Desarrollo de aplicaciones sobre Android

Application Development for Android

Carlos Alberto Vanegas*

Fecha de recepción: 9 de agosto de 2012
Fecha de aceptación: 14 de septiembre de 2012

Resumen

El sistema operativo Android es una plataforma de código abierto que admite la construcción de aplicaciones para, de manera sencilla, manejar, configurar y personalizar el dispositivo móvil; esto hace que se pueda disfrutar de una interfaz amigable en la que se puede priorizar necesidades como correo, chat, redes sociales, multimedia, mensajería, etc. En este artículo se presenta una breve descripción de Android, sus características y su arquitectura, así como el software necesario para el desarrollo de aplicaciones Android, complementado con una sencilla aplicación para realizar algunas operaciones matemáticas.

Palabras clave: aplicación, interfaz de usuario, máquina virtual, sistema operativo.

* Ingeniero de sistemas, Universidad Incca de Colombia. Especialista en Ingeniería de Software, Universidad Distrital Francisco José de Caldas. Magíster en Ingeniería de Sistemas, Universidad Nacional de Colombia. Docente investigador del grupo CompuParalela adscrito a la Facultad Tecnológica de la Universidad Distrital Francisco José de Caldas. cavanegas@udistrital.edu.co

Abstract

The Android operating system is an open source platform that supports building applications that allow managing, configuring and customizing the mobile device in an easy way. This makes it possible to enjoy a friendly interface where one can prioritize some needs such as email, chat, social networking, multimedia and messaging. In this article a brief description of the Android operating system is showed. Furthermore, its features, architecture and the software necessary for the development of Android applications are reviewed. All this complemented by a simple application to perform some mathematical operations.

Key words: Application, operating system, user interface, virtual machine.

1. Introducción

Android es un sistema operativo de código abierto para móviles basado en el núcleo de Linux, el cual permite desarrollar ilimitadas aplicaciones para teléfonos inteligentes, tabletas, reproductores MP3, televisores, cámaras, como también admite la distribución de los productos generados [1]. El proyecto Android es liderado por el grupo Open Handset Alliance, en el cual se agrupan varios fabricantes, desarrolladores de hardware y software, entre los cuales se pueden mencionar: Google, Samsung, HTC, Dell, Intel, Qualcomm, Motorola, LG, Telefónica, T-Mobile, Nvidia [2]. Mediante alianzas en el mundo, Android ha tenido un crecimiento acelerado, desde su primera versión [3] 1.0 liberada el 23 de septiembre del 2008 hasta la actual versión 4.0.3; cuenta con más de 200 millones de dispositivos móviles activados, lo que hace que cada día se activen 550.000 nuevos dispositivos en más de 137 países. Por otro lado, en el tercer semestre del 2011 se descargaron más de 2400 millones de aplicaciones para los dispositivos Android [1].

El SDK (*kit de desarrollo de software*) de Android proporciona las herramientas (correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros) y Apis (*interfaz de programación de aplicaciones*) necesarias para empezar a desarrollar aplicaciones en la plataforma Android usando el lenguaje de programación Java [4]. Entre las características de la plataforma Android se encuentran:

- *Application framework (marco de trabajo de aplicaciones)*: esta característica admite el cambio y la reutilización de cada uno de los componentes.

- *Dalvik virtual machine (máquina virtual Dalvik)* [5]: máquina virtual (VM) llamada *Dalvik* del sistema operativo Android; es el software que ejecuta las aplicaciones de los dispositivos móviles. La VM genera los archivos con extensión:

- *.apk*: archivos comprimidos con los ejecutables y los datos del programa.
- *.dex*: archivo ejecutable, similar al *.class* de Java que optimiza el rendimiento en la VM.

- *Integrated browser (navegador inteligente)*: basado en el proyecto de código abierto Webkit [6], el cual es un sistema que funciona como base para los navegadores web *Safari*, *Google Chrome*, *Epiphany*, *Maxthon*, entre otros.

- *Optimized graphics (optimizador de gráficas)*: librerías para realizar gráficas 2D y 3D sobre las especificaciones de *OpenGL 1.0*, donde se puede incluir opcionalmente un acelerador de hardware.

- *SQLite*: gestor de base de datos transaccional para sistemas operativos de móviles.

- *Media support (soporte de multimedia)*: permite trabajar con los formatos más comunes de audio, video e imágenes.

- *GSM Telephony (telefonía GSM)*: proporciona interfaces para programar aplicaciones en que se utilicen todas las características de la telefonía GSM, como texto, datos y mensajes SMS.

- *Rich development environment (entorno de desarrollo)*: esta característica incluye emuladores de dispositivos, herramientas para depuración de aplicaciones, memorias, perfiles de rendimiento y el conector para desarrollar en el entorno de Eclipse IDE.

- *Bluetooth, EDGE, 3G y Wifi, Camera, GPS, compass and accelerometer*: son características que se pueden implementar con Android, pero son dependientes del hardware.

2. Arquitectura Android

La arquitectura de Android [7], [8] está formada por 4 *capas* que permiten la generación de aplicaciones. El acceso a cada capa se realiza por intermedio de librerías, en las cuales cada capa puede utilizar los elementos de la capa inferior para realizar sus funciones. Las capas de la arquitectura Android son:

- *Linux kernel*: capa de abstracción del hardware (*hardware abstraction layer*), que permite que las aplicaciones accedan a través de controladores (*drivers*) asumiendo la administración de los recursos del teléfono y del sistema operativo. En esta capa están disponibles los controladores para *display* (pantalla), *keypad* (teclado), *camera* (cámara), *wifi* (conexión inalámbrica), *flash memory* (memoria rápida), *audio*, *binder* (cobertura) y *power management* (administrador de energía).

