Bloque 1. Introducción a Android

José A. Montenegro

Dpto. Lenguajes y Ciencias de la Computación
ETSI Informática. Universidad de Málaga
monte@lcc.uma.es twitter

22 de noviembre de 2011

¿Qué es Android?

- Android es una "pila" software para dispositivos móviles que incluyen un sistema operativo, middleware y aplicaciones esenciales.
- El Kit de desarrollo Software (SDK) de android proporciona las herramientas y los interfaces de programación (APIs) necesarias para comenzar a desarrollar aplicaciones en la plataforma Android con el lenguaje de programación Java.

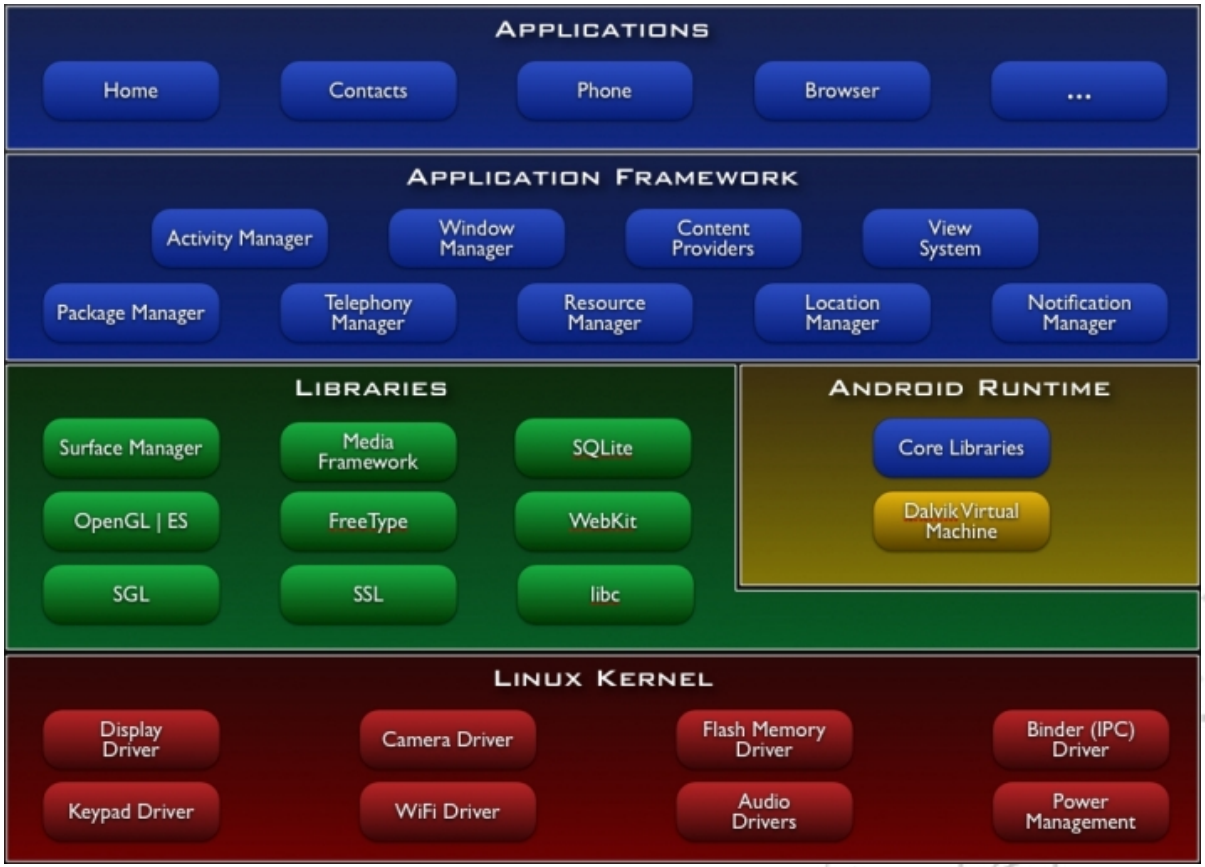


Figura 1: Arquitectura del Sistema

Características Principales

Marco de Trabajo de Aplicación permite reutilizar y reemplazar los componentes.

Máquina Virtual Dalvik máquina virtual Java optimizada para dispositivos móviles.

Navegador web integrado, basado en el proyecto opensource WebKit.

Gráficos Optimizados mediante una librería 2D y 3D basados en la especificación OpenGL ES 1.0

SQLite para el almacenamiento datos.

Soporte Multi media para audio, vídeo, y formato de imágenes (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

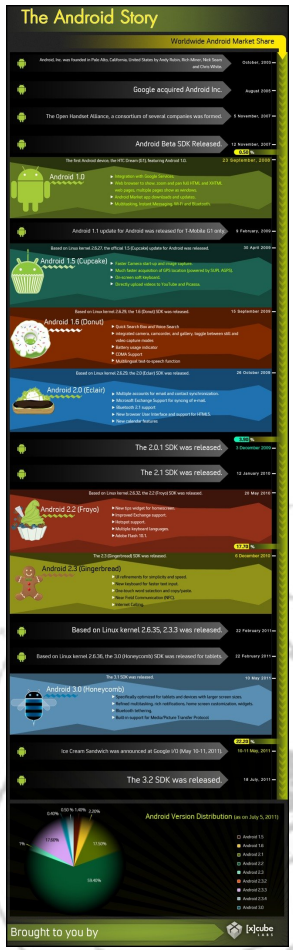
Telefonía GSM,Bluetooth, EDGE, 3G, y WiFi

Camera, GPS, Brújula, y Acelerómetro

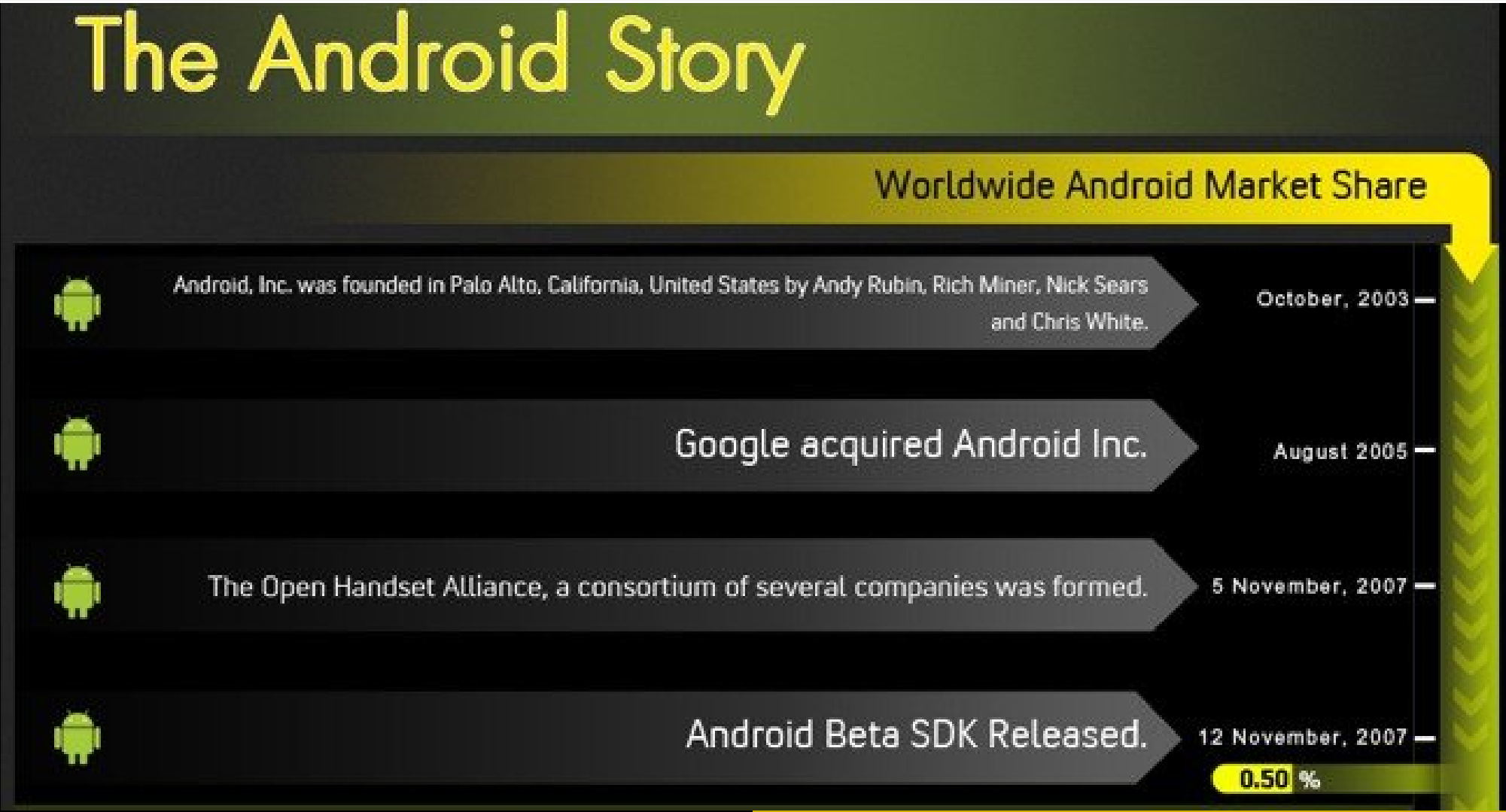
Entorno de desarrollo completo incluye un emulador de dispositivo, herramientas para debugging, memoria y perfiles de rendimiento. Ademas incluye un plugin para el IDE de Eclipse.

