Google se hace móvil.

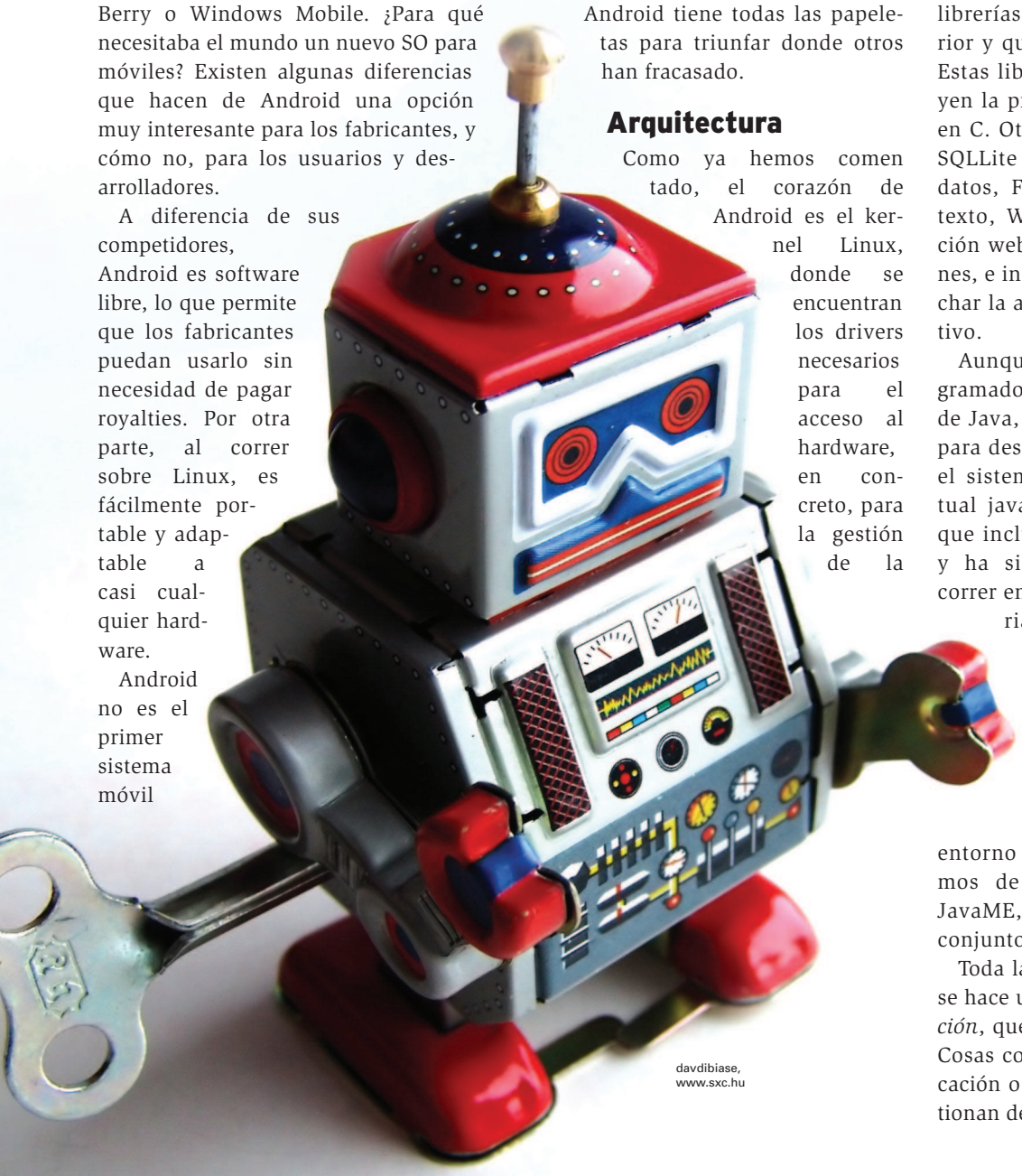
ANDROID

El nuevo sistema operativo de Google se enfrenta cara a cara con el IPhone de Apple. Promete facilidad de uso, un montón de aplicaciones interesantes, conectividad total y un modelo de programación sencillo pero muy potente. **POR ALBERTO GARCÍA SERRANO**

Con Android, Google ha hecho una apuesta arriesgada entrando en un mundo en el que ya brillan con luz propia Nokia con Symbian, Apple con su IPhone, BlackBerry o Windows Mobile. ¿Para qué necesitaba el mundo un nuevo SO para móviles? Existen algunas diferencias que hacen de Android una opción muy interesante para los fabricantes, y cómo no, para los usuarios y desarrolladores.