- *Librerías (libraries)*: bibliotecas de Android escritas en C/ C++, que se encargan de realizar la comunicación entre la capa de abstracción de hardware con la interfaz de programación de aplicaciones. Se destacan: *surface manager* (gestión gráfica), *openGL/ES* (gráficas 3D), *SGL* (gráficas 2D), *media framework* (multimedia), *freetype* (fuentes de mapas de bits), *ssl* (seguridad), *SQLite*, *Webkit*, *libc* (librerías C).

- *Entorno de ejecución Android (Android runtime)*: no es considerada una capa, pero está formada por las siguientes librerías: *Dalvik virtual machine* (máquina virtual Dalvik), *Core libraries* (núcleo de librerías).

- *Marco de trabajo de aplicaciones (application framework)*: conformado por clases y servicios que permiten las funciones básicas de los móviles y la programación de las aplicaciones. Se encuentran los siguientes elementos: *activity manager* (administrador de actividades), *windows manager* (administrador de ventanas), *content provider* (compartidor de aplicaciones), *views* (vistas), *notification manager* (administrador de notificaciones), *package manager* (administrador de paquetes), *telephony manager* (administrador telefónico), *resource manager* (administrador de recursos), *location manager* (administrador de posicionamiento), *applications* (aplicaciones).

3. Software para desarrollar aplicaciones Android desde Eclipse IDE

Para crear una aplicación Android en Windows utilizando el software Eclipse IDE para Java (recomendado por Android para crear *Apps* –aplicaciones–), es necesario instalar el siguiente software:

a) *Java Development Kit # (JDK#)*: este kit es necesario para desarrollar aplicaciones en Java (recomendable la versión 5 o superior); se puede descargar de www.oracle.com/technetwork/java/javase/downloads/index.html.

b) *SDK Android*: descargar e instalar el kit para Windows desde la página oficial http://developer.android.com/sdk/index.html. La descarga se puede realizar desde el vínculo *android-sdk r16-windows.zip* o el vínculo *installer r16-windows.exe* (recomendable).

c) *Eclipse IDE for Java Developers*: descargar el archivo *.zip* desde www.eclipse.org/downloads/. Únicamente se deberá descomprimir el archivo en la carpeta deseada y pulsar sobre el archivo *eclipse.exe* para ingresar al entorno de desarrollo de Eclipse IDE.

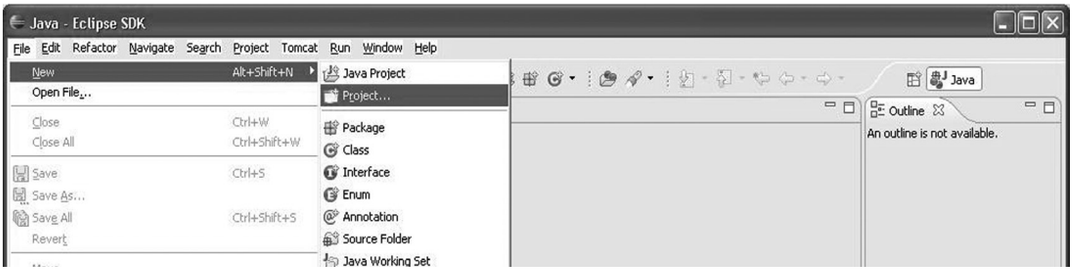
d) *Plug-in Android para Eclipse*: instalar el *plug-in* (conector) de Android para desarrollar aplicaciones Android utilizando Eclipse IDE. Se puede apoyar con los tutoriales que se encuentran en las páginas www.weterede.com/2009/06/instalacion-sdk-de-android-en-eclipse/ y http://instartius.com/blog/?p=289.

4. Crear una aplicación Android con Eclipse IDE

Para desarrollar una aplicación Android llamada *OperacionesMatematicas* que permita capturar en un campo de texto un número y en un segundo campo de texto visualizar la raíz cuadrada, la potencia, el factorial y si es un número primo o no, se deben realizar los siguientes pasos:

1. Dentro del entorno de Eclipse IDE se selecciona *File* > *New* > *Project*, como se observa en la figura 1.

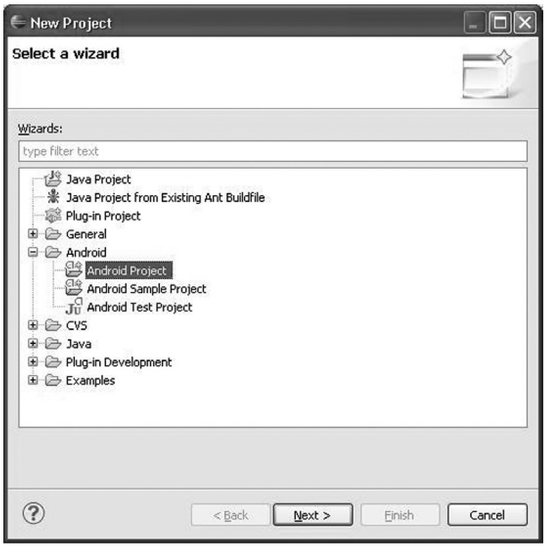
Figura 1. Entorno de desarrollo Java – Eclipse SDK



Fuente: elaboración propia.

Al pulsar la opción *Project*, se obtendrá la figura 2.

Figura 2. Asistente para crear un nuevo proyecto Android



Fuente: elaboración propia.

