EDITORIAL MACRO®

DESARROLLO DE APLICACIONES MÓVILES CON ANDROID

- Tecnologías móviles - Android Studio - Desarrollo inicial de aplicaciones
- Controles de una aplicación Android - Mejores aplicaciones para Android



Manuel Torres Remon

DESARROLLO DE APLICACIONES MÓVILES CON

ANDROID

EDITORIAL MACRO®

Perú - México - Colombia - Chile - Ecuador - España - Bolivia - Uruguay - Guatemala - Costa Rica

EDITORIAL MACRO®

Desarrollo de aplicaciones móviles con Android
Autor: Manuel Torres Remon

© Derechos de autor registrados:
Empresa Editora Macro EIRL

© Derechos de edición, arte gráfico y diagramación reservados:
Empresa Editora Macro EIRL

Coordinación de edición:
Magaly Ramon Quiroz

Diseño de portada:
Alessandra Bonilla Zapata

Corrección de estilo:
José Vásquez Espíritu

Diagramación:
Fernando Cavassa

Edición a cargo de:
© Empresa Editora Macro EIRL
Av. Paseo de la República N.° 5613, Miraflores, Lima, Perú

Teléfono: (511) 748 0560
E-mail: proyectoeditorial@editorialmacro.com
Página web: www.editorialmacro.com

Primera edición: octubre 2016
Tiraje: 1700 ejemplares

Impresión
Talleres gráficos de la Empresa Editora Macro EIRL
Jr. San Agustín N.° 612–624, Surquillo, Lima, Perú
Octubre 2016

ISBN N.° 978-612-304-517-3
Hecho el depósito legal en la Biblioteca Nacional del Perú N.° 2016-13154

Prohibida la reproducción parcial o total, por cualquier medio o método, de este libro sin previa autorización de la Empresa Editora Macro EIRL.

Manuel Torres Remon

Licenciado en Informática. Se ha dedicado a la consultoría y docencia de cursos de tecnología desde hace 15 años. A su vez, ha brindado diversos talleres de capacitación en las instituciones más importantes de Lima.

Realizó sus estudios en el Instituto de Educación Superior Manuel Arévalo Cáceres, en la Universidad Alas Peruanas y en la Universidad Peruana del Norte. En todas estas instituciones, obtuvo una esmerada formación profesional que ha podido demostrar ampliamente a lo largo de su trayectoria docente.

Actualmente, se desempeña como profesor de tecnología en los institutos Manuel Arévalo Cáceres, Cibertec y Unimaster, en donde imparte los cursos de Programación, Base de Datos y Análisis de Sistemas.

Dedicatoria

Este libro está dedicado con mucho cariño y aprecio a mis pequeñas Ángela Victoria y Fernanda Ximena, que son y seguirán siendo mi fuente de inspiración, y mucho más a mi esposa, Luz, por comprenderme en todo lo que me propongo.

Agradecimiento

Todo libro escrito es un sacrificio que tiene que ser valorado y esto solo se dará si usted aprecia este material tanto como lo aprecio yo. Escuché, alguna vez, la siguiente frase: «La lectura de un libro enriquece de conocimientos y empobrece la ignorancia», pienso que es cierto. Se sabe que hoy por hoy un libro impreso es difícil de leer, pues son pocas las personas que cargan un libro en la mano y muchos de ellos han sido reemplazados por los dispositivos móviles. Por eso, mi primer agradecimiento es para usted, amigo lector, por adquirir este material impreso, que me ha hecho sacrificar tiempo y esfuerzo.

Asimismo, deseo agradecer a la gran familia Macro por confiar nuevamente en mi persona para el desarrollo del libro *Desarrollo de aplicaciones móviles con Android*.

Índice

Introducción 17

Capítulo 1: Introducción a las tecnologías móviles

1.1 Introducción 21
1.2 Arquitectura de las tecnologías móviles 22
1.3 Tipos de aplicaciones 22
1.3.1 Aplicaciones o apps 22
1.3.2 Sistemas 23
1.3.3 Administración de la información 23
1.4 Sistemas operativos móviles 23
1.4.1 iOS (iPhone OS) 23
1.4.2 Android 24
1.4.3 Windows Phone 24
1.4.4 Tizen 24
1.4.5 WebOS 24
1.5 Componentes de un sistema operativo móvil 25
1.5.1 Las aplicaciones 25
1.5.2 Estructuras de una aplicación 28
1.5.3 Bibliotecas 29
1.5.4 El núcleo Kernel 30
1.6 Android 31
1.6.1 Historia 31
1.6.2 Características 32
1.6.3 Versiones 32
1.7 Proceso de diseño y desarrollo de una aplicación Android 34
1.7.1 Análisis 34
1.7.2 Formalización del proyecto 34
1.7.3 Maquetado 34
1.7.4 Codificación 35
1.7.5 Publicación 35

Capítulo 2: Programación orientada a objetos con Java

2.1 Introducción 39
2.2 Conceptos en programación orientada a objetos 40
2.2.1 Paquetes 40
2.2.2 Alcance de los elementos contenidos en un paquete 41
2.3 Clases en Java 42
2.3.1 Partes de una clase 42
2.3.2 Atributos de una clase 44
2.3.3 Métodos en las clases 46
2.4 Objetos en Java 48
2.5 Métodos get y set 50
2.6 Implementación de métodos get y set con Android Studio 53
2.7 Método constructor 55
2.8 Referencia this 56

Capítulo 3: Android Studio y desarrollo inicial de aplicaciones

3.1 Android Studio 2.0 61
3.2 Características del Android Studio 61
3.3 Máquina virtual 61
3.3.1 Máquina virtual ART (Android Runtime) 63
3.3.2 Tecnología AOT (AHead Of Time) 64
3.4 Requerimientos mínimos para la instalación de Android Studio 64
3.5 Proceso de instalación de Android Studio 65
3.6 Descripción de las pantallas iniciales de Android Studio 68
3.7 Android SDK 71
3.8 AVD Manager 73
3.9 Creando un nuevo proyecto 75
3.10 Descripción de la pantalla principal Android Studio 79
3.10.1 Paleta de proyectos (Project) 79
3.10.2 AndroidManifest 84
3.10.3 Características del AndroidManifest 84
3.10.4 Elementos que componen un AndroidManifest 84
3.10.5 Cambiando características de la aplicación usando AndroidManifest 87
3.11 Paleta de herramientas de actividad 89
3.12 Controles del emulador 91
3.13 Árbol de componentes 93
3.14 Propiedades de los controles 93
3.15 Ejecución de la aplicación en el emulador 94
3.16 Descripción del emulador estándar 95
3.17 Unidades de medida 96
3.18 Recursos Android 96
3.18.1 Drawable 97
3.18.2 Layout 98
3.18.3 Mipmap 106
3.18.4 Values 111

Capítulo 4: Controles de una aplicación Android

4.1 Controles layout 119
4.1.1 LinearLayout 119
4.1.2 FrameLayout 129
4.1.3 TableLayout 137
4.2 Controles de texto 144
4.2.1 TextView 144
4.2.2 EditText 148
4.3 Controles de acción 153
4.3.1 Button 153
4.3.2 ToggleButton 158
4.3.3 ImageButton 162
4.4 Intent 168

Capítulo 5: Juegos

5.1 Aspectos generales 183
5.2 Aplicación de Canvas: Líneas, rectángulo y círculos 184
5.2.1 Dibujar una línea vertical 187
5.2.2 Dibujar rectángulos 188
5.2.3 Dibujar un círculo 189
5.3 Aplicación de Canvas: Texto 190
5.4 Aplicación de fuentes externas: Texto 191
5.5 Datos 193
5.5.1 Persistencia y preferencia 193
5.5.2 Aplicacion sobre el uso de Preferencias 193
5.6 DisplayMetrics 198

Capítulo 6: Introducción a la inteligencia artificial: Agentes Android y dispositivos móviles

6.1 Introducción 221
6.2 Descripción de la aplicación 222
6.3 JADE programación para Android 222
6.4 Conceptos básicos 228
6.4.1 Web semántica 228
6.4.2 Implementar la web semántica 231
6.4.3 Componentes de programación 234
6.4.4 Herramientas 235
6.4.5 Impactos de la web semántica en la programación 236
6.4.6 Razonador 236
6.4.7 Lógica descriptiva 238
6.4.8 Lenguajes de la web semántica 239
6.4.9 Pellet 241
6.5 Jena 242
6.5.1 Características 242
6.5.2 Procesamiento de la API RDF y OWL 243
6.5.3 Mecanismo de inferencia en Jena 244

6.5.4 Consultas SPARQL (estándar de la W3C) en Jena 244
6.5.5 Procesamiento de importaciones ontológicas 256
6.5.6 Insertar un razonador al modelo 258
6.5.7 Resumen de razonadores 260
6.5.8 Consultar un modelo 260
6.5.9 Jena persistencia 261
6.5.10 Jena SDB 261
6.5.11 Instalación en Windows 261
6.5.12 Construcción del modelo persistente en la base datos Mysql y carga de los datos por la consola Cygwin 264
6.5.13 Aplicación Java utilizando la API Jena/SDB 265
6.5.14 Jena/SDB diseños de base de datos 270
6.6 Anexos 271
6.6.1 RDF (Resource Description Framework) 271
6.6.2 RDFS (rdfschema) 273
6.6.3 D2r Server 273
6.6.4 D2RQ con Jena 276

Capítulo 7: HTML 5

7.1 Introducción 279
7.2 Requerimiento de Software 279
7.2.1 PhoneGap 279
7.3 Canvas 282
7.3.1 Conceptos básicos sobre Canvas 282
7.3.2 drawimage() 289
7.4 Manejo de eventos en HTML5 290
7.4.1 Método addEventListener() 290
7.5 Ejemplo HTML5 básico 291

Capítulo 8: Google Maps Android API V3

8.1 Introducción 315
8.2 Coordenadas 321
8.3 Ejemplo Mapa 1 322
8.4 Ejemplo Mapa 2 328
8.5 Ejemplo Mapa 3 333
8.6 Marcadores 336
8.7 Ejemplo Mapa 4 337
8.8 Ejemplo Mapa 5 341

Capítulo 9: Códigos ocultos en su dispositivo móvil

9.1 Código 349
9.1.1 Lista de los diferentes códigos 349
9.1.2 Códigos para obtener información del firmware 350
9.1.3 Códigos para ejecutar diferentes tests de fábrica 350

Capítulo 10: Las mejores aplicaciones para Android

10.1 WhatsApp 353
10.2 Facebook 353
10.3 Spotbros 353
10.3.1 Características 353
10.4 Twitter 354
10.5 Otras aplicaciones 354

Referencias bibliográficas 359

Introducción

Hace algunos años el desarrollo de aplicaciones para móviles resultaba un tanto complejo, ya que se debía prever muchas cuestiones, como por ejemplo, que la aplicación pueda ejecutarse en uno u otro dispositivo. Para hoy ese problema ya no existe gracias a la aparición del sistema operativo Android, el cual no hace distinciones sobre el tipo de dispositivo que le sirva de soporte.

Se está viviendo una rápida expansión de los teléfonos inteligentes y esto ha creado una amplitud en el desarrollo de aplicaciones para móviles. Es así que este material tiene como objetivo iniciarlo a usted en el mundo del desarrollo de aplicaciones móviles. Para ello, se ha creído conveniente dividir este libro en diez capítulos para un mejor entendimiento, los cuales se detallan a continuación:

En el capítulo uno se hace referencia a los principales conceptos que se deben conocer sobre los dispositivos móviles. Se desarrollan, además, las principales características que presentan los sistemas operativos móviles y la identificación de los pasos que se deben seguir para implementar una aplicación móvil.

En el capítulo dos se realiza una breve introducción a la programación Java, donde se reconocen algunos conceptos básicos de la programación orientada a objetos. También sobre cómo se debe implementar una clase y cuáles son sus componentes, cómo crear un objeto a partir de una clase y cómo aplicar un encapsulamiento de la información.

En el capítulo tres se entra un poco más en el fondo con el tema Android, en el cual se estudian algunos conceptos básicos de este sistema operativo, así como la identificación de los componentes de su sistema y la creación de aplicaciones usando la plataforma Android Studio.

En el capítulo cuatro se realiza un estudio sobre los principales controles que permiten crear aplicaciones móviles visuales haciendo uso de la plataforma Android Studio. En el capítulo cinco se conocerán algunos aspectos básicos sobre la programación de juegos, así como del uso del Canvas para gráficos y textos. En el capítulo seis se realiza una introducción a la inteligencia artificial haciendo uso de agentes Android.

En el capítulo siete se hace un estudio del lenguaje HTML5 como plataforma de trabajo para la implementación de aplicaciones móviles. En el capítulo ocho se detallan algunos conceptos sobre los Google Maps, así como la especificación de sus coordenadas y marcadores.

En el capítulo nueve se analizan los códigos de nuestra aplicación móvil, la forma de obtener información del firmware y el test de fábrica. Finalmente, en el capítulo diez se da un vistazo a las mejores aplicaciones Android especificando sus características generales y más relevantes.



CAPÍTULO

1

Introducción a las tecnologías móviles

1.1 Introducción



Fuente: <http://www.omicrono.com/2013/05/cuantos-smartphones-hay-en-el-mundo/>

A principios de los años 90, el teléfono móvil era considerado como un aparato electrónico difícil de obtener por el costo inicial que tenía; pocas personas podían adquirirlo y era un tema de novedad tecnológica más que un medio de vida como es considerado hoy en día, ya que realiza muchas actividades en un solo lugar generando una dependencia total entre el usuario y el móvil.

Asimismo, el mundo de la telefonía móvil es uno de los más amplios y competitivos ya que existen miles de empresas fabricantes de dispositivos móviles que abastecen las preferencias y principalmente los gustos exigentes de los usuarios en el mundo.



Al respecto, se debe mencionar que hay muy pocas cosas en el mundo que han cambiado la vida de los personas como lo han hecho los dispositivos móviles, existe una manía por los móviles modernos y con muchas funcionalidades que permiten tener todo controlado desde un solo aparato. Un estudio reciente realizado por un organismo especializado de las Naciones Unidas para las Tecnologías de la Información y Comunicación menciona que en el año 2000 había en el mundo alrededor de 700 millones de líneas móviles, asimismo, menciona que hoy nos encontramos en un poco más de 7000 millones superando el total de habitantes en el planeta.

Desde la primera aparición de los dispositivos móviles en el año 1983 con el Motorola DynaTAC, el cual pesaba alrededor de 800 gramos, han pasado más de 33 años y el dispositivo móvil ha ido perdiendo peso, la duración de la batería se ha alargado, la visualización ha mejorado considerablemente y con la integración al internet deja un mundo abierto de aplicaciones nuevas que antes ni se imaginaba.

Algunos expertos manifiestan que la tecnología móvil ha llegado a un punto importante en la historia del hombre que será difícil imaginarse cómo será el teléfono inteligente del futuro. A continuación algunas manifestaciones con respecto al tema:

> *"No estamos seguros ni de si tendremos teléfonos como estos, quizás la comunicación estará en algún lugar de mi bolsillo, quizás en alguna pantalla en mis gafas, quizás la música en un dispositivo en mi oído, se darán muchas posibilidades".*
>
> Shao Yang, de la multinacional china Huawei

"Piensen en los autos. No han evolucionado tanto en 20 años, así que los teléfonos podrían verse como se ven hoy, o en un mundo alternativo, alguien solucionaría el problema de la durabilidad de las baterías, así que podremos tener aparatos más pequeños, pantallas flexibles, pero también podríamos interactuar con teléfonos de una forma diferente".

Ben Wood de la firma de análisis tecnológico CCS Insight

Sin embargo, al final se puede considerar que el dispositivo móvil va a evolucionar mucho más y que la comunicación se va a diversificar adoptando el móvil a nuestro sistema de vida como ya se viene haciendo.

1.2 Arquitectura de las tecnologías móviles

Si se toma en cuenta que los dispositivos móviles realizan muchas de las tareas que comúnmente se realiza en una laptop o computadora personal, se puede decir que los dispositivos móviles presentan características particulares que incluyen:

- Una duración finita de la batería, esto principalmente al uso constante de las notificaciones.
- Disminución constante del tamaño de la pantalla y mejora en la calidad de visualización.
- Conectividad y sincronización con otros aparatos para un control centralizado de las cosas.
- Presenta una gama de sensores que permiten recolectar información para determinados procesos.
- Administración sencilla del dispositivo que facilita la instalación y actualización de aplicaciones desde un usuario no necesariamente experto.

Si se habla de arquitectura no se puede dejar de mencionar que los usuarios adquieren dispositivos móviles para realizar descargas de aplicaciones según sus necesidades, de allí que se genera una dependencia con respecto al tiempo de descarga de dicha aplicación que podría mejorar con el tiempo pero esto se dará solo si la arquitectura del móvil evoluciona. Podría afectarse cuando los sistemas pasen completamente a los dispositivos móviles ya que estas necesitan mucho más recursos como el espacio de memoria o la velocidad en el proceso.

Existen métodos de desarrollo de arquitectura, tales como QAW, ADD o ATAM. Tal es el caso que ATAM es un método de evaluación de arquitecturas móviles desarrollado por la Software Enginnering Institute, su propósito es evaluar las consecuencias de las decisiones arquitectónicas con relación a la calidad que se ofrecen en los dispositivos móviles.