Historia de Android

1. 2008 (Sep). Android 1
2. 2009 (Abr). Android 1.5 (Cupcake)
3. 2009 (Sep). Android 1.6 (Donut)
4. 2009 (Oct). Android 2 (Eclair)
5. 2010 (May). Android 2.2 (Froyo)
6. 2010 (Dec). Android 2.3 (Gingerbread)
7. 2011 (May). Android 3 (Honeycomb)
8. 2011 (Oct). Android 4 (IceCream)

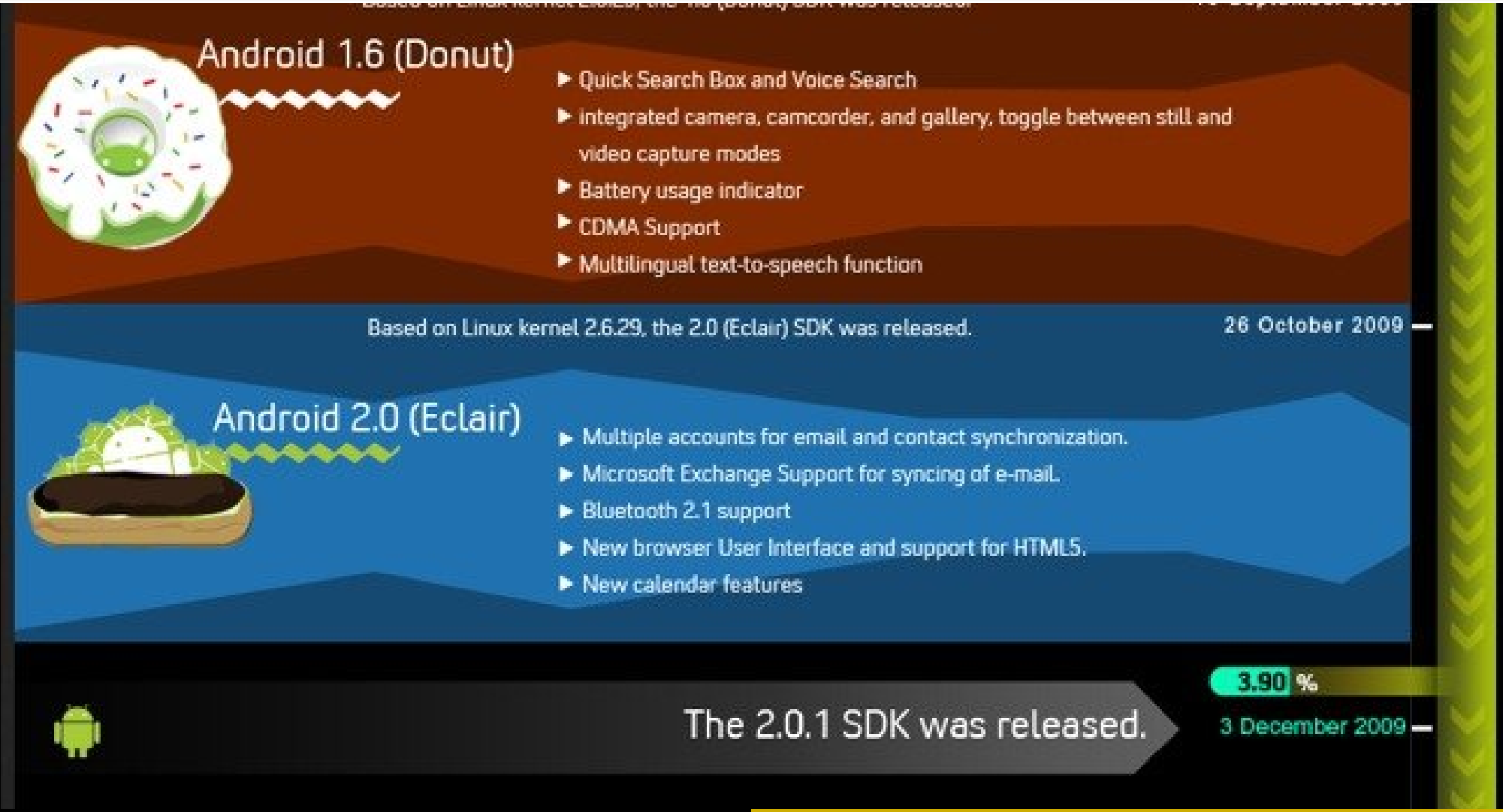


Historia de Android

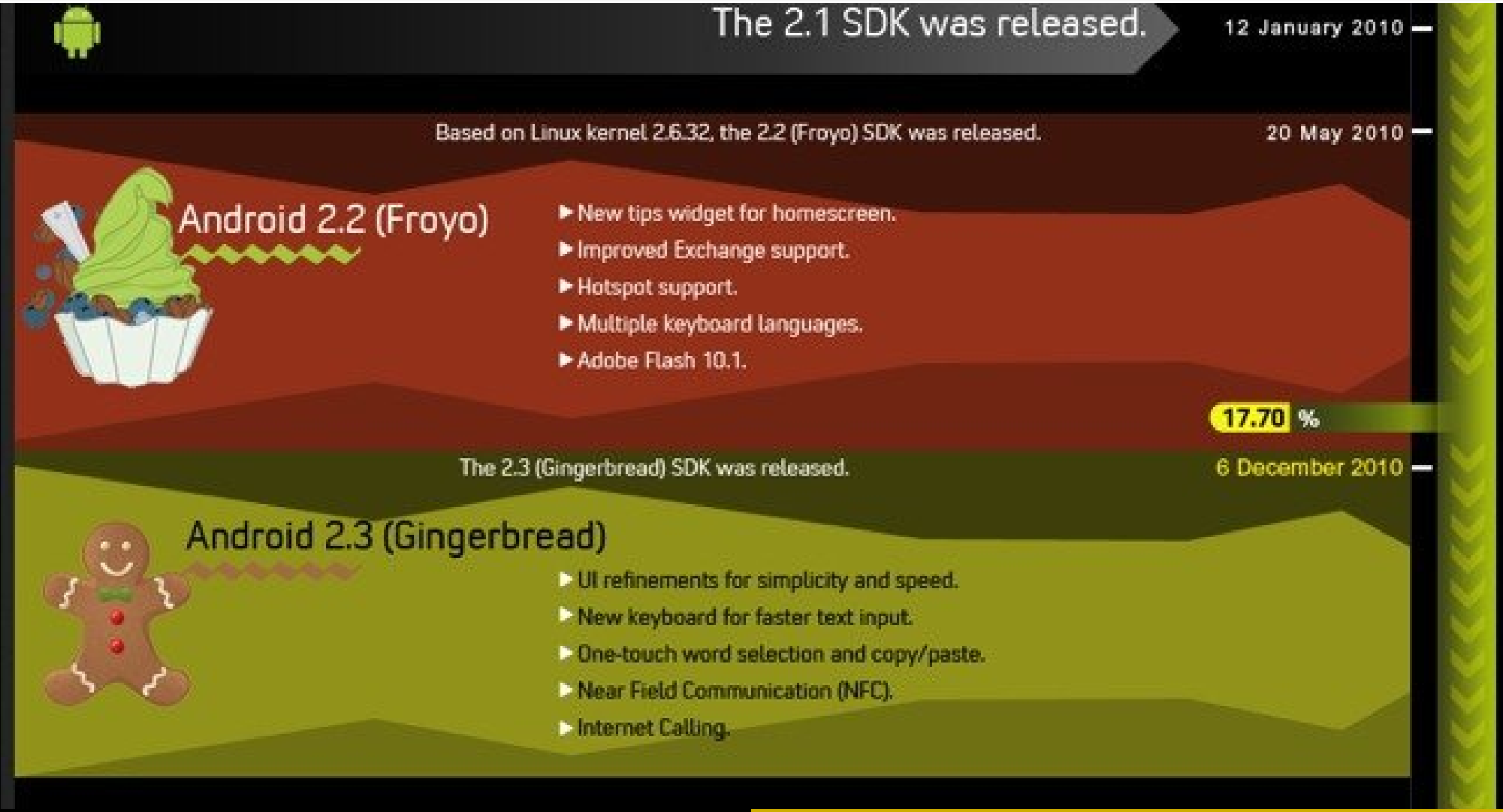


Historia de Android

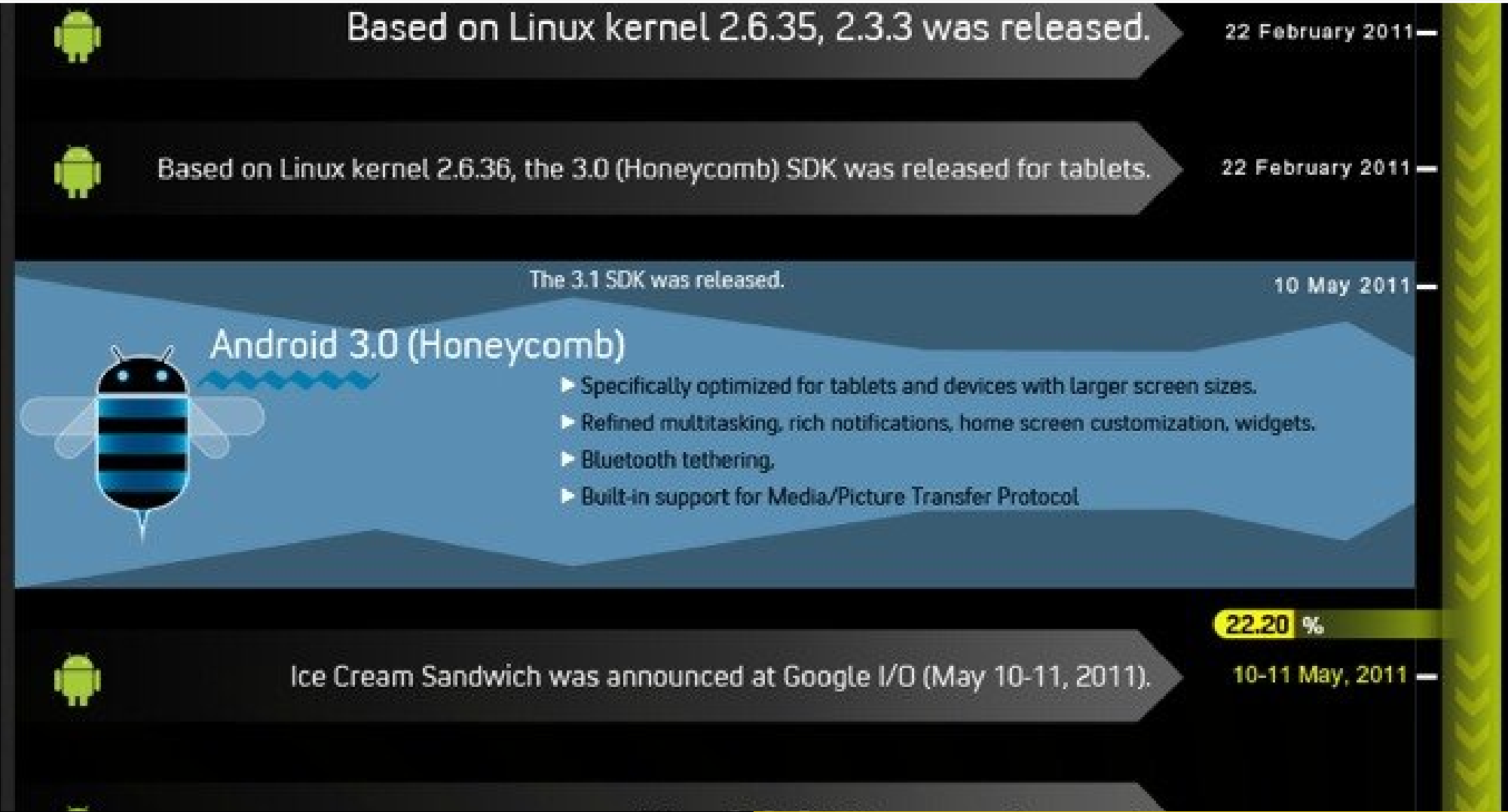
Historia de Android



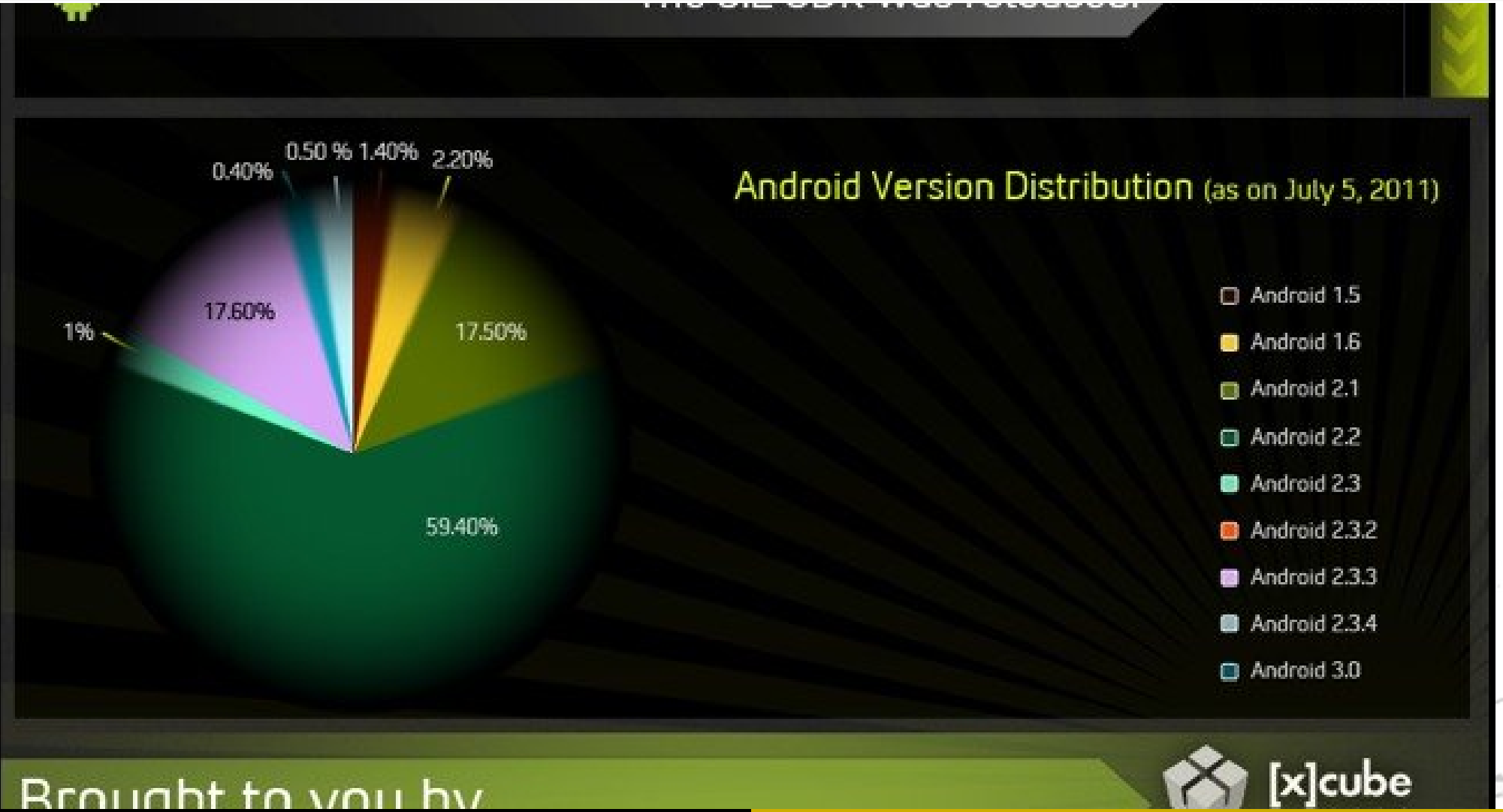
Historia de Android



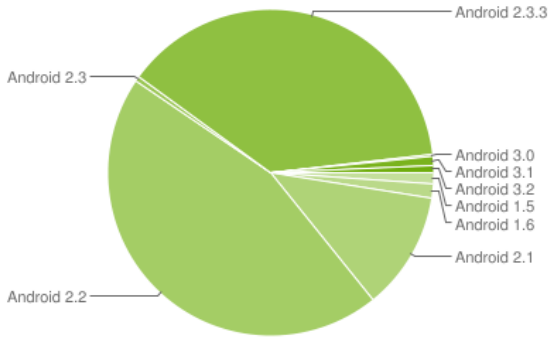
Historia de Android

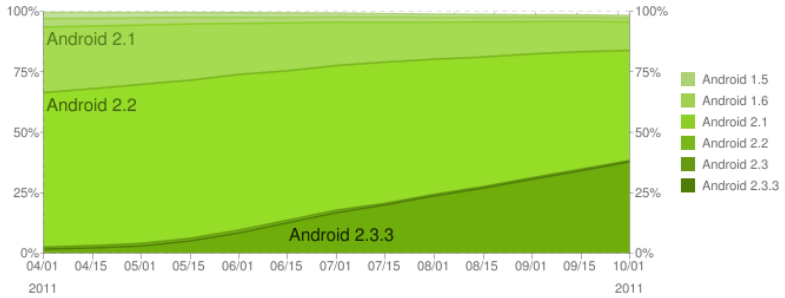


Historia de Android



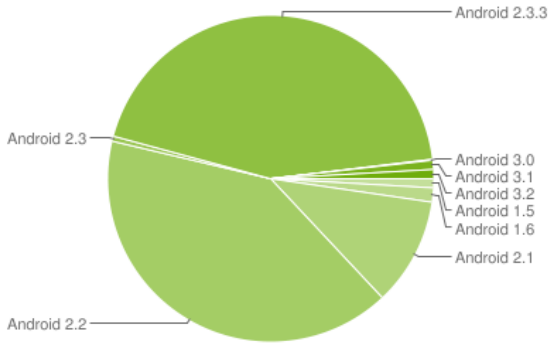
A fecha de 3 Octubre de 2011 la distribución de versiones de Android viene representada por los siguientes gráficos:

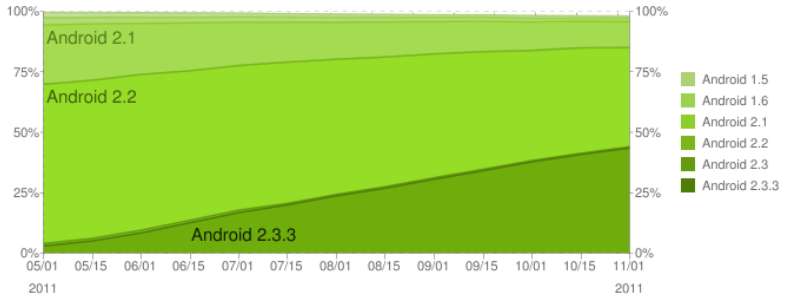




- 1.5 Cupcake $1{,}1\,\%$
- 1.6 Donut $1{,}4\,\%$
- 2.1 Eclair $11{,}7\,\%$
- 2.2 Froyo $45{,}3\,\%$
- 2.3 - 2.3.2 Gingerbread $0{,}5\,\%$
- 2.3.3 - 2.3.7 Gingerbread $38{,}2\,\%$
- 3.0 Honeycomb $0{,}2\,\%$
- 3.1 Honeycomb $0{,}9\,\%$
- 3.2 Honeycomb $0{,}7\,\%$

A fecha de 3 Noviembre de 2011 la distribución de versiones de Android viene representada por los siguientes gráficos:





- 1.5 Cupcake $0{,}9\,\%$
- 1.6 Donut $1{,}4\,\%$
- 2.1 Eclair $10{,}7\,\%$
- 2.2 Froyo $40{,}7\,\%$
- 2.3 - 2.3.2 Gingerbread $0{,}5\,\%$
- 2.3.3 - 2.3.7 Gingerbread $43{,}9\,\%$
- 3.0 Honeycomb $0{,}1\,\%$
- 3.1 Honeycomb $0{,}9\,\%$
- 3.2 Honeycomb $0{,}9\,\%$

