



# Programación multimedia y dispositivos móviles

Jacinto D. Cabrera Rodríguez



Contenido actualizado



Contenidos digitales  
[www.sintesis.com](http://www.sintesis.com)



# **P**rogramación multimedia y dispositivos móviles

Contenido actualizado

Consulte nuestra página web: [www.sintesis.com](http://www.sintesis.com)  
En ella encontrará el catálogo completo y comentado



Queda prohibida, salvo excepción prevista en la ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra sin contar con autorización de los titulares de la propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (arts. 270 y sigs. Código Penal). El Centro Español de Derechos Reprográficos ([www.cedro.org](http://www.cedro.org)) vela por el respeto de los citados derechos.

# Programación multimedia y dispositivos móviles

Jacinto D. Cabrera  
Rodríguez

Contenido actualizado



**ASESOR EDITORIAL:**

Juan Carlos Moreno Pérez

© Jacinto D. Cabrera Rodríguez

© EDITORIAL SÍNTESIS, S. A.  
Vallehermoso, 34. 28015 Madrid  
Teléfono 91 593 20 98  
<http://www.sintesis.com>

ISBN: 978-84-1357-366-3  
Depósito Legal: M-17.207-2024

Impreso en España - Printed in Spain

Reservados todos los derechos. Está prohibido, bajo las sanciones penales y el resarcimiento civil previstos en las leyes, reproducir, registrar o transmitir esta publicación, íntegra o parcialmente, por cualquier sistema de recuperación y por cualquier medio, sea mecánico, electrónico, magnético, electroóptico, por fotocopia o por cualquier otro, sin la autorización previa por escrito de Editorial Síntesis, S. A.

# índice

<b>PRESENTACIÓN .....</b>	13
<b>1. TECNOLOGÍAS MÓVILES .....</b>	15
Objetivos .....	15
Mapa conceptual .....	16
Glosario .....	16
1.1. Introducción .....	17
1.2. Tecnologías móviles: características y limitaciones .....	17
1.2.1. Características de los dispositivos móviles .....	17
1.2.2. Tipos de dispositivos móviles .....	18
1.2.3. Tecnología de comunicación móvil .....	20
1.2.4. Limitaciones de los dispositivos móviles .....	22
1.3. Sistemas operativos móviles .....	23
1.3.1. Android .....	24
1.3.2. iOS .....	25
1.3.3. Windows Phone .....	25
1.3.4. BlackBerry OS .....	26
1.3.5. Symbian .....	26
1.3.6. Palm OS P (WebOS) .....	26
1.3.7. Firefox OS .....	26
1.3.8. Ubuntu Touch .....	27
1.3.9. Harmony OS .....	27
1.4. Lenguajes de programación para dispositivos móviles .....	27
1.4.1. Desarrollo nativo .....	27
1.4.2. Desarrollo multiplataforma compilado a nativo .....	27
1.4.3. Desarrollo multiplataforma basado en HTML5 .....	28

<b>1.5. Entornos integrados de desarrollo de aplicaciones móviles .....</b>	28
1.5.1. Symbian .....	29
1.5.2. BlackBerry OS .....	29
1.5.3. Windows Phone .....	29
1.5.4. iOS .....	29
1.5.5. Android .....	30
<b>Resumen .....</b>	30
<b>Actividades de autoevaluación .....</b>	31
<b>2. INTRODUCCIÓN A ANDROID .....</b>	33
<b>Objetivos .....</b>	33
<b>Mapa conceptual .....</b>	34
<b>Glosario .....</b>	34
<b>2.1. Introducción .....</b>	35
<b>2.2. Fundamentos de una aplicación en Android .....</b>	35
<b>2.3. Componentes de una aplicación .....</b>	36
<b>2.4. Android Manifest .....</b>	36
<b>2.5. Recursos de una aplicación .....</b>	38
<b>2.5.1. Cómo acceder a los recursos .....</b>	40
<b>2.6. La clase R .....</b>	41
<b>2.7. Estructura de un proyecto .....</b>	42
<b>2.7.1. Instalación de Android Studio .....</b>	42
<b>2.7.2. Iniciando el entorno de desarrollo .....</b>	44
<b>2.7.3. Elementos de un proyecto .....</b>	46
<b>2.7.4. Configurar el dispositivo donde probar las aplicaciones .....</b>	47
<b>2.8. Los permisos en Android .....</b>	48
<b>Resumen .....</b>	50
<b>Actividades de autoevaluación .....</b>	50
<b>3. LAS ACTIVIDADES EN ANDROID .....</b>	53
<b>Objetivos .....</b>	53
<b>Mapa conceptual .....</b>	54
<b>Glosario .....</b>	54
<b>3.1. Introducción .....</b>	55
<b>3.2. Las actividades .....</b>	55
<b>3.3. Ciclo de vida de una aplicación .....</b>	57
<b>3.4. ¡Hola mundo!, de Android .....</b>	59
<b>3.5. Conservar el estado de una aplicación .....</b>	62
<b>3.6. Intents y filtros .....</b>	63
<b>3.6.1. Intents explícitos .....</b>	64
<b>3.6.2. Intents implícitos .....</b>	65
<b>3.6.3. Elementos .....</b>	65
<b>3.6.4. Propagación .....</b>	67
<b>3.6.5. Filtros .....</b>	69
<b>3.6.6. PendingIntent .....</b>	69
<b>Resumen .....</b>	70
<b>Actividades de autoevaluación .....</b>	71

<b>4. INTERFACE DE USUARIO: LOS LAYOUTS .....</b>	73
Objetivos .....	73
Mapa conceptual .....	74
Glosario .....	74
4.1. Introducción .....	74
4.2. Interfaces de usuario. Clases asociadas .....	75
4.3. Layouts .....	77
4.4. LinearLayout .....	77
4.5. FrameLayout .....	78
4.6. AbsoluteLayout .....	79
4.7. RelativeLayout .....	80
4.8. TableLayout .....	81
4.9. GridLayout .....	83
4.10. ConstraintLayout .....	84
Resumen .....	86
Actividades de autoevaluación .....	87
<b>5. INTERFACE DE USUARIO: CONTROLES BÁSICOS .....</b>	89
Objetivos .....	89
Mapa conceptual .....	90
Glosario .....	90
5.1. Introducción .....	91
5.2. TextView .....	91
5.2.1. Cambiar las propiedades del texto .....	92
5.2.2. Añadir nuevo texto .....	93
5.2.3. Uso de fuentes variadas .....	93
5.2.4. Animación del texto .....	94
5.3. Button .....	97
5.4. ToggleButton .....	99
5.5. ImageButton .....	100
5.6. EditText .....	100
5.7. AutoCompleteTextView .....	103
5.8. MultiAutoCompleteTextView .....	103
5.9. Spinner .....	104
5.10. CheckBox .....	105
5.11. RadioButton .....	106
5.12. Switch .....	107
5.13. SeekBar .....	108
5.14. RatingBar .....	109
5.15. ProgressBar .....	109
Resumen .....	110
Actividades de autoevaluación .....	111
<b>6. LISTADOS Y MENÚS .....</b>	113
Objetivos .....	113
Mapa conceptual .....	114
Glosario .....	114
6.1. Introducción .....	115