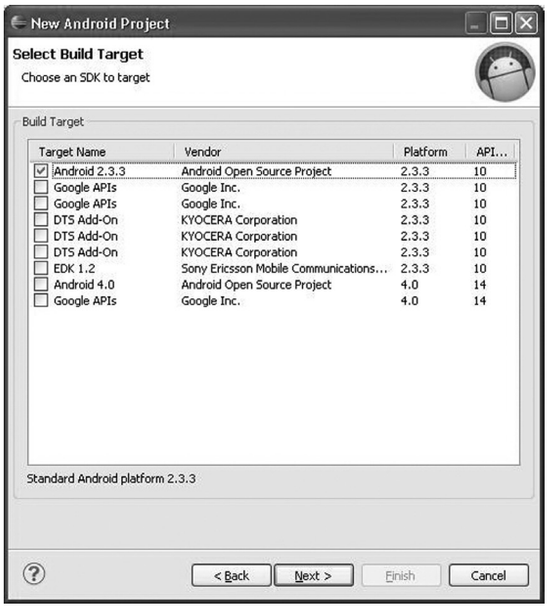
2. Se selecciona el icono *Android Project* y se pulsa el botón *Next* para visualizar la pantalla de la figura 3.

Figura 3. Selección del nombre del nuevo proyecto Android

Fuente: elaboración propia

3. En el campo de texto al frente de Project Name se escribe el nombre del proyecto, OperacionesMatematicas, y nuevamente se pulsa el botón Next para obtener la figura 4.

Figura 4. Selección de la versión de la plataforma Android



Fuente: elaboración propia.

4. Se selecciona la versión SDK de Android que previamente se haya instalado; para el ejemplo se eligió la plataforma Android 2.3.3. Se pulsa el botón Next para ver la a figura 5.

Figura 5. Configuración del nuevo proyecto Android

Fuente: elaboración propia.

En esta ventana se puede modificar todo lo concerniente a la informacion de la aplicación:

- *Application Name*: nombre de la aplicación que aparecerá en el dispositivo Android, tanto en la barra superior durante la ejecucion como en el icono que se instalará en el menú de programas.

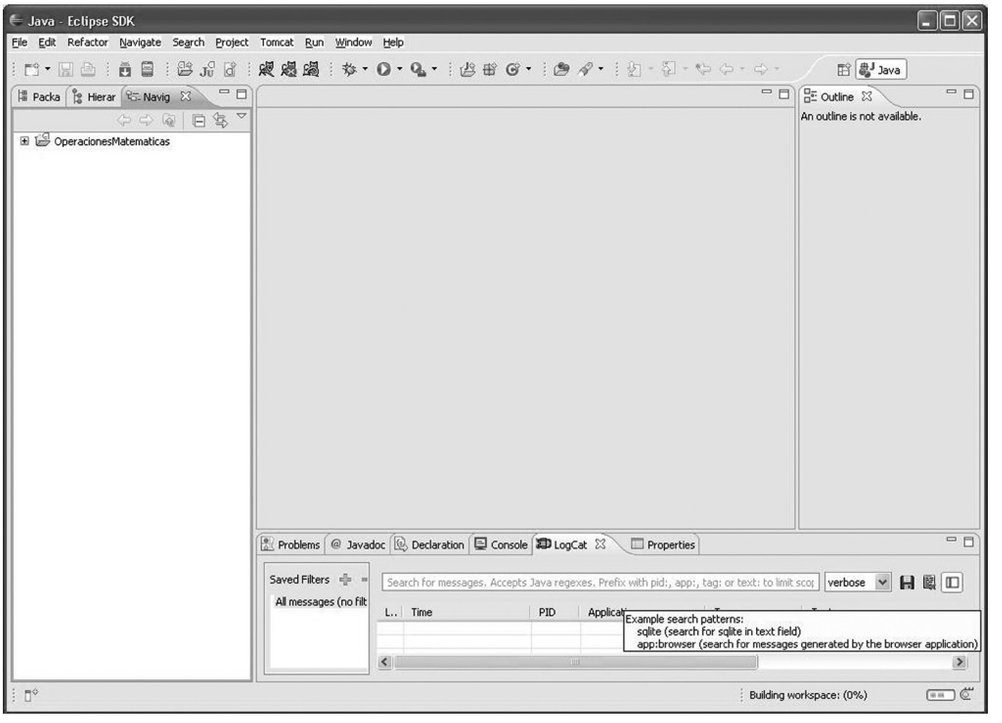
- *Package Name*: nombre del paquete del espacio de nombres donde se almacenará la aplicación generada.

- *Create Activity*: nombre de la clase que se generará. Esta será una subclase de *Activity Android*; esta clase se puede ejecutar y modificar.

- *Minimum SDK*: especifica el nivel del API Android [3] que requiere la aplicación.

Al pulsar el boton *Finish*, se obtendrá la figura 6.

Figura 6. Ventana del proyecto Android finalizado



Fuente: elaboración propia.

Si se pulsa sobre el signo + al lado del nombre de la aplicación, se observarán los elementos del proyecto Android (figura 7) [9]:

- *assets*: carpeta que contiene una serie de archivos de datos, JAR externos, fuentes, entre otros, que pueden ser utilizados por la aplicación.

- *bin*: carpeta que contiene todos los archivos *.class*.

