Curso de Android

Ramón Alcarria, UPM
Augusto Morales, UPM

Objetivos Generales del Curso

1. Introducción a la plataforma Android: estado actual, evolución de la plataforma.

1. Plataforma Android desde el punto de vista de desarrollador: entorno de programación y publicación.

1. Programación básica en Android: GUI, Ciclo de Vida, APIs etc.

1. Introducción a herramientas avanzadas de programación.

1. Introducción a las librerías de Realidad Aumentada/Mixta Existentes.

Programa

	Introducción del curso	APIs para el manejo del telefono	GUI básico y avanzado	Multimedia y almacenamiento	Aplicaciones Avanzadas I	Aplicaciones Avanzadas II
	Instalación y configuración	SMS, configuración teléfono, red, GPS, NFC y acceso a internet	Views/Layouts	Manejo de la cámara (Fotos y Video)	Listas/Adaptadores	Repaso App avanzadas. Testing y debugging
	DESCANSO					
	Diferencias con Java SE/Java EE	Parsing de mensajes: DOM, SAX y JSON	Componentes gráficos	Threads en Android	Broadcast Receivers	Modelo de negocio Publicidad, Publicación en Google Play
	"Hola mundo" y elementos Android (Activity, Intent, Service, etc)	Invocación a servicios Web y REST.	Menus/ Dialogos	Escritura de archivos, sistema de ficheros	Realidad aumentada y Java Native Interface	Ejercicios y propuesta de proyecto
	DESCANSO					
	Salida por pantalla y trazas	Localización y mapas	Widgets	Bases de datos (SQLite)	Comunicación con HTML/Javascript	

Programa – Día 1 - Introducción

- **Introducción del curso**
  - Objetivos, materiales y planificación.
  - Introducción a Android, características, versiones.
- **Instalación y configuración**
  - Instalación del entorno de programación de Android en Eclipse.
  - Configuración del simulador y conexión con el móvil.
- **Diferencias con Java SE/ Java EE**
  - Repaso de Java SE y Nuevas APIs Android
  - Cómo funciona Android: Manifiesto, ciclo de vida, manejo de trazas, compilación, permisos, etc.
- **"Hola mundo" en Android y elementos**
  - Nuestro primer programa
  - Activities, Layouts, Intents, Background Services, Notifications

Programa – Día 2 - Comunicaciones

- **APIs Generales de Android**
  - APIS para el manejo del teléfono
  - SMS, configuración teléfono, red, GPS, NFC y acceso a internet.
- **Parsing de mensajes**:
  - DOM, SAX y JSON
- **Invocación a servicios Web y REST.**
- **Servicios de localización y de mapas**
  - Librerías de Google Maps
  - Otros servicios de Mapas

Programa – Día 3 - GUI

GUI en Androids:

- Declarative User Interface
- Programmatic User Interface
- Utilides de ambas GUI

- **GUIs básicas en Android**
  - Views y layouts
  - Componentes gráficos
- **GUIs avanzadas**
  - Menus y diálogos
  - Widgets

Programa – Día 4 - Multimedia

- **Soporte Multimedia**
  - Manejo de la cámara
  - Formatos de audio
  - Reproducción de audio y video
- **Threads en Android**
  - Utilización de Threads, AsyncThreads, Threads en Servicios/Actitivies
- **Almacenamiento de contenido**
  - Escritura en archivos, sistema de ficheros.
  - Bases de datos (SQLite): Creación, arranque, inserciones y eliminaciones, búsquedas, etc.

Programa – Día 5 – Aplicaciones Avanzadas

- **Aplicaciones avanzadas**
- **Comunicación con interfaces HTML/JavaScript**
- **El NDK de Android**
  - Cuándo desarrollar utilizando NDK
  - Ventajas/Desventajas
- **Comunicación con interfaces nativas (JNI)**
- **Listas, Adaptadores y Broadcast Receivers.**
- **Realizada aumentada**
  - Librerías Qualcomm
  - NyARToolkit.EN

Programa – Día 6

- **Testing y debugging**
  - Unit Test
  - Herrameintas de desarrollo Android (ADT)
  - Versiones android “cocinadas”, bootmanager, recursos avanzados , APIs ocultos etc.
- **Modelo de negocio:**
  - Añadir publicidad a las aplicaciones
  - Subir y promocionar aplicaciones en Google Play Store.
- **Ejercicios y propuesta de proyecto**

Introducción a Android

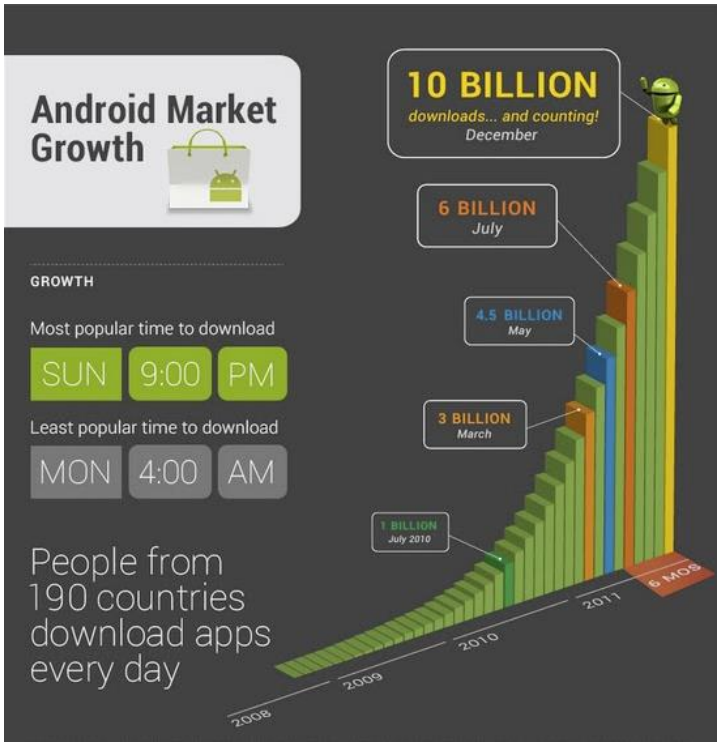
Ramón Alcarria, UPM
Augusto Morales, UPM

Historia

- Julio 2005
  - Google adquiere Android, Inc. ■Pequeña empresa que desarrolla software para móviles (hasta entonces una gran desconocida)
- Noviembre 2007 Nace la Open Handset Alliance
  - Consorcio de empresas unidas con el objetivo de desarrollar estándares abiertos para móviles
  - Texas Instruments, Broadcom co., Google, HTC, Intel, LG, Marvel Tech., Motorola, Nvidia, Qualcomm, Samsumg Electonics, Sprint Nextel, T-Mobile
  - **Se anuncia su primer producto, Android, plataforma para móviles construida sobre el kernel de Linux 2.6**

Historia

- Octubre 2008 Publicado el proyecto open source Android
  - Distribuido principalmente con licencia Apache 2.0
    Partes en otras licencias, p.e. GPL v2 para el núcleo
  - Se abre el Android Market
  - HTC Dream (G1), primer teléfono con Android

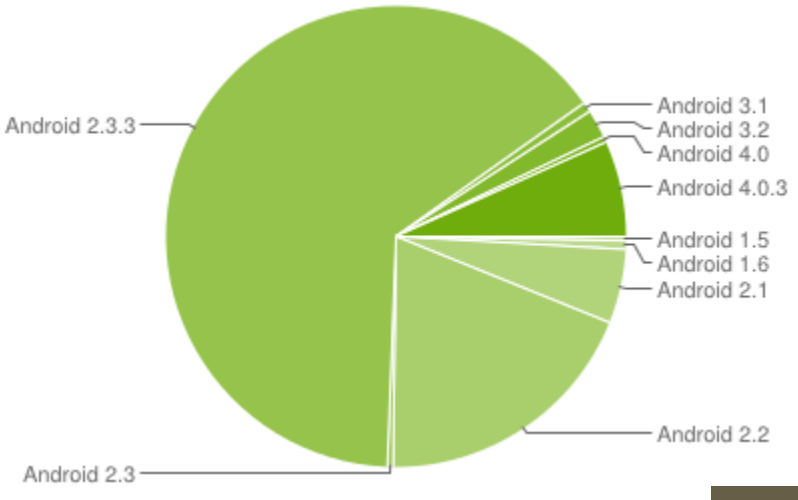


Versiones y APIs de Android

- Octubre 2008 – Android 1.0
  - API Level 1
- Febrero 2009 – Android 1.1
  - API Level 2
- Abril 2009 – Android 1.5 – Cupcake
  - API Level 3 : copy-paste, soporte mp4and stereo audio streaming via Bluetooth, widgets
- Septiembre 2009 – Android 1.6 – Donut
  - API Level 4: speech, screen resolution to 800 X 480, optimización de camera, CDMA.
- Noviembre 2009 – Android 2.0 – Éclair
  - API Level 5 : Advanced messaging, soporte HTML 5 , mejoras en contrast ratio y en la cámara.
- Diciembre 2009 – Android 2.0.1
  - API Level 6
- Enero 2009 - Android 2.1

Versiones y APIs de Android

- Diciembre 2010 – Android 2.3 - Gingerbread
  - API Level 10 - Power Management, NFC, Sensores, Mejora en recolector de basura etc.
- Febrero 2011 – Android 3.x - Honeycomb
  - API Level 11,12,13 – Enfocado en Tablets, Pantallas de gran resolución, Multicore, aceleración por hardware etc.
- Diciembre 2011 – Android 4.0.3 – Ice Cream Sandwich
  - API Levell 14 y 15: Unifica 3.x y 2.x. Mejora de notificaciones,



Características Principales

•Framework de aplicación que habilita la reutilización y reemplazo de componentes
• Máquina virtual Dalvik optimizada para móviles
• Navegador integrado basado en WebKit
• Gráficos optimizados por una librería gráfica 2D propia; gráficos 3D basados en la especificación OpenGL ES 1.0
•SQLite para almacenamiento de datos estructurados

Características Principales

- Soporte para gran variedad de archivos multimedia (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- Telefonía GSM
- Bluetooth, EDGE, 3G y WiFi (4G, WiMAX,…)
- Cámara, GPS, compás, acelerómetro,…
- Entorno de desarrollo completo incluyendo emulador, herramientas de depuración, profiling de memoria y rendimiento y plugin para el IDE Eclipse

Arquitectura de Android

Instalación y Configuración del Entorno

Materiales y Software

Entornos Principal:

•Android SDK : Provee las librerías y el entorno necesario para comenzar a programar.

•Android NDK: librerías avanzadas de programación

Sistemas Operativos:

•Windows XP, Vista, 7.