<b>6.2. Los listados .....</b>	115
<b>6.3. ListView .....</b>	115
<b>6.4. GridView .....</b>	117
<b>6.5. Spinner .....</b>	119
<b>6.6. Trabajando con adaptadores .....</b>	120
<b>6.6.1. ArrayAdapter .....</b>	121
<b>6.6.2. BaseAdapter .....</b>	124
<b>6.7. OptionsMenu .....</b>	126
<b>6.8. Submenú .....</b>	127
<b>6.9. Menú contextual .....</b>	128
<b>6.10. Menú contextual en listas .....</b>	129
<b>Resumen .....</b>	130
<b>Actividades de autoevaluación .....</b>	131
<b>7. PREFERENCIAS, DIÁLOGOS Y NOTIFICACIONES .....</b>	133
<b>Objetivos .....</b>	133
<b>Mapa conceptual .....</b>	134
<b>Glosario .....</b>	134
<b>7.1. Introducción .....</b>	135
<b>7.2. Preferencias .....</b>	135
<b>7.2.1. Categorías .....</b>	135
<b>7.2.2. CheckBoxPreference .....</b>	136
<b>7.2.3. EditTextPreference .....</b>	136
<b>7.2.4. ListPreference .....</b>	136
<b>7.2.5. MultiSelectListPreference .....</b>	137
<b>7.2.6. Acceso a las preferencias .....</b>	137
<b>7.3. Toast .....</b>	138
<b>7.4. Diálogos .....</b>	139
<b>7.4.1. DatePickerDialog .....</b>	142
<b>7.4.2. TimePickerDialog .....</b>	143
<b>7.5. Notificaciones .....</b>	143
<b>7.5.1. Crear la notificación .....</b>	144
<b>7.5.2. Establecer la actividad de destino .....</b>	145
<b>7.5.3. Construir y lanzar la notificación .....</b>	145
<b>Resumen .....</b>	147
<b>Actividades de autoevaluación .....</b>	147
<b>8. MATERIAL DESIGN .....</b>	149
<b>Objetivos .....</b>	149
<b>Mapa conceptual .....</b>	150
<b>Glosario .....</b>	150
<b>8.1. Introducción .....</b>	151
<b>8.2. Elementos del diseño .....</b>	151
<b>8.2.1. La superficie y la luz .....</b>	151
<b>8.2.2. Elevación .....</b>	151
<b>8.2.3. Colores .....</b>	152
<b>8.2.4. Movimiento .....</b>	152
<b>8.2.5. Tipografías .....</b>	153
<b>8.2.6. Iconografía .....</b>	153

8.3. Trabajar con Material Design .....	153
8.4. Layouts y ToolBar .....	154
8.4.1. ToolBar .....	155
8.5. TabLayout .....	157
8.6. TextInputLayout .....	159
8.7. Floating Action Button (FAB) .....	161
8.8. Snackbar .....	163
8.9. CardView y RecyclerView .....	164
8.9.1. RecyclerView .....	166
8.10. Navigator Drawer .....	170
Resumen .....	172
Actividades de autoevaluación .....	172
 <b>9. ANDROID AVANZADO .....</b>	 175
Objetivos .....	175
Mapa conceptual .....	176
Glosario .....	176
9.1. Introducción .....	177
9.2. Los fragmentos .....	177
9.2.1. Construyendo un fragmento .....	179
9.2.2. Intercambio de fragmentos .....	181
9.2.3. Fragmentos especiales .....	181
9.3. Timer, hilos y procesos .....	184
9.3.1. Hilos .....	185
9.3.2. Hilos secundarios .....	185
9.3.3. Tareas asíncronas .....	190
9.3.4. Timer/TimerTask .....	191
9.3.5. Planificador de tareas (Scheduler) .....	192
9.4. Los servicios .....	193
9.4.1. Ciclo de vida de los servicios .....	193
9.4.2. Started Service .....	194
9.4.3. Bound Service .....	196
9.4.4. Intent Service .....	197
9.4.5. Servicios en primer plano .....	199
9.5. Receptores de Broadcast .....	199
9.5.1. Envío de Broadcast .....	201
9.6. La conectividad en Android .....	202
9.6.1. Gestión de las comunicaciones .....	202
9.6.2. Conectividad WIFI .....	203
9.6.3. Comunicación Bluetooth .....	208
9.6.4. Comunicación de campo cercano .....	221
Resumen .....	227
Actividades de autoevaluación .....	228
 <b>10. PERSISTENCIA .....</b>	 231
Objetivos .....	231
Mapa conceptual .....	232
Glosario .....	232
10.1. Introducción .....	233

<b>10.2. Preferencias compartidas .....</b>	233
<b>10.3. Almacenamiento en la memoria interna del dispositivo .....</b>	235
<b>10.4. Lectura de un recurso de la aplicación .....</b>	236
<b>10.5. Persistencia en la tarjeta de almacenamiento externo .....</b>	236
<b>10.6. Persistencia en bases de datos (SQLite) .....</b>	238
10.6.1. Insertar registros .....	240
10.6.2. Eliminar registros .....	240
10.6.3. Actualizar registros .....	240
10.6.4. Consultar registros .....	241
10.6.5. Movimientos del cursor .....	242
<b>10.7. Almacenamiento en la Red .....</b>	243
10.7.1. Conexión HTTP .....	243
10.7.2. Conexión HTTPS .....	246
10.7.3. Conexión por Sockets .....	248
<b>10.8. Proveedores de contenidos .....</b>	250
10.8.1. Crear un proveedor de datos (Content Provider) .....	251
10.8.2. Acceder a datos de otra aplicación (Content Resolver) .....	254
<b>Resumen .....</b>	258
<b>Actividades de autoevaluación .....</b>	258