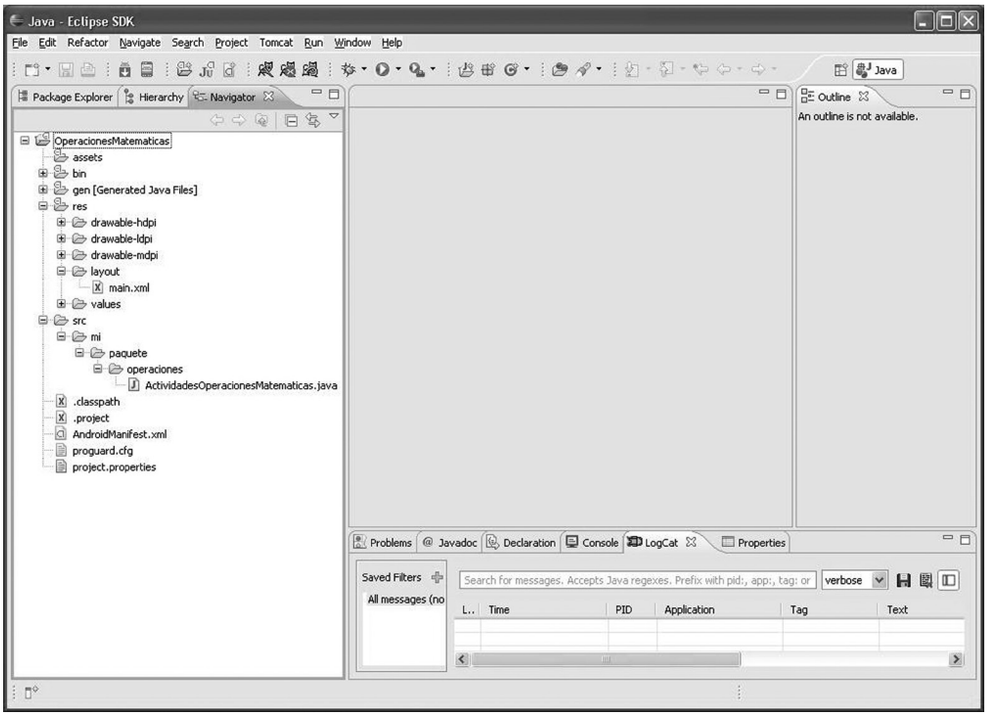
- *gen*: carpeta que contiene archivos generados automáticamente por SDK Android. Estos no deben modificarse.

- *res*: carpeta que contiene los recursos usados por la aplicación.

- *src*: carpeta que contiene el código fuente de la aplicación. Los archivos se almacenan en el espacio de nombres creado.

- *AndroidManifest.xml*: archivo que describe la aplicación donde se indican las actividades, intentos, servicios y proveedores de contenido.

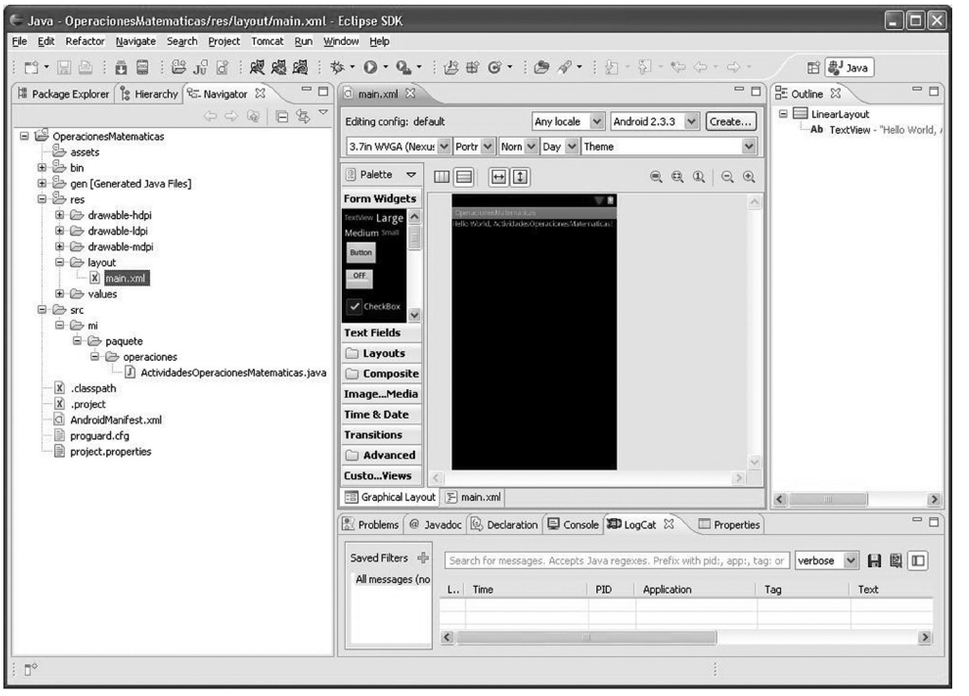
Figura 7. Elementos generados por el proyecto Android



Fuente: elaboración propia.

5. Para diseñar la pantalla principal de la aplicación se deberá abrir el archivo *main.xml* que se encuentra en la carpeta *res/layout/*. En la parte inferior de la ventana central se visualizarán dos pestañas: *Graphics Layout* y *main.xml*. En *Graphics Layout* se mostrarán en forma visual los elementos actuales de la aplicación, así como también un cuadro de elementos que se pueden adicionar a la aplicación. Esto se puede apreciar en la figura 8.