Instalación Entorno de Trabajo

- Esta sección muestra como instalar el kit de desarrollo software Android (SDK) y todo el software relacionado que necesitaremos para el desarrollo.
- A la finalización podremos ejecutar una aplicación básica en un emulador y en un dispositivo.
- El punto inicial donde encontrar información actualizada y las herramientas necesarias es el sitio de Desarrolladores de Android:

http://developer.android.com.

Software Necesario

Los Sistemas Operativos soportados son:

- Windows XP (32-bit), Vista (32-64-bit), o Windows 7 (32-64-bit)
- Mac OS X 10.5.8 o posterior (sólo x86)
- Linux
  - GNU C Library (glibc) 2.7 o posterior .
  - Ubuntu Linux, version 8.04 o posterior.

El software necesario para el desarrollo es:

- Java JDK 5 ó 6 (Descargar).
- Eclipse IDE 3.5 o mayor (Descargar).
- Android SDK (Descargar).
- ADT Plugin para Eclipse (Info).

Android SDK

- La SDK de Android es una colección de archivos: librerias, ejecutables, scripts, documentación y demás.
- Una vez descargada la sdk debemos descomprimir el archivo en una localización determinada y añadirla a la variable de entorno *PATH* para que pueda ser ejecutada fácilmente por todos los elementos del sistema.
- Por ejemplo en Mac OS debemos modificar $.profile$ en el directorio del usuario.

export PATH=$PATH: /androidsdkmac_x86/tools: /androidsdkmac_x86/platform

Figura 2: Incluyendo path en Mac OS

En el caso de windows XP es necesario cambiar la variable de entorno *PATH*. del usuario o del sistema según se desee.