•Linux

•Mac

Frameworks de Desarrollo Compatibles

•Eclipse

•IntroAndroidDevNetBeans

•IntelliJ IDEA 10

Android SDK

- Instalar Android SDK:

http://developer.android.com/sdk/index.html

  - Repaso de Funcionalidades, APIs etc.
  - Descarga de Imágenes de OS

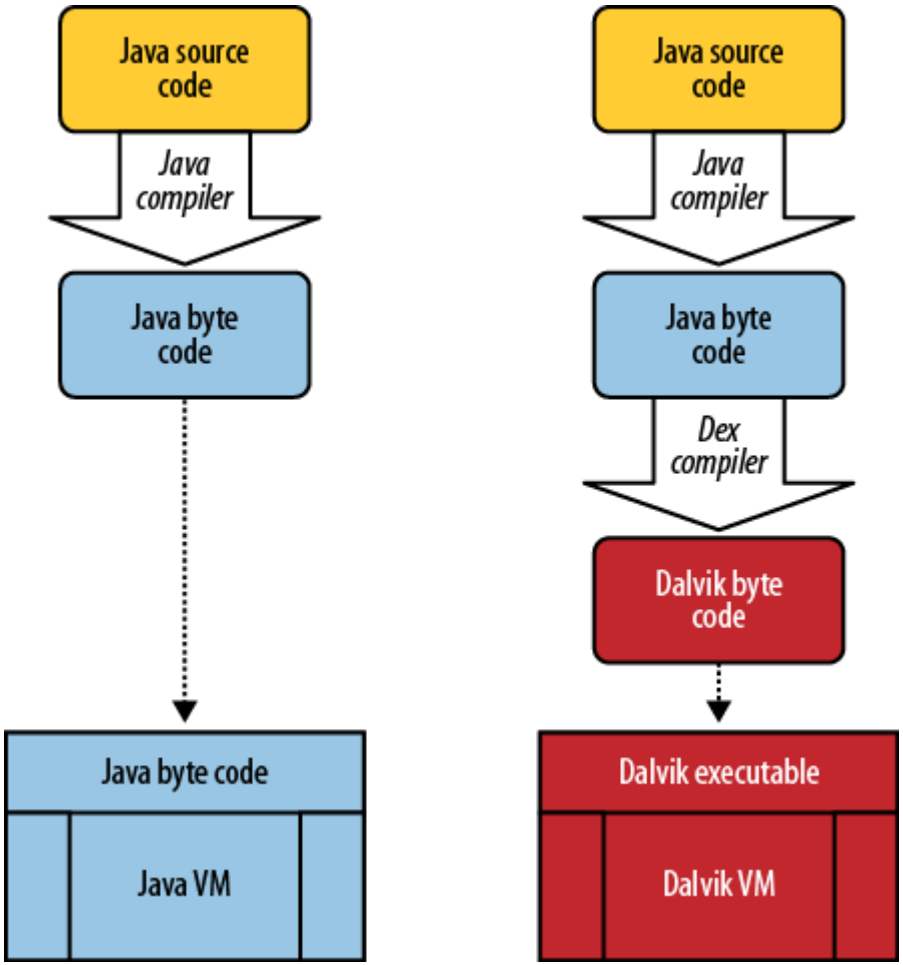
- Instalar los plugins de Eclipse, y el Android Debugging Tool
- http://developer.android.com/guide/developing/tools/adb.html
- http://developer.android.com/sdk/eclipse-adt.html

  - Toma de Contacto con Emulador

Aplicaciones en Android

Java SE vs Android

- Android utiliza Java para syntaxis & bytecode
- Las librerías más cercanas son Java Standard Edition JSE



Java SE vs Android

- Ciclo de vida distinto.
- Acceso a recursos hardware distinto (Android está pensado para ello)
- GUI. Android no utiliza SWING/AWT ni librerías de J2ME, en vez de esto tiene otros mecanismos. (Muy eficientes).

Java SE vs Android

APIS heredados:

- java.awt.font.* Para definir propiedades del estilo y tamaño de la letra. (Subconjunto de J2SE)
- java.beans.*: Para definir beans con los métodos get y set.(Subconjunto de J2SE)
- java.io.*: Conexión genérica.
- java.lang.* Clases de la VM. (Subconjunto de J2SE)
- java.math.* Para operaciones matemáticas.
- java.nio.* Para definir buf

Aplicaciones

- Las aplicaciones están escritas en Java y empaquetadas en Android package (APK)
  - Cada aplicación es independiente
  - Se ejecuta en su propio proceso de Linux
  - Cada proceso tiene su propia máquina virtual de Java
  - Cada aplicación tiene asignado un ID de usuario de Linux
  - Solo ese ID de usuario tiene permisos para acceder a los archivos de la aplicación
- Como cada uno de los procesos de los APK están separados; la seguridad en android puede volverse demasiado restrictiva.

Aplicaciones

Estructura .APK (ZIP) se compone de los siguientes archivos:

- classes.dex
  - Dalvik: código ejecutable por la máquina virtual
- res (recursos)
- META-INF (firmas de la aplicacion)
- AndroidManifest.xml (compilado)
- resources.arsc (relaciona los

Aplicaciones

Estructura .APK (ZIP) se compone de los siguientes archivos:

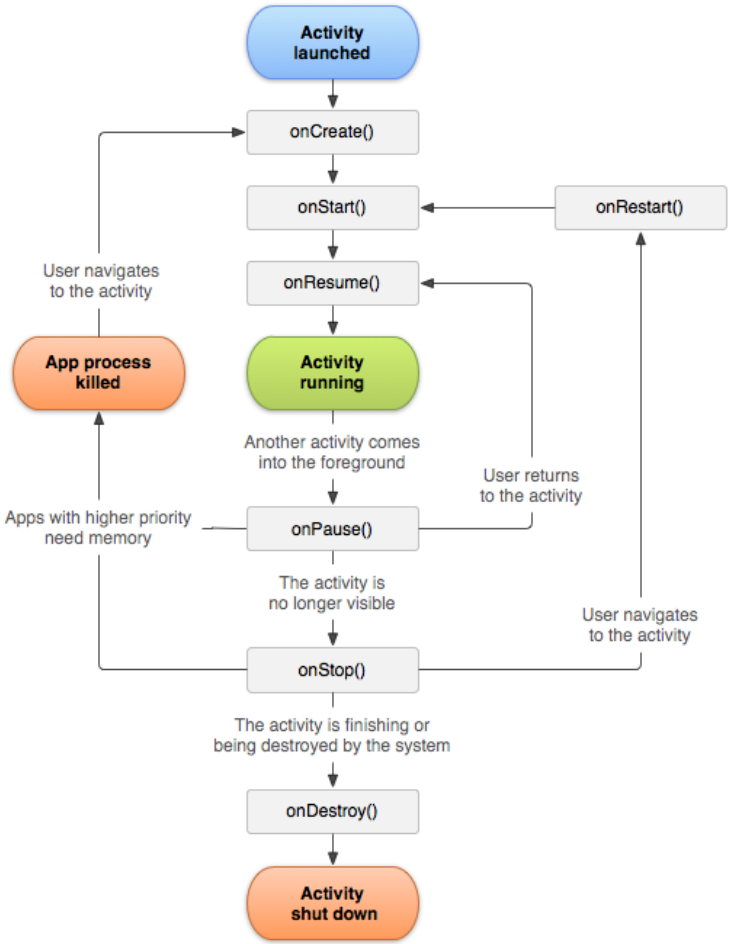
- classes.dex
  - Dalvik: código ejecutable por la máquina virtual
- res (recursos)
- META-INF (firmas de la aplicacion)
- AndroidManifest.xml (compilado)
- resources.arsc (relaciona los

Conceptos Básicos

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

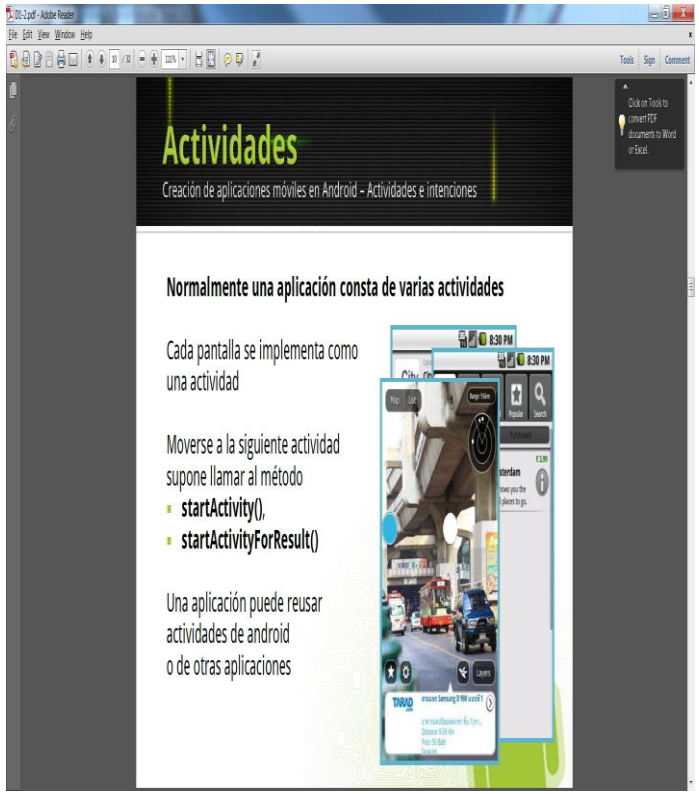
Activity

•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



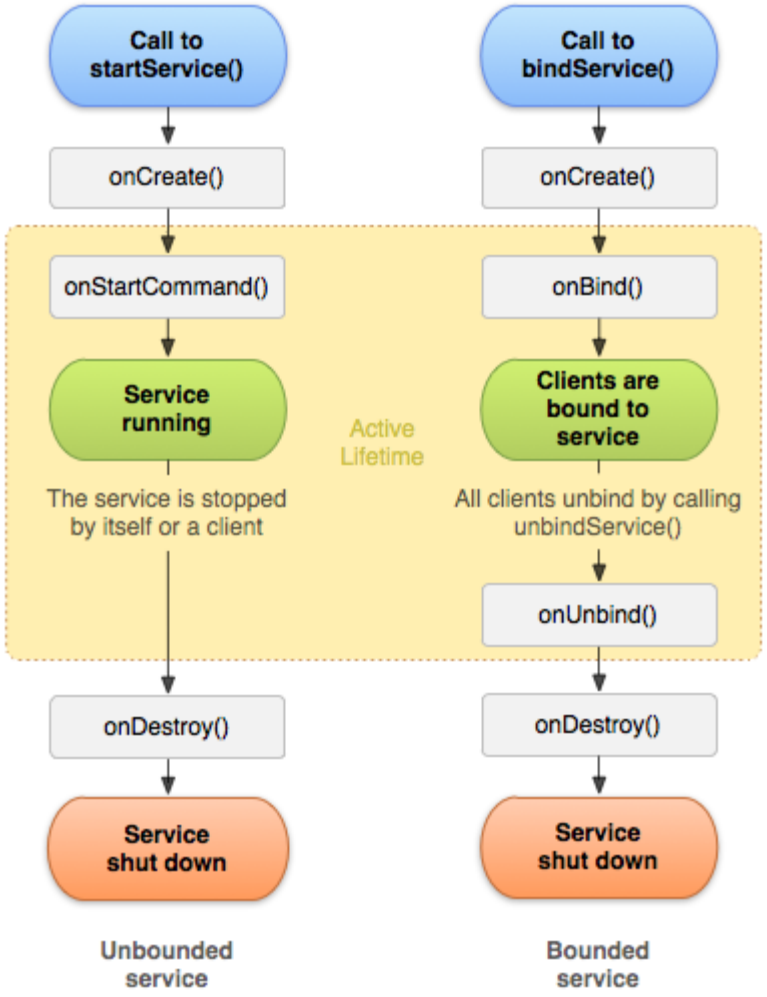
Activity Aplicaciones

- **Normalmente una aplicación consta de varias actividades** Cada pantalla se implementa como una actividad Moverse a la siguiente actividad supone llamar al método **startActivity()**, **startActivityForResult()**
- Una aplicación puede reusar actividades de android o de otras aplicaciones



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



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.
- Son descripciones abstractas de lo que se desea ejecutar

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*).

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.

Manifiesto en Android

Antes de que el sistema Android puede iniciar un componente, el sistema debe conocer la existencia del mismo mediante el . AndroidManifest.xml La aplicación debe declarar todos sus componentes en este archivo- Identificar los permisos de usuario de la aplicación que la aplicación requiere; tales como acceso a Internet o acceso de lectura a los contactos del usuario.