1.3 Tipos de aplicaciones

La tecnología móvil presenta tres variados tipos de aplicaciones que hoy se usan y que son cotidianos para el usuario. Estos son:

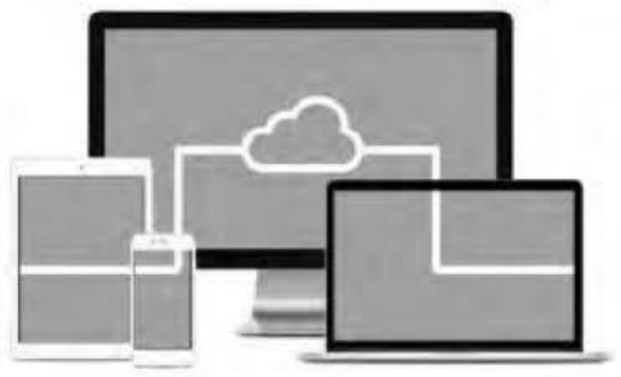
1.3.1 Aplicaciones o apps

El usuario fácilmente puede instalar estas aplicaciones en su dispositivo móvil por medio de una tienda virtual; las aplicaciones se pueden instalar directamente en el dispositivo o de forma aislada en una memoria externa, pero aun así usa todos los recursos que puede presentar el dispositivo móvil.



1.3.2 Sistemas

En este tipo de aplicación, los dispositivos móviles pasan a ser un punto de extensión de un sistema y la mayor ventaja que presenta es que no necesita una computadora de escritorio o portátil para hacer uso del mismo. Se puede mencionar, por ejemplo, que gracias al dispositivo móvil el sistema de ventas de una empresa no espera a sus clientes sino que ahora podrá interactuar directamente con él y realizar cualquier tipo de transacción.



1.3.3 Administración de la información

Cumple la misma funcionalidad que el tipo de sistemas, con la diferencia que tiene un objetivo específico y siempre realizará la misma tarea, por ejemplo, la información que emiten los GPS o la especificación de los datos de un producto gracias a un código QR.



1.4 Sistemas operativos móviles

El sistema operativo representa la lógica que tiene el dispositivo móvil tal como se presenta en una computadora personal, es decir, es un programa que permite gestionar de manera adecuada los recursos de hardware del dispositivo móvil, asimismo provee capacidad de servicios a los programas de aplicación, para que estos puedan ejecutarse. Entre las principales características que puede presentar un sistema operativo móvil se encuentran:

- Gestionar y administrar un entorno de trabajo adecuado para el usuario; tal como lo realizan las plataformas Windows o Linux.
- Administrar de manera adecuada los componentes como por ejemplo el reconocimiento de la memoria, estado de la cámara, etc.
- Administrar de forma adecuada la distribución y asignación de recursos frente a solicitudes realizadas por las aplicaciones.
- Administrar la correcta ejecución de una aplicación en cualquier tipo de dispositivo.
- Mantener una línea de actualización de las aplicaciones mediante notificaciones al usuario.
- Administrar la sincronización de los dispositivos móviles con otros dispositivos.

Entonces, se puede decir que de acuerdo a sus características un sistema operativo móvil se orienta más a la conectividad inalámbrica del dispositivo. Finalmente, es de vital importancia que un sistema operativo disponga de una gran variedad de aplicaciones en donde el usuario pueda administrarlo sin tener conocimientos avanzados de programación ya que siempre se le dará la posibilidad de personalizar su dispositivo móvil como crea conveniente.

Entre los principales sistemas operativos para móviles están:

1.4.1 iOS (iPhone OS)

iOS es un sistema operativo que cumple con todas las características que poseen los sistemas operativos móviles; lo que diferencia a este sistema frente a otros competidores es que se considera como un sistema cerrado; ya que la compañía

Apple no permite la modificación de las características originales que presenta el sistema, el usuario solo podrá realizar modificaciones necesarias para el uso del entorno del sistema, pero no puede modificar su estructura interna.

Parece una desventaja considerar que el sistema es cerrado pero en realidad el sistema garantiza una estabilidad óptima del sistema frente a los usuarios; es decir, las modificaciones o actualizaciones que pueda realizar el usuario tienen un límite; el cual viene implementado por el fabricante.

Por otro lado, el sistema no permite la libre distribución ya que no permite la licencia múltiple y solo dicho sistema solo puede instalarse en un dispositivo móvil iPhone.

1.4.2 Android

Es considerado el sistema operativo móvil más popular con un gran 85% del mercado móvil frente a sus competidores. Una de sus características es que es de código abierto y está disponible para cualquier tipo de dispositivo como teléfonos, tabletas, etc.



Se le llama código abierto a la posibilidad de adaptar el sistema a las necesidades del usuario, esta característica fue heredada de Linux de donde basa toda su estructura.

Entre las principales características que se puede mencionar es que Android se adapta a las diferentes resoluciones que presentan las pantallas de los dispositivos móviles. Asimismo tiene un soporte para el renovado HTML5; para la realización de videollamadas incorpora la aplicación Google Talk. Tiene la disponibilidad para adaptarse a diferentes teclados y la facilidad de personalizar el escritorio de su dispositivo.

1.4.3 Windows Phone

La empresa Microsoft reacciona tarde ante los más grandes sistemas operativos como Android y iOS, por ello, en los últimos años impulsó financieramente su sistema operativo móvil para ser la competencia ante los grandes del mercado haciendo alianzas estratégicas con Nokia y otras empresas; pero hoy ese esfuerzo se califica como no productivo y se dará de baja a dicho sistema para concentrarse en el sistema Windows 10 Mobile el cual estará disponible para todos tipo de dispositivo móvil como teléfonos, tabletas, computadoras, etc.

1.4.4 Tizen

Inicialmente fue distribuido como un sistema de código abierto basando su tecnología en Linux. Se ha desarrollado a partir de la plataforma Linux de Samsung llamándose Tizen 2, el cual es considerado un código cerrado como lo hace iOS.

La estrategia de la compañía Samsung es incorporar en sus dispositivos dicho sistema, el dispositivo más conocido es el reloj Smartwatch Gear S, el cual presenta una sincronización completa con su dispositivo móvil.



1.4.5 WebOS

Así como la compañía Samsung cuenta con Tizen, WebOs pertenece a LG, el cual usa dicho sistema operativo en sus televisores inteligentes llamados Smart TV.

1.5 Componentes de un sistema operativo móvil

El sistema operativo resulta muy importante para el dispositivo móvil ya que proporciona al usuario un conjunto de posibilidades o servicios que generará una diferencia entre los diferentes sistemas operativos. Dichos componentes también son llamados capas como se verá en la siguiente gráfico:



1.5.1 Las aplicaciones

La capa más alta dentro de la arquitectura es de las aplicaciones ya que esta interactúa con el usuario concentrándose solo en el manejo y ejecución de la aplicación mas no en su implementación.

Cada sistema operativo desarrolla sus propias aplicaciones y esto hace la diferencia cuanto se adquiere un dispositivo móvil, a continuación se verá en qué plataforma se desarrollan los principales sistemas operativos:

- Todas las aplicaciones **Android** se desarrollan en Java.
- Todas las aplicaciones **Windows Phone** se desarrollan en **Visual Studio**.
- Todas las aplicaciones **iOS** se desarrollan en **Objetive-C**.

Ahora, se verán los tipos de aplicaciones que se usan en nuestros dispositivos móviles:

A. Aplicaciones nativas

Se denomina así a las aplicaciones que han sido desarrolladas en su mismo software, son creadas exclusivamente para un determinado tipo de dispositivo. Es decir, las aplicaciones nativas de Android solo podrán ser ejecutadas en dispositivos móviles que tengan dicho sistema operativo.

Inicialmente el dispositivo móvil ya viene precargado de aplicaciones nativas, pero el usuario puede actualizarlas descargando la última versión desde la tienda virtual Play Store. Una de las principales características que presentan las aplicaciones nativas es que no será necesario estar conectados



a una red para ejecutar dichas aplicaciones y que sin necesidad de abrir el aplicativo este puede mostrar notificaciones como lo realiza Facebook u otras aplicaciones.

Entonces, se mencionan algunas ventajas y desventajas que presentan las aplicaciones nativas:

Ventajas:

- Tienen un absoluto control de los componentes hardware y software.
- Cuentan con una mejor performance y alto rendimiento en su dispositivo de origen.
- Envían notificaciones o avisos a los usuarios sin importar una conexión segura de internet.
- Cada vez se desarrollan más aplicaciones nativas que permiten una mejor experiencia al usuario.
- Los fabricantes ofrecen las actualizaciones o mejoras mediante la tienda virtual Play Store ya que permiten publicarlas para su distribución.
- Muchas de estas aplicaciones no necesitan conexión a internet para ejecutarse.
- La actualización de las aplicaciones nativas es constante cuando detecta una conexión a internet estable.

Desventajas:

- Tienen un alto costo frente a otros tipos de aplicaciones tanto para su desarrollo como para la publicación en la tienda virtual.
- Se desarrollan en diferentes lenguajes de programación y solo pueden ser ejecutadas en un mismo sistema operativo; si se quiere utilizar aplicaciones en otros sistemas se tendrá que buscar emuladores.
- Hay un consumo alto de tiempo para su desarrollo.
- Si el desarrollador de la aplicación no actualiza su código siempre se tendría la misma funcionalidad en la aplicación.

B. Aplicaciones web o web-apps

Se denomina así a las aplicaciones que se pueden visualizar en los dispositivos móviles bajo un navegador. Estas aplicaciones son desarrolladas mediante una integración entre el lenguaje HTML5, CSS3, JavaScript y otros que permitan implementar una aplicación web. A diferencia de las aplicaciones nativas esta no se preocupa del tipo de sistema operativo, pues solo se debe contar con un navegador web.

Entonces, se mencionarán algunas ventajas y desventajas que presentan las webapps:

Ventajas:

- Puede ser usado en cualquier dispositivo móvil que cuente con navegador web.
- El código de su desarrollo puede ser distribuible entre otras aplicaciones webapps.
- No necesita instalar una aplicación webapps.
- El costo para su desarrollo es variable si se compara con las aplicaciones nativas.
- Permite mostrar todo tipo de información al usuario como si estuviera frente a una computadora.

Desventajas:

- Debe tener una conexión a internet todo el tiempo de uso de la aplicación webapps.
- Por su naturaleza no pueden ser distribuidas en tiendas virtuales.
- No tienen necesidad de usar recursos del dispositivo móvil y mucho menos del sistema operativo.

Un ejemplo de webapps está en la versión responsiva de algunos sitios web; al ingresar al sitio web este se adecua al dispositivo móvil tanto en tamaño como distribución de los elementos que componen la web.



Fuente: <http://flywebmedia.com/wp-content/uploads/2015/04/Responsive-Web-Design-Hilton-Head.jpg>

C. Aplicaciones híbridas o webapps nativa

Se denomina así a las aplicaciones que combinan la tecnología de las aplicaciones nativas y las webapps. Estas aplicaciones son desarrolladas en integración con HTML5, CSS3 y otros pero además usan un marco de trabajo (framework) específico como por ejemplo PhoneGap, Titanium appAcelerator y otros.



Las aplicaciones híbridas son compatibles con todos los sistemas operativos móviles y especialmente estas pueden ser publicadas en la tienda virtual.

Entonces, se mencionarán algunas ventajas y desventajas que presentan las aplicaciones híbridas:

Ventajas:

- Se desarrollan de manera independiente del sistema operativo.
- Son las únicas aplicaciones que se distribuyen en diferentes tiendas virtuales ya que son multiplataforma.
- Tienen acceso a los recursos hardware y software del dispositivo móvil.

Desventajas:

- El usuario cree estar usando una aplicación web y no una nativa.
- No se cuenta con documentación de software.

1.5.2 Estructuras de una aplicación

Esta capa se encuentra debajo de las aplicaciones porque gestiona su ejecución y encima de la biblioteca y núcleo del sistema operativo, su objetivo es brindar bloques de construcción de alto nivel que permitirán crear aplicaciones para nuestro sistema. Las principales partes son:

A. Gestor de actividades

Es el encargado de controlar el ciclo de vida de cualquier tipo de aplicaciones que se ejecute en un dispositivo móvil.

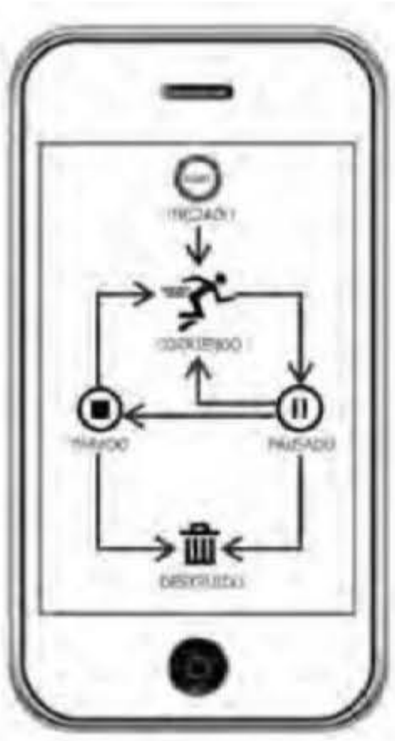


Figura 1.1 Ciclo de vida de una actividad en un dispositivo móvil

B. Gestor de ventanas

Es el encargado de gestionar las ventanas bajo el sistema operativo Android, el cual permite administrar las ventanas que se muestran a los usuarios al hacer uso de cualquier tipo de aplicación en su dispositivo móvil.

C. Proveedores de contenido

Es el encargado de encapsular datos o información que pueden ser compartidos entre diferentes aplicaciones; estos se encapsulan en una lógica interna llamada API el cual brinda un conjunto de servicios a las aplicaciones que lo soliciten. Por ejemplo, hay aplicaciones que usan la información de los contactos registrados en nuestro dispositivo móvil.

D. Gestor de telefonía

Es el encargado de controlar todo lo referente a la telefonía del dispositivo móvil siempre y cuando este tenga el servicio de telefonía activo; es así que existen Tablets con chip para móvil y aquí también se administra la gestión de telefonía.

E. Gestor de paquetes

Es el encargado de administrar las carpetas con que cuenta el sistema operativo móvil, es así que en un dispositivo móvil nuevo se encuentra una estructura de carpetas predeterminada.

F. Gestor de visualización

Es el encargado de gestionar la visualización de las diferentes aplicaciones en el dispositivo móvil.

G. Gestor de notificaciones

Es el encargado de controlar las notificaciones que realizan las diferentes aplicaciones en un determinado dispositivo móvil, por ejemplo los mensajes de texto, alertas, etc.

H. Gestor de recursos

Es el encargado de la gestión de los recursos que podrían solicitar las diferentes aplicaciones desde el dispositivo móvil. Se le llama recurso a cualquier elemento no código dentro del sistema.

I. Gestor de sensores

Es el encargado de gestionar los sensores dependiendo de la actividad que se realiza en el móvil, por ejemplo todos los teléfonos cuentan con un sensor de proximidad el cual apaga la pantalla mientras usted acerca el teléfono a su oído.

J. Gestor de ubicación

Es el encargado de determinar la ubicación geográfica del dispositivo móvil mediante el GPS del equipo.

1.5.3 Bibliotecas

Es la capa que contiene aplicaciones nativas de Android. Su desarrollo está basado en el lenguaje C y C++; estas ya se encuentran compiladas y listas para ser usadas por cualquier componente Android en los dispositivos móviles. A continuación algunas de la características que presentan las bibliotecas o librerías de la arquitectura Android.

A. Librería C - LibC

La biblioteca estándar de C es una evolución de la librería BSD de C estándar; tiene la misión de recopilar todo tipo de archivos de cabecera y biblioteca con rutinas establecidas las cuales permiten un control de las operaciones que realiza el sistema operativo como I/O (Input/Output).

B. Bilioteca de navegación - WebKit

Es una tecnología moderna usada por Google en sus navegadores que permite mostrar información desde aplicaciones web basadas en HTML.

C. Librerías gráficas - SGL

Gracias a esta librería se puede combinar elementos que contengan 2D y 3D en una misma interfaz gracias a sus motores de gráficos, en algunas casos se necesitará renderizar solo si su dispositivo no dispone del adecuado.

D. OpenGL ES

Librería que ofrece servicios de acelerados de hardware para 3D.

E. Gestor de superficie

Encargado de manejar el acceso al subsistema de representación gráfica en 2D y 3D, esto permite administrar el diseño mostrado al usuario usando una combinación de mapa de bits y así obtener mejores efectos en diferentes aplicaciones.

F. Multimedia

Cada versión del sistema operativo para Android cuenta con una gran variedad de codecs que permite visualizar todo tipo de video y de la misma manera para el tema de audio e imágenes.

G. SQLite

Android incluye un gestor de base de datos llamada SQLite el cual proporciona un conjunto de servicios basados en la administración de la información que puede ser usado por las aplicaciones dentro del dispositivo móvil.

H. Puertos seguros - SSL (Secure Socket Layer)

Proporciona un conjunto de servicios basados en seguridad y especialmente de encriptación de datos el cual permite al usuario tener protegido sus dispositivos mediante el uso de contraseñas, asimismo proporciona capacidades para otros tipos de aplicaciones de uso seguro.