A diferencia de sus competidores, Android es software libre, lo que permite que los fabricantes puedan usarlo sin necesidad de pagar royalties. Por otra parte, al correr sobre Linux, es fácilmente portable y adaptable a casi cualquier hardware.

Android no es el primer sistema móvil basado en Linux y que es software libre. La propia Nokia abandera el proyecto Maemo, e incluso Ubuntu desarrolla Ubuntu Mobile, pero no parecen alcanzar la masa crítica necesaria. Android tiene todas las papeletas para triunfar donde otros han fracasado.



davdibiase, www.sxc.hu

Arquitectura

Como ya hemos comentado, el corazón de Android es el kernel Linux, donde se encuentran los drivers necesarios para el acceso al hardware, en concreto, para la gestión de la pantalla, el teclado, la cámara, la red Wi-Fi, el audio y la memoria Flash, entre otros. En principio, el desarrollador no accederá directamente a esta capa, sino que utilizará una serie de librerías que están en un nivel superior y que nos abstraen del hardware. Estas librerías, entre las que se incluyen la propia libc, están programadas en C. Otras librerías de este nivel son SQLLite para la gestión de bases de datos, FreeType para las fuentes de texto, WebKit y SSL para la navegación web y el cifrado de comunicaciones, e incluso OpenGL ES para aprovechar la aceleración gráfica del dispositivo.

Aunque estén escritas en C, el programador accede a ellas desde un API de Java, que es el lenguaje que se usa para desarrollar en Android. Para ello, el sistema incluye una máquina virtual java (JVM). La máquina virtual que incluye Android se llama Dalvik, y ha sido creada por Google para correr en dispositivos con poca memoria y poca capacidad de proceso. A diferencia de la JVM de SUN, Dalvik ejecuta archivos *.dex* en lugar de los clásicos archivos *.class* de Java. Los archivos *.dex* son más compactos y están más optimizados para el entorno del teléfono. No dispondremos de toda la API de JavaSE o JavaME, sino que se incluye un subconjunto llamado *Core Libraries*.

Toda la programación del dispositivo se hace usando el *Framework de aplicación*, que nos ofrece todo lo necesario. Cosas como el ciclo de vida de la aplicación o la gestión de recursos, se gestionan desde aquí.

Listado 1: main.xml.

```
01 <?xml version="1.0"
   encoding="utf-8"?>
02 <LinearLayout
   xmlns:android="http://schemas.
   android.com/apk/res/android"
03
   android:orientation="horizonta
   l"
04
   android:layout_width="fill_par
   ent"
05
   android:layout_height="fill_pa
   rent" />
06
07 <EditText
08     android:id="@+id/url_text"
09
   android:layout_width="wrap_con
   tent"
10
   android:layout_height="wrap_co
   ntent"
11     android:layout_weight="1"
12     android:lines="1" />
13
14 <Button
15
   android:id="@+id/nav_button"
16
   android:layout_width="wrap_con
   tent"
17
   android:layout_height="wrap_co
   ntent"
18
   android:text="@string/go_url"
   />
19 </LinearLayout>
```

Finalmente, en la capa superior están las aplicaciones de usuario, desde el navegador web hasta la propia aplicación que nos permite llamar por teléfono pasando por las aplicaciones de Google, como Maps o Gmail.

Teléfonos

El primer teléfono en utilizar Android ha sido el HTC G1. Desgraciadamente, en Europa sólo lo comercializa T-Mobile, que no ofrece servicio en España. Sin embargo, en el momento de escribir este artículo, Telefónica Movistar ha comunicado que va a comercializarlo (bajo el nombre de HTC Dream) y Vodafone está a punto de comercializar el HTC Magic (también conocido como HTC G2). Probablemente, cuando lea estas líneas ambos teléfonos estarán ya disponibles.

En principio, son muy parecidos. El HTC Dream incluye un teclado físico que no incluye el HTC Magic, a cambio, HTC Magic incorpora más capacidad de memoria ROM y una batería más longeva. En el Cuadro 1 se describen las características de ambos.