## **11. LIBRERÍAS MULTIMEDIA EN ANDROID .....** 261

<b>Objetivos .....</b>	261
<b>Mapa conceptual .....</b>	262
<b>Glosario .....</b>	262
<b>11.1. Introducción .....</b>	263
<b>11.2. Las imágenes en Android .....</b>	263
11.2.1. La pantalla .....	263
11.2.2. Las unidades de medida en Android .....	265
11.2.3. La clase ImageView .....	265
11.2.4. La clase BitMap .....	266
<b>11.3. Los dibujos en Android .....</b>	267
11.3.1. Vistas personalizadas .....	267
11.3.2. La clase Canvas y la clase Paint .....	268
11.3.3. La clase Path .....	271
11.3.4. La clase Drawable .....	272
<b>11.4. Grabación/reproductor de audio .....</b>	272
11.4.1. La clase SoundPool .....	272
11.4.2. La clase MediaPlayer .....	274
11.4.3. La clase MediaRecorder .....	276
<b>11.5. Grabación/reproductor de vídeo .....</b>	277
11.5.1. La clase VideoView .....	278
11.5.2. La clase MediaController .....	278
11.5.3. La clase MediaPlayer .....	279
11.5.4. La clase MediaRecorder .....	280
<b>11.6. La clase MediaStore .....</b>	281
11.6.1. Obtener información de MediaStore .....	282
11.6.2. Añadir contenidos a MediaStore .....	283
<b>11.7. La pantalla táctil .....</b>	283
11.7.1. Eventos de pantalla .....	284
11.7.2. Eventos en pantallas Multi-Touch .....	285

11.7.3. Las gesturas .....	285
<b>11.8. Los sensores .....</b>	<b>285</b>
11.8.1. Acelerómetro .....	287
11.8.2. Giroscopio .....	288
11.8.3. Magnetómetro .....	288
11.8.4. Sensor de orientación .....	289
11.8.5. Sensor de proximidad .....	289
11.8.6. Sensor de iluminación .....	289
<b>11.9. La geolocalización .....</b>	<b>290</b>
11.9.1. Obtener la ubicación del dispositivo mediante la API de Android .....	290
11.9.2. Google Maps .....	293
<b>Resumen .....</b>	<b>296</b>
<b>Actividades de autoevaluación .....</b>	<b>297</b>
 <b>12. LOS JUEGOS EN ANDROID .....</b>	 301
<b>Objetivos .....</b>	<b>301</b>
<b>Mapa conceptual .....</b>	<b>302</b>
<b>Glosario .....</b>	<b>302</b>
<b>12.1. Introducción .....</b>	<b>303</b>
<b>12.2. Animación 2D y 3D .....</b>	<b>303</b>
12.2.1. Animación 2D .....	303
12.2.2. Animación 3D .....	304
<b>12.3. La animación en Android .....</b>	<b>305</b>
12.3.1. Transiciones de escenas .....	307
12.3.2. Animar drawables .....	307
12.3.3. Animación del Canvas .....	309
<b>12.4. Los sprites .....</b>	<b>310</b>
<b>12.5. Las colisiones .....</b>	<b>311</b>
<b>12.6. Propiedades de los objetos: materiales, texturas, luces y sombras .....</b>	<b>312</b>
12.6.1. Los materiales .....	312
12.6.2. Las texturas .....	312
12.6.3. Las luces .....	313
12.6.4. Las sombras .....	314
<b>12.7. La clase SurfaceView .....</b>	<b>314</b>
<b>12.8. La clase GLSurfaceView .....</b>	<b>315</b>
<b>12.9. Arquitectura de un juego .....</b>	<b>319</b>
<b>12.10. Motores de juegos .....</b>	<b>322</b>
12.10.1. Framework libGDX .....	322
12.10.2. Otros motores de juegos .....	326
<b>12.11. Entornos de desarrollo para juegos .....</b>	<b>328</b>
<b>Resumen .....</b>	<b>329</b>
<b>Actividades de autoevaluación .....</b>	<b>330</b>
 <b>BIBLIOGRAFÍA .....</b>	 333



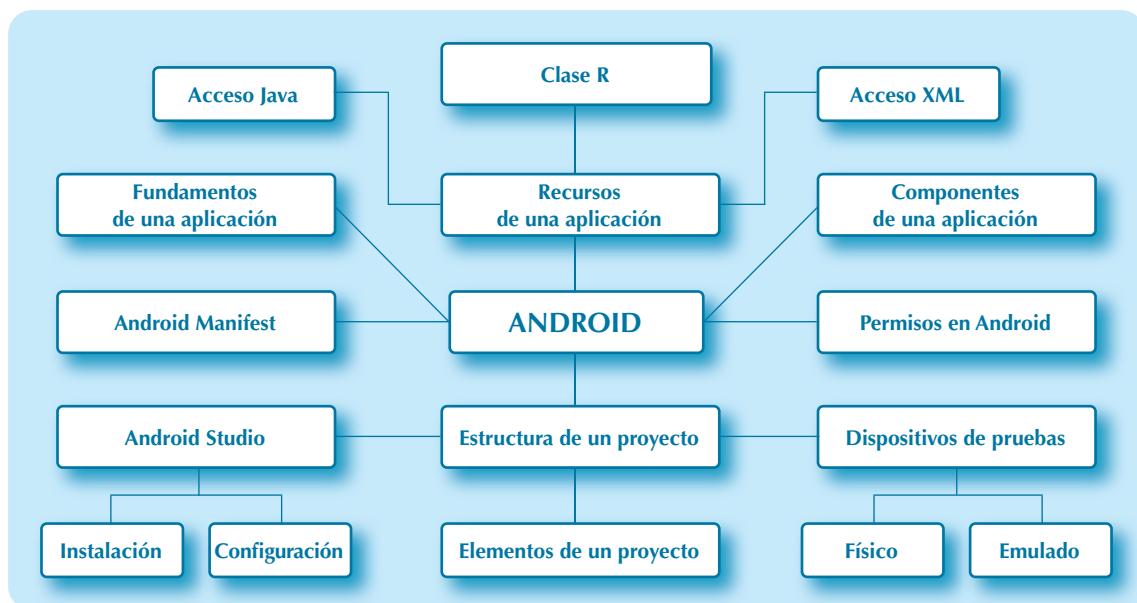
Este libro contiene material adicional para ampliar información y realizar las actividades propuestas

# Introducción a Android

## Objetivos

- ✓ Conocer los fundamentos de las aplicaciones en Android.
- ✓ Distinguir los componentes de una aplicación.
- ✓ Saber valorar la utilidad del fichero Android Manifest.
- ✓ Identificar y usar con fluidez los recursos de una aplicación.
- ✓ Comprender la funcionalidad de la clase R.
- ✓ Instalar y configurar Android Studio como entorno de desarrollo.
- ✓ Reconocer y manejar los permisos en Android.