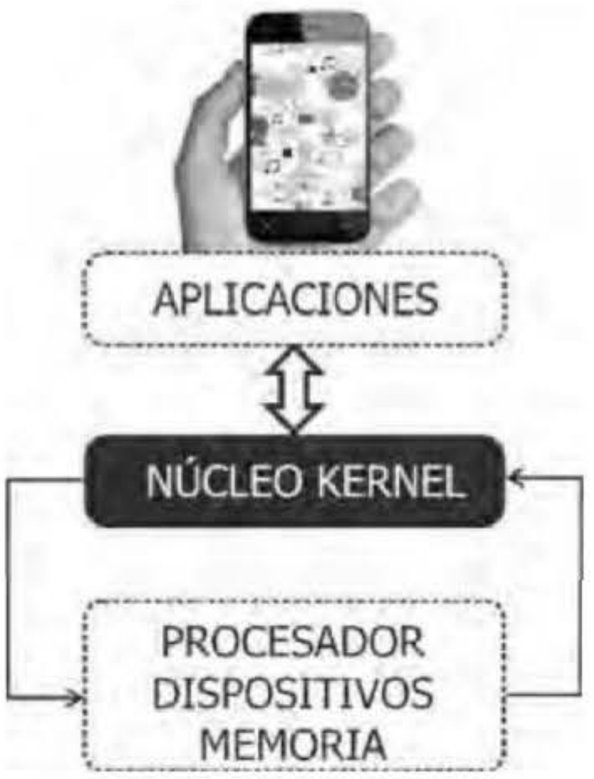
I. Fuentes - FreeType

Proporciona el uso de fuentes bitmap y renderizados de forma vectorial que permite tener una mejor visualización de los textos en diferentes formatos del dispositivo móvil.

1.5.4 El núcleo Kernel

Kernel es el componente más importante que puede tener un sistema operativo móvil ya que gestiona una integración entre el hardware del dispositivo móvil y el software del mismo.

Asimismo, una de sus principales características es que brinda servicio a los demás componentes como los controladores del dispositivo, gestión y administración de archivos, gestión y administración de procesos y principalmente la gestión y administración de la memoria del dispositivo. En el caso de Android presenta la particularidad de contar con un motor Java en el desarrollo de sus núcleos.



1.6 ANDROID

1.6.1 HISTORIA

Es un sistema operativo de código abierto que basa toda su tecnología en Linux y está orientado a cualquier tipo de dispositivo móvil. En octubre del 2003 Andy Rubin, Richard Miner, Nick Sears y Chris White fundaron la compañía Android Inc. Inicialmente crearon un sistema operativo para cámaras fotográficas y con el objetivo de ampliar su mercado de ventas, implementaron mejoras en el sistema haciendo que el sistema sea de código abierto adaptando todo el proyecto en dispositivos móviles.

En julio de 2015, la empresa Google compra Android Inc, dando un gran paso a la nueva era de los dispositivos móviles, crea un sistema operativo muy útil, de libre acceso y sin licencias como hasta hoy se viene utilizando. En noviembre de 2007 se anuncia la creación de una alianza (Open Handset Alliance) entre fabricantes de tecnologías como dispositivos y semiconductores, así como entre compañías de software, operadores de telefonía y otras empresas con el objetivo específico de implementar el proyecto Android como un sistema de código libre y completo; una plataforma completa de aplicaciones para el usuario de dispositivos móviles sin necesidad de adquirir licencias o software propietario. Las empresas que más destacan en la alianza son Google, HTC, Samsung, Intel, entre otros.

En octubre de 2008 nace la primera tienda virtual llamada Android Market para usuarios del sistema Android aproximadamente con unas cincuenta aplicaciones de libre acceso y descarga. Finalmente, en octubre de 2012 se modifica el nombre de la tienda por Play Store. Es tan popular la tienda en el planeta que hasta mayo de 2016 tiene alrededor de 65 billones de descargas, como se muestra en la siguiente gráfica:

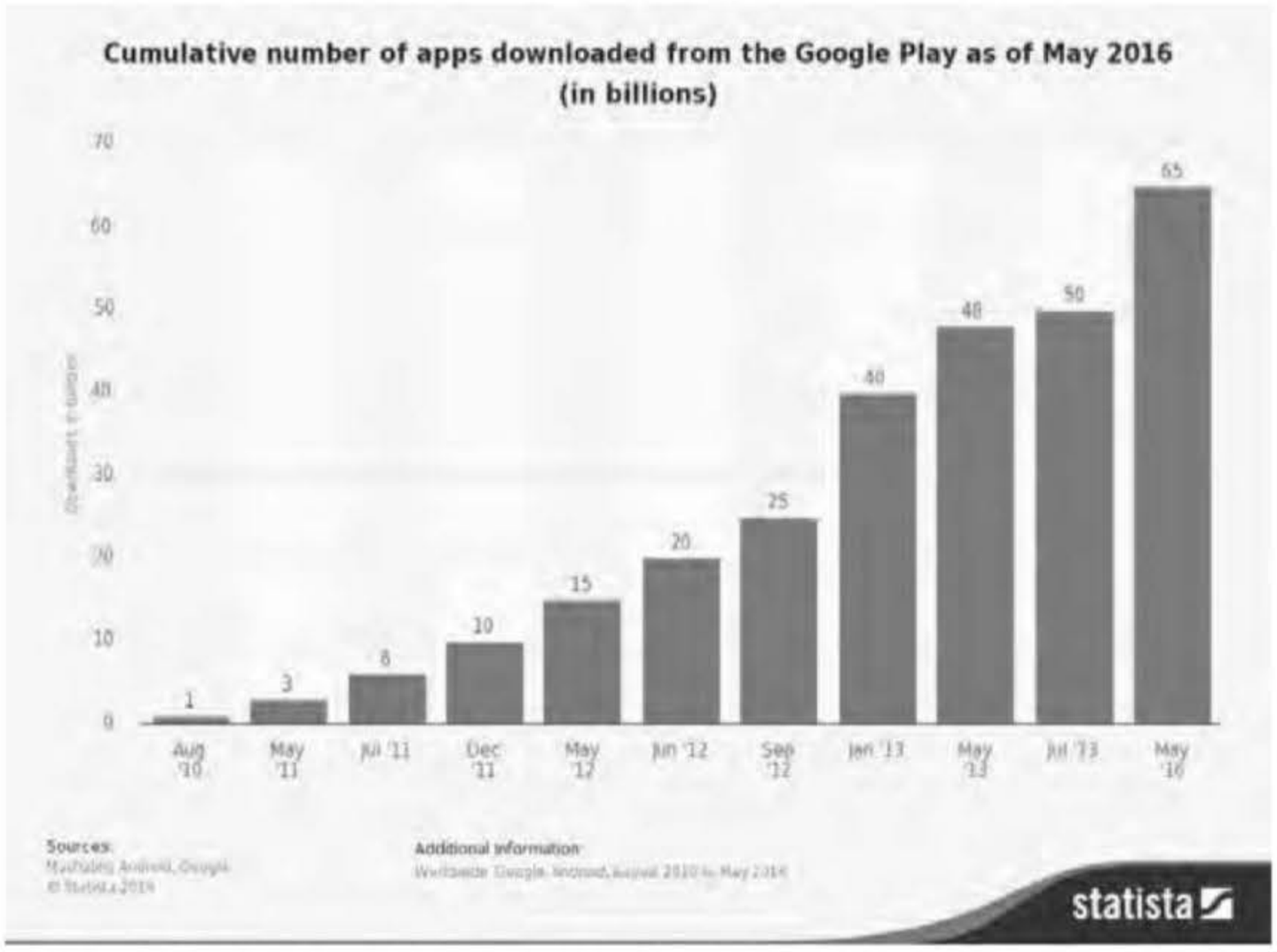


Figura 1.2 Cantidad de descargas realizadas desde Play Store (agosto 2010 - mayo 2016)

Fuente: <http://www.statista.com/graphic/5/281106/number-of-android-app-downloads-from-google-play.jpg>, <http://signum.org.es/tag/android/>

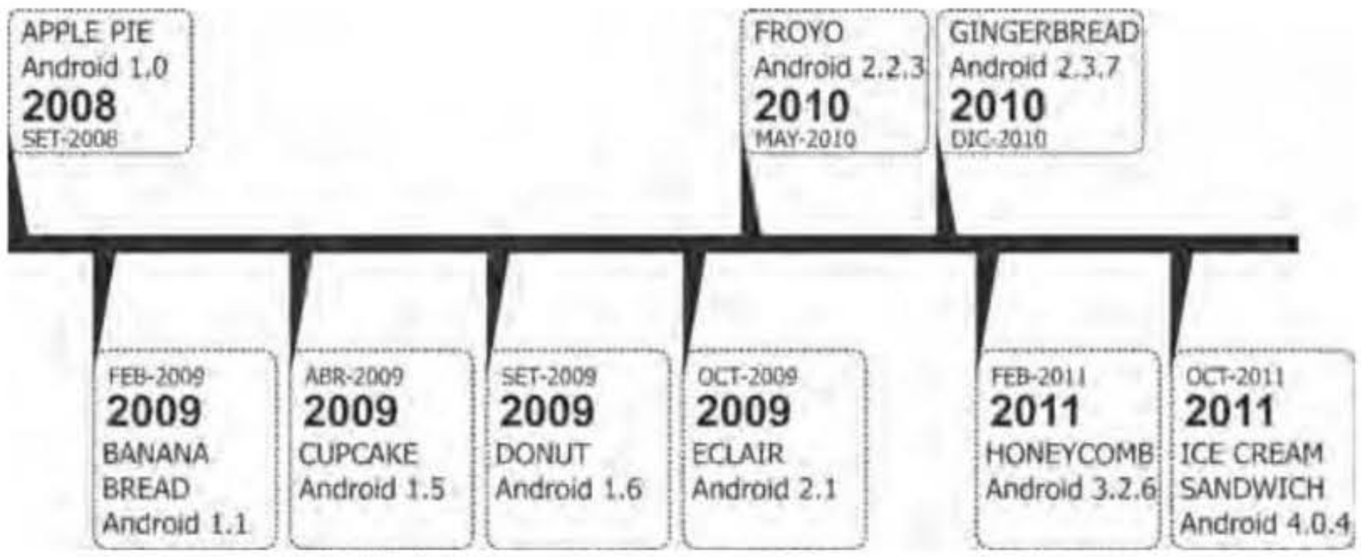
1.6.2 Características

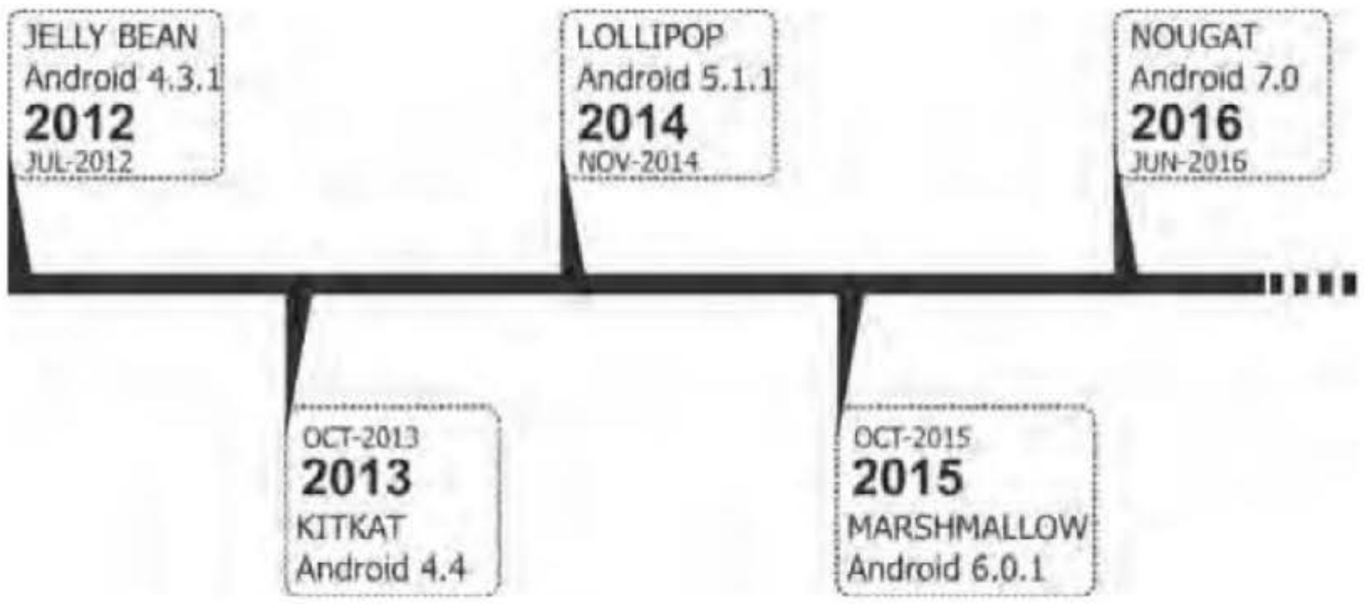
Cuando se hizo un estudio de los componentes de la arquitectura Android se hacía referencia a sus características, por lo tanto, ahora se resumirá las características más importantes que presenta Android:

- Presenta una plataforma totalmente adaptable a todo tipo de pantallas sin importar el tamaño de estas.
- Cuenta con una librería de gráficos 3D que permiten tener una mejor experiencia en el manejo de videos o juegos. Está basada en las especificaciones de la OpenGL ES.
- Posee su propio gestor de base de datos llamado SQLite el cual se caracteriza por su simplicidad, rapidez y usabilidad que permiten un desarrollo de aplicaciones basado en datos de forma profesional.
- Posee una amplia de conectividad nativa como por ejemplo GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, WiMAX, etc.
- Posee una administración de mensajería basada en SMS, MMS y Android Cloud to Device Messaging Framework.
- Tiene un navegador web nativo que permite visualizar páginas web basadas en HTML.
- Posee una máquina virtual que le permite ejecutar sus aplicaciones llamada Dalvik.
- Soporta la mayoría de los formatos multimedia tanto para video, audio o grabación.
- Presenta un gran soporte para la transmisión de video y audio por secuencias más conocido como streaming.
- Posee un soporte para hardware adicional como cámara de fotos, de video, pantallas táctiles, GPS, etc.
- Soporta la tethering, el cual permite al dispositivo móvil ser usado como un punto de acceso para permitir a un computador portátil usar la conexión 3G.

1.6.3 Versiones

Se verá la evolución con respecto a las versiones que presenta Android desde su primera aparición en setiembre de 2008 hasta junio de 2016.





Los aspectos más importantes en las versiones de Android son:

Versión	Descripción
apple Pie	El primer dispositivo móvil en incorporarlo fue el HTC Dream, era también conocido como Google Phone. También incluía la primera versión de la tienda virtual llamada Android Market.
Banana Bread	Se presentó en dispositivos como el T-Mobile G1, el cual presentaba algunas mejoras importantes con respecto a la versión inicial.
CupCake	Introducción del teclado virtual y la posibilidad de agregar algunas widgets.
Donut	Mejora en los cuadros de búsqueda y renueva el aspecto inicial que presentaba la tienda virtual.
Eclair	Integra los contactos del dispositivo móvil con las principales redes sociales como Facebook o Twitter.
Froyo	Destaca la velocidad de todo su sistema al ejecutar las aplicaciones y realizar navegaciones web. Además incluye la posibilidad de convertir al dispositivo móvil en un hotspot.
GingerBread	Renueva la interfaz de usuario con mejoras en su resolución y agrega a su teclado virtual la función de corregir textos mientras se escribe.
HoneyComb	Renueva la interfaz del usuario incorporando una barra en la parte superior con acceso rápido a notificaciones y otras características del sistema.
Ice Cream Sándwich	Los widgets se separan de las aplicaciones mostrándose independientes. También incluye el reconocimiento facial de los usuarios como tema de seguridad.
Jelly Bean	Mejora en las notificaciones al usuario e incorpora la aplicación Google Now como una características importante de esta versión.
KitKat	Introduce el tema de reducción de sus elementos mejorando considerablemente el aspecto. Implementa la funcionalidad de la aplicación Hangouts como un servicio oficial de Google.
Lollipop	Destaca la forma de funcionamiento de las notificaciones y el consumo de la batería. Modifica la máquina virtual Dalvik por ART (Android Runtime).
Marshmallow	Incorpora un mejor rendimiento y estabilidad en el sistema operativo. Añade un soporte de huellas dactilares.
Nougat	Es la nueva versión de Android el cual incorpora el manejo de varias aplicaciones en un mismo entorno, se puede abrir dos o más ventanas y distribuirlas de manera adecuada. Mejora en la duración de la batería. Permite personalizar la barra de navegación en los dispositivos. Ahora se podrá responder desde las notificaciones. Mejora en el explorador de archivos.

1.7 Proceso de diseño y desarrollo de una aplicación Android

Se verá cuál es el proceso adecuado para un desarrollo de aplicaciones para dispositivos móviles usando Android ya que inicialmente se recomienda que tenga algunos conocimientos de programación Java y especialmente orientada a objetos.

1.7.1 Análisis

En esta fase se analiza el proyecto y se determina qué soluciones se le brindará al usuario final, recuerde que toda aplicación debe estar centrada en el usuario y no en el desarrollador; finalmente, este etapa termina cuando se tiene una idea clara del tipo de aplicación que desea desarrollar.

Se podría mencionar que aquí se debe trazar ciertos objetivos que debe cumplir la aplicación como, por ejemplo, "tener el control del registro de productos en una tienda y mantenerlos actualizados".