El Entorno de Desarrollo

La última versión del SDK de Android es la 1.1. Puede descargarse de [1]. El SDK nos ofrece, además de un emulador, todas las aplicaciones y librerías que vamos a necesitar para desarrollar aplicaciones. El emulador recrea perfectamente un móvil Android, por lo que, en principio, casi todo el desarrollo podremos hacerlo sin usar un terminal real. Para instalar el SDK sólo hay que descomprimir el archivo que hemos descargado en un directorio (por ejemplo, */opt/androidSDK/*), y ya podremos empezar a trabajar con el SDK; sin embargo, usarlo directamente puede ser un poco engorroso.

Afortunadamente, han pensado en todo y han creado un plugin para Eclipse que nos va a facilitar mucho el trabajo. Instalar el plugin en Eclipse es sencillo. Lo primero es descargar Eclipse de [2] en caso de que no lo tengamos ya instalado. Lo iniciamos, y seleccionamos el menú *Help* > *Software updates*... Tras seleccionar la pestaña *Available Software*, pulsamos el botón *Add Site*... y añadimos la URL *https://dl-ssl.google.com/android/eclipse/*. Pulsamos *OK*. Nos aparecerá el nuevo sitio en el listado de la pestaña *Available Software*. Lo marcamos y pulsamos el botón *Install*. A partir de aquí sólo hay que seguir los pasos que nos va indicando Eclipse. Una vez instalado, seleccionamos *Window* > *Preferences* > *Android*, y pulsando el botón *Browse*, seleccionamos el directorio donde hemos instalado el SDK. Ya podemos empezar a crear nuestra primera aplicación.

Modelo de Programación

Antes de crear nuestra primera aplicación Android, vamos a exponer algunos conceptos que habrá que tener presentes a la hora de crear aplicaciones. Empecemos con las **Actividades** (Activities). Una Actividad se corresponde con una pantalla de la aplicación, es decir, que tendremos tantas actividades como pantallas tenga nuestro programa. Cada Actividad es responsable de mantener su estado, de forma que puedan integrarse en el ciclo de vida de la aplicación, que es gestionado por el propio framework de aplicación. En Android podemos crear las interfaces de usuario de dos formas, desde la propia actividad usando código Java, o usando un fichero XML para describirla como si fuera una página HTML. Esta última es la manera más sencilla y cómoda.

Otro elemento importante son los **Intents**. Un Intent es un mecanismo para describir una acción, por ejemplo, hacer una foto o enviar un email. Todas las acciones que queramos realizar la haremos mediante Intents. Este mecanismo es muy flexible ya que, por ejemplo, podríamos crear una actividad para envío de mensajes SMS y registrarla de forma que maneje el Intent correspondiente. De esta forma, cuando alguien quiera enviar un SMS mediante este Intent, se invocará nues-

Figura 1: Arquitetura de Android.

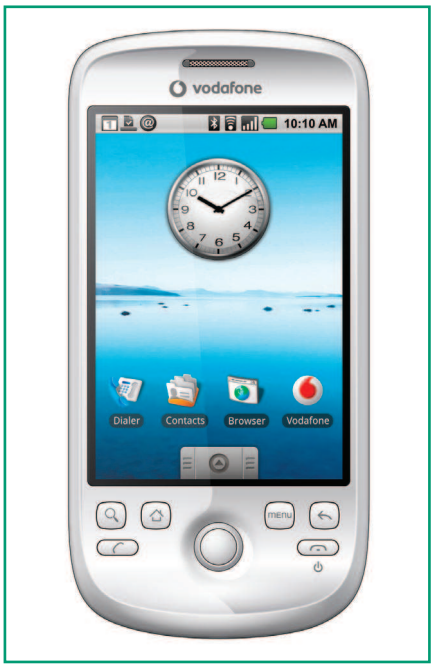


Figura 2: Nuevo HTC Magic de Vodafone.

tra actividad en vez de la actividad por defecto para envío de mensajes. En el ejemplo que desarrollaremos en este artículo, le pediremos a Android que abra una página web mediante un Intent, y el sistema operativo sabrá qué aplicación tiene que utilizar para ello.