- Declarar el nivel mínimo exigido por la API de la aplicación.
- Declarar funciones de hardware y software utilizados o requeridos por la aplicación, como una cámara, los servicios de Bluetooth o una pantalla multitáctil.
- Librerías que deben ser enlazadas. (Google Map lib)

http://developer.android.com/guide/topics/

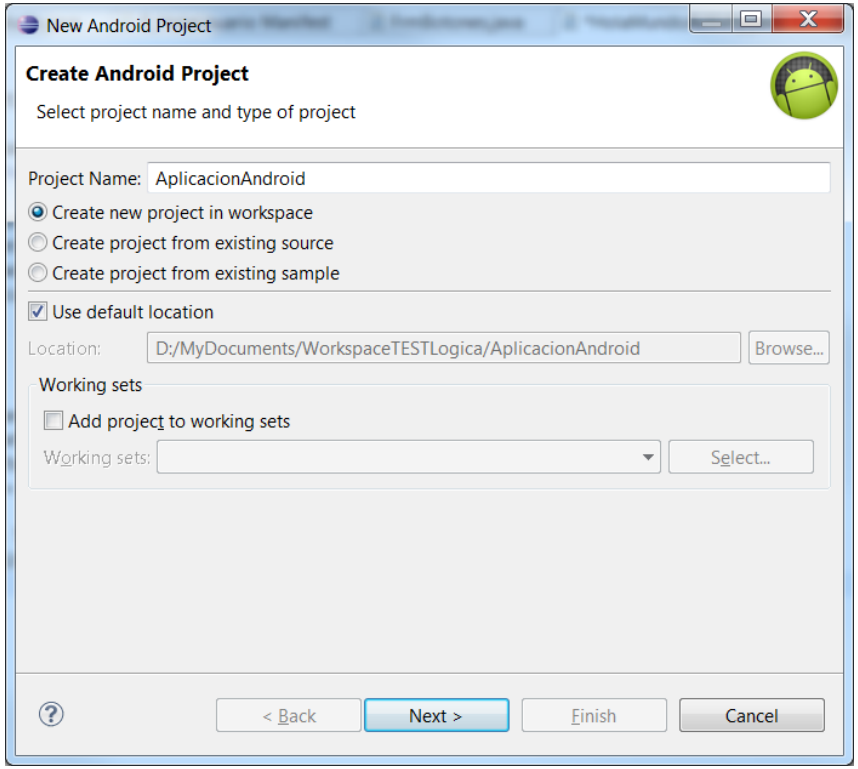
Permisos

- El modelo de seguridad de las aplicaciones de android obliga a que se declaren explícitamente los recursos requeridos por una aplica
- Ejemplos:
- <uses-permission android:name="android.permission.INTERNET"></uses-permission>
- <uses-permission android:name="android.permission.VIBRATE"></uses-permission>
- <uses-permission android:name="android.permission.CAMERA"></uses-permission>
- <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
- <uses-permission android:name="android.permission.NFC" />

Ejemplo de Aplicación

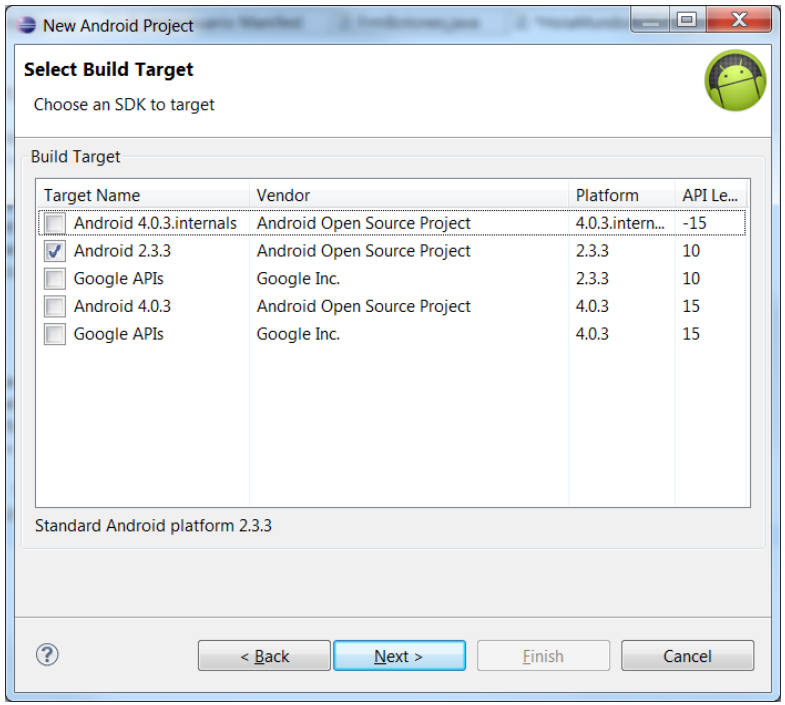
Ejemplo 1 – Hello World

1. Crear un proyecto Android: en El Eclipse – New - Android Project



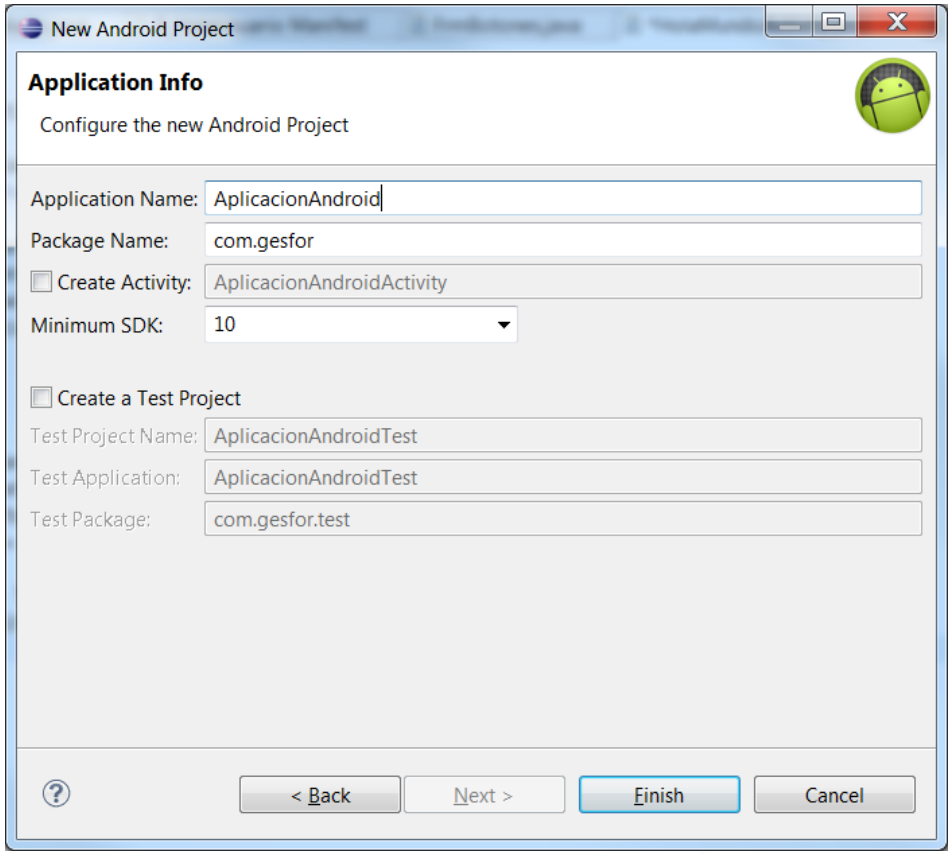
Ejemplo 1 – Hello World

1. Elección de API



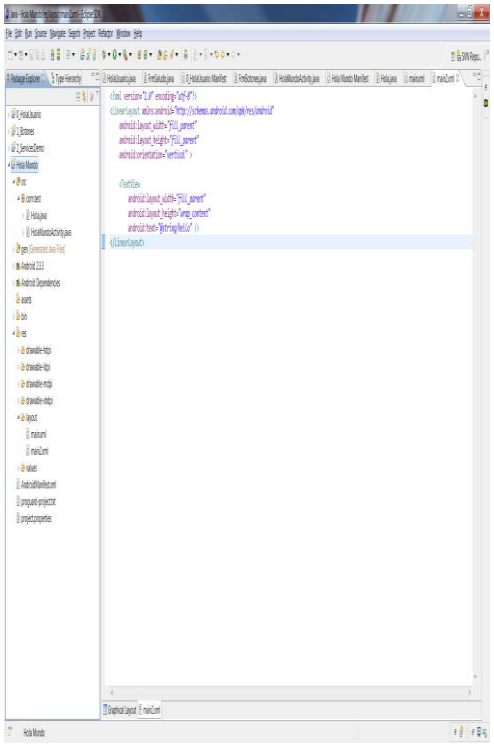
Ejemplo 1 – Hello World

1. Configuración de Aplicacion



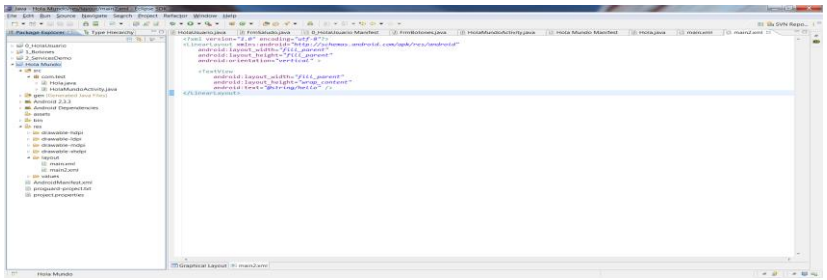
Estructura de Directorio

- Cuando creamos un nuevo proyecto Android en Eclipse se genera automáticamente la estructura de carpetas necesaria para poder generar posteriormente la aplicación. Esta estructura será común a cualquier aplicación, independientemente de su tamaño y



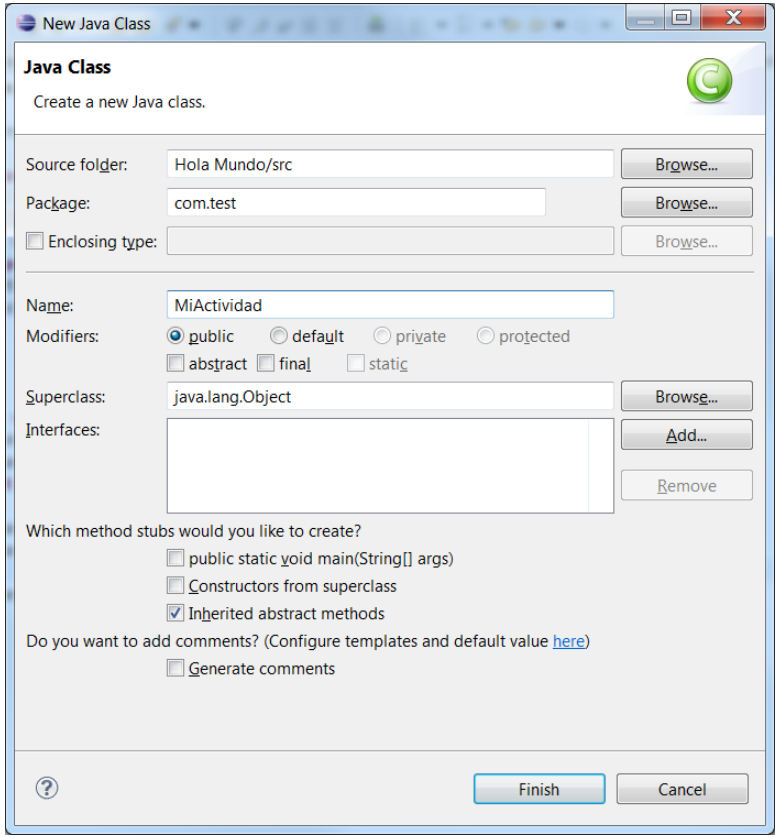
Estructura de Directorio – Carpeta SRC

- Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc.