1.7.2 Formalización del proyecto

En esta etapa se define la funcionalidad que debe tener la aplicación, el cual determina las reglas de negocio que debe cumplir el desarrollador de la aplicación. Todo esto debe estar especificado en un documento usando algunos diagramas que demuestren el funcionamiento de la aplicaciones haciendo que se entienda a quien está dirigida la aplicación.

1.7.3 Maquetado

Esta etapa define prototipos sobre cómo será el proyecto al finalizarlo; esto le dará una idea clara al desarrollador sobre cómo implementará dicha aplicación, asimismo se podría visualizar cómo se relaciona el usuario con la aplicación. Existen muchas aplicaciones que permiten maquetar de manera profesional una aplicación, por ejemplo, está Balsamiq Mockups (https://balsamiq.com/products/mockups/).



Fuente: <https://bbvaopen4u.com/sites/default/files/img/embed/new/bbva-open4u-wireframes.png>

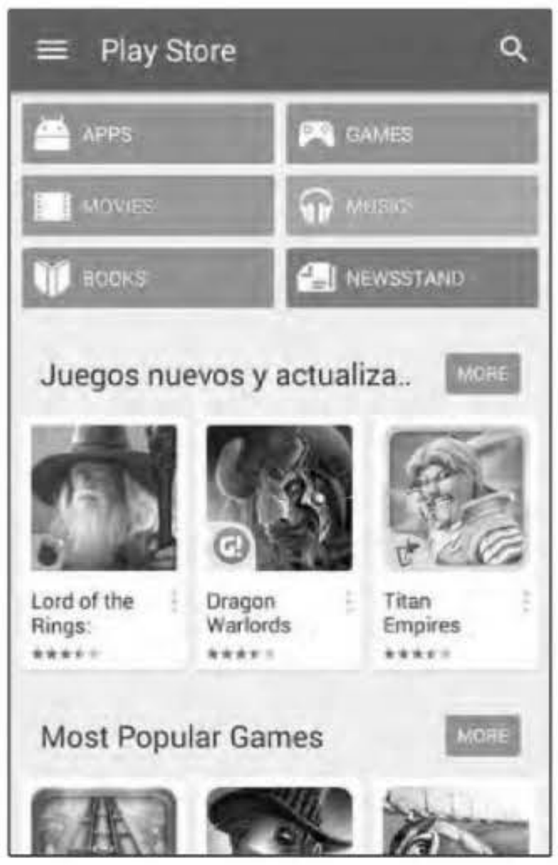
1.7.4 Codificación

El desarrollador de aplicaciones se encargará de llevar el maquetado a un lenguaje de programación agregando códigos que permitirá darle la lógica necesaria a la aplicación para que cumpla todas las reglas de negocio especificadas en los documentos anteriores. En este caso se usará Android Studio como soporte para el desarrollo de aplicaciones con Android.



1.7.5 Publicación

La publicación de aplicaciones móviles en la tienda virtual tiene un costo de $25 y se puede realizar una vez que se termina con las aplicaciones móviles.



Fuente: <http://screenshots.de.sftcdn.net/de/scrn/3336000/3336209/google-play-store-12-342x535.jpg>



CAPÍTULO

2

Programación orientada a objetos con Java

2.1 Introducción

La programación orientada a objetos surge como un intento de dominar la complejidad que tiene la construcción de un software. A partir de esto se desarrollarán aplicaciones grandes o pequeñas duraderas en el tiempo y factible de modificar. Tradicionalmente, la forma de enfrentarse a la complejidad del software era usando la programación estructurada; el cual hacía que una aplicación compleja se descomponga en porciones pequeñas de códigos simples y fáciles de codificar. Entonces, se puede decir que la programación orientada a objetos es otra forma de descomponer el código por medio de la descomposición de objetos.

La programación orientada a objetos (POO) en java es una forma especial de implementar aplicaciones más cercanas a cómo se expresarían las cosas en la vida real que otros tipos de programación. Con la POO se tiene que aprender a pensar las cosas de una manera diferente. Para implementar nuestras aplicaciones en términos de objetos, propiedades y métodos en el desarrollo de este material se usarán las clases ya implementadas dentro de las aplicaciones móviles con Android.

En la actualidad programar orientado a objetos es algo común entre los programadores ya que los lenguajes de programación promueven este estilo. Durante varios años, los desarrolladores se han dedicado a construir aplicaciones muy parecidas que permitían resolver problemas haciendo que el programador sea una persona indispensable en la organización ya que solo él conoce el código que implementó, la POO se creó con la idea principal de no depender de una persona particular de manera que otras personas puedan utilizarlas y adelantar su trabajo, de manera que se consiga que el código se pueda reutilizar.

Con todo lo manifestado de la POO no piense que es difícil programar en orientada a objetos, por el contrario, se volverá una cultura en muy poco tiempo para usted ya que desarrollar una aplicación móvil promueve el uso de Java como lenguaje de programación. Ahora toca pensar en términos de objetos para poder entender mejor los términos usados hasta el momento. A continuación se verá cómo encontrar algunas características de un dispositivo móvil:



Concepto	El teléfono móvil es un dispositivo inalámbrico electrónico para acceder y utilizar los servicios de la red de telefonía celular o móvil. Se denomina celular en la mayoría de países latinoamericanos debido a que el servicio funciona mediante una red de celdas, donde cada antena repetidora de señal es una célula.
Características	Sistema Operativo: Android 4.0.4 Banda: 3G – 850MHz Memoria RAM: 1GB Pantalla capacitiva 5.2 pulgadas Resolución 800 × 600 pixeles Doble cámara 13 Megapíxeles
Funcionalidad	Llamada a otros teléfonos Captura fotos Captura videos Ejecuta juegos 3D Reproduce música y videos Accede a redes sociales

En programación orientada a objetos el teléfono móvil sería un objeto de la clase teléfono, las propiedades serían las características y los métodos serían las funcionalidades que tiene el dispositivo móvil.

2.2 Conceptos en programación orientada a objetos

2.2.1 Paquetes

Una aplicación móvil en Android cuenta con paquetes predeterminados los cuales son usados como un mecanismo para administrar activitys, imágenes, valores, íconos y principalmente clase que compone una aplicación móvil; el propósito es tener organizado dichos componentes dentro de un proyecto.

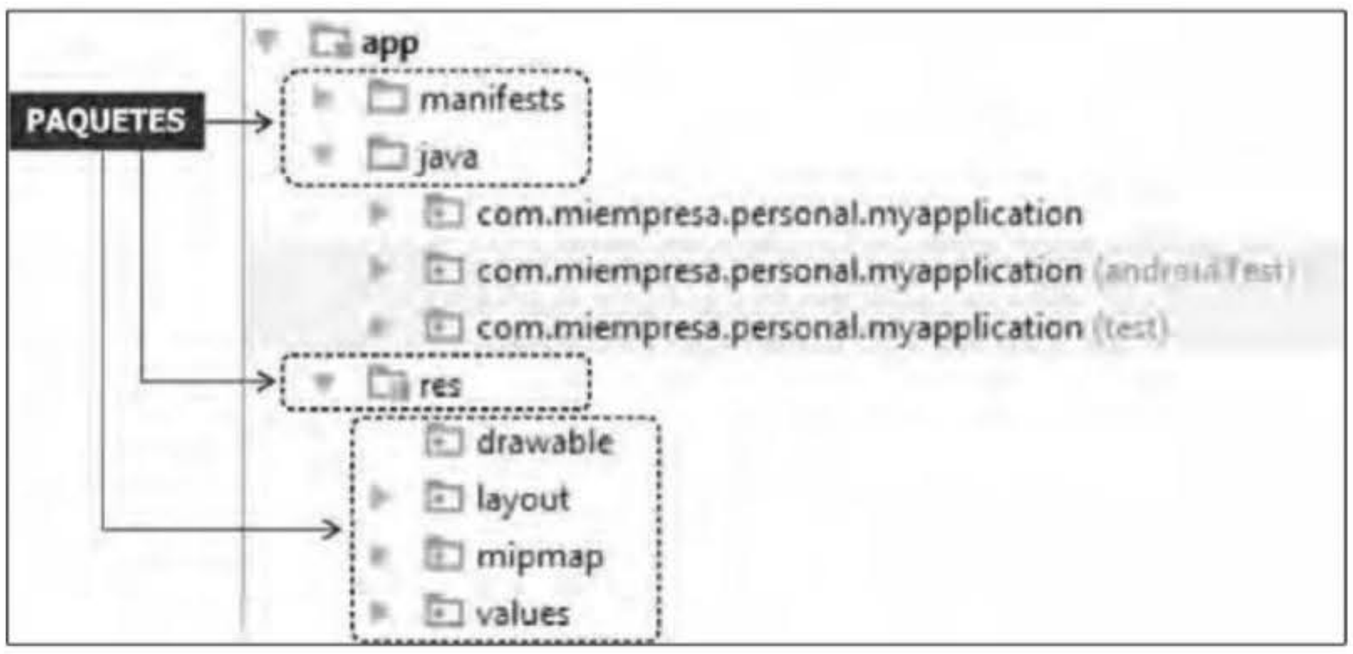


Figura 2.1 Estructura básica de una aplicación Android (Android Studio)

En la figura se puede observar que una aplicación Android está organizada principalmente por paquetes que tienen un aspecto muy parecido a las carpetas de Windows, asimismo se podría decir que estos tienen las mismas características. Pero lo que le diferencia es que a nivel de código para agregar un componente dentro del paquete se usa la cláusula “package”, por ejemplo, vea el siguiente código:

```
package com.miempresa.personal.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Para visualizar la definición de los paquetes usted debe aperturar todos los archivos de la aplicación con extensión .java. Entonces, la actividad que se muestra se encuentra definida dentro del paquete "com. miempresa.personal.myapplication".

Para poder hacer referencia a algún componente dentro del paquete debe usar la siguiente sentencia "import paquete.subpaquete", por ejemplo, en el siguiente código:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

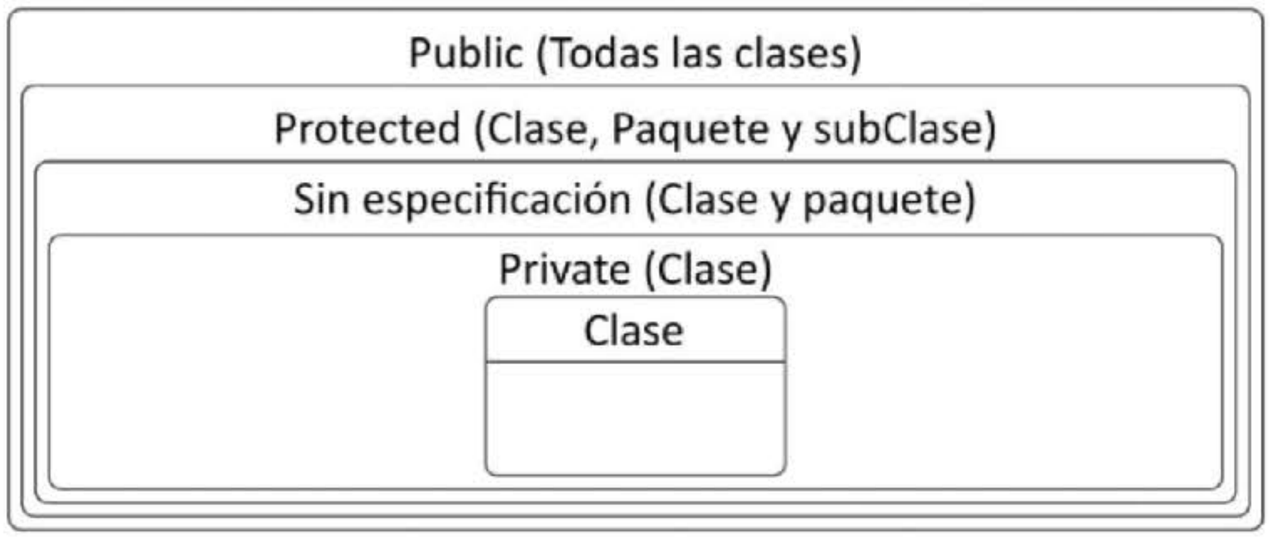
La cláusula import permite hacer referencia a los paquetes y subpaquetes que pudiera tener la aplicación; en el código anterior el paquete Android tiene muchos subpaquetes que pueden referenciarse mediante puntos. Para nuestro caso al agregar algún código que necesite un paquete particular solo debe presionar Alt + Enter y automáticamente se codificará la sentencia import haciendo referencia a los paquetes necesarios.

2.2.2 Alcance de los elementos contenidos en un paquete

Si se tiene en cuenta que la construcción de un paquete conlleva una organización de clases, entonces también se debe considerar el alcance o visibilidad que tienen frente a los demás paquetes, por ejemplo; si se necesita hacer referencia de un paquete a otro, se debe considerar que dicho elemento sea privado, público, protegido o no tenga un alcance definido. A continuación un cuadro que muestra la situación que puede tener un elemento dentro de un paquete:

Situación del elemento contenido en el paquete	Visibilidad privada (private)	Visibilidad pública (public)	Visibilidad protegida (protected)	Sin visibilidad
Dentro de la misma clase	SÍ	SÍ	SÍ	SÍ
Desde una subclase	NO	SÍ	SÍ	NO
Desde una clase implementada dentro del mismo paquete	NO	SÍ	SÍ	SÍ
Desde el exterior de la misma clase	NO	SÍ	NO	NO

En la siguiente gráfica se muestra el alcance que pueden tener los miembros de una clase, usted como programador debe tener en cuenta que tanto atributos como métodos trabajan con un fin en común y que la definición del alcance es muy importante para la aplicación.

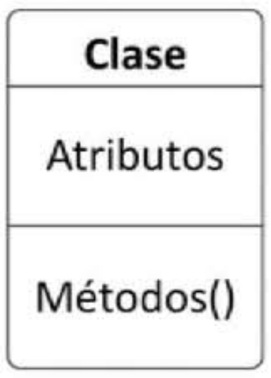


2.3 Clases en Java

Una clase es elemento básico de la programación orientada a objetos, representa a una plantilla desde la que se pueden crear objetos. Técnicamente una clase es una declaración de objetos, también se podría definir como una abstracción de objetos; esto quiere decir que la definición de un objeto es la clase. Cuando se programa sobre un objeto y se le define características y funcionalidades, en realidad lo que se hace es programar en una clase.

2.3.1 Partes de una clase

- Técnicamente una clase está compuesta por un nombre identificativo, atributos (los cuales representan las características que puede tener la clase) y los métodos que son funciones que puede realizar la clase usando ciertos atributos:



- En Java se visualiza con el siguiente script:

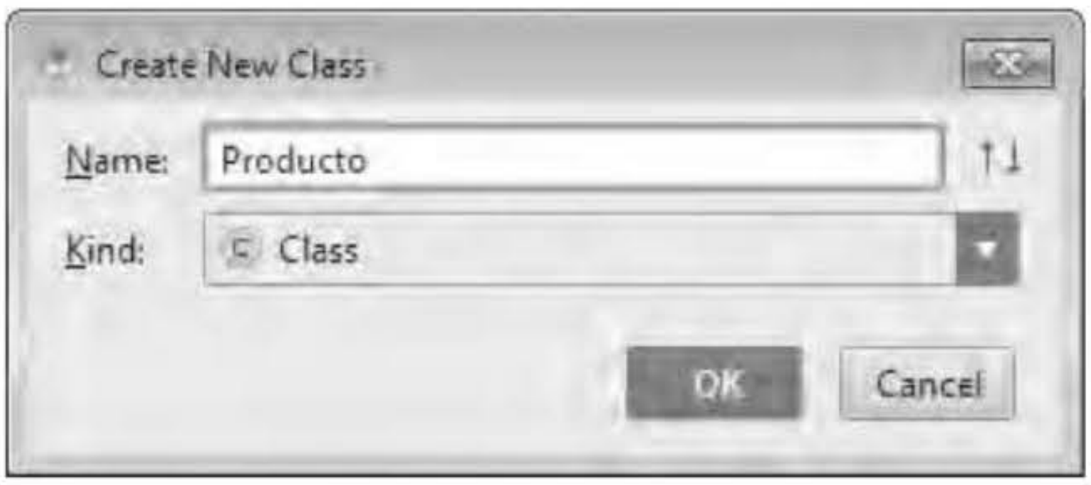
Nivel de la clase

```
public class Clase
{
    Atributos
    Métodos()
}
```

Si se necesita implementar la clase Producto en un sistema de control de ventas se tendrá:

Producto
- codigo: **int** - descripcion: String - categoria: **int** - precioUnitario: **double**
+producto(codigo:**int**,descripcion:String,categoria:**int**,precioUnitario:**double**) +getCodigo():**int** +getDescripcion():String +getCategoria():**int** +getPrecioUnitario():**double** +setCodigo(codigo:**int**):**void** +setDescripcion(descripcion:String):**void** +setCategoria(categoria:**int**):**void** +setPrecioUnitario(**double** precioUnitario):**void**

En Android Studio, para crear una clase se hace clic derecho sobre un subpaquete del paquete *Java > New > java class* y se muestra la siguiente ventana:



Se asigna un nombre y solo se presiona el botón Ok para iniciar con la creación de la clase, inicialmente la clase se muestra de la siguiente manera:

```
package com.miempresa.personal.myapplication;

public class Producto {

}
```

La cláusula package hace referencia al paquete donde se guarda la clase Producto. Hay que tener en cuenta que toda clase se crea pública y las llaves describen el límite de la misma.