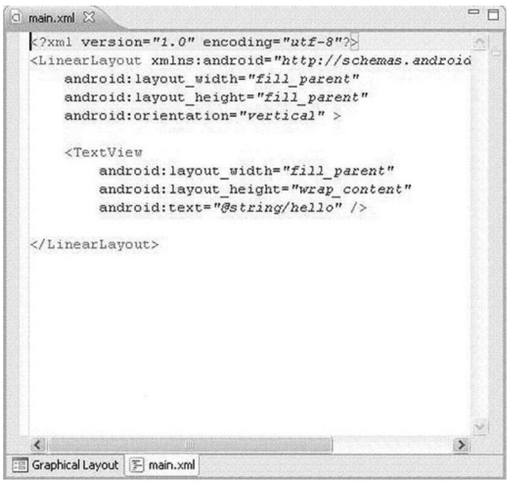
Figura 8. Vista visual del archivo main.xml



Fuente: elaboración propia.

En la parte izquierda se visualizarán los elementos actuales de la aplicación, en este caso un *LinearLayout* (contenedor) y un *TextView* (campo de texto). Además, en la ficha *main.xml* se podrá ver el código XML, el cual puede ser editado directamente. La figura 9 describe el código del archivo *main.xml*.

Figura 9. Código XML del archivo main.xml

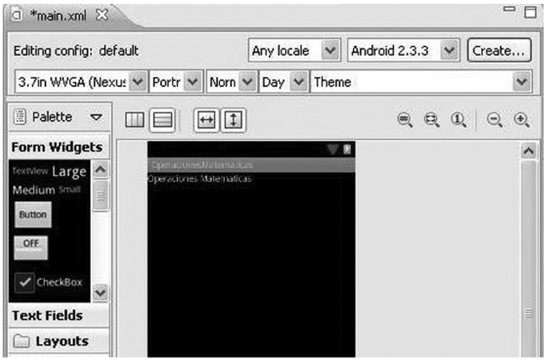


Fuente: elaboración propia.

Como se puede observar, existen los elementos *LinearLayout, TextView* (dentro del elemento *LinearLayout* se encuentra el elemento *TextView*). En dichos elementos se definen una serie de atributos como son: el ancho (*layout_width*), el alto (*layout_height*), la orientación (*orientation*) y el texto (*text*), el cual es una referencia definida en el archivo *res/values/strings.xml*.

6. En la pestaña del código XML se modifica la propiedad *android:text="@string/hello"* por *android:text="Operaciones Matemáticas"* para cambiar el título actual de la pantalla principal. En la figura 10 se podrá observar cómo queda la vista visual de la aplicación.

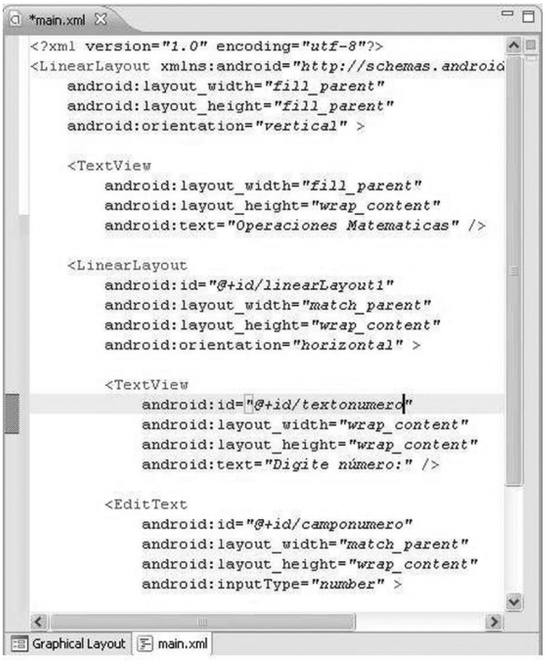
Figura 10. Vista del objeto *TextView* con la propiedad *Text* modificada



Fuente: elaboración propia.

7. Desde la ficha *Layouts* del cuadro de elementos se adiciona a la aplicación un elemento *LinearLayout*; desde la ficha *Form Widgets,* un elemento *TextView*, y desde la ficha *Text Fields*, un elemento *EditText*. A partir del código XML se modifican los atributos de los elementos de acuerdo con lo que se observa en la figura 11.

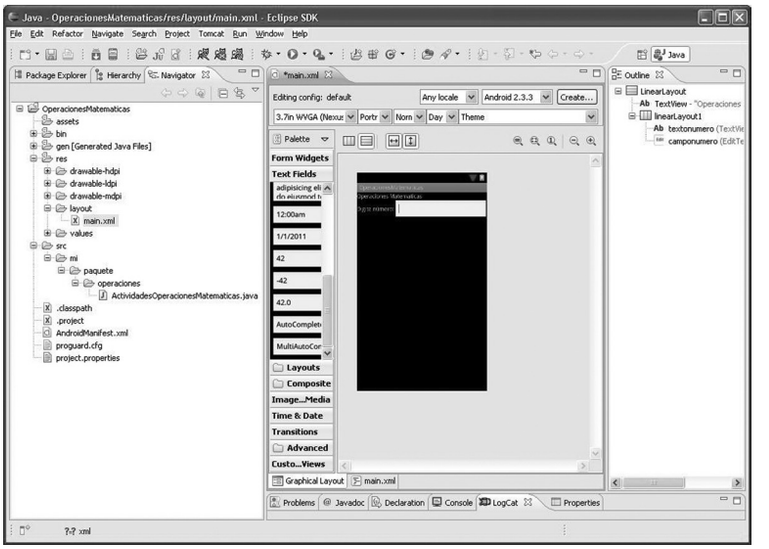
Figura 11. Código modificado de los objetos *TextView*, *EditText* y *LinearLayout*



Fuente: elaboración propia.

Con la adición de los elementos y la modificación del código se deberá observar la figura 12.