Ejemplo 1 – Hello World

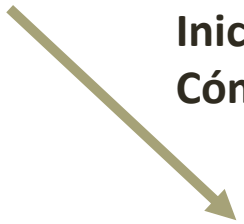
1. Crear Código de actividad – New- Class



Ejemplo 1 – Hello World

```
package com.test;

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

public class Hola extends Activity{

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

}

}
```

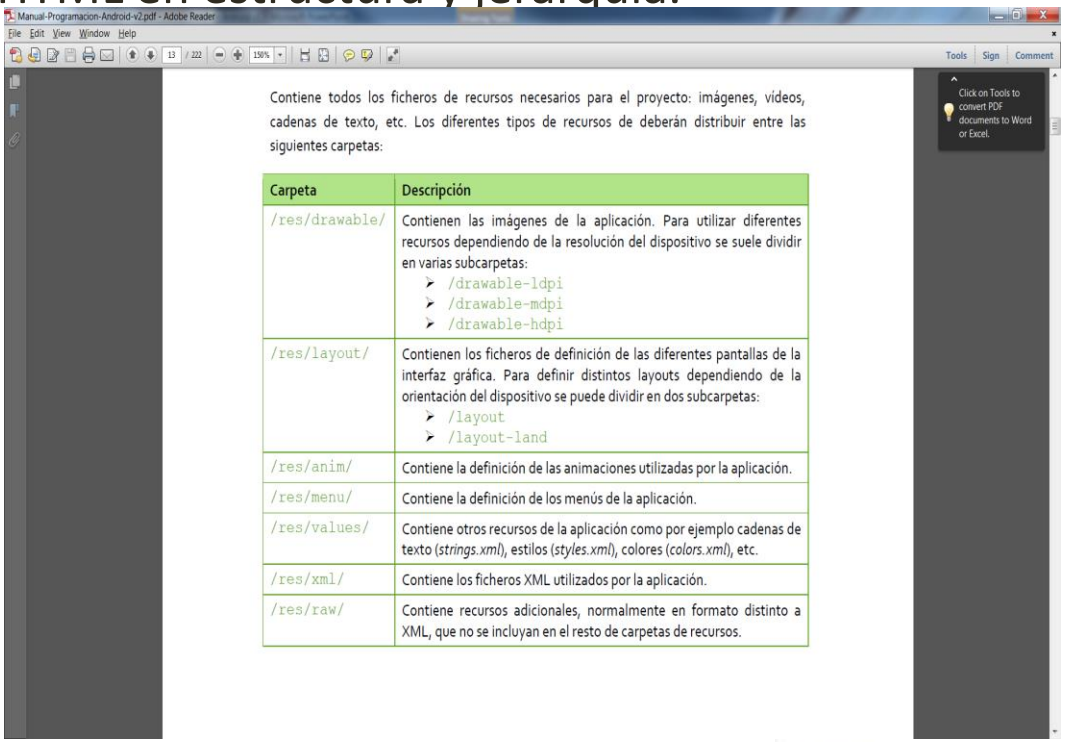
Establece el layout
Inicial de la aplicacíon
Cómo se verá



Ejemplo 1 – Hello World

¿Qué es un Layout?

- Es una representación en XML de los objetos que aparecerán en la GUI.
- Similar a HTML en estructura y jerarquía.



Manual-Programacion-Android-v2.pdf - Adobe Reader

File Edit View Window Help

Tools Sign Comment

Click on Tools to convert PDF documents to Word or Excel.

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos de deberán distribuir entre las siguientes carpetas:

Carpeta	Descripción
/res/drawable/	Contienen las imágenes de la aplicación. Para utilizar diferentes recursos dependiendo de la resolución del dispositivo se suele dividir en varias subcarpetas: ➢ /drawable-ldpi ➢ /drawable-mdpi ➢ /drawable-hdpi
/res/layout/	Contienen los ficheros de definición de las diferentes pantallas de la interfaz gráfica. Para definir distintos layouts dependiendo de la orientación del dispositivo se puede dividir en dos subcarpetas: ➢ /layout ➢ /layout-land
/res/anim/	Contiene la definición de las animaciones utilizadas por la aplicación.
/res/menu/	Contiene la definición de los menús de la aplicación.
/res/values/	Contiene otros recursos de la aplicación como por ejemplo cadenas de texto (*strings.xml*), estilos (*styles.xml*), colores (*colors.xml*), etc.
/res/xml/	Contiene los ficheros XML utilizados por la aplicación.
/res/raw/	Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.

Ejemplo 1 – Hello World

1.Irse a RES – LAYOUT - y crear main.xml

•<?xml version="*1.0*" *encoding="utf-8"*?>
•<LinearLayout
xmlns:android="*http://schemas.android.com/apk/res/android*"
- android:layout_width="*fill_parent*"
- android:layout_height="*fill_parent*"
- android:orientation="*vertical*" >

- <TextView
- android:layout_width="*fill_parent*"
- android:layout_height="*wrap_content*"
- android:text="*@string/hello*" />

•</LinearLayout>

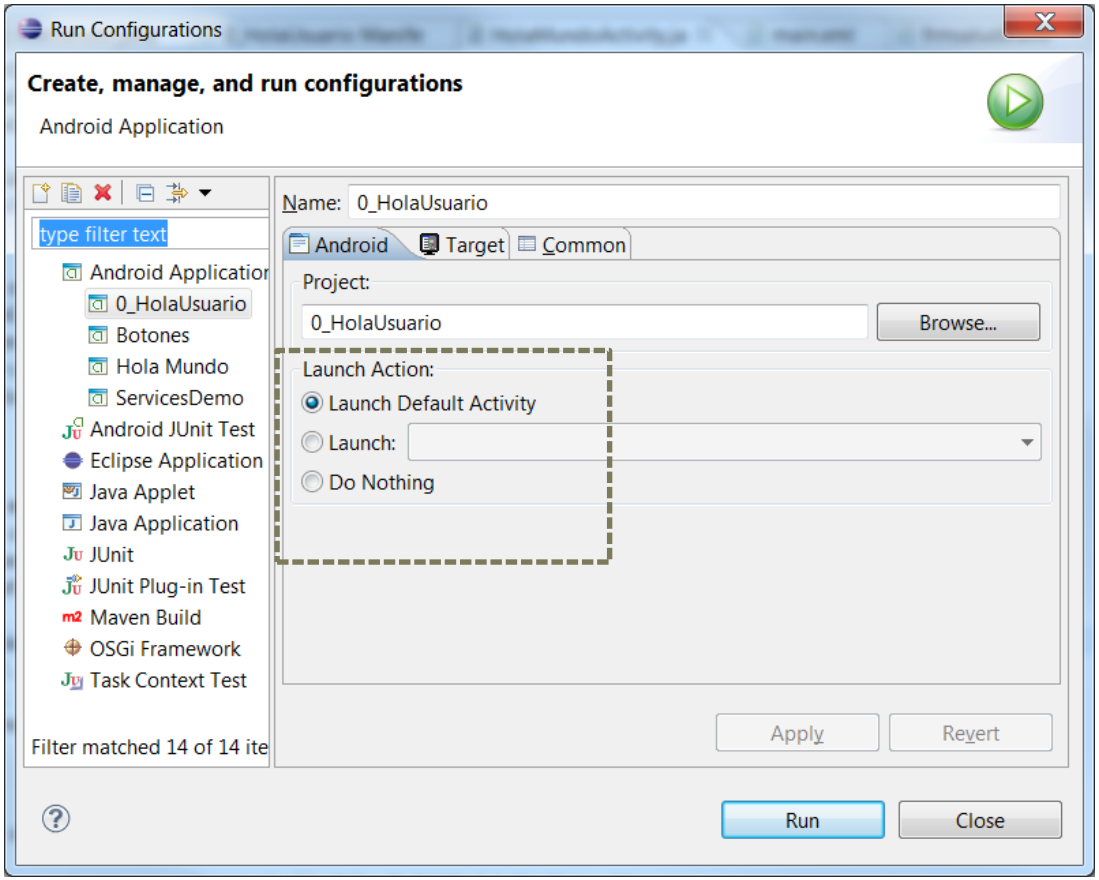
Ejemplo 1 – Hello World

1.Irse a RES – LAYOUT - y verificar strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="hello">Hola Logica!</string>
    <string name="app_name">Aplicacion Hola Mundo</string>