```
package com.miempresa.personal.myapplication;

public class Producto{
    private int codigo;
    private String descripcion;
    private int categoria;
    private double precioUnitario;

    public int getCodigo(){
        return codigo;
     }

    public String getDescripcion(){
        return descripcion;
    }

    public int getCategoria(){
        return categoria;
    }

    public double getPrecioUnitario(){
        return precioUnitario;
    }


    public void setCodigo(int codigo){
        this.codigo=codigo;
    }

    public void setDescripcion(String descripcion){
        this.descripcion=descripcion;
    }

    public void setCategoria(int categoria){
        this.categoria=categoria;
    }

    public void setPrecioUnitario(double precioUnitario){
        this.precioUnitario=precioUnitario;
    }
}
```

2.3.2 Atributos de una clase

Un atributo de clase es la característica que puede tener un objeto. Se puede decir que las propiedades son como variables donde se almacenan datos relacionados con los objetos. Se recomienda implementar atributos con visibilidad privada y que para exponerlos en la aplicación se usen los métodos get y set.

Formato:

```
Visibilidad tipoDatos nombreAtributo;
```

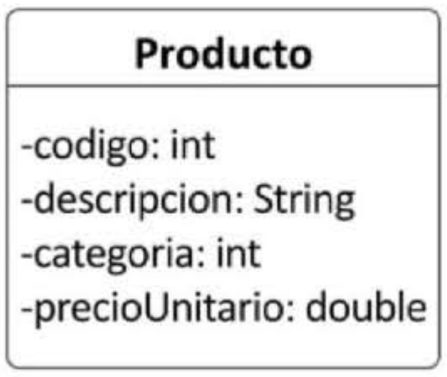
Donde:

- **Visibilidad:** Puede ser private, public, protected o sin definición.
- **TipoDatos**: Son los tipos de datos primitivos pero también se puede hacer referencia a clases específicas.
- **NombreAtributo**: Es el nombre que se le asignará al atributo de clase.

Ejemplos de declaración de atributos: Suponga que tiene la clase Vendedor y se necesite declarar el atributo código, ¿cuántas posibilidades tiene para declarar y cuál es la consecuencia de ello?

Declaración de un atributo	Explicación
`int codigo;`	Al declarar código sin visibilidad solo podrá ser accedida desde la misma clase y no desde el exterior de la clase.
`public int codigo;`	Al declarar código como público todas las clases del mismo proyecto podrán hacer referencia al valor contenido en dicho atributo.
`private int codigo;`	Al declarar como private solo podrá ser accedida desde la clase que se definió. Hay que mencionar que la implementación del atributo como privado es la más recomendada, puesto que se protege la información contenida en el atributo; es decir, no será manipulada la información.
`protected int codigo;`	Al declarar como protected solo podrá ser accedida desde la clase que se definió y las clases descendientes de esta.

Vea el script que declara los atributos de la clase Producto como privados:

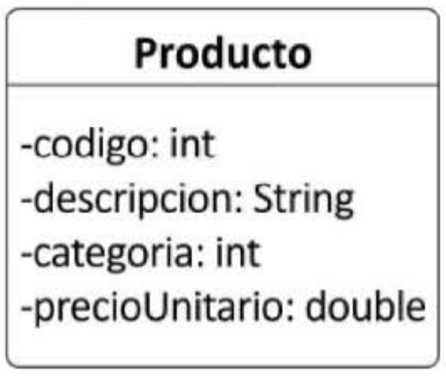


En java tendrá el siguiente script:

```
package com.miempresa.personal.myapplication;

public class Producto{
    private int codigo;
    private String descripcion;
    private int categoria;
    private double precioUnitario;
}
```

Vea el script que declara los atributos de la clase Producto como públicas:



En java tendría el siguiente script:

```
package com.miempresa.personal.myapplication;

public class Producto{
    public int codigo;
    public String descripcion;
    public int categoria;
    public double precioUnitario;
}
```

2.3.3 Métodos en las clases

Los métodos de la clase tienen el propósito de encapsular código con un objetivo específico, esto permite tener una organización dentro del código y así poder reutilizar el código.

Formatos:

```
Visibilidad tipoDatos nombreMetodo();
Visibilidad tipoDatos nombreMetodo(Parámetros);
```

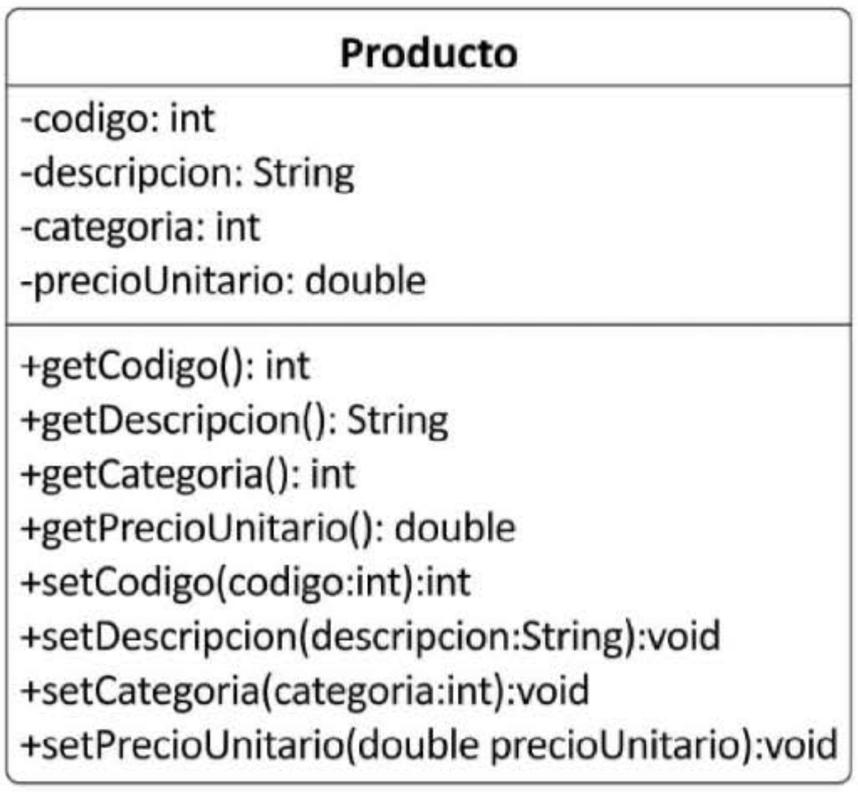
Donde:

- *Visibilidad*: Puede ser private, public, protected o sin definición.
- *TipoDatos:* Es el tipo de datos que retornará el método, cuando no retorna algún valor se colocará la cláusula void.
- *NombreMetodo:* Es el nombre que se le asignará al método de clase, hay que tener en cuenta que dependiendo del trabajo que realice el método se deberá implementar los parámetros.

Ejemplos de declaración de métodos: Suponga que tiene la clase Vendedor y se necesite declarar el método getCodigo:

Declaración de un método	Explicación
`int getCodigo(){ }`	Se declara el método getCodigo sin visibilidad por tanto solo será accesible dentro de la clase.
`public int getCodigo(){ }`	Al declararse como público el método getCodigo podrá ser accesible desde todas las clases de la misma aplicación.
`private int getCodigo(){ }`	Al declararse como privado el método getCodigo solo será accesible dentro de la misma clase.
`protected int getCodigo(){ }`	Al declararse como protegido el método getCodigo será accesible desde la misma clase y las clases descendientes.

Si necesita implementar los métodos de la clase Producto en un sistema de ventas tendría:



En java tendría el siguiente script de la clase Producto:

```
package com.miempresa.personal.myapplication;

public class Producto{
    private int codigo;
    private String descripcion;
    private int categoria;
    private double precioUnitario;

    public int getCodigo(){
        return codigo;
    }
    public String getDescripcion(){
        return descripcion;
    }

    public int getCategoria(){
        return categoria;
    }

    public double getPrecioUnitario(){
        return precioUnitario;
    }

    public void setCodigo(int codigo){
        this.codigo=codigo;
    }

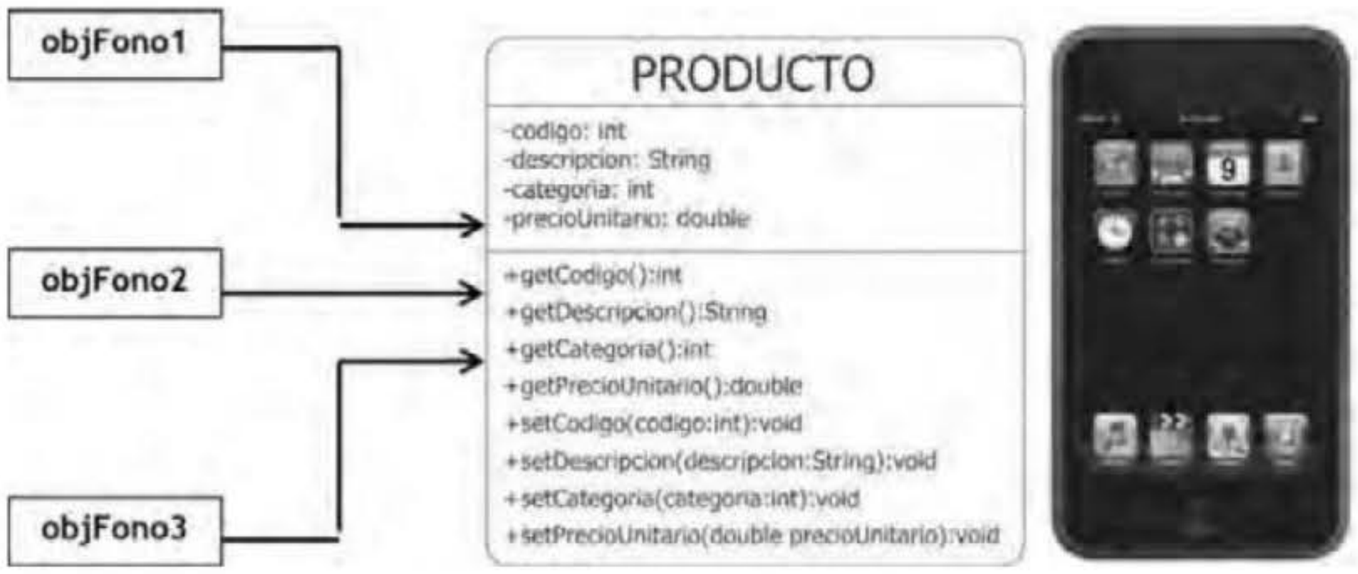
    public void setDescripcion(String descripcion){
        this.descripcion=descripcion;
    }

    public void setCategoria(int categoria){
        this.categoria=categoria;
    }

    public void setPrecioUnitario(double precioUnitario){
        this.precioUnitario=precioUnitario;
    }
}
```

2.4 Objetos en Java

Los objetos son copias o ejemplares de una clase. El hecho de crear un objeto permite tener acceso a los atributos y métodos definidos en la clase. La acción de crear un objeto de una clase es llamada instancia de clase. Véase la siguiente gráfica:



En la gráfica anterior se muestra la implementación de la clase Producto con sus respectivos atributos y métodos. Además, hay 3 instancias de clase mediante los objetos objFono1, objFono2 y objFono3.

Formato de creación de objetos en Java:

Formato una sola línea:

```
Clase Objeto = new Clase();
```

Formato múltiples líneas:

```
Clase Objeto;
Objeto = new Clase();
```

Donde:

- *Clase:* Aquí se debe especificar la clase de donde provendrá el objeto, hay que tener en cuenta que si nos encontramos en otro paquete primero se deberá importar usando la instrucción import.
- *Objeto:* Es el nombre que se le asigna al objeto, este será el enlace con los elementos que compone la clase.
- *New:* Es el operador usado por Java para crear un objeto.
- *Clase():* Es la especificación al método constructor de la clase; consideremos que si no se ha implementado el método constructor en la clase, Java implementa uno no visible para el usuario.

Vea la creación de los tres objetos de la gráfica anterior:

```
Producto objFono1 = new Producto();
Producto objFono2 = new Producto();
Producto objFono3 = new Producto();
```

O también podría realizarse de la siguiente forma:

```
Producto objFono1,objFono2,objFono3;
objFono1 = new Producto();
objFono2 = new Producto();
objFono3 = new Producto();
```

Formato para referenciar a los atributos de una clase:

Cuando se necesite usar los atributos de una clase siempre se realizará por un objeto que instancie a la clase, por tanto, un requisito es que primero se cree el objeto de la clase.

```
Objeto.atributo = valor o Expresión;
```

Donde:

- *Objeto.atributo:* El objeto es la referencia a la clase, el punto es obligatorio al momento de invocar algún atributo público de la clase.
- *Valor o expresión:* Al parámetro se le puede asignar un valor o el resultado de una expresión, solo hay que tener en cuenta que el tipo de datos devuelto por el valor o por la expresión sea el mismo especificado en el parámetro.

Se verá cómo asignar un valor a los parámetros de la clase Producto, se debe tener en cuenta que este código obedece solo si los atributos tienen visibilidad pública:

```
objFono1.codigo = "C0001";
objFono1.descripcion = "LG Smartphone 4G";
objFono1.categoria = 1;
objFono1.precioUnitario = 2250.00;
```

Formato para referenciar a los métodos de una clase:

```
Objeto.metodo();
Variable = Objeto.Metodo();
```

Donde:

- *Objeto.metodo:* El objeto es la referencia a la clase, el punto es obligatorio al momento de invocar algún método público de la clase, tiene que considerar el tipo de valor devuelto por método.

- *Variable:* Esta variable tiene la misión de obtener el resultado de invocar al método de la clase, hay que tener en cuenta que este tipo de instrucción se da solo cuando el método devuelve un valor, es decir, no es de tipo void.

Vea como invocar un método de la clase Producto:

```
String codigo = objFono1.getCodigo();
String descripcion = objFono1.getDescripcion();
int categoria = objFono1.getCategoria();
double precio = objFono1.getPrecioUnitario();
```

Este script muestra la invocación de métodos que tiene un valor de retorno específico, ahora se verá cómo se invocan a los métodos de tipo void:

```
objFono1.setCodigo("C0001");
objFono1.setDescripcion("LG Smartphone 4G");
objFono1.setCategoria(1);
objFono1.setPrecioUnitario(1250.00);
```

2.5 Métodos get y set

Los métodos set permiten actualizar el valor de un parámetro asignado con visibilidad privada, si una clase no tiene implementada un método constructor, los métodos set pueden suplir su trabajo haciendo la asignación inicial de los parámetros.

Los métodos get permiten devolver el valor asignado a un parámetro declarado como privado.

Formato para implementar un método set:

```
Visibilidad void setNombre(Parámetro){
    atributoPrivado = Parámetro;
}
```

Donde:

- *Visibilidad:* Representa el alcance del método set, casi siempre se declara como public ya que los valores provienen del exterior de la clase que lo implementa.
- *Void:* Se le asigna el tipo void ya que los métodos void solo asignan un valor al parámetro privado de la clase sin devolver valor alguno, su trabajo es solo asignación de valor.
- *setNombre:* Es el nombre que se le asigna al método set, casi siempre se inicia con la palabra set seguido del nombre del parámetro privado.
- *Parámetro:* Es el valor recibido desde el exterior que será asignado al parámetro privado de la clase.
- *atributoPrivado=Parámetro:* Esta asignación resuelve el valor que debe tener un parámetro privado, haciendo que el valor que viene de la clase exterior sea asignada directamente al parámetro privado de la clase.

Por ejemplo: Si se tiene la siguiente declaración:

```
public class Empleado {
    private String codigo;
}
```

Se puede implementar el método set del atributo código de una de las siguientes formas:

Primero:

```
public void setCodigo(String codigo){
    this.codigo = codigo;
}
```

Segundo:

```
public void setCodigo(String cod){
    codigo = cod;
}
```

Tercero:

```
public void setCodigo(String sCodigo){
    this.codigo = sCodigo;
}
```

Se usará cualquiera de ellos ya que todos los métodos mostrados realizan la misma actividad; seguidamente se mostrará el script completo para la clase.

```
public class Empleado {
    private String codigo;

public void setCodigo (String codigo){
        this.codigo = codigo;
    }
}
```

Formato para implementar un método get:

```
Visibilidad tipoDatos getNombre(){
    Return atributoPrivado;
}
```

Donde:

- *Visibilidad:* Representa el alcance del método get, casi siempre se declara como public ya que este método será invocado desde fuera de la clase que se implementó.
- *TipoDatos:* Es el tipo de datos del valor que devolverá el método, hay que tener en cuenta que el parámetro privado y su método get deben tener el mismo tipo de datos.
- *getNombre:* Es el nombre que se le asigna al método get, casi siempre se inicia con la palabra get seguido del nombre del parámetro privado.
- *atributoPrivado:* Aquí se especifica qué valor devolverá el método, casi siempre se coloca el nombre del parámetro directamente; pero algunos programadores usan el operador this para hacer referencia a un miembro de la clase.

Por ejemplo: Si se tiene la siguiente declaración:

```
public class Empleado {
    private String codigo;
}
```

Se podría implementar el método get del atributo código de una de las siguientes formas:

Primero:

```
public String getCodigo(){
    return this.codigo;
}
```

Segundo:

```
public String getCodigo(){
    return codigo;
}
```

Se usará cualquiera de ellos, ya que todos los métodos mostrados realizan la misma actividad; seguidamente se mostrará el script completo para la clase.