Posible Ruta:

Inicio → Configuración → Panel de Control → Sistema → Pestaña Opciones Avanzadas → Variables de Entorno

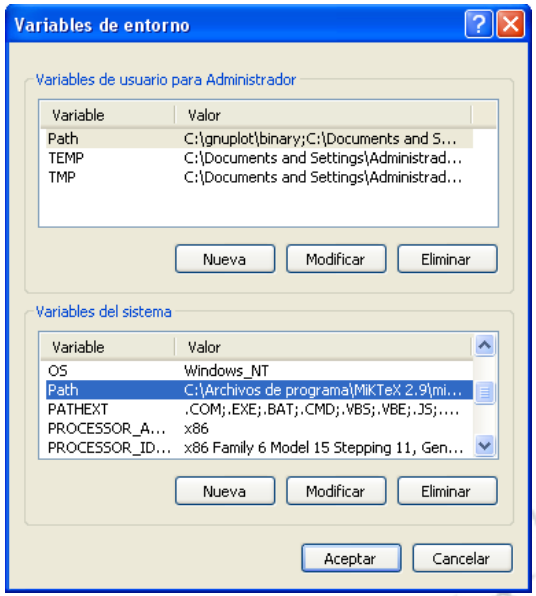


Figura 3: Modificando *PATH* en WindowsXP

- Antes de comenzar a crear un proyecto es necesario instalar uno o más plataformas de destino (targets).
- Para ello utilizaremos la SDK y AVD Manager. Esta herramienta permite instalar en el SDK multiples versiones de Android y niveles de API. Más adelante veremos como puede ser invocada desde Eclipse.
- Inicialmente lo realizaremos mediante la línea de comando, utilizando el comando: `android`

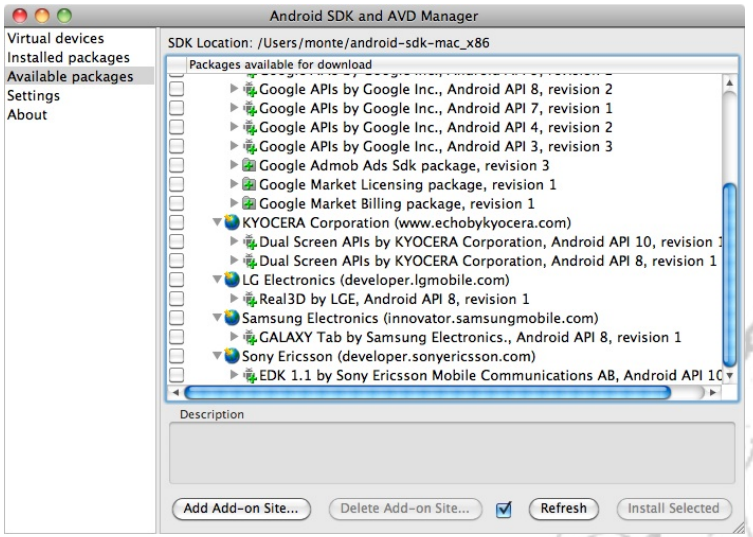


Figura 4: SDK y AVD Manager. Paquetes Disponibles

Android Development Toolkit (ADT) Plug-in para Eclipse

- Una vez que tenemos la SDK, Eclipse y JDK instaladas, es necesario instalar: el plug-in Android Developer Toolkit (ADT). El plug-in ADT añade a Eclipse las funcionalidades de Android.
- El software en del plug-in permite a Eclipse construir aplicaciones Android, lanzar el emulador de Android, conectar los dispositivos a servicios de depuración en el emulador, editar Android archivos XML, editar y compilar archivos Android Interface Definition Language (AIDL), crear paquetes de aplicación (archivos .apk).
- Para instalar debemos irnos al entorno Eclipse y en la acción del menú `Help --> Install New Software` añadir el siguiente destino:

```
https://dl-ssl.google.com/android/eclipse/
```

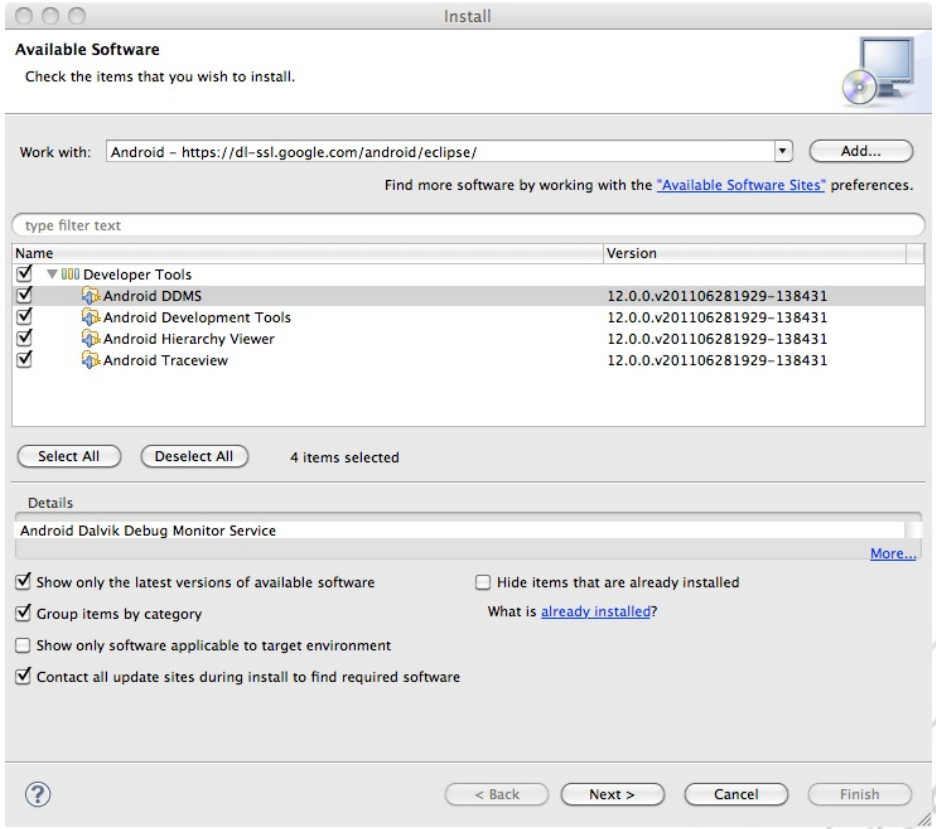


Figura 5: Instalando Android Development Toolkit (ADT) Plug-in

Una vez terminado la instalación del plug-in es necesario configurarlo.

1. Ejecutando el dialogo de las Preferencias utilizando la opción del menu `Window --> Preferences` (Linux y Windows) o `Eclipse-->Preferences` (Mac) .
2. Selecciona el elemento Android en el panel izquierdo del dialogo de Preferencias.

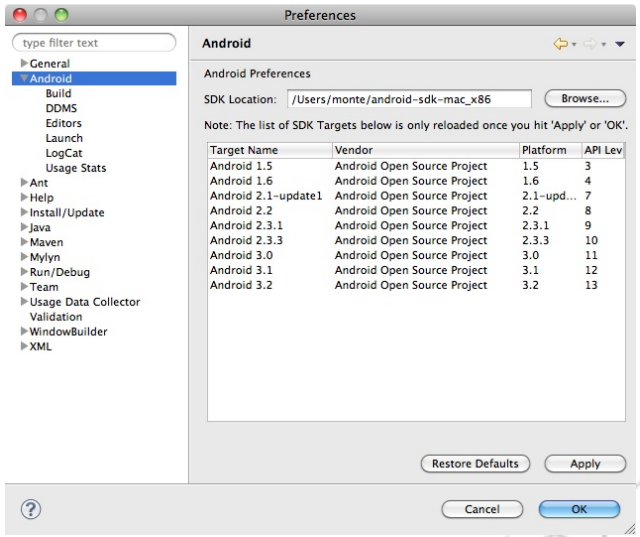


Figura 6: Configuración ADT Plug-in

Primer Proyecto Android

Una vez que tenemos ejecutado Eclipse es necesario ejecutar Wizard mediante `File --> New Project`.
Podemos ver en la figura la selección de `Android Project`.

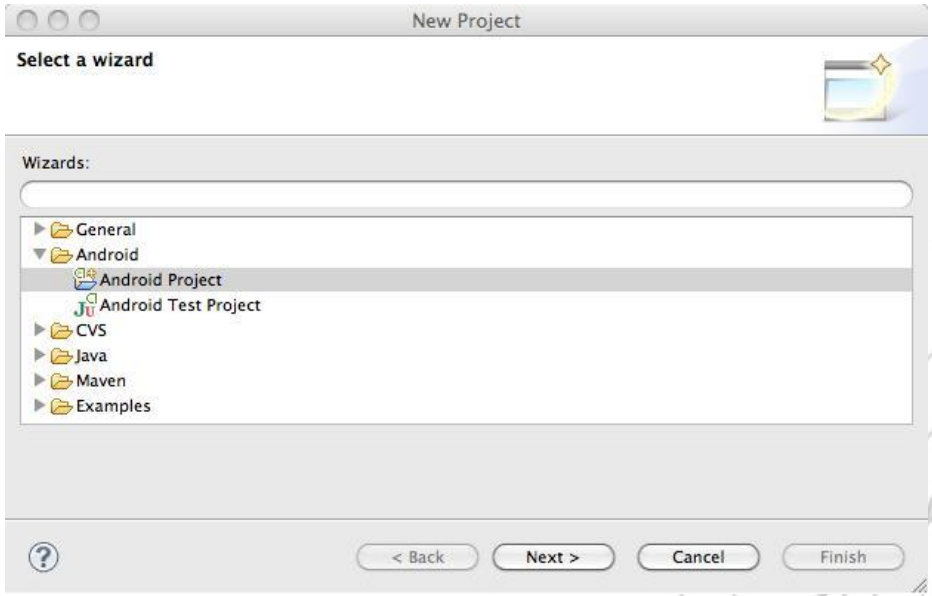


Figura 7: Ejecución del Wizard para crear Proyecto Android

El siguiente paso es la configuración de algunos de los valores del proyecto.