</resources>
```

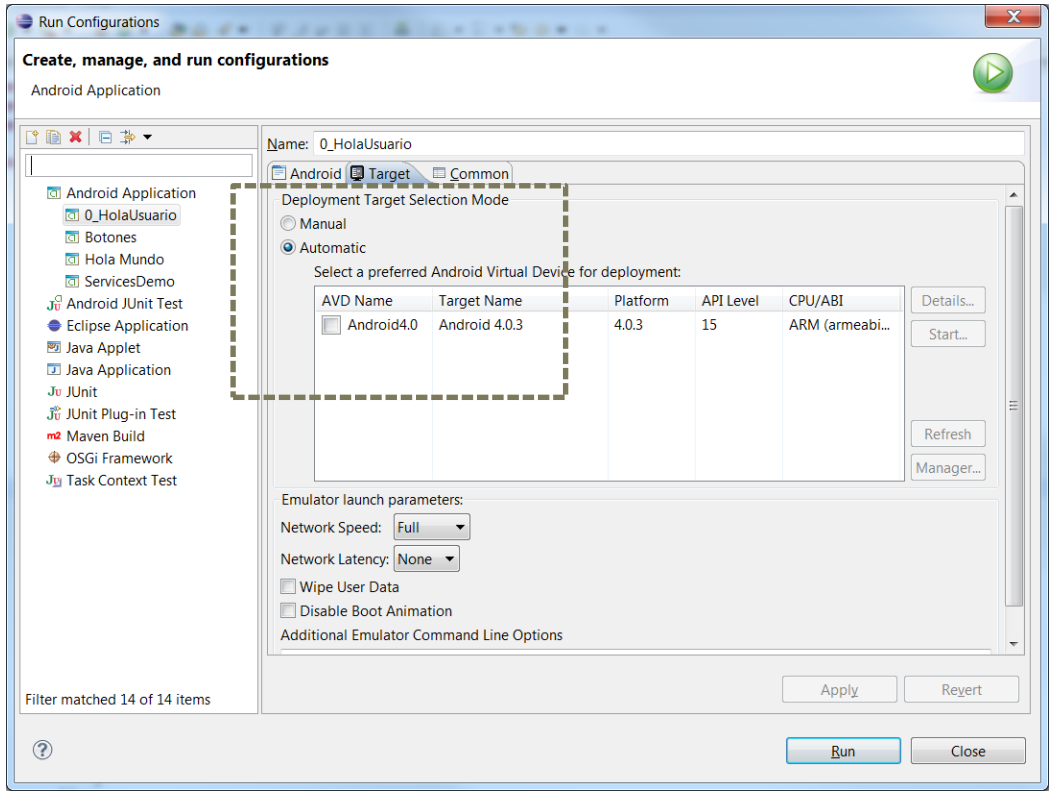
Ejemplo 1 – Ejecución

1.Irse al proyecto - Click derecho – Run Configuration:



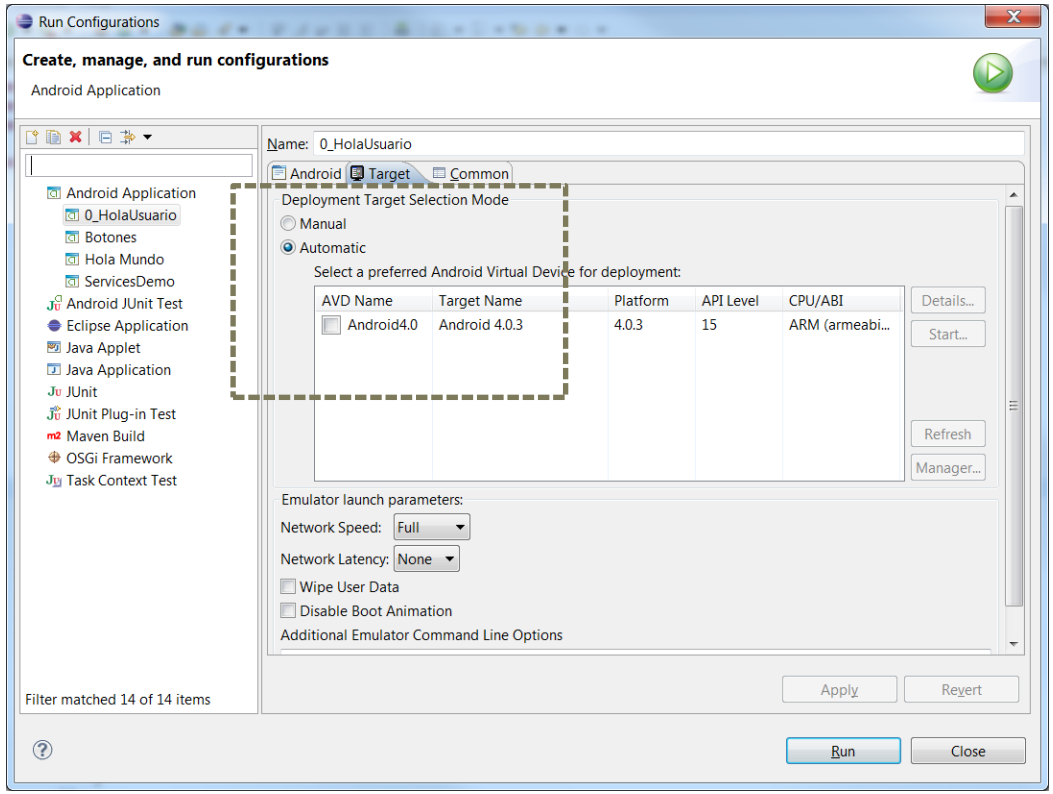
Ejemplo 1 – Ejecución

1.Irse al proyecto - Click derecho – Run Configuration:



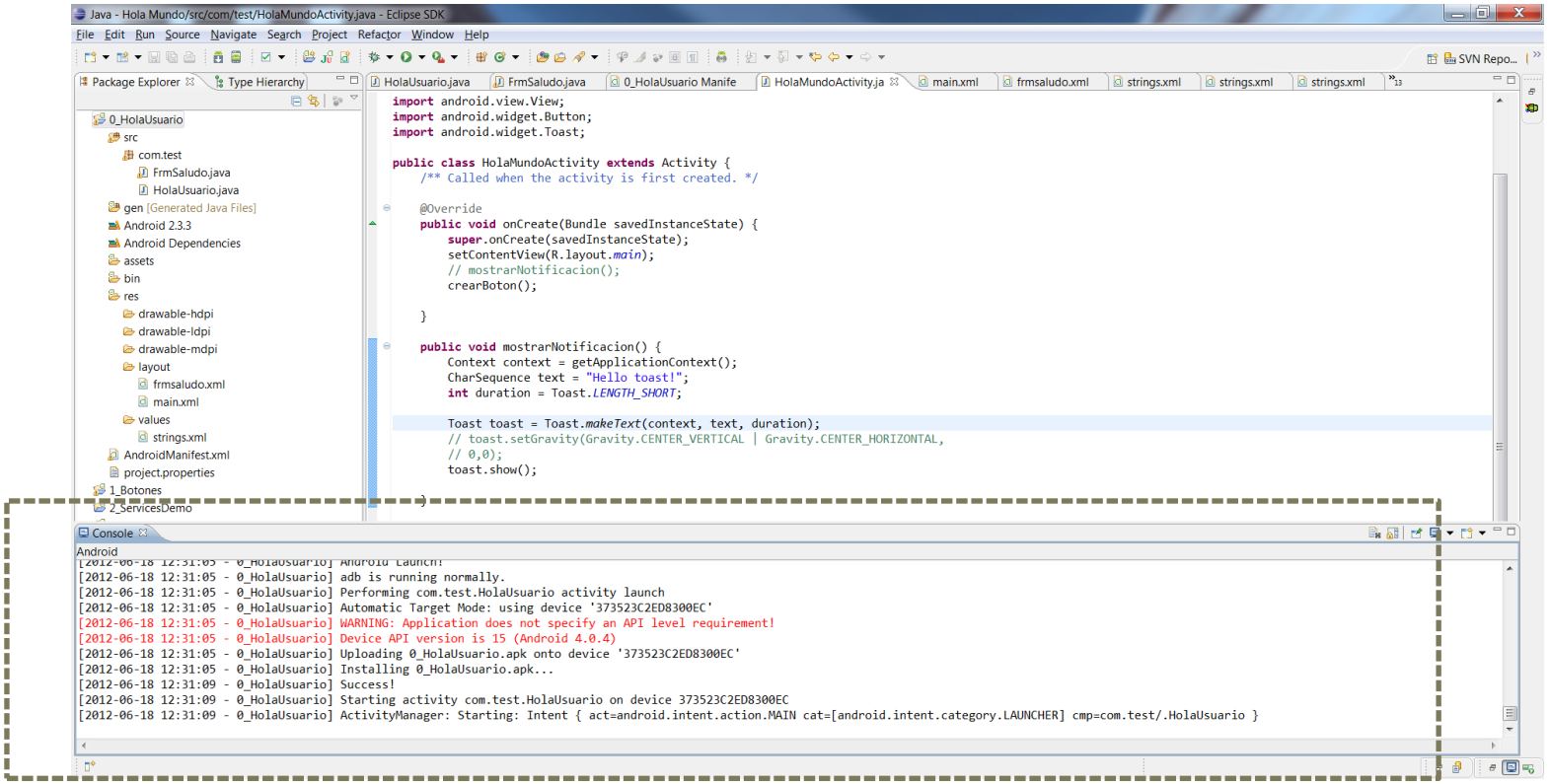
Ejemplo 1 – Ejecución

1.Irse al proyecto - Click derecho – Run Configuration:



Ejemplo 1 – Ejecución

1.Log



Ejemplo 2 – Hello World con Boton

1.Definir identificador Boton en layout

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

2. Código Java

```
final Button button = (Button) findViewById(R.id.BtnBoton1);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //Código a Ejecutar… Ej.  Llamar a servidor web, capturar datos etc..   button.setEnabled(false);

            }
        });
```

Ejemplo 3 – Hello World con Boton y Notificación

1.Notificaciones

```
Context context = getApplicationContext();
CharSequence text = "Hello Nuevamente!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
 toast.show();
```

Ejemplo 3 – Hello World con Boton y Notificación

```
package com.test;


import android.app.Activity;


public class HolaMundoActivity extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
mostrarNotificacion();
crearBoton();
}
public void mostrarNotificacion() {
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;
Toast toast = Toast.makeText(context, text, duration);
toast.show();
}
public void crearBoton() {
final Button button = (Button) findViewById(R.id.BtnBoton1);
button.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
// Perform action on click
mostrarNotificacion();
// button.setEnabled(false);
}
}); }}
```

Recursos compilados

Carpeta: /gen/
Contiene una serie de elementos de código generados automáticamente al compilar el proyecto.

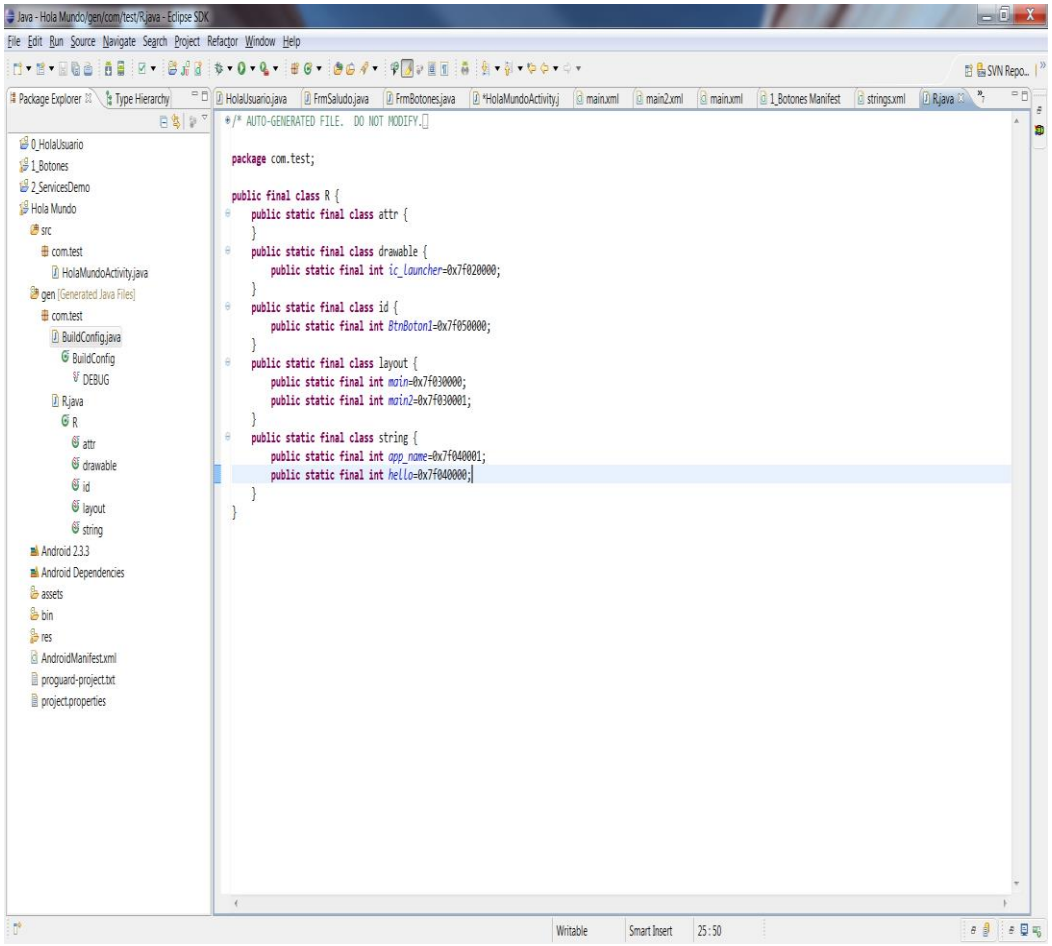
• Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente en java dirigidos al control de los recursos de la aplicación.

•Esta clase R contendrá en todo momento una serie de constantes con los ID de todos los recursos de la aplicación incluidos en la carpeta /res/,

•*NO SE DEBE MODIFICAR MANUALMENTE*

•*PROVEE PISTAS PARA SABER SI EL PROYECTO SE HA COMPILADO AUTOMÁTICAMENTE*

Recursos compilados



Propuesta de Ejercicio

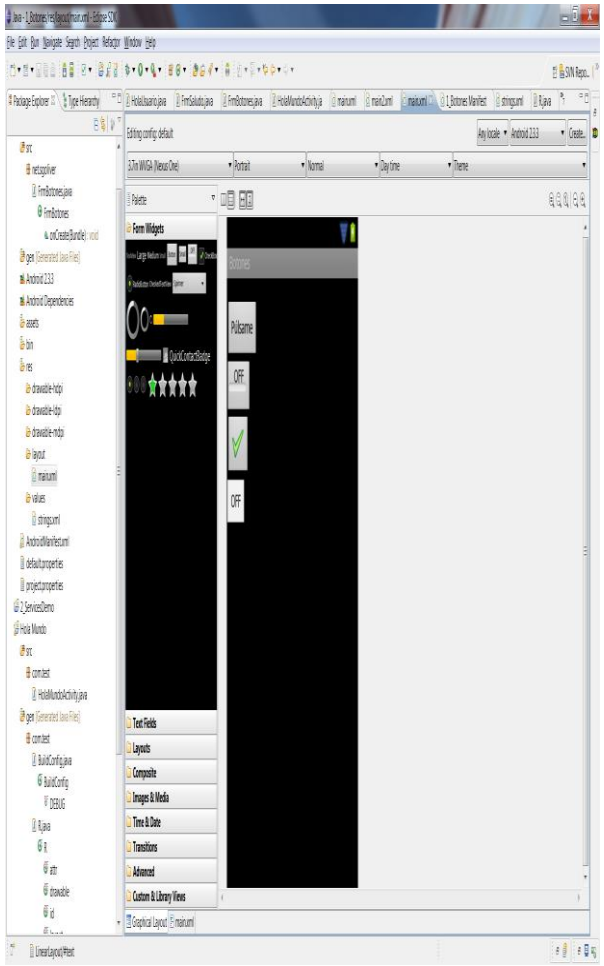
Explorar las variantes de botones:

•*Button:*
http://developer.android.com/reference/android/widget/Button.html

•*ToggleButton*
http://developer.android.com/reference/android/widget/ToggleButton.html

•*ImageButton*
http://developer.android.com/reference/android/widget/ImageButton.html

GUÍA DE FORMAS:
http://developer.android.com/resources/tutorials/views/hello-formstuff.html

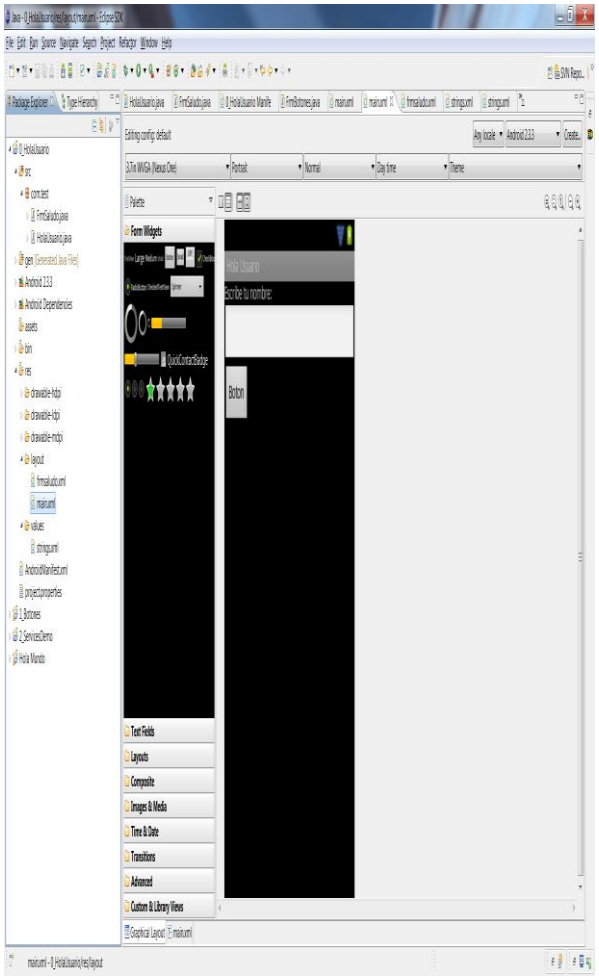


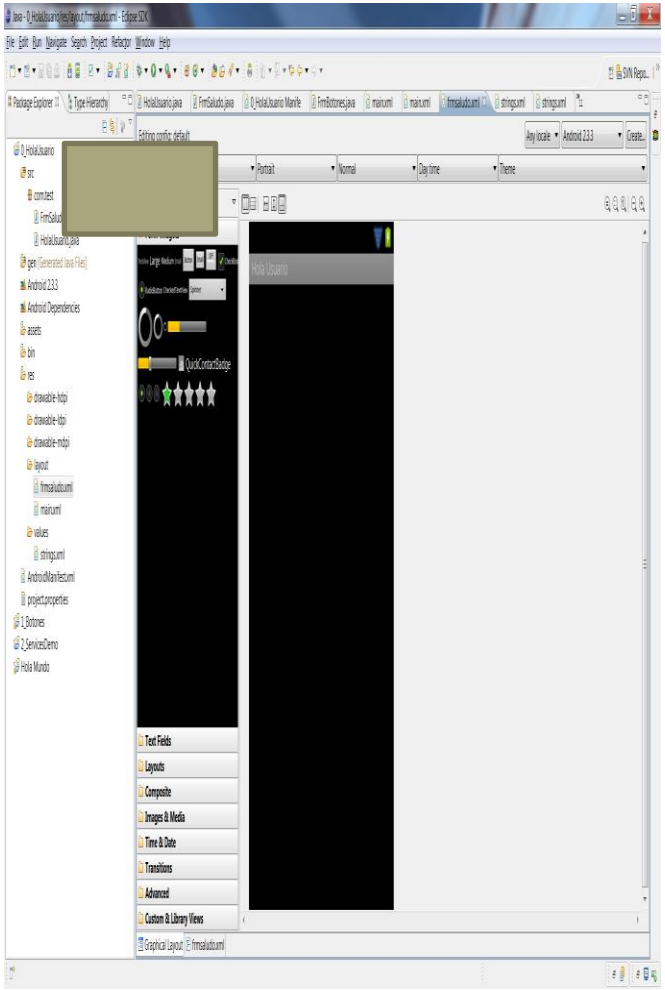
Ejemplo 4 – Crear diferentes pantallas

- Las diferentes pantallas de una aplicación Android se definen mediante objetos de tipo Activity.
- *Se definirán 2 pantallas/Pantallas y se definirán los parámetros de conexión entre ellas*
- *Crear un nuevo proyecto con los siguientes elementos:*

Actividad 1: HolaUsuario.java
Actividad 2: FrmSaludo.java
Layout 1: frmsaludo.xml
Layout2: main.xml
String: strings.xml

Ejemplo 4 – Crear diferentes pantallas





Ejemplo 4 – Crear diferentes pantallas

1.Agregar a: main.xml