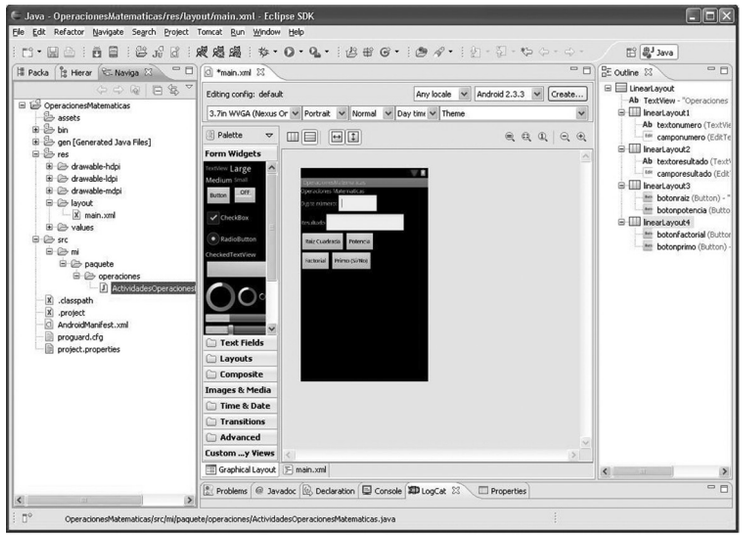
Figura 12. Vista de los objetos *TextView*, *EditText* y *LinearLayout* al modificar sus propiedades



Fuente: elaboración propia.

8. Ahora se adicionan tres elementos *LinearLayout*, dos *TextView*, dos *EditText* y cuatro *Button*. Se modifica el código para obtener la figura 13.

Figura 13.Vista final de la aplicación



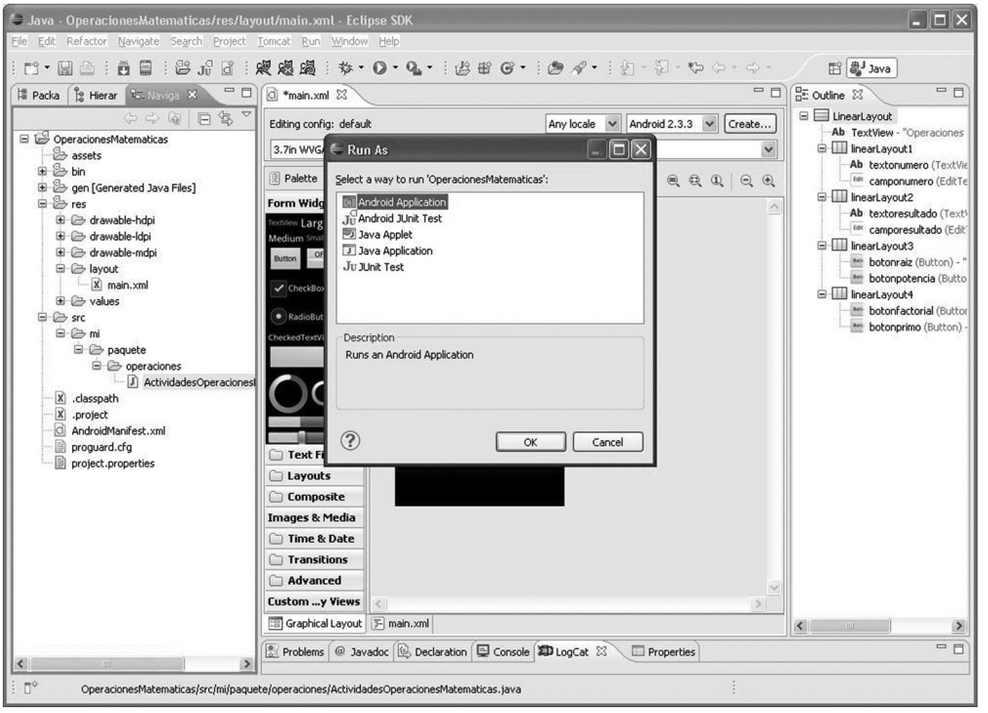
Fuente: elaboración propia.

El código final del archivo *main.xml* es el siguiente:

```
<?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="Operaciones Matematicas"
/>
<LinearLayout
android:id="@+id/linearLayout1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal" >
<TextView
android:id="@+id/textonumero"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Digite número:" />
<EditText
android:id="@+id/camponumero"
android:layout_width="102dp"
android:layout_height="wrap_content"
android:inputType="number" >
<requestFocus />
</EditText>
</LinearLayout>
<LinearLayout
android:id="@+id/linearLayout2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal" >
<TextView
android:id="@+id/textoresultado"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Resultado" />
<EditText
android:id="@+id/camporesultado"
android:layout_width="200dp"
android:layout_height="wrap_content"
android:inputType="number" />
</LinearLayout>
<LinearLayout
android:id="@+id/linearLayout3"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal" >
<Button
android:id="@+id/botonraiz"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Raiz Cuadrada" />
<Button
android:id="@+id/botonpotencia"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Potencia" />
</LinearLayout>
<LinearLayout
android:id="@+id/linearLayout4"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal" >
<Button
android:id="@+id/botonfactorial"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Factorial" />
<Button
android:id="@+id/botonprimo"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Primo (Si/No)" />
</LinearLayout>
</LinearLayout>
```

9. Para visualizar la aplicación en un emulador Android, se sitúa sobre el nombre del archivo .java (*ActividadesOperaciones-Matematicas*) y se pulsa *Ctrl + F11*, a fin de visualizar la figura 14.