## Mapa conceptual



## Glosario

**Aplicaciones responsivas.** Son aquellas que se adaptan a diferentes tamaños y proporciones de pantalla mediante un cambio o reorganizando del contenido en el interface, que puede suponer el escalado de imágenes, cambio de orden o posición en los menús.

**Broadcast.** En Android, la difusión amplia, difusión ancha o broadcast es la transmisión de datos que serán recibidos por todos los elementos que se encuentren activos y operativos en el dispositivo móvil.

**Bundle.** Son sistemas de almacenaje (paquetes) que se usan para pasar información entre diferentes actividades de Android.

**Gradle Scripts.** Carpeta de un proyecto de Android que contiene información necesaria para la compilación del proyecto.

**Intent.** Importante elemento de Android que es utilizado para cambiar de una pantalla o actividad a otra.

**Kit de desarrollo de software (SDK).** Se denomina así al conjunto de herramientas de desarrollo de software que permite al programador crear una aplicación informática para un sistema concreto.

**Platform.** Término que se aplica generalmente en sistemas operativos a una o varias tecnologías que se utilizan como base sobre la que desarrollar aplicaciones, establecer procesos o implementar nuevas tecnologías.

## 2.1. Introducción

Para el desarrollo nativo de aplicaciones en Android no basta con ser un buen conocedor del código (básicamente Java) y tener buena pericia en fundamentos de programación. La peculiaridad de los dispositivos móviles, la variabilidad que entre estos existen (como ya se estudió en el capítulo anterior) y, sobre todo, la multiplicidad de recursos que son capaces de aportar estos dispositivos, hacen necesario que antes de iniciar las tareas de codificación se conozcan los fundamentos y la estructura de componentes en los que se basa una aplicación en Android.

Además, el correcto manejo de algunos de estos componentes (Android Manifest, clase R, permisos...) es importante si se desea que las aplicaciones desarrolladas tengan alguna proyección de mayor alcance que el límite que pueda imponer el propio equipo informático y las herramientas de desarrollo.

En este capítulo se asentarárán fundamentos de obligado conocimiento para que el avance en los próximos no encuentre mayor dificultad que la del aprendizaje del código.

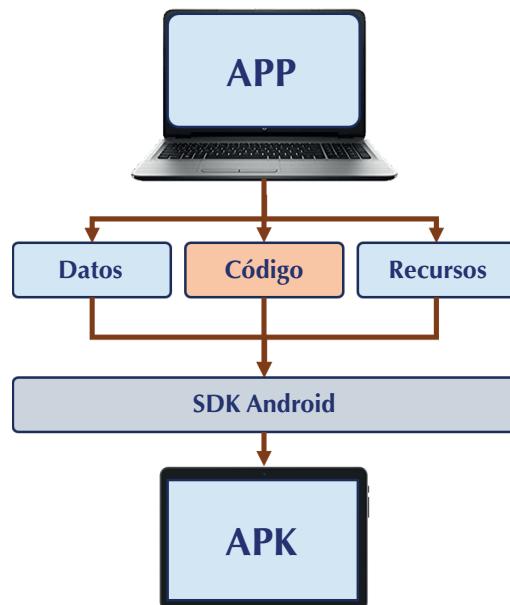
## 2.2. Fundamentos de una aplicación en Android

Como ya se ha visto, Android es un sistema operativo multiusuario basado en Linux, donde cada aplicación es considerada propiedad de un usuario distinto. El sistema asigna un identificador de usuario diferente a cada aplicación, por lo que los archivos incluidos en cada una tendrán permisos solo para ese usuario y solo él podrá tener acceso a ellos.

Esto hace que cada aplicación tenga su propio sistema de seguridad, con usuarios diferentes para cada una de ellas, permisos propios para cada usuario con máquinas virtuales propias y un proceso de Linux propio. De tal manera que Android utiliza el *principio de menor privilegio*, dando los permisos justos a cada aplicación. Todo ello lo convierte en un sistema operativo seguro.

No obstante, una aplicación puede solicitar permisos para acceder a datos (agenda), recursos (SD, Bluetooth) o funcionalidades (SMS, cámara) del dispositivo. Estos permisos asociados a la aplicación se conceden en la instalación de la misma o en el momento de usarlos (para versiones más modernas de Android), y los otorga el usuario.

Las aplicaciones Android están escritas en Java, un lenguaje de programación orientado a objetos, apoyado en determinados aspectos por XML. El SDK (kit de desarrollo de software) de Android compila el código, datos y recursos, incluyéndolo todo en un fichero APK.



**Figura 2.1**  
Paso de APP a APK.

## 2.3. Componentes de una aplicación

Los componentes de una aplicación son bloques de código mediante los cuales el sistema puede relacionarse con esta. Cada componente tiene una identidad propia y cumple un papel específico.

Hay cuatro tipos distintos de componentes; cada uno tiene un fin específico y un ciclo de vida diferente (más adelante, serán estudiados con mayor profundidad). Para entender los fundamentos de una aplicación en Android, los primeros componentes con los que es necesario familiarizarse son los siguientes:

- a) *Activity*: es el principal componente de una aplicación de Android y se encarga de gestionar gran parte de las interacciones con el usuario. Representa una pantalla independiente con una interfaz de usuario. Una aplicación puede tener varias actividades que, aunque independientes, colaboran entre sí en el funcionamiento de la aplicación.
- b) *Service*: son aplicaciones que corren de fondo para hacer operaciones de larga duración o trabajo en procesos remotos. Un servicio no tiene interfaz de usuario. Una actividad puede iniciar el servicio y permitir que se ejecute o enlazarse con él para desarrollar su cometido.
- c) *Content Provider*: se ocupa de gestionar un conjunto de datos de una aplicación para compartir. A través del proveedor de contenido, otras aplicaciones pueden consultar o incluso modificar información (si el proveedor lo permite).
- d) *Broadcast Receiver*: es una utilidad de Android que permite responder a anuncios broadcast (difusión) del sistema. Frecuentemente, un receptor de mensajes es simplemente un enlace con otros componentes realizando una cantidad mínima de trabajo.