```
<TextView android:id="@+id/LblNombre"
android:layout_height="wrap_content"
android:layout_width="fill_parent"
   android:text="@string/nombre" />

<EditText android:id="@+id/TxtNombre"
android:layout_height="wrap_content"
android:layout_width="fill_parent" />
```

Ejemplo 4 – Crear diferentes pantallas

1.Crear una nueva Layout: frmsaludo.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">

<TextView android:id="@+id/TxtSaludo"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />

</LinearLayout>
```

Ejemplo 4 – Crear diferentes pantallas

1.Modificar res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="nombre">Escribe tu nombre:</string>
    <string name="app_name">Hola Usuario</string>
<string name="hola">Boton</string>
</resources>
```

Ejemplo 4 – Crear diferentes pantallas

1.HolaUsuario.java contendrá en el onCreate:

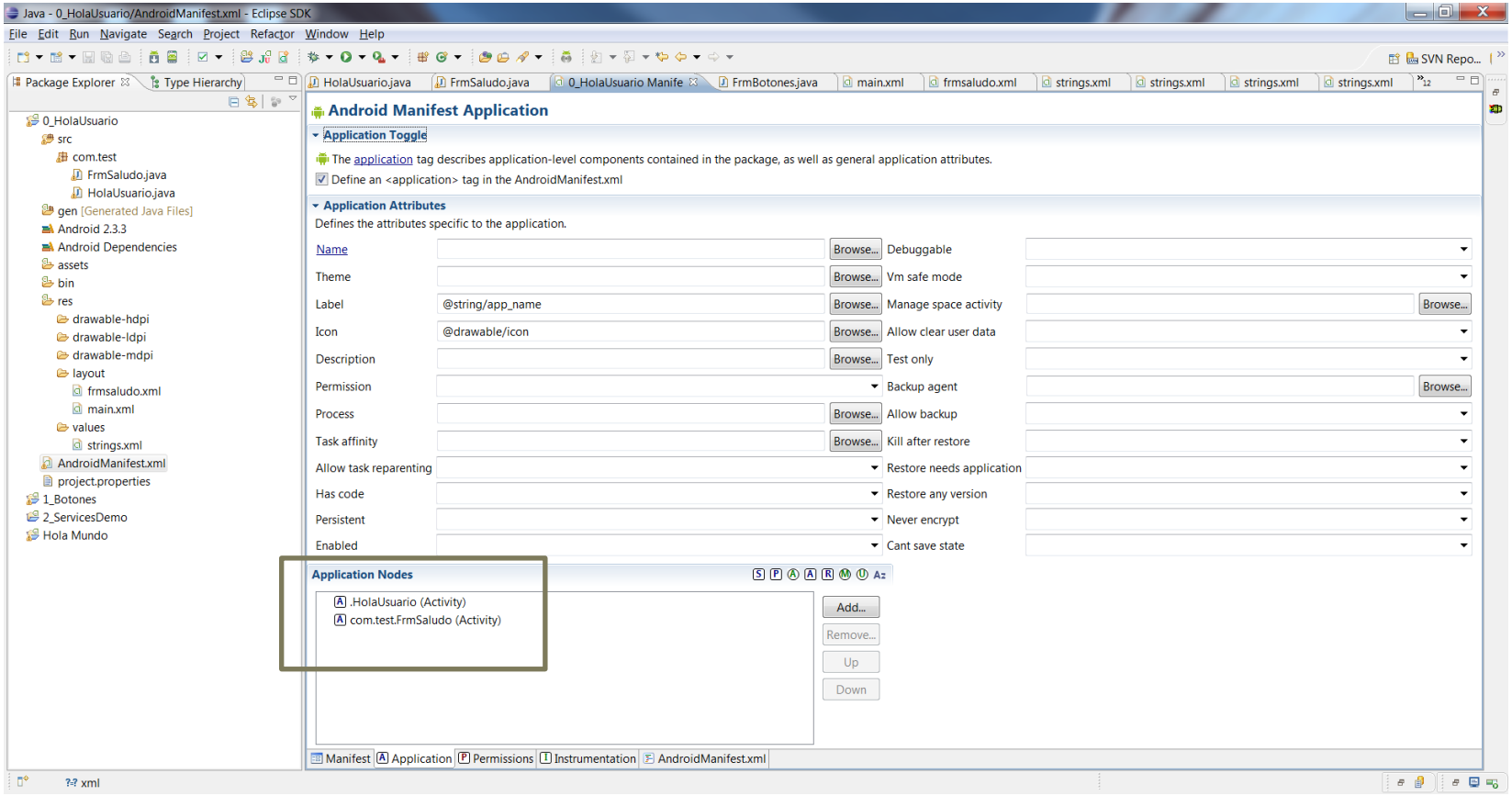
```
final EditText txtNombre =
(EditText)findViewById(R.id.TxtNombre);

final Button btnHola = (Button)findViewById(R.id.BtnHola);
      btnHola.setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View v) {
            //Creamos el Intent
         Intent intent = new Intent(HolaUsuario.this,
FrmSaludo.class);