Los **Servicios** (Services) son tareas que corren en segundo plano, como si se tratara de un demonio Unix. Imaginemos que queremos hacer sonar un MP3, pero mientras escuchamos, queremos poder seguir usando el teléfono. Mediante un servicio, podemos dejar sonando la música de fondo mientras usamos otras aplicaciones. Una actividad puede luego acoplarse a un servicio para, por ejemplo, parar o cambiar la canción.

Por último, los **Proveedores de contenido** (Contents Providers) almacenan datos que son compartidos por todas las aplicaciones. El acceso a esos datos se hace a través del API definida para cada proveedor de contenidos. Un ejemplo de proveedor de contenidos que ofrece Google son los contactos que tenemos almacenados en nuestro teléfono. Desde nuestra aplicación podremos acceder a los contactos usando el proveedor de contenidos correspondiente.

Nuestro Primer Programa

Vamos a crear nuestro primer programa Android, y para ello vamos a usar Eclipse. Para mostrar las capacidades de la plataforma, vamos a crear un sencillo navegador web. Seleccionamos el menú *File* > *New* > *Project...* y en la ventana que aparece a continuación seleccionamos *Android Project*. Eclipse abrirá un asistente que generará el esqueleto de una aplicación sencilla. Simplemente rellenamos los campos de esta nueva ventana con los mismos que pueden observarse en la figura 3 y pulsamos el botón *Finish*. En la figura 4 podemos ver cómo queda el entorno de desarrollo y la estructura de directorios que ha creado el asistente. Vamos a echar un vistazo más a fondo: El archivo principal de nuestra actividad es *AndroidBrowser.java*. Cada una de las actividades (pantallas) del programa hereda de la clase *Activity*. El framework se encarga de llamar al método *onCreate()* de nuestra actividad cuando se lanza la aplicación, y en este caso, lo único que se hace es, mediante *setContentView()*, decirle a Android qué vista queremos mostrar en la pantalla. La vista se referencia mediante un archivo de recursos llamado *R.java*. En principio, la gestión de este archivo es automática y nosotros no tendremos que preocuparnos por él, ya que según se añadan recursos, Eclipse actualizará la clase. Todos los recursos de nuestra aplicación se almacenan dentro del directorio *res*. En nuestro ejemplo, se hace referencia a un recurso llamado *main*, que está en la carpeta *layout* del directorio *res*. En este archivo se describe la interfaz de usuario como fichero XML, de forma parecida a como lo haríamos con un archivo HTML. En este caso se utiliza un layout llamado *LinearLayout* (que dispone todos los elementos de la interfaz de forma lineal) y dentro una etiqueta *TextView*, que muestra un texto en pantalla. Por supuesto hay una larga lista de tipos de layouts y de componentes de interfaz, como botones, listas, etc...

En la etiqueta *TextView*, podemos ver la propiedad *android:text="@string/hello"* que hace referencia a una etiqueta XML en el archivo de recursos *values/strings.xml* y que se llama *hello*. En este archivo almacenamos todos los literales de la aplicación, haciendo más sencilla la internacionalización de las aplicaciones.

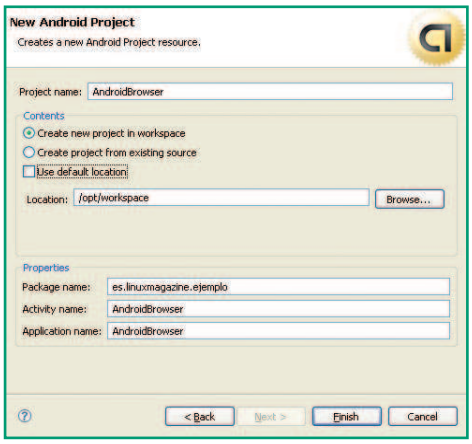


Figura 3: Asistente para la creación de una aplicación Android en Eclipse.

Por último, hablaremos del archivo *androidManifest.xml*, que contiene una gran cantidad de información acerca de la aplicación, como por ejemplo, nombre de la aplicación, versión, actividades que la componen (tendremos que registrar cada actividad del programa aquí), información de permisos y firmas, documentación, etc...

Después de este recorrido por este esqueleto de aplicación, es hora de ejecutarlo. En el explorador de paquetes de Eclipse, pulsamos con el botón derecho del ratón sobre la carpeta raíz de la aplicación y seleccionamos *Run as* > *Android application*. Esto lanzará el emulador y nos permitirá ver a Android saludándonos. Pero no vamos a quedarnos aquí, vamos a continuar creando nuestro navegador.