A lo largo de este libro, se trabajará cada uno de estos componentes y se verá que para activarlos e iniciar su ciclo de vida, desde otro elemento de la aplicación e incluso desde una aplicación distinta, se hará mediante *Intent*, un elemento esencial que utiliza Android para moverse de una pantalla (Activity) a otra.

### RECUERDA

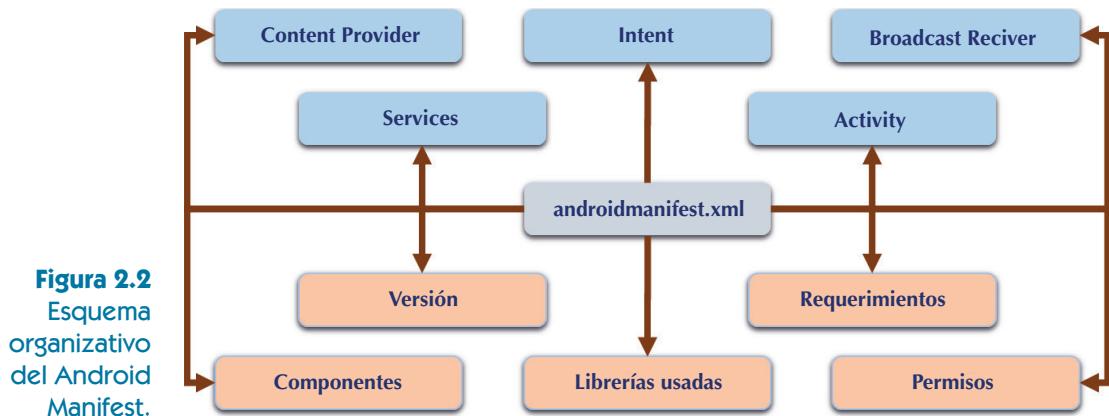
- ✓ Las aplicaciones Android están compuestas por uno o más componentes de aplicación (actividades, servicios, proveedores de contenido y emisores de notificaciones).

## 2.4. Android Manifest

Para incluir algunos de los anteriores componentes en las aplicaciones, será necesario que el sistema reconozca la existencia de ese componente, y eso lo hará leyendo el archivo Manifest (AndroidManifest.xml) de la APP. El archivo de manifiesto proporciona información esencial sobre la aplicación al sistema Android, información que el sistema debe tener para poder ejecutar el código de la APP. Todas las aplicaciones tienen este archivo, cuya gestión será fácil a

través del entorno de desarrollo; una aplicación debe tener declarados todos sus componentes en este archivo.

La principal tarea es informar al sistema acerca de los componentes de la aplicación. Otras funciones de este archivo serán registrar los permisos asociados a la aplicación, librerías que utiliza, declarar las necesidades de software/hardware, así como el nivel de API mínimo que requiere la aplicación.



Este archivo está escrito en lenguaje de marcas y tiene un aspecto similar al que se puede ver a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="es.libro.ejemplo">
    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity android:name=".Fragmentos">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Aunque existen otras (que, dependiendo del apartado que se trabaje, se irán viendo), las etiquetas que es necesario conocer inicialmente son las siguientes:

- *Manifest*. Engloba a las demás etiquetas. Define el espacio de nombres, el nombre del paquete y atributos del mismo.
- *Application*. Contiene metadatos de la aplicación (título, icono, tema), etiquetas de actividades, servicios, proveedores de contenidos y receptores de broadcast.
- *Uses-Sdk*. Indica las versiones del SDK sobre las que podrá ejecutarse, el nivel mínimo de API y el utilizado para su desarrollo.

- *Uses-Permission.* Declara los permisos que la aplicación necesita para operar. Serán presentados al usuario durante la instalación para que los acepte o deniegue. Algunos de los permisos que podemos declarar aquí son los siguientes:
  - INTERNET: conexión a Internet.
  - READ\_CONTACTS: leer en la lista de contactos.
  - WRITE\_CONTACTS: escribir en la lista de contactos.
  - SEND\_SMS: enviar SMS.
  - ACCESS\_COARSE\_LOCATION: localización mediante telefonía.
  - ACCESS\_FINE\_LOCATION: localización mediante GPS.
  - BLUETOOTH: permite el uso del Bluetooth.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-permission>
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

### Actividad propuesta 2.1



Busca en la página de Android Developer qué utilidad puede tener la inclusión del siguiente código en el Manifest:

```
<uses-permission android:name = "permiso" android:maxSdkVersion = "entero"/>
```

- *Permission.* Define un permiso que se requiere para que otras aplicaciones puedan acceder a partes restringidas de la aplicación.
- *Instrumentation.* Nos permite definir un test de ejecución para las actividades y servicios.



#### PARA SABER MÁS

La administración de permisos en Android es un aspecto muy cuidado por Google y que ha ido evolucionando con las diferentes versiones de Android, especialmente desde Android 6.0 Marshmallow. En un apartado específico de este capítulo se estudiará la gestión de permisos un poco más a fondo.

## 2.5. Recursos de una aplicación

Este término hace referencia a los datos que utiliza la aplicación, entendiendo como tal las imágenes, textos, estilos. Estos, junto con el código, configuran una aplicación.

Los recursos pueden estar predeterminados para la aplicación que se haya programado y se usarán siempre sin importar el tipo o la configuración del dispositivo sobre el que se vaya a ejecutar. O bien pueden crearse recursos alternativos, que son aquellos que se diseñan para usar con una configuración específica de dispositivo (tablet, pantallas grandes...), colaborando, en parte, con la responsividad de la aplicación.

## TEN EN CUENTA

- ✓ Las aplicaciones elaboradas se ejecutarán en dispositivos con diferentes características y configuraciones. Se deberán tener en cuenta, especialmente, los apartados de densidad de pantalla (usando recursos gráficos -ldpi, -mdpi, -hdpi, según resolución), idiomas del usuario (almacenando los recursos de string en directorios values-es, values-it), orientación y tabletas (dando soporte -sw600dp, -sw720dp), entre otros.