         //Creamos la información a pasar entre actividades
         Bundle b = new Bundle();
         b.putString("NOMBRE", txtNombre.get
```

Ejemplo 4 – Crear diferentes pantallas

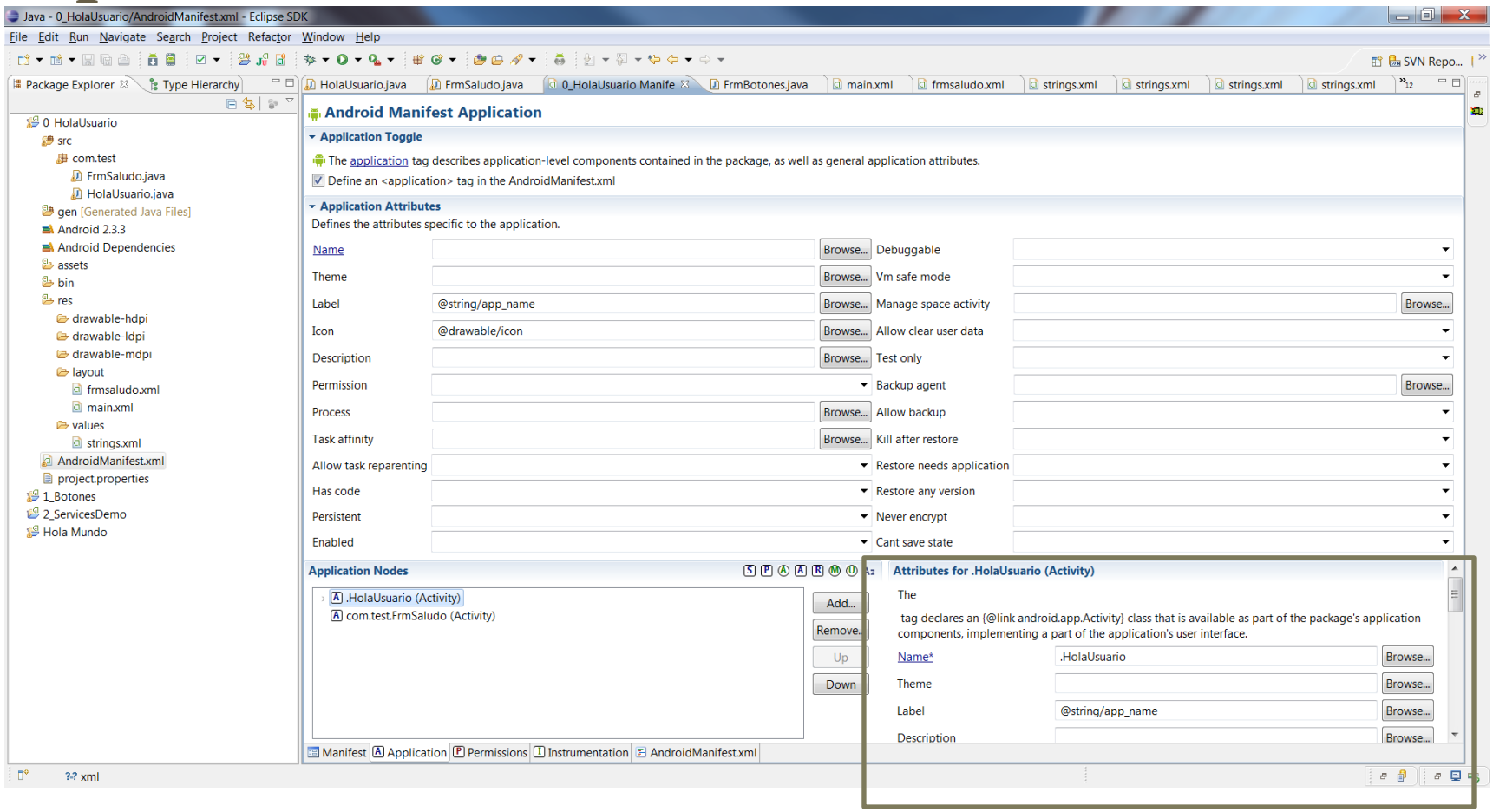
2. FrmSaludo.java contendrá:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.frmsaludo);

    //Localizar los controles
    TextView txtSaludo = (TextView)findViewById(R.id.TxtSaludo);

    //Recuperamos la información pasada en el intent
    Bundle bundle = this.getIntent().getExtras();

    //Construimos el mensaje a mostrar
    txtSaludo.setText("Hola " + bundle.getString("NOMBRE"));
}
```

Ejemplo 4 – Crear diferentes pantallas



Ejemplo 4 – Crear diferentes pantallas



Ejemplo 4 – Crear diferentes pantallas

1.El manifiest deberá reflejar las diferentes actividades:

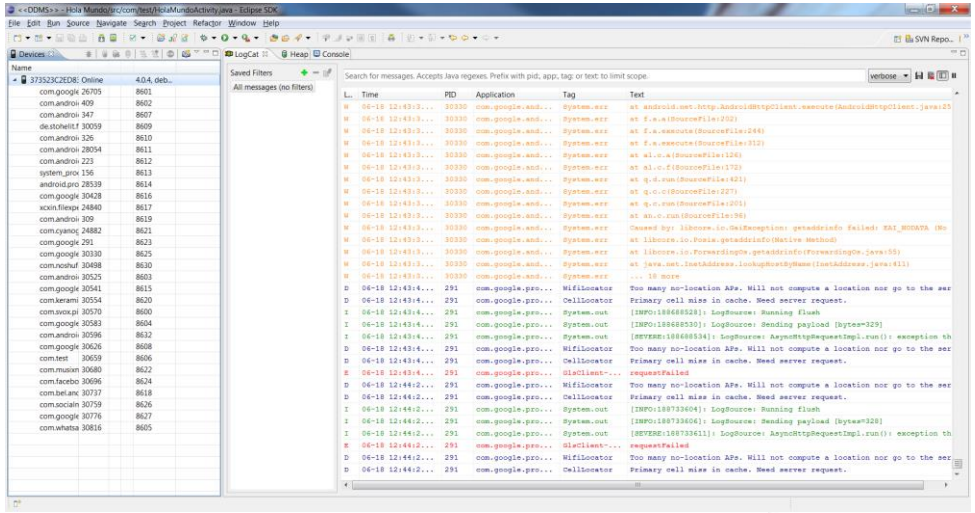
```
<?xml version="1.0" encoding="utf-8"?>

<manifest

xmlns:android="http://schemas.android.com/apk/res/android
"
    package="com.test"
    android:versionCode="1"
    android:versionName="1.0">

  <application android:icon="@drawable/icon"
  android:label="@string/app_name">

    <activity android:name=".HolaUsuario"
          android:label="@string/app_name"
```

Depuración y Logging en Android

Depuración y Logging

•Una de las técnicas más útiles a la hora de depurar y/o realizar el seguimiento de aplicaciones sobre cualquier plataforma es la creación de *logs* de ejecución. Android por supuesto no se queda atrás y nos proporciona también su propio servicio y API de *logging* a través de la clase **android.util.Log**

•**Windows-OpenPerspective-Other-DDMS (Dalvik Debug Monitor Service)**

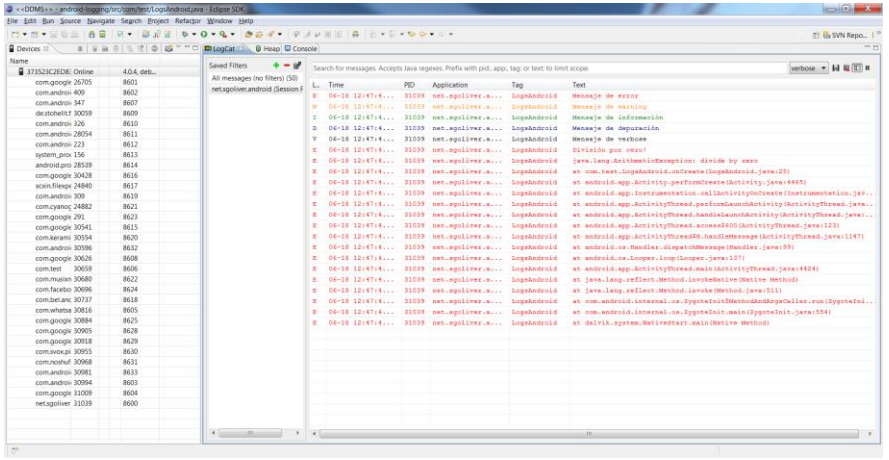


Depuración y Logging

- De forma similar a como ocurre con otros frameworks de logging, en Android los mensajes de log se van a clasificar por su criticidad, existiendo así varias categorias (ordenadas de mayor a menor criticidad):
  - Error
  - Warning
  - Info
  - Debug
  - Verbose
- Para cada uno de estos tipos de mensaje existe un método estático independiente que permite añadirlo al log de la aplicación. Así, para cada una de las categorías anteriores tenemos disponibles los métodos e(), w(), i(), d() y v() respectivamente

Ejemplo Debugging

```
Log.e(LOGTAG, "Mensaje de error");
    Log.w(LOGTAG, "Mensaje de warning");
    Log.i(LOGTAG, "Mensaje de información");
    Log.d(LOGTAG, "Mensaje de depuración");
    Log.v(LOGTAG, "Mensaje de verbose");

    try
    {
    int a = 1/0;
    }
    catch(Exception ex)
    {
    Log.e(LOGTAG, "División por cero!", ex);
    }
```



DDMS y sus capacidades adicionales

Además de controlar Logs, DDMS permite:

- Monitorización de procesos mediante PID, detenerlos etc..
- Creación de Filtros
- Visualizar Heap Size, Memoria disponible etc…
- Hacer tracking de tráfico de red de una aplicación
- Volcado de memoria en t

Aplicaciones tipo Activity+Service

Servicios

•Un servicio es una aplicación que se ejecuta de forma automática, sin interacción con el usuario. Desarrollan tareas importantes para el resto de las **aplicaciones** o para el sistema.
•Ejemplos de servicios son: servidores web, sistemas de comunicaciones, antivirus, etc…
•Un detalle muy importante. En Android los servicios no son independientes, corren en el mismo proceso que la aplicación que los consume. Es un cambio de filosofía, pero una vez que se asuma, no implica mayor complejidad.

http://developer.android.com/guide/topics/fundamentals/services.html

Ejemplo 5

1.Modificar el Manifiesto

<service android:enabled="*true*" *android:name=".MyService" />*

2. Crear una actividad y llamar un servicio mediante el comando:

startService(**new Intent(this, MyService.class));**

3. STICKY, NOT SICKY ETC…

Ejemplo 5

```
public class MyService extends Service {
private static final String TAG = "MyService";
MediaPlayer player;


@Override
public IBinder onBind(Intent intent) {
return null;
}


@Override
public void onCreate() {
Toast.makeText(this, "My Service Created", Toast.LENGTH_LONG).show();
Log.d(TAG, "onCreate");


player = MediaPlayer.create(this, R.raw.braincandy);
player.setLooping(false); // Set looping
}


@Override
public void onDestroy() {
Toast.makeText(this, "My Service Stopped", Toast.LENGTH_LONG).show();
Log.d(TAG, "onDestroy");
player.stop();
}


@Override
public void onStart(Intent intent, int startid) {
Toast.makeText(this, "My Service Started", Toast.LENGTH_LONG).show();
Log.d(TAG, "onStart");
player.start();
}
}
```

Broadcasts en Android

Broadcast

Un **Broadcast Receiver** es una especie de receptor de los eventos que produce el sistema operativo Android. Típicamente, un **Broadcast Receiver** se utiliza para mostrar notificaciones de los eventos que ocurren en nuestro teléfono móvil, como por ejemplo el descubrimiento de una red wifi o el agotamiento de la batería.

Ejemplo

1.Crear un Nuevo Proyecto, y una Actividad con un layout que tenga un textview.

1.Agregar Permiso Battery Statt

1.Al final del onCreate

IntentFilter filter = **new IntentFilter(Intent.*ACTION_BATTERY_CHANGED);***

registerReceiver(battery_receiver, filter);

Ejemplo

1.Ahora Crear el Broadcastreceiver

private BroadcastReceiver battery_receiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
int nivel = intent.getIntExtra(BatteryManager.*EXTRA_LEVEL, 0);*
String tecnologia = intent.getStringExtra(BatteryManager.*EXTRA_TECHNOLOGY);*
int estado = intent.getIntExtra(BatteryManager.*EXTRA_STATUS, 0);*

String info = "Nivel: " + nivel + "%\n";

info += ("Tecnologia: " + tecnologia + "\n");

Actualizar el TextView con la info
}};

Ejemplo

1.En el onDestroy no olvidar el

unregisterReceiver(battery_receiver);

1.Opcional
int estado = intent.getIntExtra(BatteryManager.*EXTRA_STATUS, 0);*

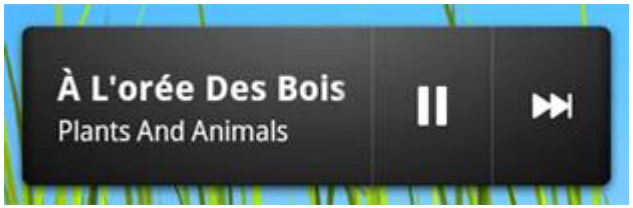
```
private String obtenerEstado(int estado){
String plugType = "Unknown";

switch(estado)
{
case BatteryManager.BATTERY_PLUGGED_AC: plugType = "AC";break;
case BatteryManager.BATTERY_PLUGGED_USB: plugType = "USB";break;
case BatteryManager.BATTERY_STATUS_DISCHARGING: plugType = "Descargando";break;
}

return plugType;
}
```

Widgets en Android

Widgets

Los **Widgets** son aplicaciones en miniaturas que pueden ser “embebidas” en otras aplicaciones (Como la pantalla HOME) y pueden recibir actualizaciones periódicas.