Project name Nombre del Proyecto no de la aplicación

Workspace Carpeta donde están los proyectos Eclipse.

Target name Imágenes de Android instaladas. Es necesaria escoger una.

Application name Nombre Aplicación.

Package name Nombre del paquete de java.

Activity El elemento principal de una aplicación Android.

Minimum SDK version Mínima versión del SDK necesaria para ejecutar la aplicación.

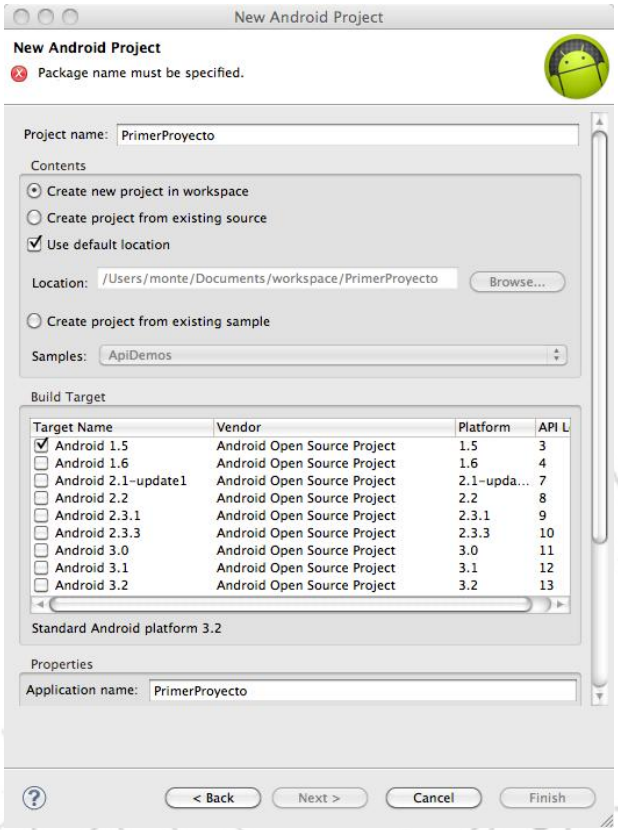


Figura 8: Cfg Proyecto

Una vez configurado los valores del proyecto pulsamos el botón `Next >`. Eclipse generará un proyecto básico con la estructura que muestra la figura.

PrimerProyectoActivity.java Archivo por defecto que ejecuta la aplicación (`Activity`)

Android Library/ Carpeta que contiene la referencia a SDK Android.

assets/ Directorio que contiene Multimedia y otros elementos necesarios.

res Directorio que almacena los recursos utilizados por el UI.

AndroidManifiest.xml Archivo que contiene la descripción de la aplicación.

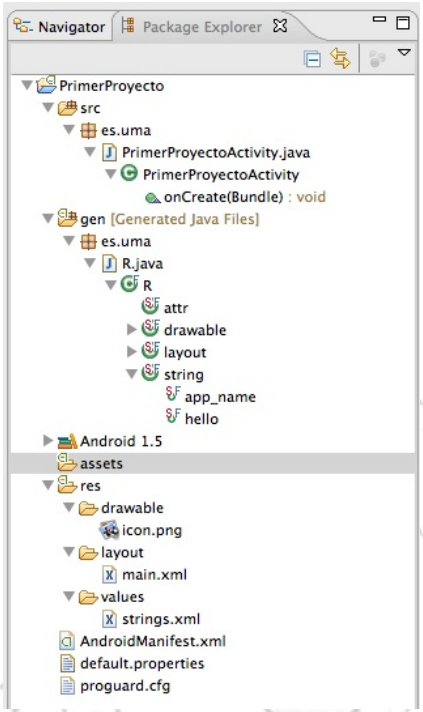


Figura 9: Contenido Proyecto

La aplicación ejecuta el código de `PrimerProyectoActivity.java`

```
1  package es.uma.PrimerProyecto;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5
6  public class PrimerProyectoActivity extends Activity {
7      /** Called when the activity is first created. */
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.main);
12     }
13 }
```

El archivo `AndroidManifest.xml` comunica al sistema que `PrimerProyectoActivity` es el primer elemento a ejecutar cuando inicie la aplicación (`android.intent.action.MAIN` `android.intent.category.LAUNCHER`) .