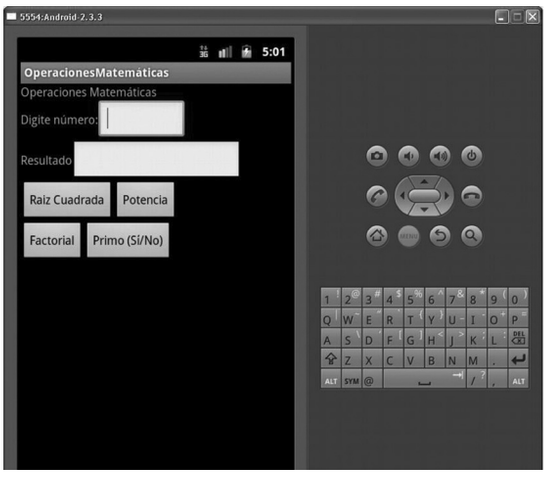
Figura 14. Ejecución de la aplicación Android



Fuente: elaboración propia.

10. Se selecciona la opción *Android Application* y se pulsa el botón *Ok* para visualizar la aplicación en el emulador Android seleccionado. Cuando se cargue el emulador se pulsa el botón *MENU* para apreciar la figura 15.

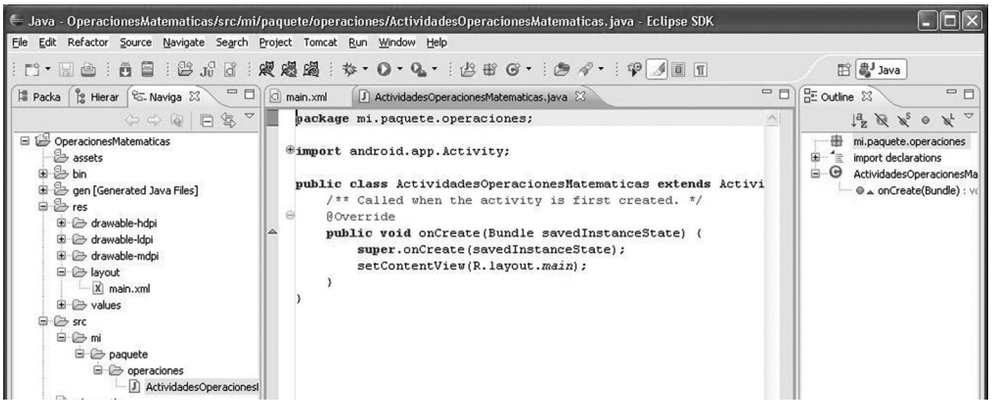
Figura 15. Vista de la aplicación en el emulador Android



Fuente: elaboración propia.

11. Por último se debe modificar el código del archivo *ActividadesOperacionesMatematicas.java,* con el fin de escribir el código necesario para que cada botón realice la operación esperada. El código que se genera automáticamente se puede observar en la figura 16.

Figura 16. Código generado en el archivo de Java



Fuente: elaboración propia.

La clase *ActividadesOperacionesMatemticas* hereda (*extends*) las propiedades de la clase *Activity*, lo que permite ejecutar acciones. También se creó el método *onCreate()*, el cual será llamado por la aplicación Android cuando se inicie su ejecución. En este método se debe realizar la inicialización y la configuración de la interfaz del usuario.

Entonces, antes del método *onCreate()* se crean las siguientes variables con el fin de asignarles posteriormente valores.

```
private TextView nuevotexto;
private EditText numerodigitado;
private EditText resultadoobtenido;
```

Después de la instrucción *setContentView(R.layout.main)* se asignan a las variables creadas los valores que contengan los objetos *textoresultado*, *camponumero*, *camporesultado*, respectivamente, como se muestra a continuación:

```
nuevotexto=(TextView)findViewById(R.
id.textoresultado);
numerodigitado=(EditText)
findViewById(R.id.camponumero);
resultadoobtenido=(EditText)
findViewById(R.id.camporesultado);
```

En las variables se almacenan las referencias de los tres objetos. El método *findViewById()* permite enlazar las variables con los objetos creados en el archivo *main.xml*. Se pasa como argumento la constante creada en la clase *R* (se genera automáticamente), el identificador del objeto (*id*) y el nombre del objeto. Como la clase *findViewById* retorna una clase tipo *View* se debe realizar el operador *cast* para cada objeto.

A continuación se crean cuatro variables tipo *Button* y se le asigna a cada una los objetos *Button* definidos en el archivo *main.xml*.

```
Button verraiz=(Button)findViewById(R.
id.botonraiz);
Button verpotencia=(Button)
findViewById(R.id.botonpotencia);
Button verfactorial=(Button)
findViewById(R.id.botonfactorial);
Button verprimo=(Button)findViewById(R.
id.botonprimo);
```

Finalmente, a cada variable tipo *Button* se le añaden *listeners* (oidores) para que, cuando el usuario haga clic sobre algún botón, realice un determinado proceso. Además se implementa un método *onClick()* para el *listener*.

```
verraiz.setOnClickListener(new View.On-
ClickListener() {
public void onClick(View v) {
nuevotexto.setText("La raiz es:");
resultadoobtenido.setText(""+Math.
sqrt(Double.parseDouble(
numerodigitado.getText().toString())));
}});
```