Además, los recursos pueden estar gestionados por ficheros XML, deben tener asignado un identificador (id) y suelen estar alojados en una subcarpeta dentro de la carpeta principal denominada *res*. Los nombres de estas subcarpetas son limitados y predefinidos por Android.

## TOMA NOTA



Nunca se deben guardar archivos de recursos directamente dentro del directorio *res*/ ya que se produciría un error en la compilación.

En el listado que se muestra a continuación se relacionan las más frecuentes y el tipo de recurso que pueden contener:

- *Carpeta res/drawable*: recursos dibujables, ficheros de bitmaps o de imágenes “escalables”. También pueden definirse aquí formas y colores de objetos dibujados.
- *Carpeta res/layout*: definidos como ficheros XML. Engloban los elementos visuales que definen el interfaz de nuestra aplicación.
- *Carpeta res/animator*: permite crear animaciones sencillas sobre uno o varios gráficos, como rotaciones, fading, movimiento y estiramiento. Cada animación se define en un fichero XML.
- *Carpeta res/mipmap*: con contenido similar a drawable, el directorio mipmap alberga elementos bitmap, aunque se suele utilizar específicamente para ubicar el icono de la aplicación.
- *Carpeta res/menu*: son ficheros XML donde se definen las diferentes opciones de los menús, submenús, barras de navegación incluidos en las aplicaciones.
- *Carpeta res/values*: es una carpeta con carácter genérico con ficheros XML que configuran diferentes aspectos de la aplicación, como es el color, dimensiones, cadenas de texto, estilos.

## RECUERDA

- ✓ Un *estilo* es el conjunto de propiedades (tamaño, relleno, color...) que definen la apariencia de un objeto visual. Un *tema* es un estilo que se aplica a toda una *Activity* o a la aplicación es su conjunto. Si un estilo se usa como un tema, se verán implicadas todas las vistas de la actividad o de la aplicación.

- *Carpeta res/xml*: almacena archivos XML utilizados por Android para dar soporte a tareas múltiples, como la que permite configurar búsquedas.
- *Carpeta res/raw*: se usa para almacenar los archivos raw de la aplicación (audio y vídeo). Este directorio tiene ciertas limitaciones en cuanto al nombre de archivos que puede tener (mayúsculas, números, caracteres especiales), por lo que, si se necesita usar el nombre real del archivo, entonces se deben colocar los archivos raw en el directorio asset de Android.



## Investiga

Accede a la página sobre tópicos (dedicada a recursos) en Android Developer a través de este QR.

Investiga en ella cómo debe hacerse la provisión de recursos y la agrupación de los mismos. Observa la importancia del uso de recursos alternativos que permiten admitir configuraciones de dispositivos con características propias y específicas.



### 2.5.1. Cómo acceder a los recursos

Como se ha visto anteriormente, todos los recursos de tu aplicación tienen asignado un identificador (generalmente, Android lo hace de manera automática), por lo que quedan el tipo de recurso y su id registrados en el archivo de la clase R de la aplicación. Esto puede ser utilizado para acceder al recurso desde el código Java o XML de la manera que a continuación se expone.

Para acceder a los recursos en XML se debe escribir el nombre del recurso y la dirección del archivo donde este se encuentra. En el ejemplo siguiente, se ha definido en ficheros XML el color, el estilo y el contenido de la cadena de texto. Esto hace muy cómodo el realizar variaciones de las propiedades de este objeto (o aplicárselas a otros) sin tener que retocar el código Java de la aplicación.

```
<TextView
    android:id="@+id/texto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/colorPrimario"
    android:text="@string/titulo"
    style="@style/EstiloTitulo"/>
```

En Java, para acceder a un recurso (getResources), se ha de localizar este a través de la clase R, al igual que en XML, indicando directorio (o fichero XML) y nombre del recurso.

```
getResources().getColor(R.color.colorPrimario);
getResources().getDrawable(R.drawable.dibujo);
miImagenView.setImageResource(R.drawable.imagen);
```

## Investiga



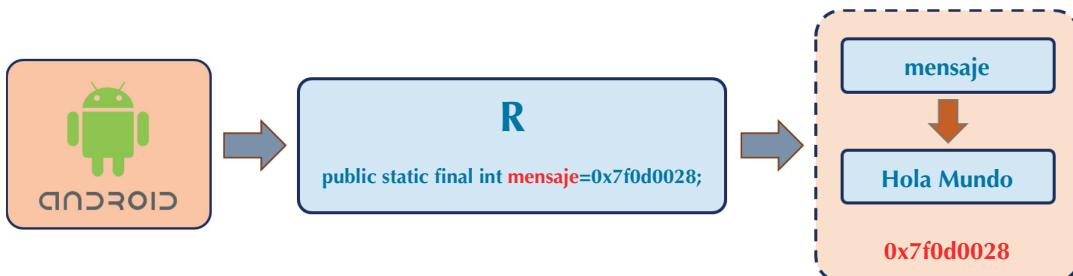
Como se irá viendo, Android nos facilita multitud de recursos, que muchas veces costará trabajo localizar o identificar. Para acceder a ellos, se debe hacer de manera similar a la que ya se ha visto, pero anteponiendo al recurso la palabra *android*.

Aquí se muestra un ejemplo de cómo acceder al color negro que nos proporciona Android, tanto desde XML como desde Java.

```
android:textColor="@android:color/black"
getResources().getColor(android.R.color.black);
```

## 2.6. La clase R

Se trata de una clase que contiene variables estáticas en las que se identifica cada tipo de recurso. Android lee el fichero XML, carga todas las estructuras solicitadas en memoria y mantiene el fichero R como referencia directa a los recursos cargados. Por tanto, cada atributo tiene una dirección de memoria asociada referenciada a un recurso en específico. En la figura 2.3, *mensaje* contiene la cadena “Hola Mundo”, que está almacenado en la dirección de memoria 0x7f0d0028.



**Figura 2.3**  
Organización de la clase R.

Aunque Android Studio ha mejorado mucho la estabilidad de la clase R, en más de una ocasión se presentan errores en esta, que impedirán, mientras no se solucionen, el seguir avanzando en el desarrollo de la aplicación. Y aunque no existe una pauta que seguir para reparar los errores en la clase R, es importante asegurarse que no existan enlaces rotos, es decir, que no exista ningún error en los archivos XML, y si todo aparece estar correcto, se pueden utilizar herramientas del entorno de desarrollo, como Build/Clean Project o Build/Rebuild Project.