La pantalla de inicio (HOME), en el fondo es una aplicación que puede contener varios Widgets.Pueden existir aplicaciones como los “Launchers” que pueden contener varios widgets. A estas aplicaciones se les llama App Widget host.



Widgets

Para crear un Widget se necesita:

•**AppWidgetProviderInfo :** es un objeto que describe la metadata del widget, su periodo de actualización, el layout etc… Este es definido en XML.

•**AppWidgetProvider:** La implementación del widget que permitirá programar la lógica del widget.

•**Layout:** se debe definir un layout inicial para el widget.

•Definirlo en el manifiesto…

```
<receiver android:name="ExampleAppWidgetProvider" >
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>
    <meta-data android:name="android.appwidget.provider"
               android:resource="@xml/example_appwidget_info" />
</receiver>
```

Widgets

•onUpdate() : Es llamado cada x tiempo. También se llama por primera vez cuando se agrega el widget a la pantalla.

•onDeleted(Context, int[]): Se llama cada vez que el widget se destruye.

•onEnabled(Context): Es llamado cuando se agrega la primera instancia del Widget. Los widgets puede ser agregados varias veces.

•onDisabled(Context): Es llamado cuando la última instancia de un widget se destruye.

•onReceive(Context, Intent): Es llamado cada vez que hay un Broadcast, y antes de los métodos anteriores.

Widgets

•onUpdate() : Es llamado cada x tiempo. También se llama por primera vez cuando se agrega el widget a la pantalla.

•onDeleted(Context, int[]): Se llama cada vez que el widget se destruye.

•onEnabled(Context): Es llamado cuando se agrega la primera instancia del Widget. Los widgets puede ser agregados varias veces.

•onDisabled(Context): Es llamado cuando la última instancia de un widget se destruye.

•onReceive(Context, Intent): Es llamado cada vez que hay un Broadcast, y antes de los métodos anteriores.

Ejemplo

1.Crear una actividad comun

1.Crear un widget

```
public class MiWidget extends AppWidgetProvider {
@Override
public void onUpdate(Context context,
        AppWidgetManager appWidgetManager,
        int[] appWidgetIds) {
//Actualizar el widget
//...
}
}
```

Ejemplo

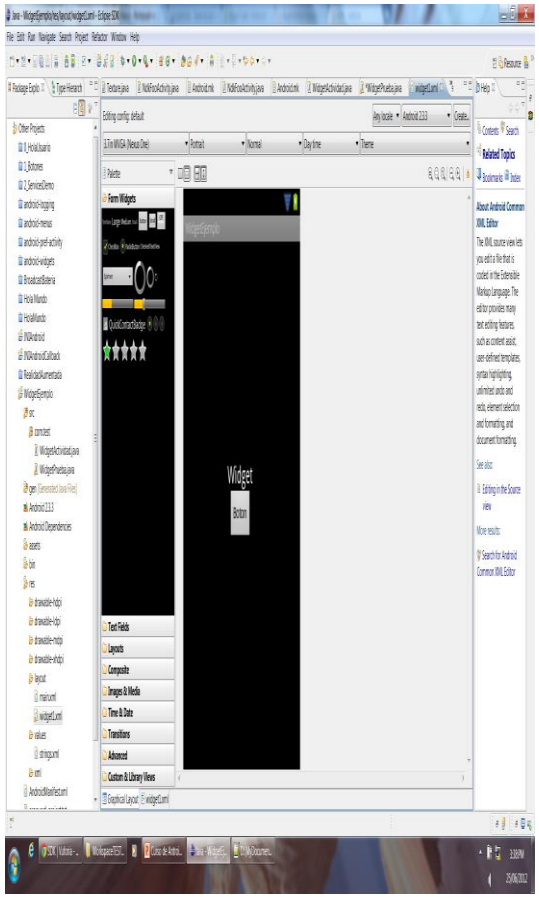
3. Definir manifiesto

```
<receiver android:name="com.test.MiWidget" android:label="Mi Primer Widget">
<intent-filter>
<action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
<action android:name="com.test.ACTUALIZAR" />
</intent-filter>
<meta-data
android:name="android.appwidget.provider"
android:resource="@xml/miwidget_wprovider" />
</receiver>
```

Ejemplo 2

1.Crear una actividad y un widget
2.En el manifiesto del widget colocar un intentfilter tipo:
<action android:name="*com.test.ACTUALIZAR*" />

1.Crear un nuevo widget y crear un layout así:



Ejemplo 2

```
public void onReceive(Context context, Intent intent)
{super.onReceive(context, intent);
if (WIDGET_UPDATE.equals(intent.getAction())) {
//Obtenemos el ID del widget a actualizar
int widgetId = intent.getIntExtra(
AppWidgetManager.EXTRA_APPWIDGET_ID,
AppWidgetManager.INVALID_APPWIDGET_ID);

//Obtenemos el widget manager de nuestro contexto
AppWidgetManager widgetManager =
AppWidgetManager.getInstance(context);

//Actualizamos el widget
if (widgetId != AppWidgetManager.INVALID_APPWIDGET_ID) {
actualizarWidget(context, widgetManager, widgetId);
}
}
```

Ejemplo 2

1.En el onUpdate() del Widget colocar:

public void onUpdate(Context context, AppWidgetManager appWidgetManager, **int[] appWidgetIds)**
{

```
//Iteramos la lista de widgets en ejecución
for (int i = 0; i < appWidgetIds.length; i++)
{
//ID del widget actual
int widgetId = appWidgetIds[i];

//Actualizamos el widget actual
actualizarWidget(context, appWidgetManager, widgetId);
}
```

Ejemplo 2

```
Public static void actualizarWidget(Context context, AppWidgetManager appWidgetManager, int widgetId)
{
//Obtenemos la lista de controles del widget actual
RemoteViews controles =   new RemoteViews(context.getPackageName(), R.layout.widget1);
//Asociamos los 'eventos' al widget
Intent intent = new Intent(WIDGET_UPDATE);
intent.putExtra(
    AppWidgetManager.EXTRA_APPWIDGET_ID, widgetId);
PendingIntent pendingIntent =
PendingIntent.getBroadcast(context, widgetId,
intent, PendingIntent.FLAG_UPDATE_CURRENT);
controles.setOnClickPendingIntent(R.id.boton, pendingIntent);
//Obtenemos la hora actual
Calendar calendario = new GregorianCalendar();
String hora = calendario.getTime().toLocaleString();
//Actualizamos la hora en el control del widget
controles.setTextViewText(R.id.texto, hora);
//Notificamos al manager de la actualización del widget actual
appWidgetManager.updateAppWidget(widgetId, controles);
}
```

Ejemplo 2

Ejercicio de integración.

Crear un Widget con dos botones. El primer botón actualizará la hora del Widget, el segundo botón irá abrirá una pantalla que mostrará:

1.El tiempo en ese instante

2.El estado de la batería en ese instante.

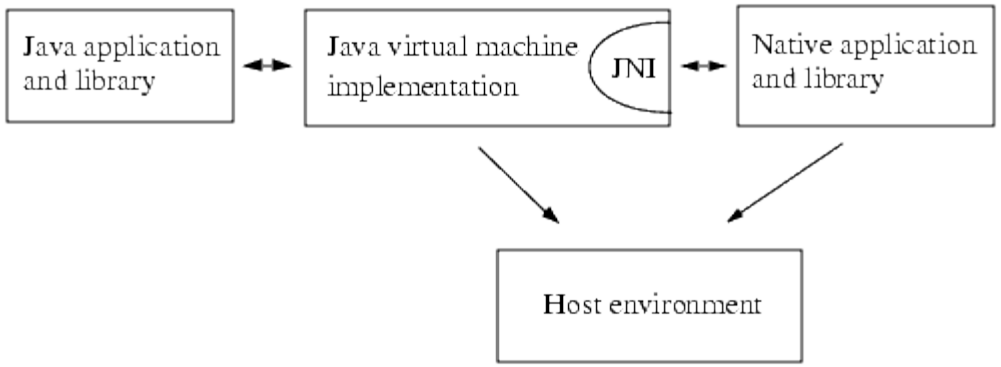
3.Dejar un boton libre para Realidad Aumentada

JNI en Android

JNI

•El JNI se usa para escribir métodos nativos que permitan solventar situaciones en las que una aplicación no puede ser enteramente escrita en Java, como por ejemplo en el caso de que la biblioteca estándar de clases no proporcione soporte para funcionalidades dependientes de la plataforma.

•También se usa para modificar programas existentes escritos en algún otro lenguaje, permitiéndoles ser accesibles desde aplicaciones Java.



JNI

Cuando No Usarlo

- **Cuando se puede interoperar dos librerías en dos lenguajes mediante TCP/IP. Ej. Por XML-RPC, Web-REST etc..**
- **Cuando el rendimiento no es prioritario.**
- **Cuando la portabilidad es necesaria.**

Cuando Usarlo:

- **Cuando se desea implementar código a bajo nivel. Por ejemplo aplicaciones 3-D.**
- **Cuando se desean soportar operaciones específicas que no son soportadas por Java.**

JNI en Android

- **Android soporta JNI, sin embargo se tiene que tener cuidado ya que vuelve la JVM suceptible al código en C.**

- **JNI está soportado por el NDK de Android**

- Descargar y Descomprimir el archivo: http://developer.android.com/tools/sdk/ndk/index.html
- **Poner la carpeta android-ndk-r* en el Path del sistema.**

JNI en Android

- **Android MK file: tiene la configuración del JNI, cómo el compilador ndk debe tratar los archivos… versión de APIS, etc jni/Android.mk**

- **Código en C o C++ . Ej. Micodigo.c**

JNIEXPORT jstring JNICALL
Java_com_test_NdkFooActivity_invokeNativeFunction(JNIEnv env, jobject javaThis) {*

JNI en Android

- **Las Librerías se cargan con:**

```
// Cargar la libreria que hace matching con: jni/Android.mk
static {
System.loadLibrary("ndkfoo");
}
```

- **Llamadas a librerías Nativas en Java:**
  - ***Private native String obtenerDatos();***
  - ***Private native void iniciarEntorno();***

JNI en Android

1.Descargar CYGWIN en la dirección:

http://www.cygwin.com/setup.exe y seleccionar "Install from the Internet!“

1.All -> Devel -> "make: The GNU version of the 'make' utility“

1.Configurar las variables de entornos:
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/libexec:/System/Library/CoreServices:/Developer/usr/bin:
Development/Android/android-ndk-r7:export PATH

1.Probar el comando ndk-build

JNI en Android

1.Descargar CYGWIN en la dirección:

http://www.cygwin.com/setup.exe and select "Install from the Internet!“

1.All -> Devel -> "make: The GNU version of the 'make' utility“

1.Configurar las variables de entornos:
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/libexec:/System/Library/CoreServices:/Developer/usr/bin:
Development/Android/android-ndk-r7:export PATH

1.Probar el comando ndk-build

JNI en Android

1.Crear una actividad sencilla y declarar el método:

private native String invokeNativeFunction();

1.Cargar la librería:

// Cargar la libreria que hace matching con: jni/Android.mk
static {
System.*loadLibrary("ndkfoo");*
}

1.Crear un Boton y en el onclicListener capturar el valor de la invocación ; y luego visualizarlo con un TOAST

Ej: String miinfo = invokeNativeFunction();

JNI en Android

*Crear el archivo N**dkfoo.c***

```
#include <string.h>
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_test_NdkFooActivity_invokeNativeFunction(JNIEnv* env, jobject
javaThis) {
  return (*env)->NewStringUTF(env, "Hola del codigo en C!");
}
```

JNI en Android

Agregar en el Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

# Here we give our module name and source file(s)
LOCAL_MODULE    := ndkfoo
LOCAL_SRC_FILES := Ndkfoo.c

TARGET_PLATFORM := android-10

include $(BUILD_SHARED_LIBRARY)
```

JNI en Android

1.Irse a la carpeta del projecto e invocar:
ndk-build