A la variable *verraiz* se le adiciona el *listener* utilizando el método *setOnClickListner*. En el método *onClick()*a la variable *nuevotexto* en su propiedad *setText* se le asigna el texto *"La raíz es:"* y a la variable *resultadoobtenido* la raíz cuadrada del valor obtenido de la variable *numerodigitado* utilizando el método *sqrt()* de la clase *Math* de Java. A continuación se muestra el código de los demás botones.

```
// código para obtener la potencia.
verpotencia.setOnClickListener(new View.On-
ClickListener() {
public void onClick(View v) {
nuevotexto.setText("La potencia es:");
        resultadoobtenido.setText(""+Math.
pow(Double.parseDouble(
numerodigitado.getText().toString()),3.0));
    }});
// código para obtener el factorial.
verfactorial.setOnClickListener(new View.On-
ClickListener() {
public void onClick(View v) {
nuevotexto.setText("El factorial es:");
double numero,factorial=1;
                    numero=Double.
parseDouble(numerodigitado.getText().
toString());
for (int i=2;i<=numero;i++)
    factorial*=i;
resultadoobtenido.setText(""+factorial);
    }})
;  // código para obtener si el número es o no
primo.
verprimo.setOnClickListener(new View.On-
ClickListener() {
public void onClick(View v) {
    nuevotexto.setText("El numero:");
    int seguir=0,divisor=2, numero;
    numero=Integer.
parseInt(numerodigitado.getText().toString());
    while(seguir==0)
    {
    if(numero%divisor==0)
      { resultadoobtenido.setText("No es
Primo");
    seguir=1; }
    else
    divisor++;
    if(numero==divisor)
        {  resultadoobtenido.setText("Es
Primo");
    seguir=1; }
    }
    }});
```

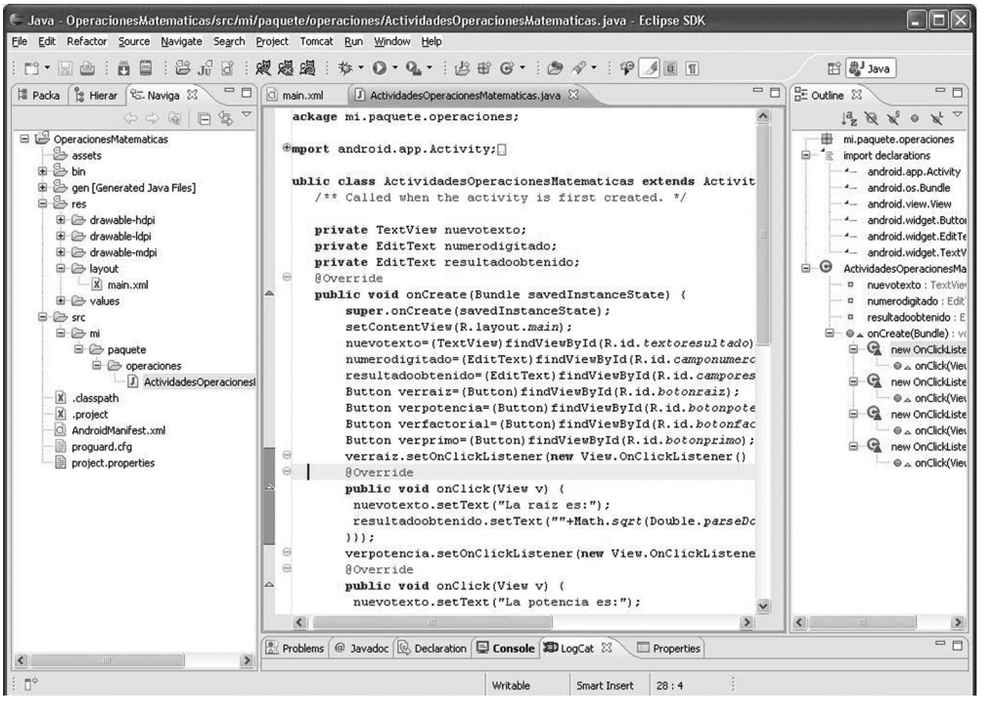
Para poder utilizar los diferentes objetos es necesario importar los paquetes específicos antes de la creación de la clase *ActividadesOperacionesMatematicas*. Existen dos maneras de hacerlo: escribir en forma manual cada paquete que se va a utilizar o simplemente

pulsar las teclas *Ctrl+Shift+O*, para que se añadan automáticamente. Los paquetes utilizados en la aplicación se escriben a continuación:

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

La figura 17 muestra cómo quedaría el código de la aplicación.

Figura 17. Código de la aplicación en el archivo de Java



Fuente: elaboración propia.

Si se ejecuta nuevamente la aplicación, y se digita el número 5 en el campo de texto respectivo y se pulsa el botón *Raíz Cuadrada*, se observará en el emulador la figura 18.

Figura 18. Aplicación final de Android ejecutada

Fuente: elaboración propia.

5. Conclusiones

- El sistema operativo Android permite el desarrollo de aplicaciones sencillas de aprender y entender.

- Con Android es fácil el manejo, la configuración y la personalización de un dispositivo móvil.

- El sistema operativo Android posibilita desarrollar aplicaciones utilizando instrucciones de Java.

6. Referencias

[1] Android everywhere. Disponible en www.android.com/developers/

[2] Open Handset Alliance. Disponible en www.openhandsetalliance.com/android_overview.html

[3] Apis Level. Disponible en http://developer.android.com/guide/appendix/api-levels.html

[4] What is Android. Disponible en http://developer.android.com/guide/basics/what-is-android.html

[5] La máquina virtual Dalvik. Disponible en http://gdroid.com.mx/blog/2011/06/12/la-maquina-virtual-dalvik/

[6] The Webkit open source Project. Disponible en http://www.webkit.org/

[7] Arquitectura de Android. Disponible en http://androideity.com/2011/07/04/arquitectura-de-android/

[8] Software de comunicaciones – Arquitectura Android. Disponible en https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

[9] Desarrollo de aplicaciones Android. Disponible en http://tednologia.com/introduccion-android/