## TOMA NOTA

Nunca debe modificarse el archivo R.java manualmente. Este se genera a través de la herramienta AAPT (Android Asset Packaging Tool) cuando se compila el proyecto, por tanto, todos los cambios que se hagan manualmente se anularán tras cada compilación.

## 2.7. Estructura de un proyecto

Hasta el momento, se han visto los fundamentos y elementos en los que se basa una aplicación desarrollada en Android; ahora debemos de organizarlos y estructurarlos dentro del proyecto. Afortunadamente, toda esa labor recaerá sobre la herramienta de desarrollo utilizada (en nuestro caso, Android Studio), que a través de una serie de preguntas o decisiones iniciales nos dejará un entorno preparado para que iniciemos las labores de codificación.

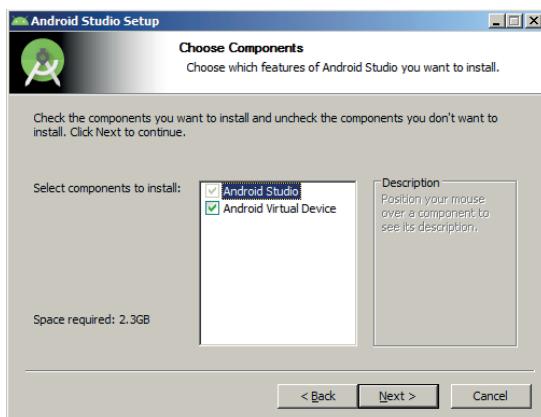
Para poder entender de manera correcta cómo se estructura un proyecto en Android, y para poder iniciarnos en la práctica de código, se ha de tener instalado el entorno de desarrollo con el que se vaya a trabajar. Actualmente, el proceso es extraordinariamente simple, en especial si se utiliza Android Studio (lo que se hará en este libro), ya que Google ha simplificado todas las tareas de instalación mediante un Bundle (paquete) que se ocupa de gestionar todo el proceso, y tan solo recurre al usuario para algunas cuestiones básicas de ubicación y configuración.

Por otro lado, el proceso tiene algunas variaciones, dependiendo del sistema operativo utilizado y, en su caso, de la versión del mismo; aquí veremos algunas nociones básicas de la instalación, comunes a cualquiera de las variantes utilizadas, dejando para otros módulos (Entornos de Desarrollo) la profundización en cada uno de los aspectos de esta herramienta.

### 2.7.1. Instalación de Android Studio

Este entorno de desarrollo integrado presentado en 2013 para la plataforma Android reemplazó a Eclipse como IDE oficial para el desarrollo de aplicaciones para Android. Se basa en IntelliJ IDEA de JetBrains, es de licencia gratuita y puede obtenerse para cualquier sistema operativo. La última versión estable es la 3.4.2, de abril de 2019.

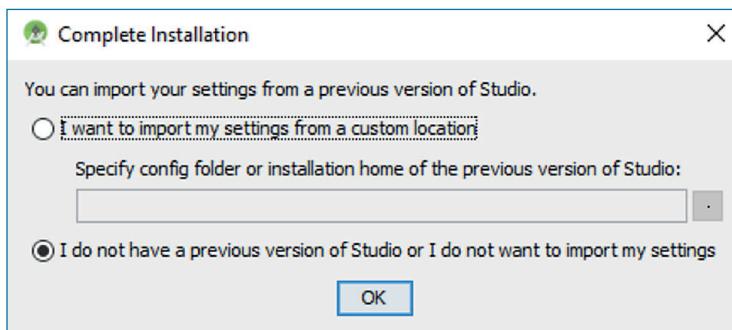
Aunque ha mejorado mucho, es un IDE con gran necesidad de requisitos del sistema. Para la versión 3 se necesita un mínimo de 3GB de RAM (8GB recomendado), además de 1 GB adicional si se utiliza el emulador. En cuanto al espacio en disco, se debe contar al menos con 4GB (IDE, SDK, emulador, caches) y, lógicamente, tener instalado el Java Development Kit (JDK).



**Figura 2.4**  
Elección de componentes.

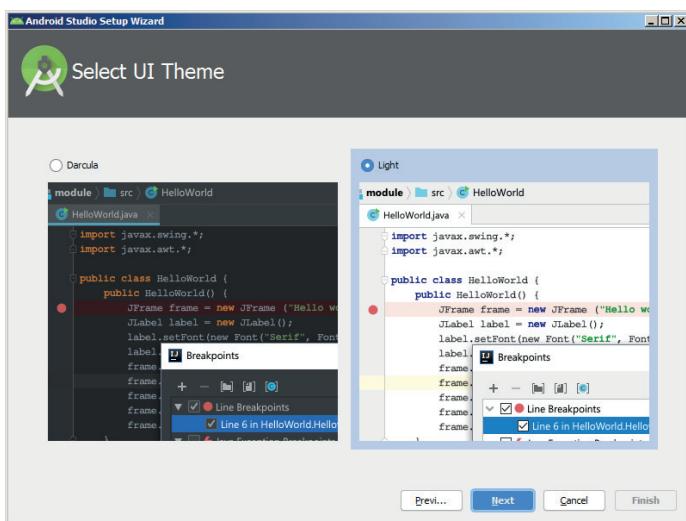
Su instalación es sumamente sencilla; la descarga se puede hacer de la web oficial de Android (<http://bit.ly/2TLAYro>). El proceso de instalación puede tener pequeñas modificaciones, según la versión de Android Studio que se vaya a instalar: en la versión 3.5 de agosto de 2019, una vez lanzada la instalación tan solo hay que seguir una serie de pantallas que, en su mayoría, solo habrá que pulsar siguiente (next), como son la de aceptar términos, bienvenida, componentes que se van a instalar. En esta, además del entorno, puede marcarse la instalación del emulador AVD (AndroidVirtual Device), que aunque no es obligatorio, sí es recomendable, pese a que se tenga pensado realizar el desarrollo sobre dispositivos reales.

También preguntará dónde ubicar el entorno. Se debe poner especial atención en este apartado y procurar ubicarlo en directorios y unidades fácilmente accesibles (nunca utilizar caracteres “extraños”) y, a ser posible, en unidades especialmente rápidas. Una vez descargados los paquetes y terminada la instalación, se pasa al proceso de configuración, donde se nos consultará si buscar e importar configuraciones anteriores del entorno.