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3        package="es.uma"   android:versionCode="1" android:versionName="1.0">
4      <uses-sdk android:minSdkVersion="3" />
5      <application android:icon="@drawable/icon"
6       android:label="@string/app_name">
7          <activity android:name=".PrimerProyectoActivity"
8              android:label="@string/app_name">
9              <intent-filter>
10                 <action android:name="android.intent.action.MAIN" />
11                 <category android:name="android.intent.category.LAUNCHER" />
12             </intent-filter>
13         </activity>
14     </application>
15 </manifest>
```

Establecimiento de un Dispositivo Virtual Android (AVD)

- Antes de Ejecutar la aplicación que hemos creado es necesario configurar un dispositivo virtual Android.
- Podemos establecer AVD mediante la ejecución de la aplicación `android` desde la línea de comando o dentro de Eclipse.
- El estado inicial de AVD es el que muestra la figura sin ningún elemento creado.

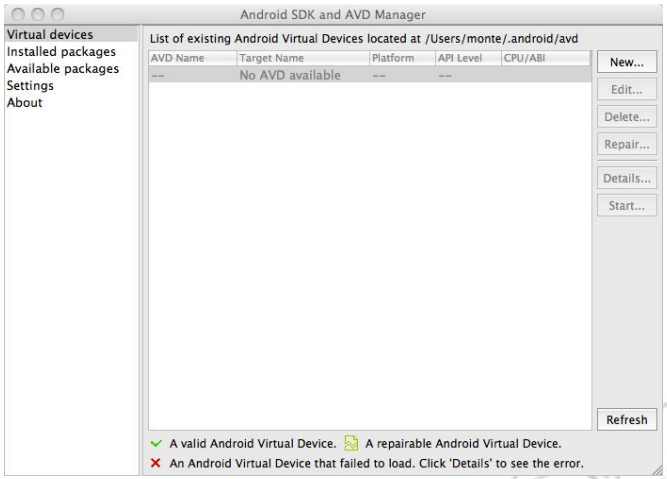


Figura 10: Estado Inicial

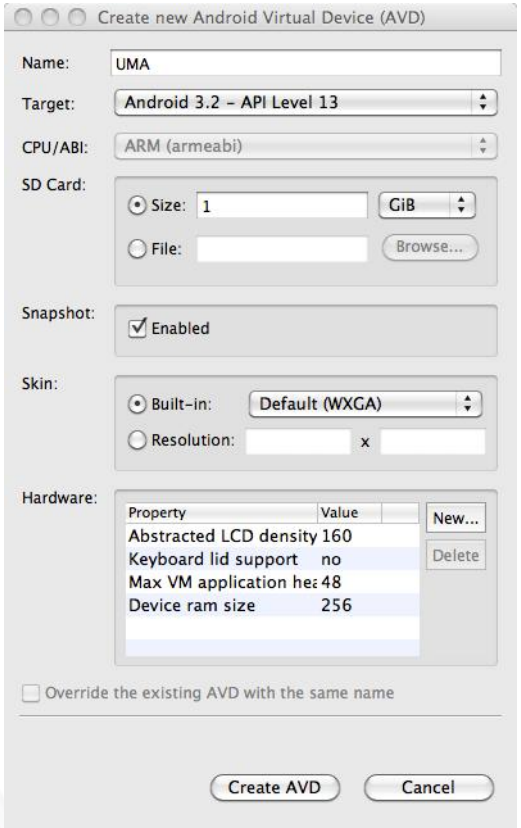


Figura 11: Configuración

Figura 12: Mensaje Final

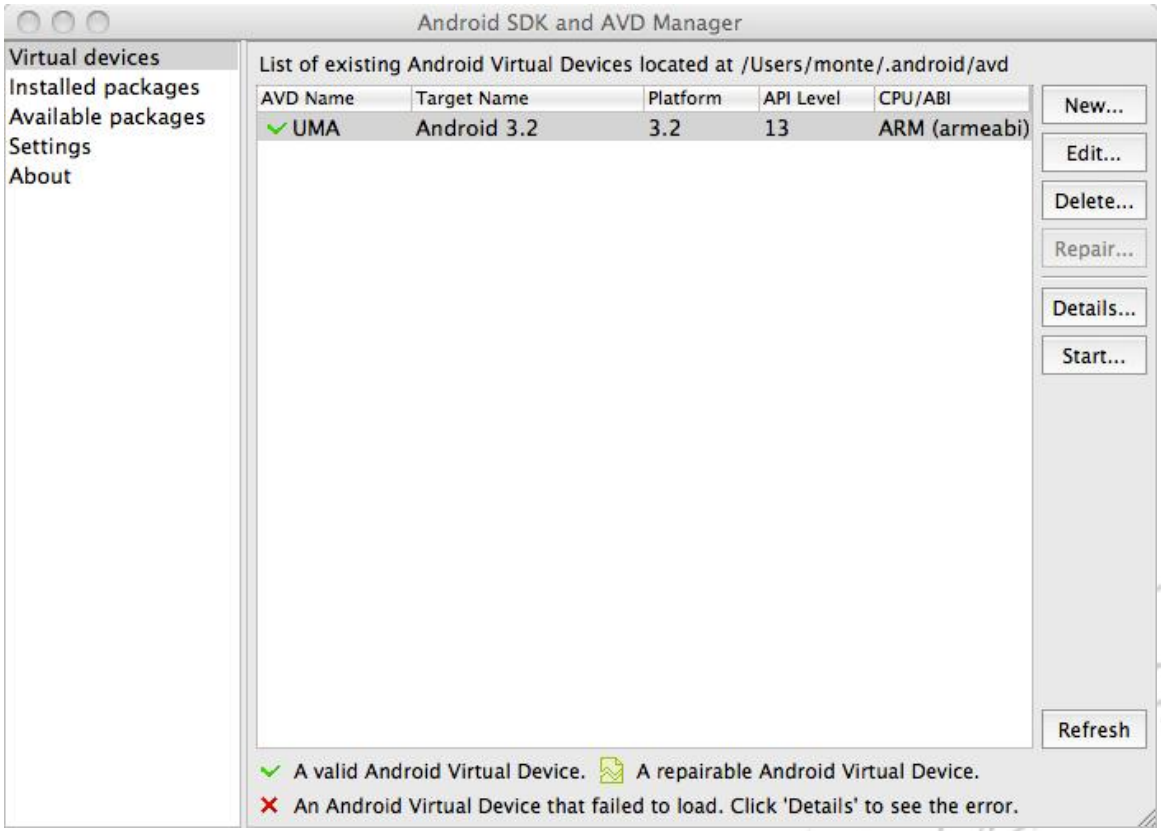


Figura 13: AVD Creada

Selección manual de varios dispositivos de ejecución

La siguiente figura muestra la ventana de selección de varios dispositivos para ejecutar las aplicaciones. En este caso tenemos dos dispositivos virtuales y uno real en la parte superior.

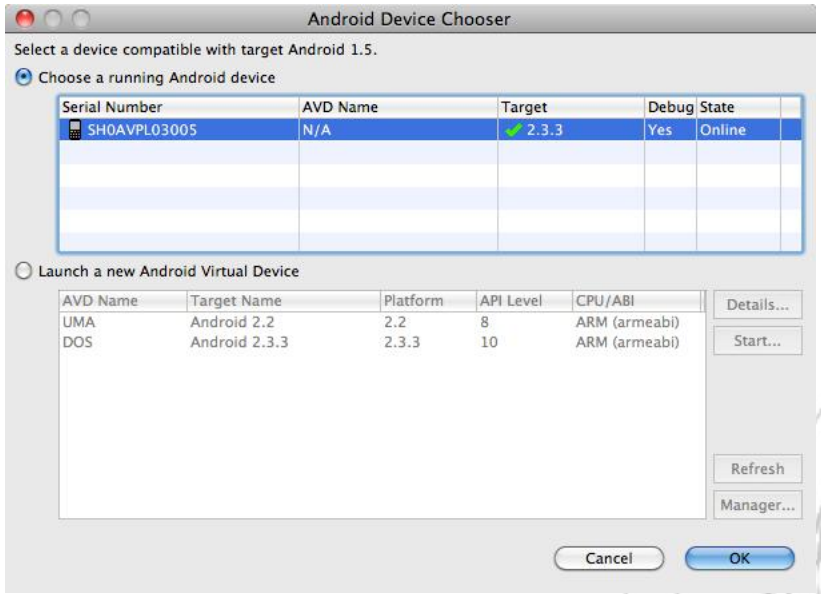


Figura 14: Elección de la ejecución en Varios Dispositivos

PrimerProyecto en Simulador Android 1.5

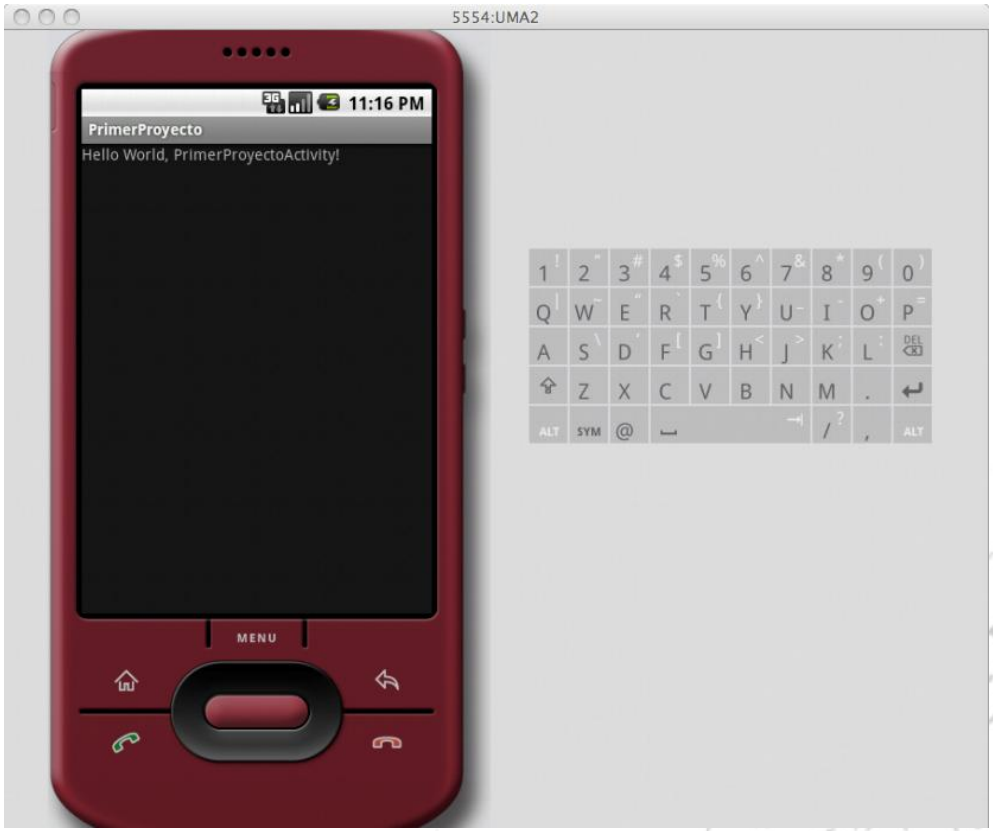


Figura 15: PrimerProyecto en Android 1.5

PrimerProyecto en Simulador Android 1.6