Comenzaremos definiendo la interfaz de usuario sustituyendo el archivo *res/layout/main.xml* por el que aparece en el listado 1. Estamos creando un cuadro de texto para introducir la URL y un botón para navegar. Todo de forma declarativa.

Al archivo */res/values/strings.xml* le añadimos la línea que define el texto "Ir..." para el botón, tal y como se observa en el listado 2.

Listado 2: strings.xml.

```
01 <?xml version="1.0"
   encoding="utf-8"?>
02 <resources>
03     <string
   name="app_name">AndroidBrowser
   </string>
04     <string
   name="go_url">Ir...</string>
05 </resources>
```

Por último, vamos a añadir el código que aparece en el listado 3 al archivo AndroidBrowser.java. Podemos ejecutarlo igual que lo hicimos antes.

Lo que realmente ocurre en este código, es que cuando pulsamos el botón "Ir…", Android trata de lanzar un *Intent* que ya tiene predefinido (ACTION_VIEW), y al que se le pasa la URL (como objeto de tipo Uri). Este *Intent* ya sabe que para ver este tipo de contenidos necesita lanzar el navegador del teléfono.

En [1] podemos acceder a tutoriales y a toda la documentación necesaria para continuar profundizando.

Android Market

Ya llevas algún tiempo programando para Android y has creado tu *killer application*. ¿Y ahora qué? ¿Cómo la distribuyo? Google ha pensado en todo y ha creado Android Market, un sistema de distribución de aplicaciones, tanto gratuitas como de pago. Gracias a Market cualquiera puede acceder y descargar nuestra aplicación. Para poder ofrecerla sólo hay que subscribirse como desarrollador, lo que tiene un coste de 25$ (un solo pago).

Figura 4: Estructura de directorios de la aplicación.

Por otra parte, a diferencia de App Store para IPhone, no existen requisitos para subir una aplicación, es decir, que apenas hay filtros, con lo que, en principio, nadie va a rechazar nuestro programa. A cambio, es probable que se inunde de aplicaciones mal hechas o que directamente no funcionen, pero Market ofrece un sistema de puntuaciones por votación que pretende corregir este problema y facilitarnos el acceso a aplicaciones de calidad.

Listado 3: AndroidBrowser.java.

```
01 package
   es.linuxmagazine.ejemplo;
02
03 import android.app.Activity;
04 import android.content.Intent;
05 import
   android.view.View.OnClickListe
   ner;
06 import android.net.Uri;
07 import android.os.Bundle;
08 import android.view.View;
09 import android.widget.Button;
10 import
   android.widget.EditText;
11
12 public class AndroidBrowser
   extends Activity {
13
14   private EditText url;
15   private Button botonIr;
16
17   @Override
18   public void onCreate(Bundle
   savedInstanceState) {
19
   super.onCreate(savedInstanceSt
   ate);
20
   setContentView(R.layout.main);
21
22     url = (EditText)
   findViewById(R.id.url_text);
23     botonIr = (Button)
   findViewById(R.id.nav_button);
24
25
   botonIr.setOnClickListener(new
   OnClickListener() {
26     public void onClick(View
   view) {
27         Uri uri =
   Uri.parse(url.getText().toStri
   ng());
28         Intent browseIntent =
   new Intent(Intent.ACTION_VIEW,
   uri);
29
   startActivity(browseIntent);
30       }
31     });
32   }
33 }
```

Conclusiones

Cualquiera que haya programado para otras plataformas, como Symbian, sabe lo complicado que es todo el proceso. En especial, Symbian, además de tener un modelo de aplicación bastante complejo, la propia Nokia pone obstáculos a los desarrolladores de software libre con su modelo de firmado y restricciones de capacidades.

En Android todo es muy diferente. El propio sistema es libre, y tanto el entorno de desarrollo como el framework son extremadamente simples en cuanto se coge algo de práctica. Para el desarrollador, Android ofrece sencillez y acceso a todas las capacidades del terminal. ■

RECURSOS

[1] Página desarrolladores Android: *http://developer.android.com/*

[2] Página del proyecto Eclipse: *http://www.eclipse.org/*

[3] Página de Android Market: *http://www.android.com/market/*

[4] Página principal de Android: *http://www.android.com/*