**Figura 2.5**  
Importación  
configuración.

También, tras la ventana de bienvenida, nos consultará sobre el tipo de dispositivo para el que se programará y el estilo (tema) con el que queremos que se muestre la interfaz de usuario.



**Figura 2.6**  
Selección de tema del IU.

Por último, hará una revisión y actualización de algunos componentes necesarios para las tareas de desarrollo, como son la SDK (platform), el Intel @ HAXM, o los AVD, entre otros.



### SABÍAS QUE...

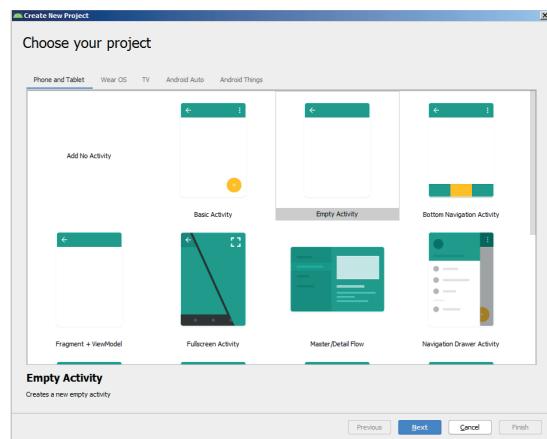
El Intel HAXM (Hardware Accelerated Execution Manager) es un sistema de virtualización que nos ayudará a mejorar el rendimiento del emulador de Android cuando se utilice este tipo de procesadores.

## 2.7.2. Iniciando el entorno de desarrollo

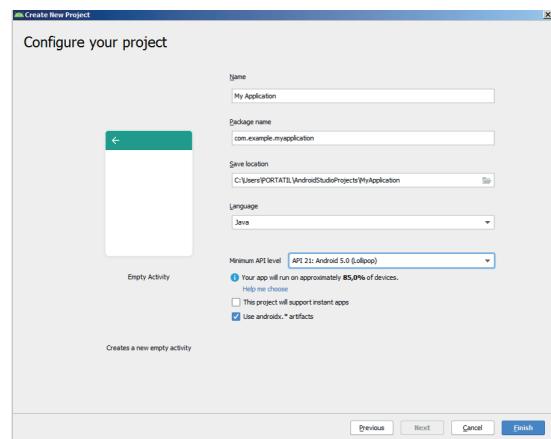
Terminada la instalación, el proceso de iniciar la primera aplicación consistirá en una serie de pasos que definirán algunas características generales del mismo.

En la primera pantalla preguntará el tipo de vista que se aplicará a la actividad principal. Aunque se puede elegir entre numerosas variantes, resulta aconsejable escoger en los primeros pasos del aprendizaje la denominada *Empty Activity*. En la siguiente pantalla habrá que dar un nombre a la aplicación (debe empezar por una letra en mayúscula), el dominio (Package) de la misma (importante a la hora de empaquetarla) y dónde se ubicará.

En el dominio es necesario separar las palabras por punto y normalmente se comienza colocando el prefijo “com” o “es”. Luego se pone cualquier nombre (el de la empresa o proyecto) y, por último, el nombre de la aplicación (es.libro.programa).

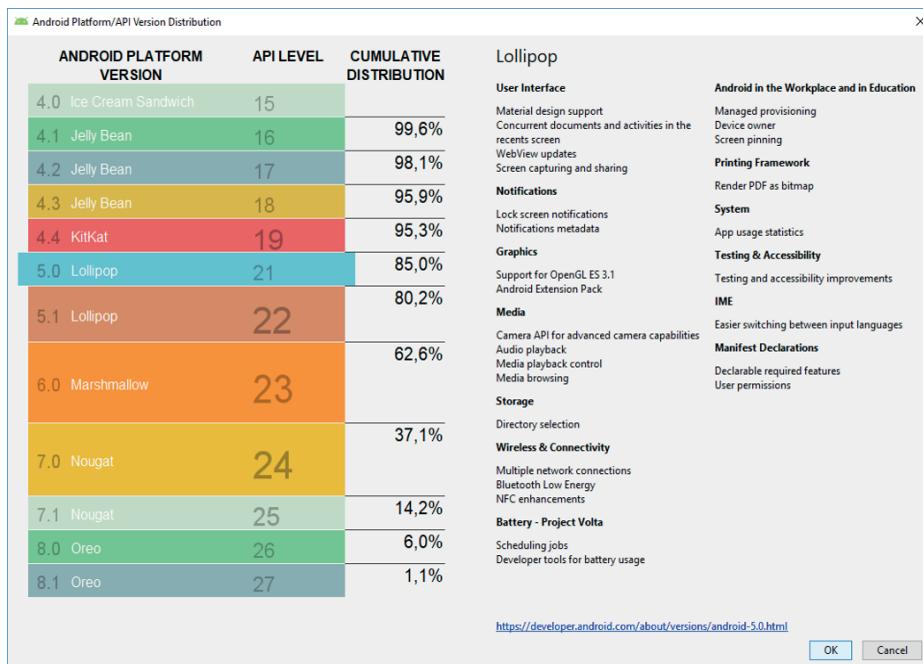


**Figura 2.7**  
Selección de proyecto.



**Figura 2.8**  
Configuración de proyecto.

Además, se ha de elegir lenguaje (Java o Kotlin). La mínima API que aceptará la aplicación (recomendable Lollipop o superior), pulsando el enlace “Help me choose” mostrará un gráfico con los dispositivos que aceptarán la aplicación en caso de subirla al Play Store.



**Figura 2.9**  
Porcentaje de distribución según versiones de Android.

El activar Android Instant App Support permite lanzar las aplicaciones, en fase de desarrollo o prueba, disminuyendo los tiempos de espera. También se puede marcar el uso del complemento de Android X, que permite utilizar la biblioteca de esta versión.

#### RECUERDA

- ✓ Android está fundamentado en Java y, por tanto, se ha de tener instalando el JDK (kit de desarrollo de Java). Además, Android debe saber localizarlo mediante una correcta configuración de las variables del entorno. Lo normal es que estas estén automáticamente configuradas, pero, en caso contrario (Android Studio no reconocería Java), se deberán configurar a mano.

En Linux, etc/enviroment e incorporar:

```
JAVA_HOME="/usr/java/jdk1.8.0_25"
```

En Windows, dentro de propiedades del sistema:



**Figura 2.10**  
Edición de la variable del sistema.