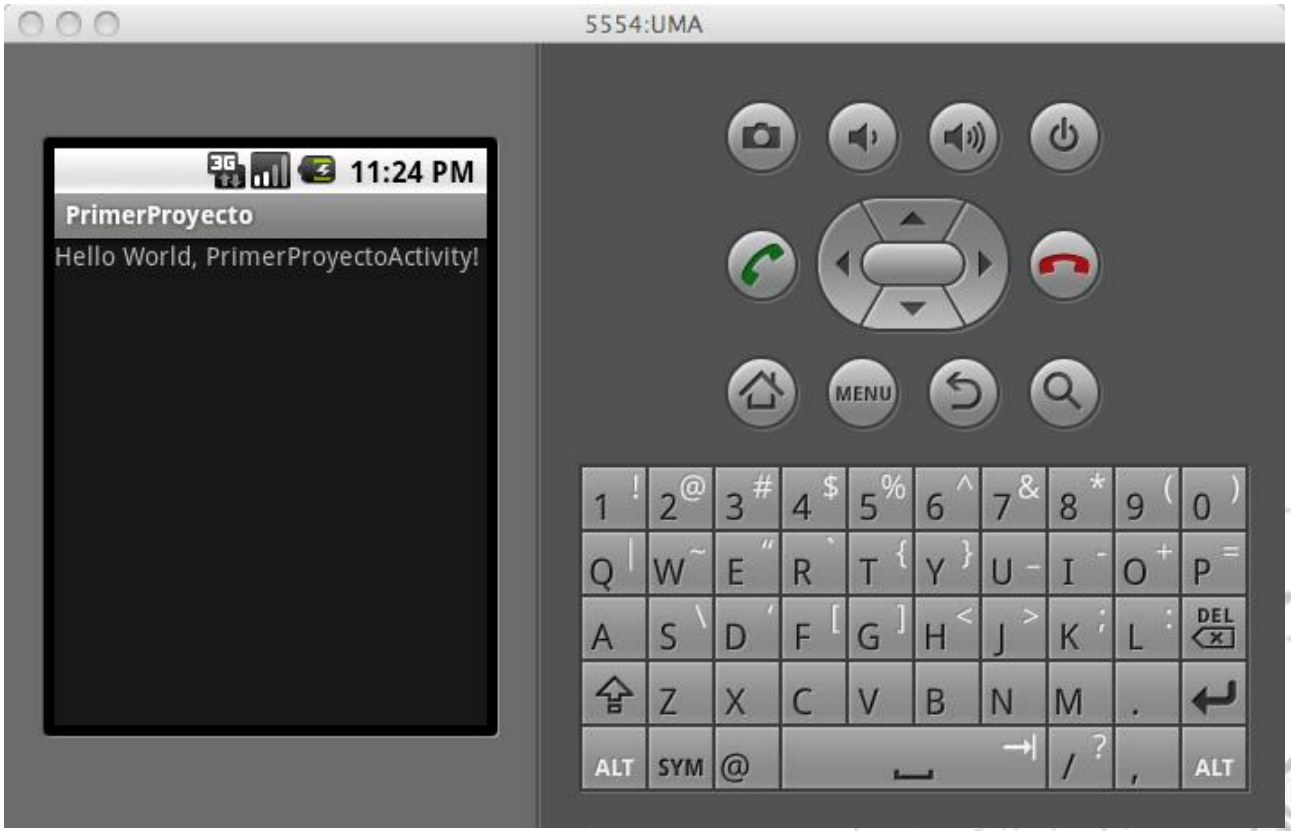


Figura 16: PrimerProyecto en Android 1.6

PrimerProyecto en Simulador Android 2.3.1

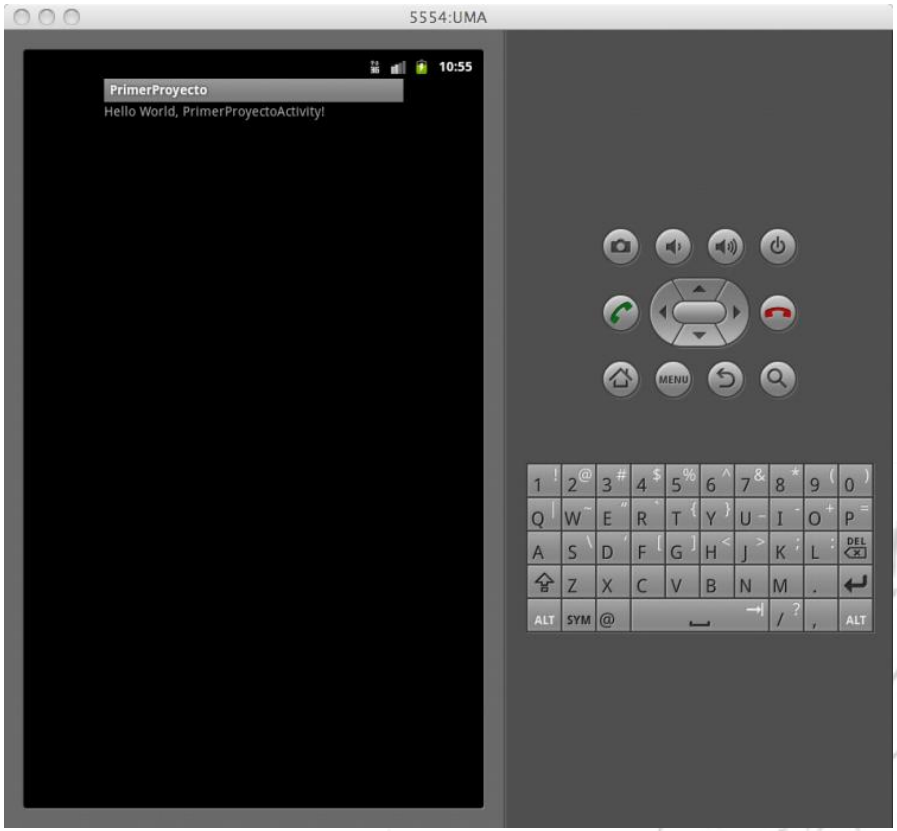


Figura 17: PrimerProyecto en Android 2.3.1

PrimerProyecto en Simulador Android 3.2.

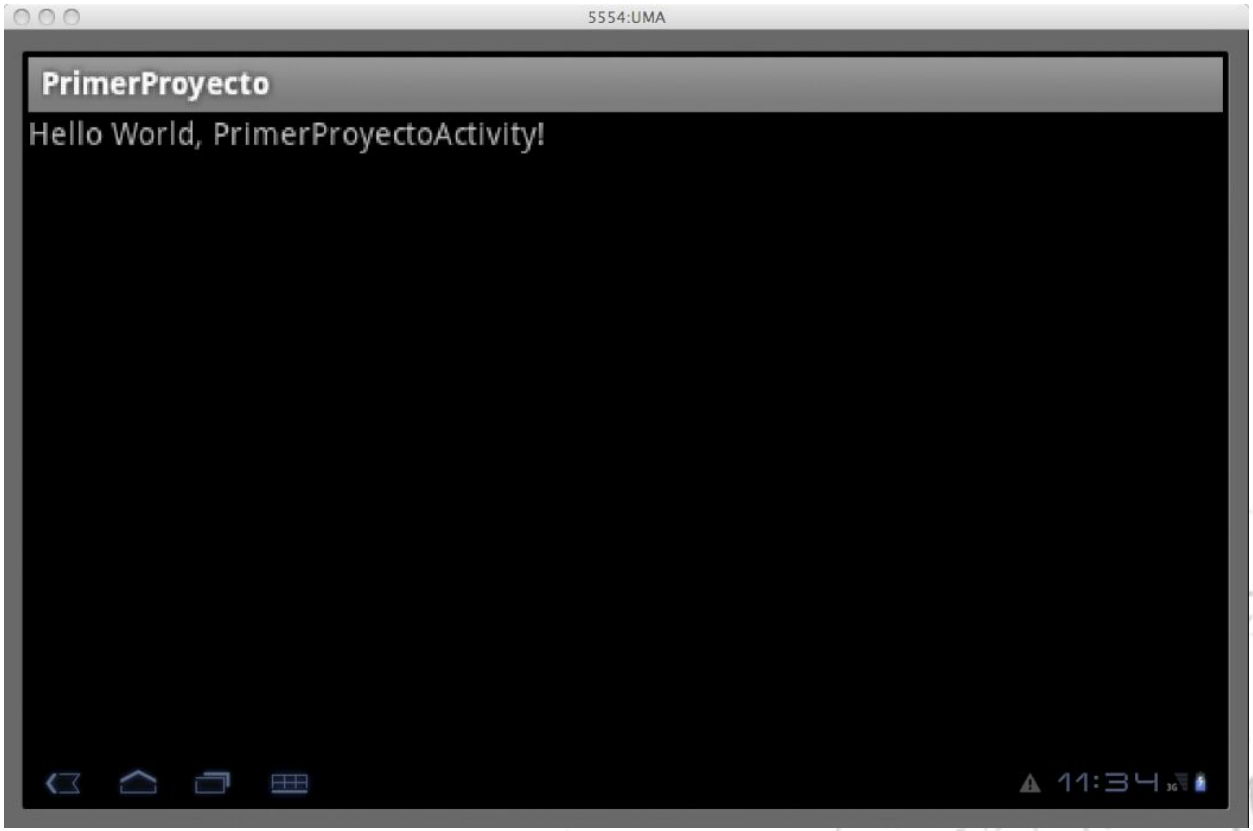


Figura 18: PrimerProyecto en Android 3.2

PrimerProyecto en Dispositivo Android HTC Desire

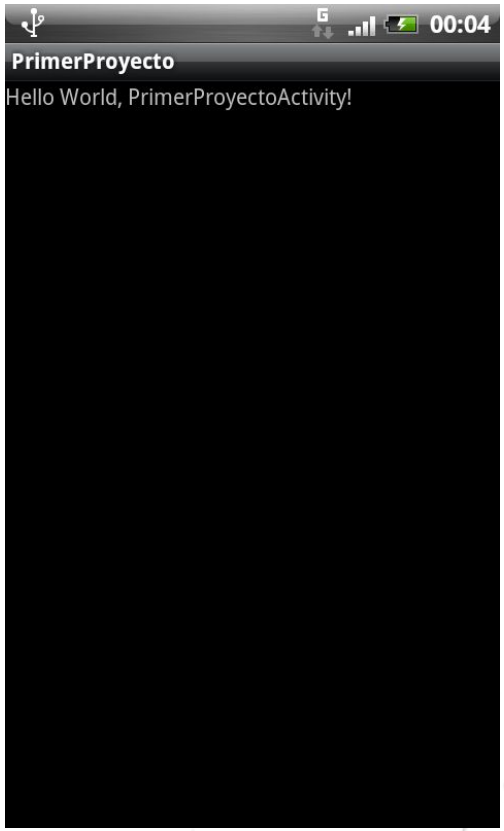


Figura 19: PrimerProyecto en HTC Desire

Abrir Proyectos Prácticas

Para abrir los proyectos de las prácticas, tenemos que irnos al menú de Eclipse: $File \rightarrow Import$ y seguir los pasos de las dos imágenes adjuntas:

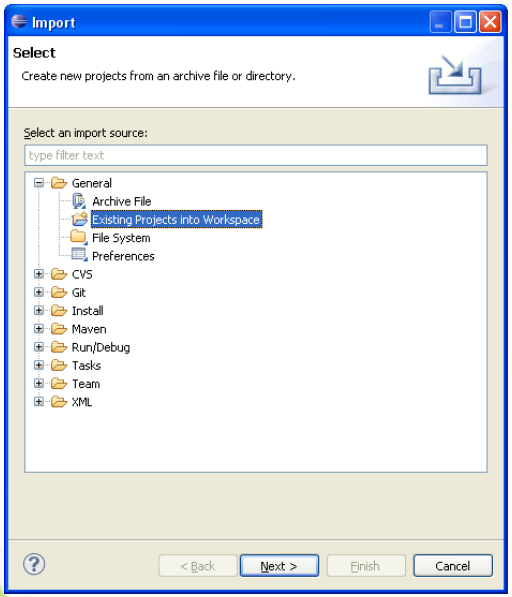


Figura 20: Importar Proyecto Paso 1

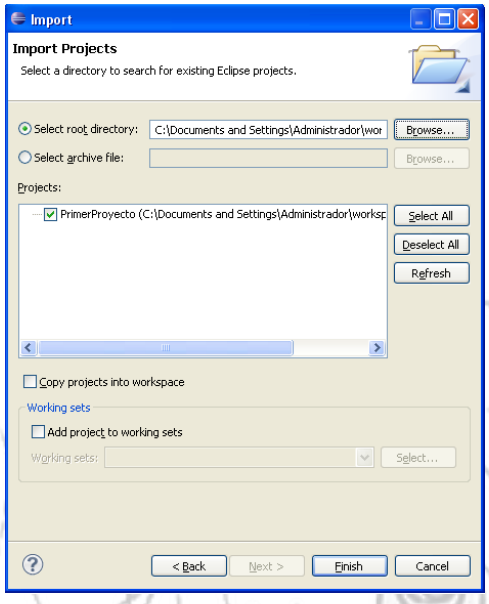


Figura 21: Importar Proyecto Paso 2

Abrir Proyectos Ejemplo de Eclipse

Android SDK nos ofrece una serie de proyectos de ejemplo con los cuales podemos practicar. Para abrir los proyectos de ejemplo, tenemos que irnos al menú de Eclipse: $File \rightarrow NewProject$ y seguir los pasos de las dos imágenes adjuntas:

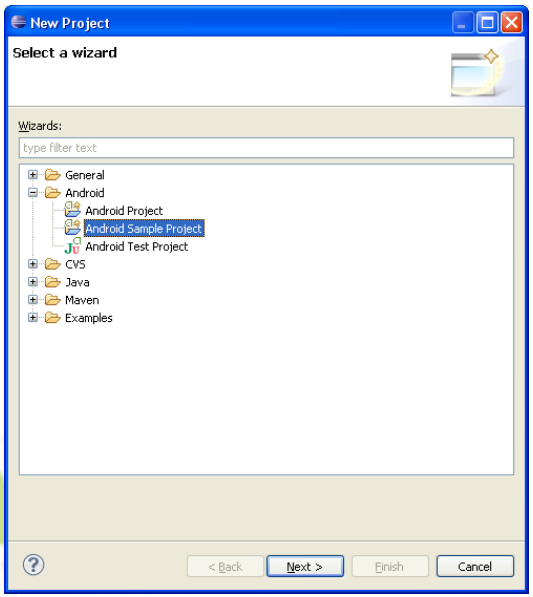


Figura 22: Abrir Ejemplos Paso 1

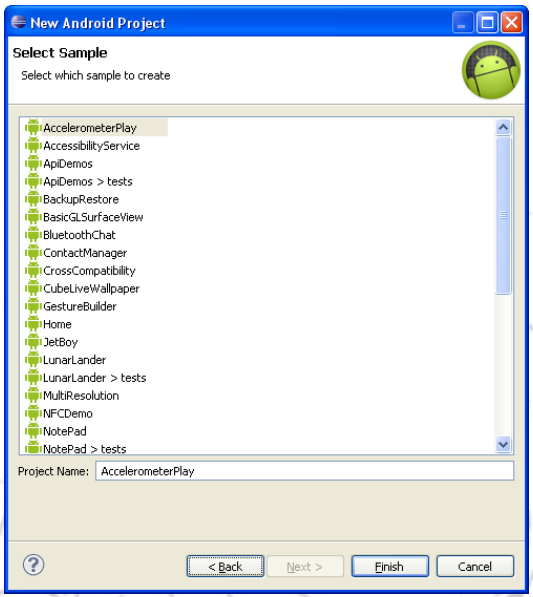


Figura 23: Abrir Ejemplos Paso 2

Otras Herramientas del SDK

El SDK nos ofrece un conjunto de herramientas muy útiles para el desarrollo de aplicaciones android, a continuación detallaremos algunas de ellas.

- Android Debug Bridge (adb)
- Emulador
- DDMS

Android Debug Bridge (adb)

adb es una herramienta ejecutada en línea de comando que posibilita comunicarse con una instancia del emulador o un dispositivo Android.

El ejecutable está en la carpeta `<sdk>/platform-tools/`, por lo que debemos incluirla también en la variable $PATH$.

El patrón de uso del comando es:

```
adb [-d|-e|-s <NumeroSerie>] <comando>
```

Información detallada en: Android sdk.

El primer comando que deberíamos ejecutar es:

```
adb devices
```

La salida nos muestra que dispositivos están conectados y disponibles para interactuar. La respuesta es una lista de dispositivos con el par de información de cada dispositivo (número de serie, estado).

En el caso que exista más de un dispositivo conectado deberemos incluir su número de serie con la opción `-s`.

```
adb -s <NumeroSerie> <comando>
```

Comandos más usuales de adb:

`install <path_to_apk>` Instala una aplicación (.apk) en el dispositivo.

```
adb install PrimerProyecto.apk
```

`pull <remoto> <local>` Copia un archivo o directorio **desde** el emulador o dispositivo

`push <local> <remoto>` Copia un archivo o directorio **al** emulador o dispositivo

```
adb push openssl /sdcard/openssl
```

`shell` Establece una consola remota en el dispositivo o emulador. Nos permite ejecutar tanto comandos que están ubicados en `/system/bin/` como aplicaciones.

A modo de ejemplo la siguiente sentencia produce la salida mostrada en la figura 5.

```
adb shell am start http://www.marca.es
```

Figura 24: Ejecución Remota navegador Web

Emulador Android

Anteriormente hemos visto la ejecución de las aplicaciones en el emulador de Android que ha sido ejecutado desde la aplicación `Eclipse`. También es posible su ejecución desde línea de comandos como herramienta del SDK.

```
emulator -avd <avd_name> [-<option> [<value>]] ... [-<qemu args>]
```

En nuestro caso tenemos un dispositivo virtual denominado UMA tal y como muestra la figura 13.

La ejecución de las siguientes sentencias ejecutaría la AVD creada.

```
emulator -avd uma
```

```
emulator @uma
```

Información detallada en: Android sdk.

Monkey

Es un programa que es ejecutado en el emulador o dispositivo y genera cadenas pseudo aleatorias de eventos de los usuarios así como eventos del sistema.

La sintaxis es:

```
adb shell monkey [options] <event-count>
```

Por ejemplo la siguiente sentencia enviaría 500 eventos a nuestra aplicación

```
adb shell monkey -p es.uma.PrimerProyecto -v 500
```

Información detallada en: Android sdk.

DDMS desde Eclipse

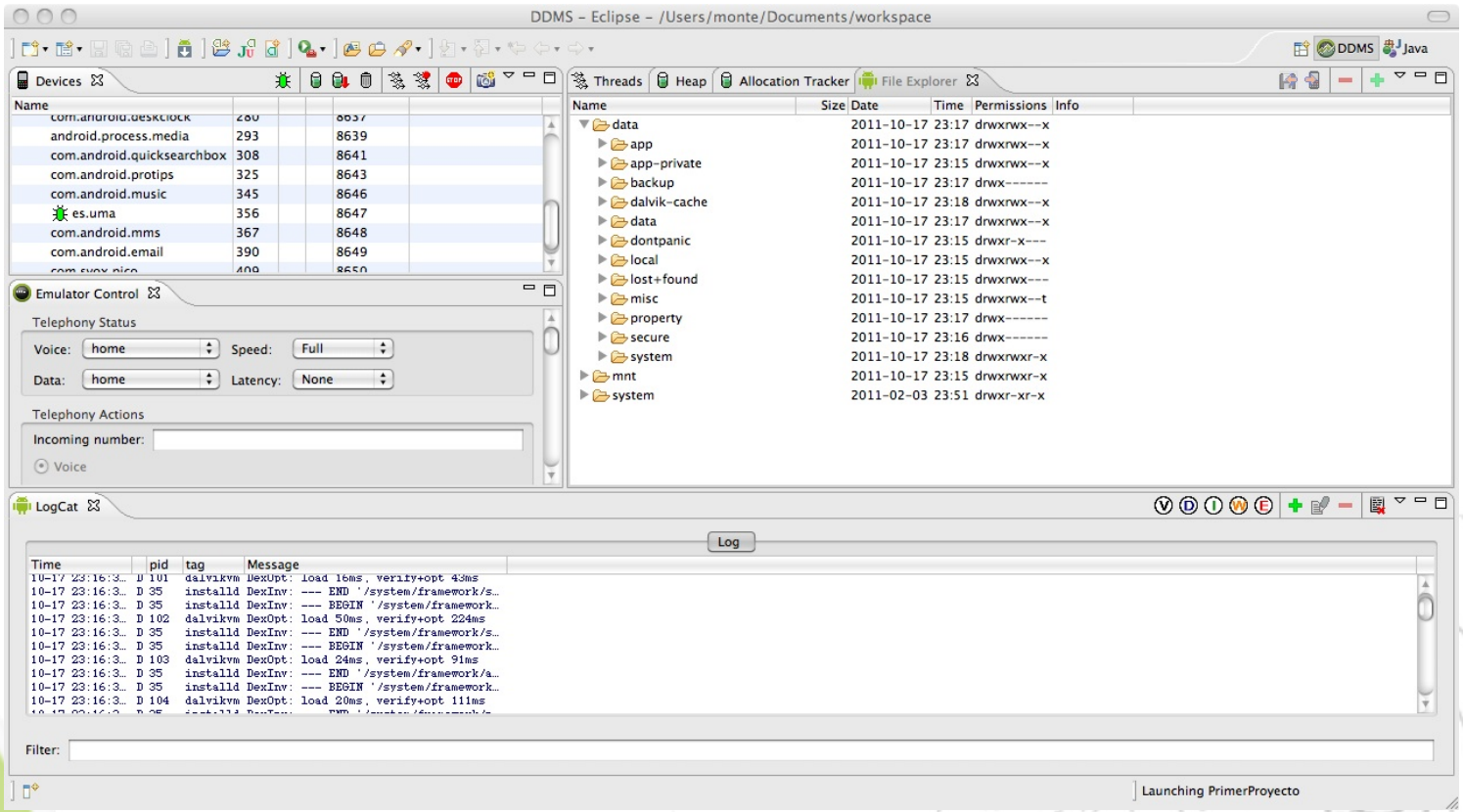


Figura 25: Ejecución DDMS en Eclipse

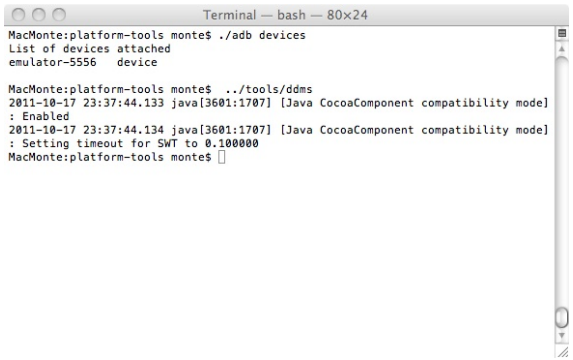


Figura 26: adb devices

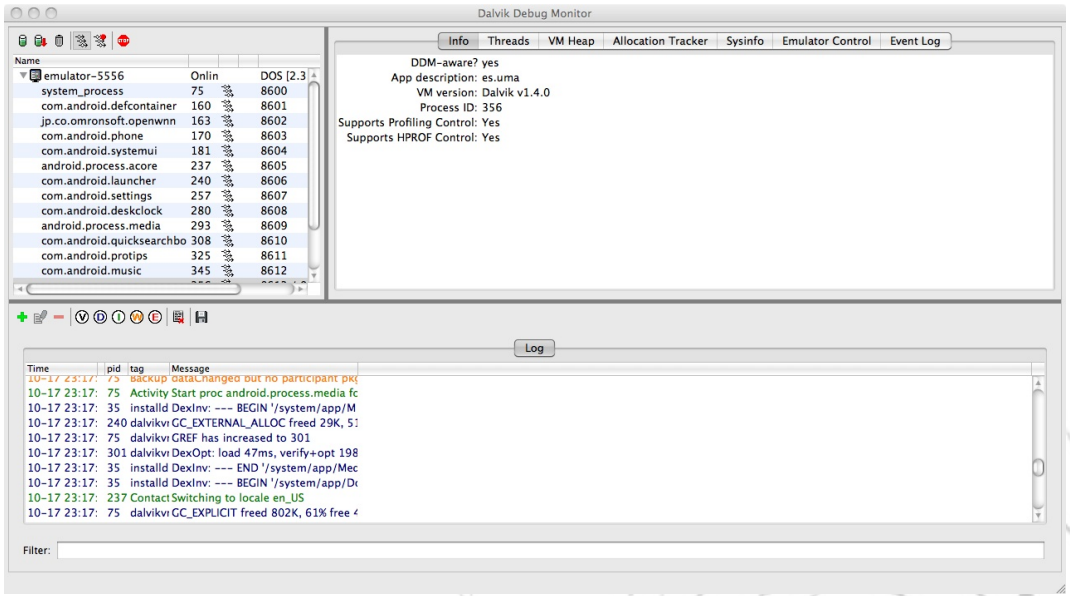


Figura 27: DDMS ejecutado línea de comandos

Android NDK

Android NDK es una herramienta adicional al SDK que permite realizar aplicaciones en código nativo.

Proporciona las cabeceras y librerías que permiten crear actividades, administrar las entradas de los usuarios, utilizar los sensores, ... etc, en `C` o `C++`.

Actualmente existe una serie de proyectos para permitir ejecutar aplicaciones nativas de linux. Nosotros veremos el caso concreto de `openssl`.

Más información en Android SDK.

José A. Montenegro Montes
Dpto. Lenguajes y Ciencias de la Computación
ETSI Informática. Universidad de Málaga
monte@lcc.uma.es