```
public class Empleado {
    private String codigo;

    public String getCodigo(){
        return codigo;
    }
}
```

2.6 Implementación de métodos get y set con Android Studio

Para poder crear los métodos get y set en Android Studio se debe realizar los siguientes pasos:

1. Tener una clase dentro del paquete respectivo de un proyecto móvil.
2. Declarar los atributos de la clase con visibilidad privada como se muestra en el siguiente código:

```
package com.miempresa.personal.myapplication;

public class Producto{
    private int codigo;
    private String descripcion;
    private int categoria;
    private double precioUnitario;
}
```

3. Hacer clic derecho sobre el código de la clase *Refactor > Encapsulate Fields...* y se muestra la siguiente ventana:



De la ventana marque todas las casillas correspondientes a los campos de la clase (Fields), asegúrese de que los checks de Get Access y Set access se encuentren activas; asimismo, la opción private para "Encapsulated Fields Visibility" y la opción public de "Accessors Visibility" deben encontrarse seleccionadas, para finalmente presionar el botón Refactor y visualizar el siguiente código:

```
package com.miempresa.personal.myapplication;

public class Producto{
    private int codigo;
    private String descripcion;
    private int categoria;
    private double precioUnitario;

    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

    public int getCategoria() {
        return categoria;
    }

    public void setCategoria(int categoria) {
        this.categoria = categoria;
    }

    public double getPrecioUnitario() {
        return precioUnitario;
    }

    public void setPrecioUnitario(double precioUnitario) {
        this.precioUnitario = precioUnitario;
    }
}
```

2.7 Método constructor

Un constructor es un método especial dentro de la clase, este se encargará de asignar de valores iniciales a los parámetros privados de la clase. Hay que tener en cuenta que esta actividad solo lo realiza una vez por ejecución, vale decir que si necesita actualizar o modificar el valor asignado por el constructor; tendrá que usar los métodos set.

Formato para la implementación de un método constructor:

```
Visibilidad nombreConstructor(Parámetro){
    atributoPrivado = Parámetro;
}
```

Donde:

- *Visibilidad:* Representa el alcance del método constructor, casi siempre se declara como public ya que este método será invocado desde fuera de la clase que se implementó.
- *nombreConstructor:* Es el nombre asignado al método constructor, para poder ser reconocido como constructor debe tener el mismo nombre de la clase.
- *Parámetro:* Aquí se especifican los parámetros según la cantidad los atributos a inicializar.
- *atributoPrivado=parámetro:* Esta asignación permite enviar el valor del parámetro hacia el atributo privado.

Por ejemplo: Si se tiene la siguiente declaración:

```
public class Empleado {
    private String codigo;
    private String nombres;
    private int edad;
}
```

La implementación del método constructor es de la siguiente forma:

```
public Empleado(String codigo,String nombres, int edad){
    this.codigo=codigo;
    this.nombres=nombres;
    this.edad=edad;
}
```

A continuación se verá la implementación completa de la clase empleado:

```
public class Empleado {
    private String codigo;
    private String nombres;
    private int edad;

    public Empleado(String codigo,String nombres, int edad){
        this.codigo=codigo;
        this.nombres=nombres;
        this.edad=edad;
    }
}
```

Creando un objeto de la clase Empleado usando método constructor:

Primera forma: Enviando los valores al método constructor directamente.

```
Empleado objEmp = new Empleado("E0001","Juan López G.",50);
```

Segunda forma: Enviando variables locales como valores al método constructor.

```
String codigo = getCodigo();
String nombres=getNombres();
int edad=getEdad();
Empleado objEmp = new Empleado(codigo, nombres, edad);
```

Tercera forma: Enviando métodos que capturan valores que son enviados al constructor de la clase.

```
Empleado objEmp = new Empleado(getCodigo(),getNombres(),getEdad());
```

2.8 REFERENCIA THIS

Permite resolver ambigüedades entre los atributos de una clase y los parámetros de los métodos, muchos métodos de un objeto necesitan acceder a los datos miembros de este, y a otras propiedades del objeto. Para hacer referencia explícitamente a una referencia desde el método se utiliza la palabra reservada this.

Formato de referencia this:

```
this.atributo;
this.metodo();
```

Donde:

- *this.Atributo:* Es la referencia a una variable de clase.
- *this.Método:* Es la referencia a un método de clase.

Por ejemplo: Si se tiene el método constructor Empleado que tiene como parámetros el código, nombres y edad; que casualmente tienen el mismo nombre que los atributos privados de la clase, se debe referir a ellos por medio de la referencia this:

```
public Empleado(String codigo, String nombres, int edad) {
    this.codigo=codigo;
    this.nombres=nombres;
    this.edad=edad;
}
```



CAPÍTULO

3

Android Studio y desarrollo inicial de aplicaciones

3.1 Android Studio 2.0

Actualmente, es considerado la interfaz de desarrollo de aplicaciones oficial para Android promocionado por Google. El desarrollo de aplicaciones Android se viene implementando en un conjunto de ID; entre los más usados se tiene al IDE Eclipse y Android Studio. Este último ofrece un entorno integrado de desarrollo con nuevas herramientas.



Entre las principales características que presenta Android Studio es la edición de códigos de primer nivel, la depuración, las herramientas de rendimiento, un sistema de compilación flexible y un sistema instantáneo de compilación e implementación que logran implementar aplicaciones de alta calidad para el usuario.

Cuando se crea un nuevo proyecto en Android Studio el contenido estructural del proyecto aparece dentro de la carpeta SRC. También cuenta con un sistema de emulación integrado, el cual permite visualizar cualquier cambio en la aplicación en tiempo real.

Por otro lado, Android Studio es publicado de forma gratuita a través de la licencia Apache y está disponible para plataformas Windows, Linux y Mac OS X. La última versión de Android Studio es la 2.1.3 lanzada en agosto del 2016.

3.2 Características del Android Studio

Android Studio se distingue de sus competidores por la simpleza y profesionalismo en el desarrollo de aplicaciones. Sus principales características son:

- Cuenta con un conjunto de herramientas de empaquetado y etiquetado de código que permite tener una mejor organización de los bloques de código generados en una aplicación. Así, pareciera que el usuario programara en capas.
- Presenta una renderización de su código en tiempo real, ganando tiempo en el desarrollo de una aplicación móvil.
- Cuenta con un sistema Drag and Drop que permite movilizar los componentes a través de la interfaz de usuario, dando una mejor experiencia en el desarrollo.
- Cuenta con la tecnología Google Cloud Messaging, el cual permite enviar datos desde el servidor a terminales Android a través de la nube.
- Permite la importación de aplicaciones desarrolladas en Eclipse.
- Cuenta con un conjunto de plantillas que permitirán crear aplicaciones bajo diseños comunes de Android.
- Permite previsualizar una aplicación en diferentes dispositivos y resoluciones.

3.3 Máquina virtual

Una máquina virtual es un software que se instala en una computadora, el cual permite simular la ejecución de aplicaciones pertenecientes a otros sistemas operativos. Es decir, cuando se ejecuta una de estas aplicaciones la máquina virtual toma la administración de dicha ejecución y virtualiza a la aplicación. Dicho de otra manera, la máquina virtual permitirá ejecutar varios sistemas operativos de manera simultánea usando la misma plataforma hardware.

La máquina virtual realiza una serie de actividades para que la aplicación pueda ser ejecutada correctamente en cualquier dispositivo móvil:

- Al implementar un código fuente (.java) de la aplicación esta se debe compilar.
- Se analizan los posibles errores sintácticos o de otro tipo para que el usuario pueda corregirlos, ya que, de otra manera, la aplicación no se ejecutará.
- A partir del código fuente se genera un archivo (.class) que contiene un código binario llamado código de bytes, el cual es posible entenderlo solo por la máquina virtual.
- Luego, a partir del código (.class) se pasa por un proceso de optimización que permite comprimir los archivos en bloques de menor capacidad. Esto se da porque estas aplicaciones se ejecutarán en un dispositivo móvil que no cuentan con una gran capacidad de memoria como sí tiene una computadora. Entonces, todos los archivos .class son agrupados o comprimidos para un mejor desempeño de la aplicación frente a un dispositivo móvil.
- Finalmente, todo el código generado a partir de la optimización es enviado al núcleo del sistema operativo para poder ejecutar la aplicación y mostrar los resultados esperados por el usuario.

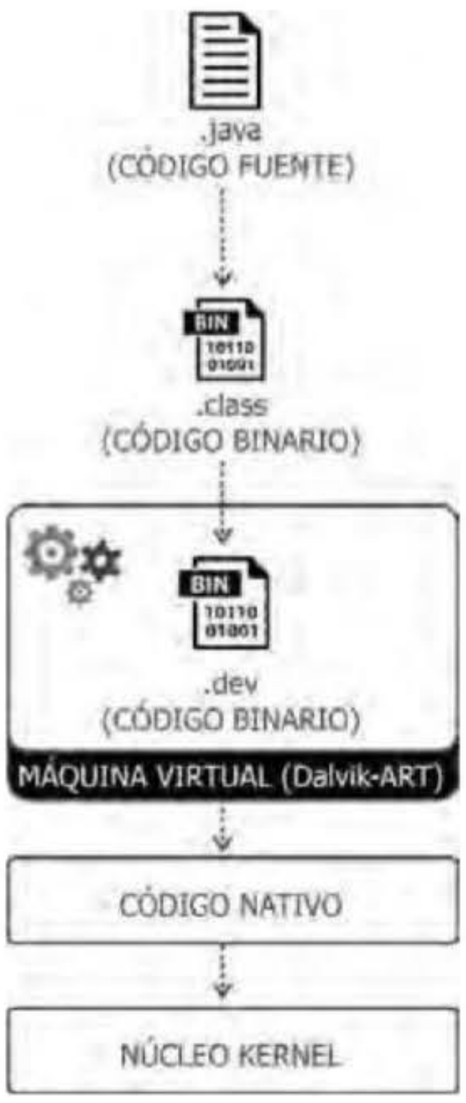


Figura 3.1 Trabajo de la máquina virtual Android.

El sistema de archivos generados por la máquina virtual Android son los siguientes:

Tipo de archivo	Descripción
.dex	Davilk Executable Format (Formato ejecutable para máquinas virtuales Davilk): Es un archivo que contiene código binario comprimido en un archivo de tipo apk. Este archivo tiene la característica de no ser exclusivo para un tipo de dispositivo, sino más bien puede ser interpretado en cada dispositivo que se ejecute, dando así portabilidad al código. En el proceso de interpretación se genera un archivo de tipo ODEX (Máquina Virtual Dalvik) o ELF (Máquina Virtual ART).
.odex	Optimized DEX (Archivo DEX optimizado): Es un archivo proveniente del .dex que contiene un conjunto de sentencias que solo son interpretados en la Máquina Virtual Dalvik. Al iniciar una aplicación en cualquier dispositivo automáticamente se traduce dinámicamente dicho archivo y lo registra en un archivo de tipo .odex; se le llama dinámico porque a medida que va traduciendo lo va guardando en el archivo .odex.
.elf	Executable and Linkable Format (Formato ejecutable y enlazable): Es un archivo proveniente del .dex que contiene un conjunto de sentencias que solo son interpretadas por la Máquina Virtual ART. La mejora que se realiza es en la forma en que se guarda la traducción del archivo .dex realizándolo una sola vez; mientras que .odex va traduciendo y va registrando.

3.3.1 Máquina virtual ART (Android Runtime)

Es la nueva máquina virtual del sistema operativo el cual cumple las funciones clásicas de la máquina virtual de Java (JVM), con algunas diferencias por el hecho de compilar aplicaciones para dispositivo móviles.



Figura 3.2 Trabajo de la máquina virtual ART.

En la máquina virtual ART existe un archivo llamado .dex2aot que permite convertir un archivo .dex en un archivo compilado .elf, usando la tecnología Ahead of Time (Antes de Tiempo). Esto permite entender la aplicación y mostrarla al usuario en un emulador. Gráficamente, para ejecutar una aplicación móvil se deberían seguir las siguientes actividades:

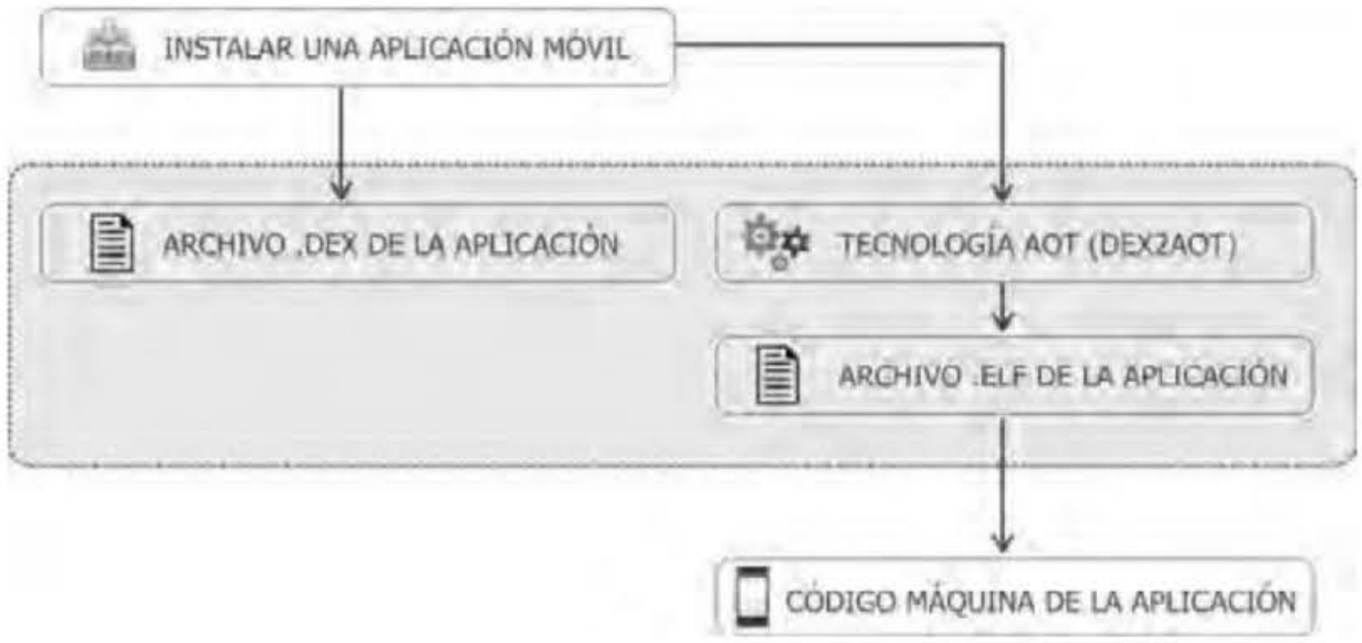


Figura 3.3 Actividades al ejecutar una aplicación móvil

Finalmente, se mencionan las características de la máquina virtual ART:

- Consume menos tiempo al ejecutar una aplicación, ya que no crea máquinas virtuales por cada aplicación.
- Mejora el rendimiento en la ejecución de las aplicaciones móviles, ya que administra mejor el garbage collector y el mismo sistema en sí.
- Compila un archivo .dex en un archivo .elf eficientemente, ahorrando tiempo en el compilador, ya que precarga un conjunto de clases en la memoria.
- Existe una compatibilidad con la máquina virtual Dalvik que fue nativa de Android.

3.3.2 Tecnología AOT (AHead Of Time)

Su función principal es convertir un código de bytes (Bytecode) en código máquina para sea entendible por el sistema operativo Android y este ejecute la aplicación. Todo este trabajo se realiza usando Ahead Of Time, es decir, Antes de Tiempo, a diferencia de la máquina virtual que usa Just And Time (Justo a Tiempo). Entre los principales beneficios que tiene al ejecutarse antes de tiempo están:

- Menor tiempo de ejecución de las aplicaciones.
- Menor consumo de recursos del procesador, memoria RAM y espacio en disco.
- Menor consumo de energía en los dispositivos móviles.

3.4 Requerimientos mínimos para la instalación de Android Studio

Los requerimientos mínimos para una buena performace de Android Studio en las diferentes plataformas de trabajo son:

- Memoria: 2 GB de RAM o superior.
- Disco duro: 500 MB o superior.
- 1.5 GB o superior para la instalación de Android SDK.
- Resolución: 1280 × 800 píxeles por pulgada.
- JDK - Java Development Kit 7 o superior.

Adicionalmente, se debe tener en cuenta:

- Cuando use Linux debe contar con la librería C GNU conocido como glibC o posterior.
- Cuando use MacOS X debe tener instalado el Java Runtime Environment (JRE), mínimo la versión 6.0.

3.5 Proceso de instalación de Android Studio

Para iniciar la instalación, primero se debe instalar el JDK en su computadora, para luego instalar el Android Studio 2:

Software	Capacidad en disco	Url de descarga
jdk-8u101-windows-x64.exe	193 MB	http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html?ssSourceSiteId=otnes
android-studio-bundle-143.3101438-window,..	1.23 GB	https://developer.android.com/studio/index.html

Pasos para la instalación:

Paso 1: Dentro de la instalación del Android Studio, una vez ya instalado el JDK en su computadora. Al iniciar la misma se nos presenta la pantalla de bienvenida de instalación del Android Studio, aquí solo debe seleccionar el botón Next:



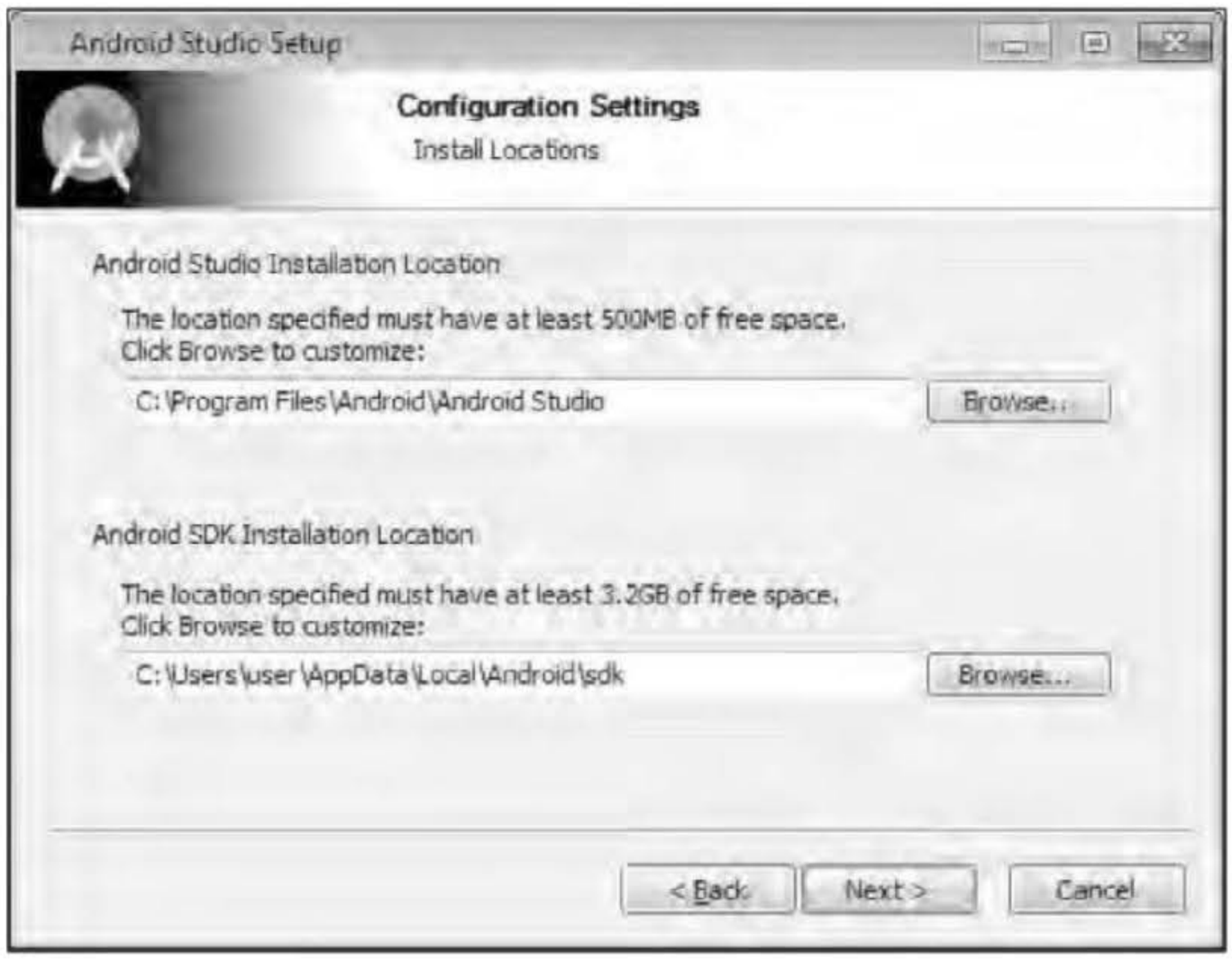
Paso 2: La siguiente ventana hace referencia a la selección de los componentes que necesita instalar. Solo mantenga seleccionadas las opciones Android SDK y el Android Virtual Device:



Paso 3: Se presenta esta ventana, aquí solo seleccione el botón I Agree para continuar con el proceso de instalación:



Paso 4: La siguiente ventana corresponde a la de configuración de la instalación, donde debe seleccionar una ubicación de donde se registrarán todos los archivos del Android Studio. Por ahora solo se observarán los valores predeterminados que muestra el asistente de instalación:



Paso 5: Antes de comenzar con la instalación de los componentes el cuadro pregunta por lo íconos de acceso directo que genera la aplicación. Para este caso se va a mantener la configuración como se muestra en el asistente. Luego para dar inicio a la instalación seleccione el botón Install:



Paso 6: El proceso de instalación se inicia con la descarga de los archivos necesarios para la aplicación y termina con la creación de los íconos de acceso directo. Cuando finalice la instalación se mostrará un mensaje de Installation Complete. Aquí debe seleccionar el botón Next para mostrar la última ventana de instalación:



Paso 7: Finalmente, se muestra la ventana de Completing Android Studio Setup, el cual indica que toda la instalación se desarrolló con normalidad y que no hay ningún error que impida que la aplicación funcione correctamente. Aquí debe seleccionar el botón Finish y, si ya quiere empezar a usar la aplicación Android Studio, active el check Start Android Studio:



3.6 Descripción de las pantallas iniciales de Android Studio

Actualmente es considerado la interfaz de desarrollo de aplicaciones oficial para Android promocionado por Google. El desarrollo de aplicaciones Android se viene implementando en un conjunto de ID; entre los más usados está el IDE Eclipse y Android Studio, este último ofrece un entorno integrado de desarrollo con nuevas herramientas.

Si es la primera ejecución del Android Studio, se presentarán las siguientes pantallas:

PANTALLA DE INICIO DEL ANDROID STUDIO



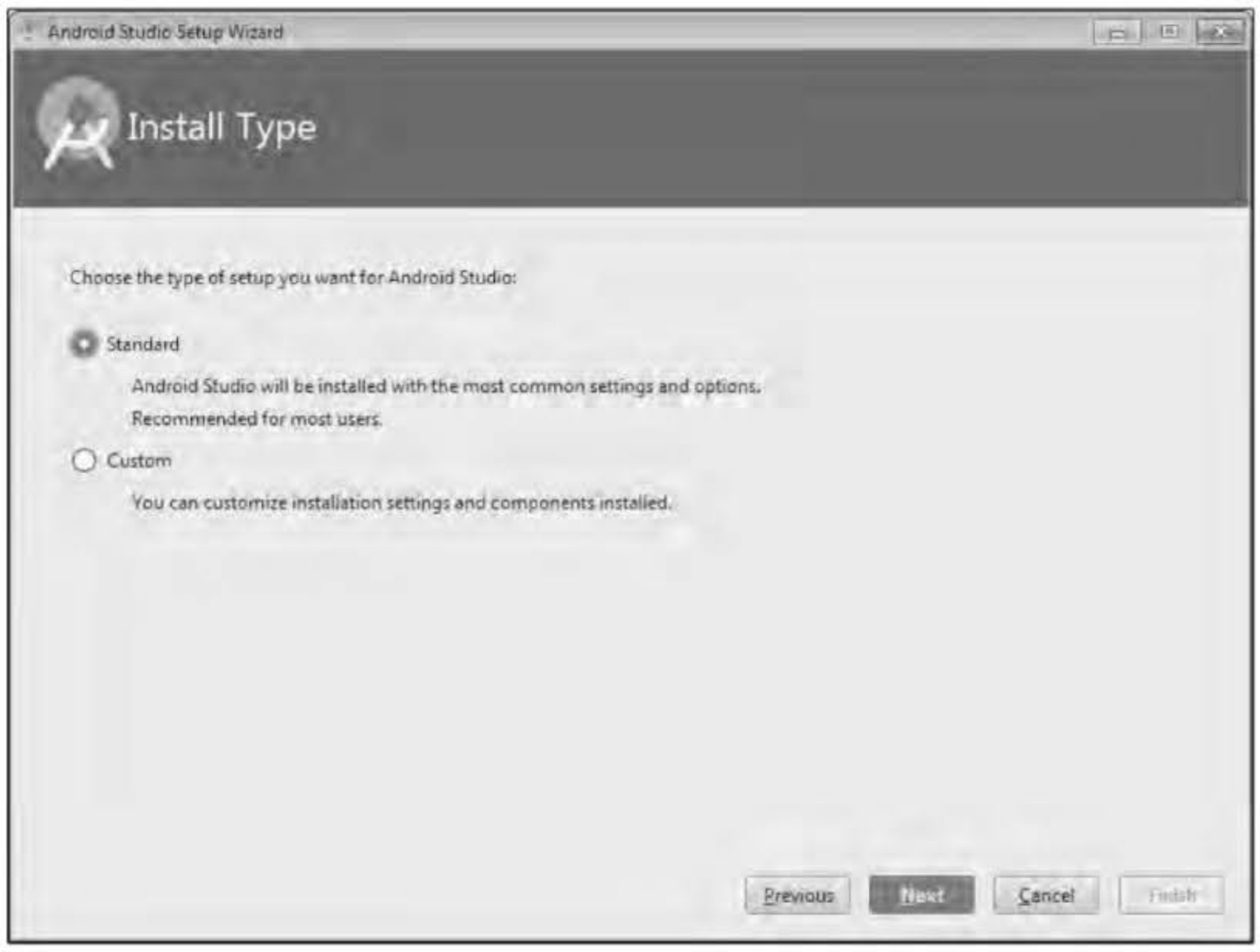
Paso 1: Si ya había tenido una experiencia con una versión anterior a Android Studio 2.0, entonces se muestra una ventana que indica que puede importar la configuración. Para este caso seleccione la opción I do not have a previous versión of Studio y presione el botón Ok.



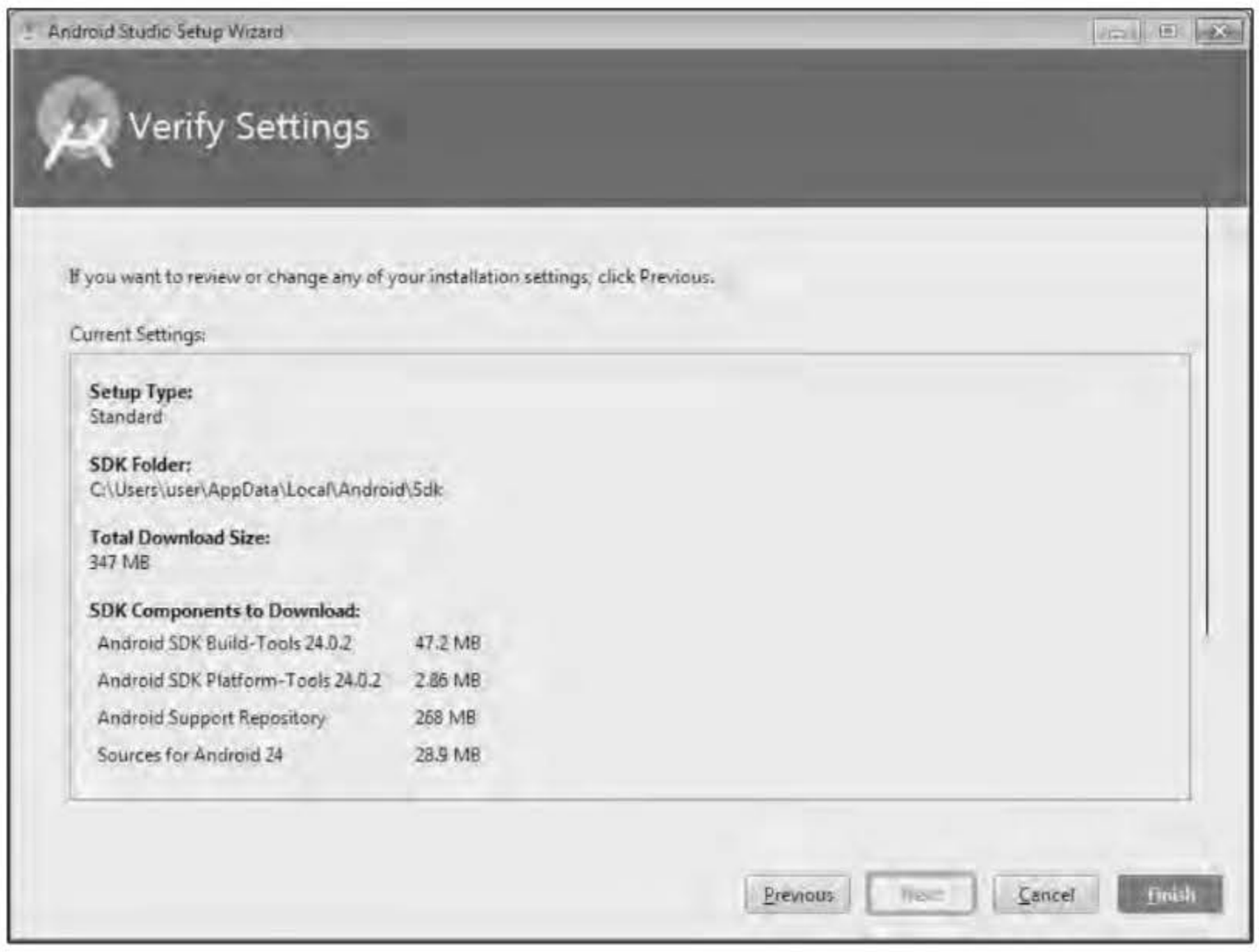
Paso 2: La siguiente ventana en presentarse es la de bienvenida, desde el cual solo debe presionar el botón Next:



Paso 3: A continuación, debe seleccionar un tipo de instalación. Estos tipos de aplicaciones ya vienen pre-configuradas con lo necesario para iniciar con las aplicaciones, así es que, para este caso, seleccione la opción Standar y el botón Next para continuar.



Paso 4: A continuación, se muestra una estadística de las configuraciones que se realizó de forma estándar como la ubicación del Software Development Kit (SDK) de Android, el tamaño total usado para la configuración y los tamaños de los componentes mínimos para descargar. Ahora presione el botón Finish para iniciar la descarga de dichos componentes.



Paso 5: El proceso de descarga de componentes suele demorar un poco de tiempo, por eso debe esperar, ya que, a parte de descargar los componentes necesarios, los está instalando.



Una vez finalizada la descarga e instalación de los componentes presione el botón Finish y por fin comenzará con el desarrollo de sus aplicaciones Android usando el IDE Android Studio 2.0.

3.7 Android SDK

Una aplicación Android cuenta con muchos elementos que no son reconocidos a simple vista, pues el desarrollador usará sus controles en una aplicación, pero, internamente, hay una plataforma que da soporte a dichos componentes, que es el SDK (Software Development Kit). Google ofrece de forma gratuita el SDK oficial, el cual cuenta con una serie de drivers, herramientas y recursos diversos para iniciar el desarrollo de aplicaciones con Android.

SDK incluye un conjunto de herramientas que permitirán crear aplicaciones móviles con Android, el cual incluye:

- Servicios y las API para el funcionamiento del dispositivo.
- Depurador de aplicaciones.
- Emulador para testear las aplicaciones.
- Documentación sobre los componentes Android.

Una característica de Android es que al momento de aparecer una nueva versión, Google libera el código fuente y publica el SDK con la nueva versión de Android. Esto sirve para que los desarrolladores puedan adaptar sus aplicaciones a la nueva versión.

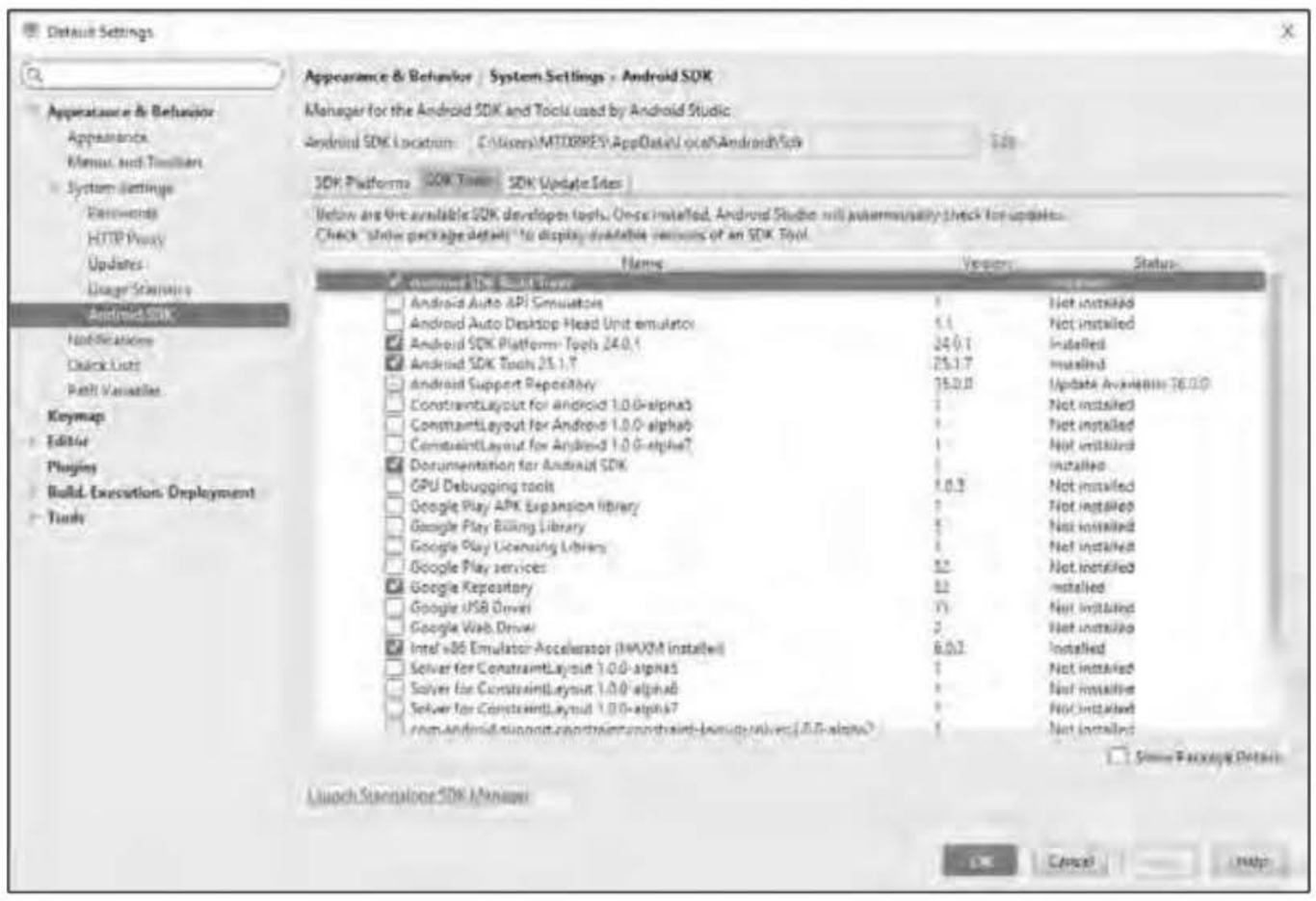
En Android Studio no pueden estar habilitadas todas las características que presenta el SDK, pero sí se puede activar para su uso con los siguientes pasos:

Paso 1: *Tools > Android > SDK Manager*

Paso 2: De la siguiente ventana, la ficha (SDK Plataforms) muestra una lista de plataformas que se pueden agregar al Android Studio para su desarrollo. Verá que algunos presentan un estado de Not installed y que pueden ser instalados desde esta ventana.



Paso 3: Luego, en la ficha SDK tools se presentan un conjunto de herramientas propias del SDK que no fueron habilitadas y que solo serán instaladas cuando el desarrollador necesite de ellas. Solo debe indentificar en qué estado se encuentran.



3.8 AVD Manager

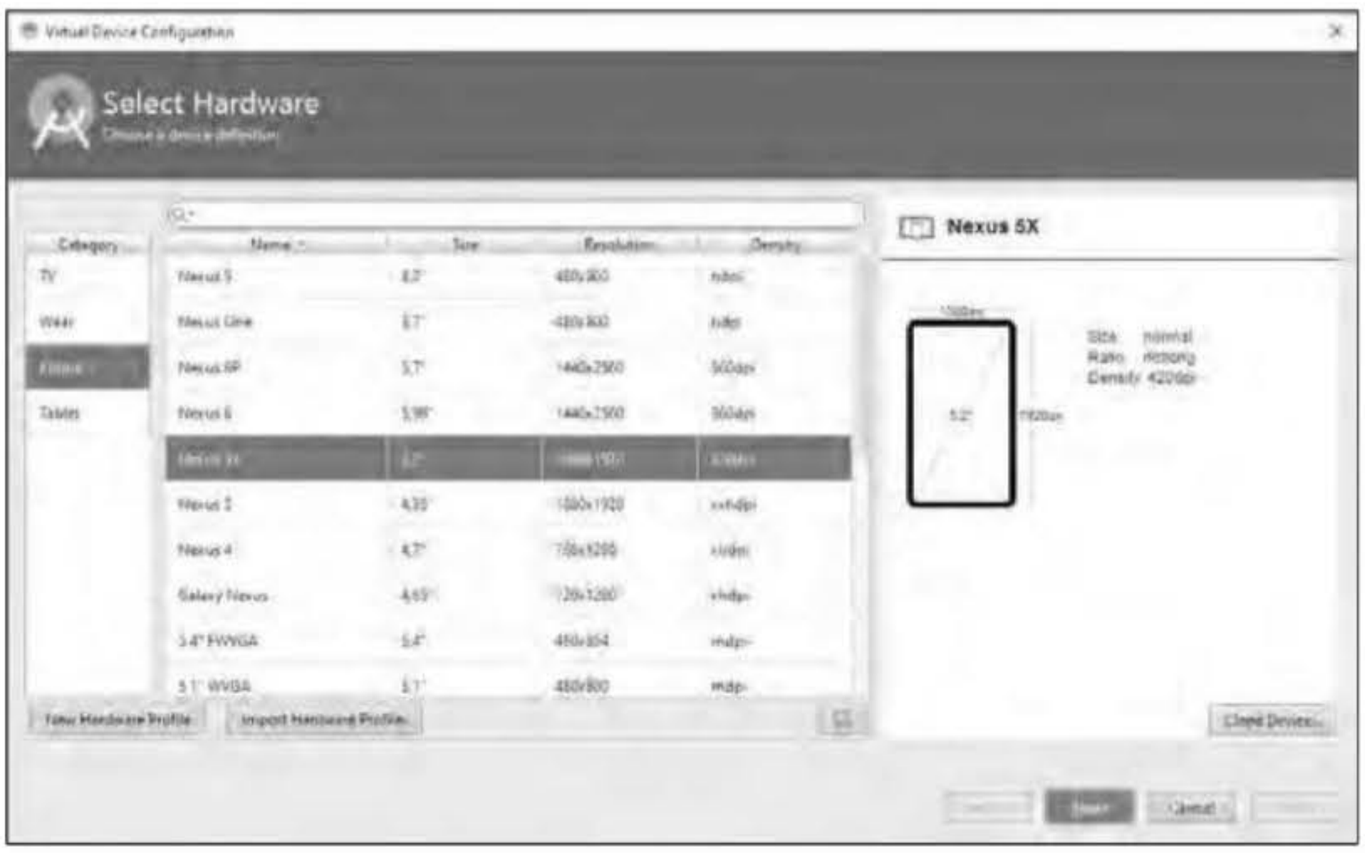
Representa al dispositivo virtual de Android. Se hace esta especificación para realizar las pruebas de las aplicaciones en un emulador. Desde aquí puede realizar todas las pruebas correspondientes a una aplicación sin la necesidad de un dispositivo físico.

Paso 1: *Tools > Android > AVD Manager*

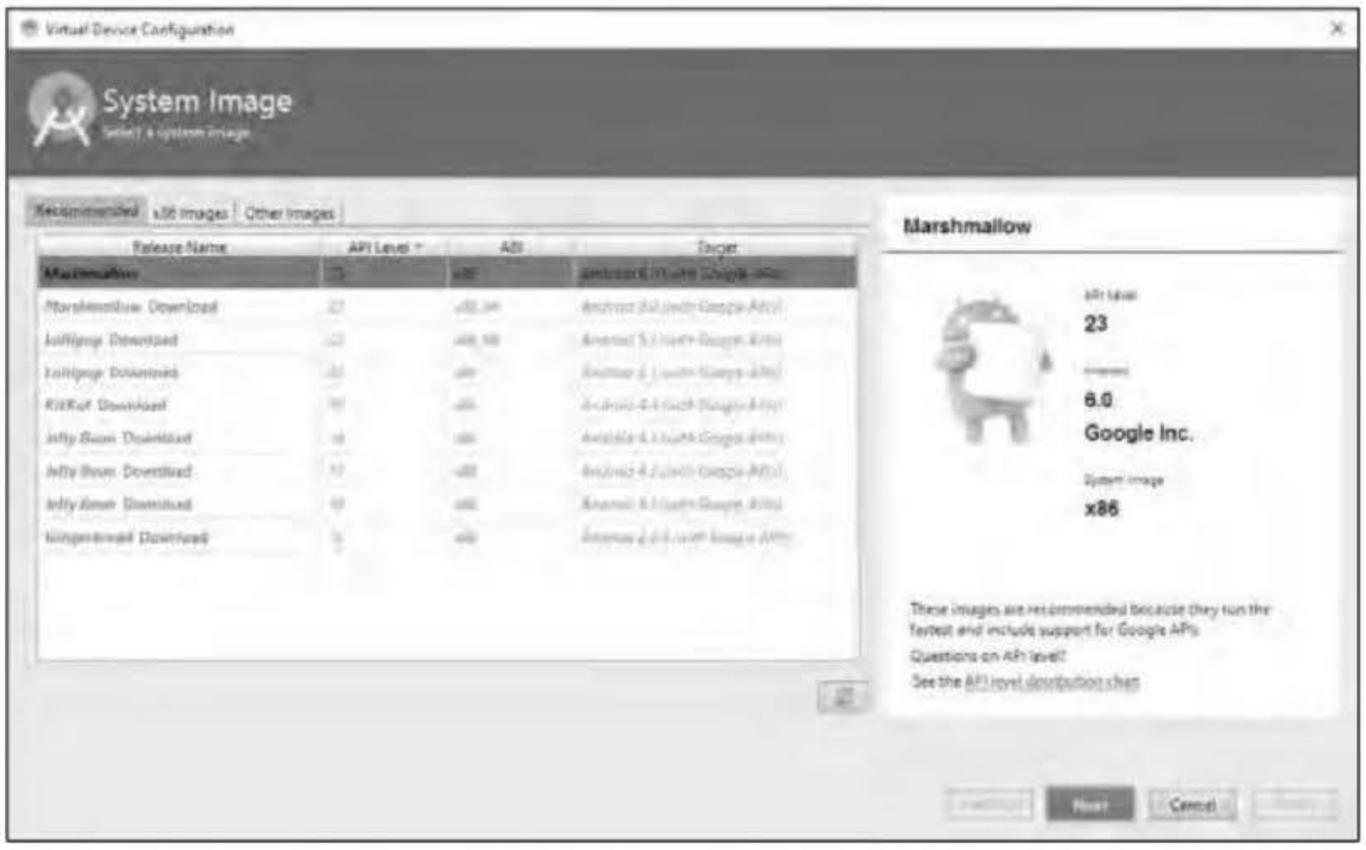
Paso 2: La primera vez que se invoca se mostrará la ventana Android Virtual Device Manager. Desde aquí podrá seleccionar el tipo de dispositivo que representará el emulador.



Paso 3: Al seleccionar el botón Create Virtual Device se muestra la ventana de seleccionar un dispositivo. Se presenta una lista de dispositivos según su categoría, ya sea para televisión, relojes, teléfonos o tablet. Se recomienda que seleccione el dispositivo que se marca por defecto. Para este caso se ha seleccionado Nexus 5x de 5.2 pulgadas con una resolución de 1080 x 1920 y 420 de densidad. Luego, debe seleccionar el botón Next para realizar la instalación.

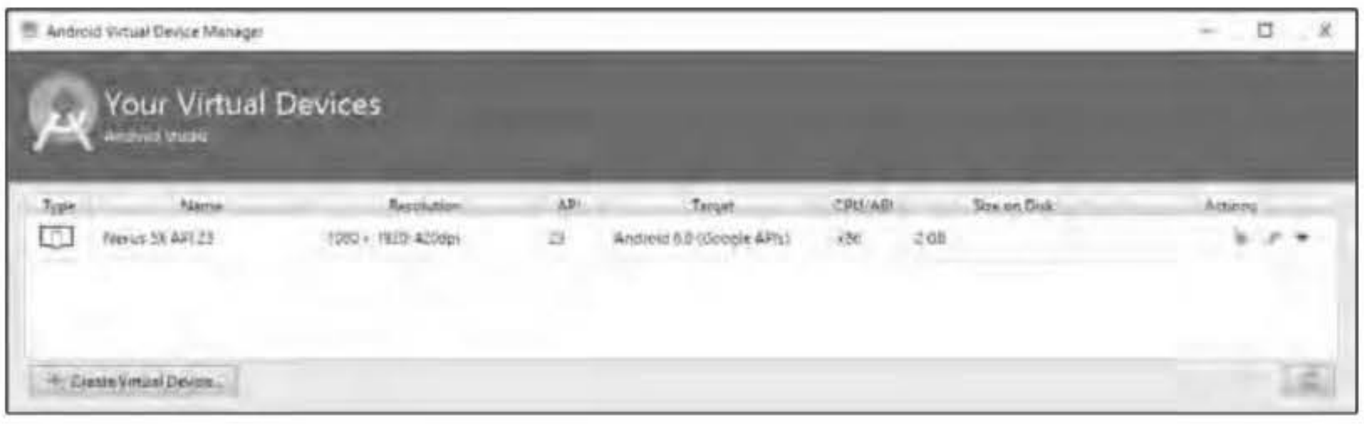


Paso 4: En la ventana System Image debe seleccionar el sistema operativo con el que trabajará el AVD. Para este caso, trabaje con el que se encuentra seleccionado por defecto, Marshmallow y, finalmente, presione el botón Next para realizar las últimas configuraciones al dispositivo.



Paso 5: Si usted decide mantener la configuración inicial, entonces presione, el botón Finish para terminar la instalación y configuración del AVD. Finalmente, se mostrará la ventana con el dispositivo instalado. Se puede instalar muchos dispositivos sin ninguna restricción.





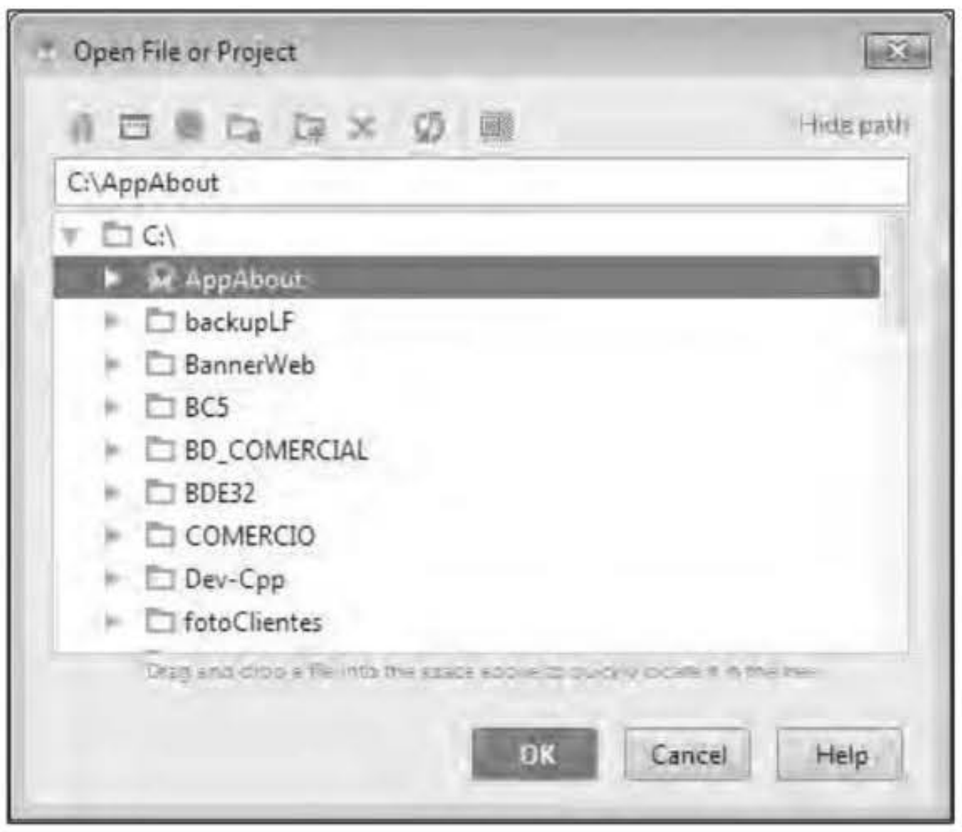
3.9 CREANDO UN NUEVO PROYECTO

Al iniciar el IDE Android Studio siempre mostrará la siguiente ventana inicial:



Desde aquí puede se pueden realizar las siguientes acciones:

- *Crear una nueva aplicación*: Seleccione Start a new Android Studio project.
- *Abrir una aplicación Android existente*: Seleccione Open an existing Android Studio project, luego esta opción mostrará la ventana de apertura de proyecto para buscar y abrir el proyecto. Normalmente, estos archivos presentan un ícono similar al de acceso directo al Android Studio, como se muestra en la siguiente imagen:



- *Configurar el entorno del Android Studio*: Seleccione *Configure > Settings* y realice las modificaciones que crea conveniente en apariencia, color, etc.



Pasos para crear un nuevo proyecto:

Paso 1: Seleccione la opción Start a new Android Studio Project.



Paso 2: Asigne un nombre a la aplicación, por ejemplo, AppAbout. Se recomenda iniciar con una letra mayúscula el nombre de la aplicación.

Paso 3: Adicionalmente, puede asignar un nombre de dominio de la empresa donde se está desarrollando la aplicación, aquí podría dejarlo con el valor por defecto o colocar el nombre de su organización.

- *Configurar el entorno del Android Studio*: Seleccione *Configure > Settings* y realice las modificaciones que crea conveniente en apariencia, color, etc.



Pasos para crear un nuevo proyecto:

Paso 1: Seleccione la opción Start a new Android Studio Project.

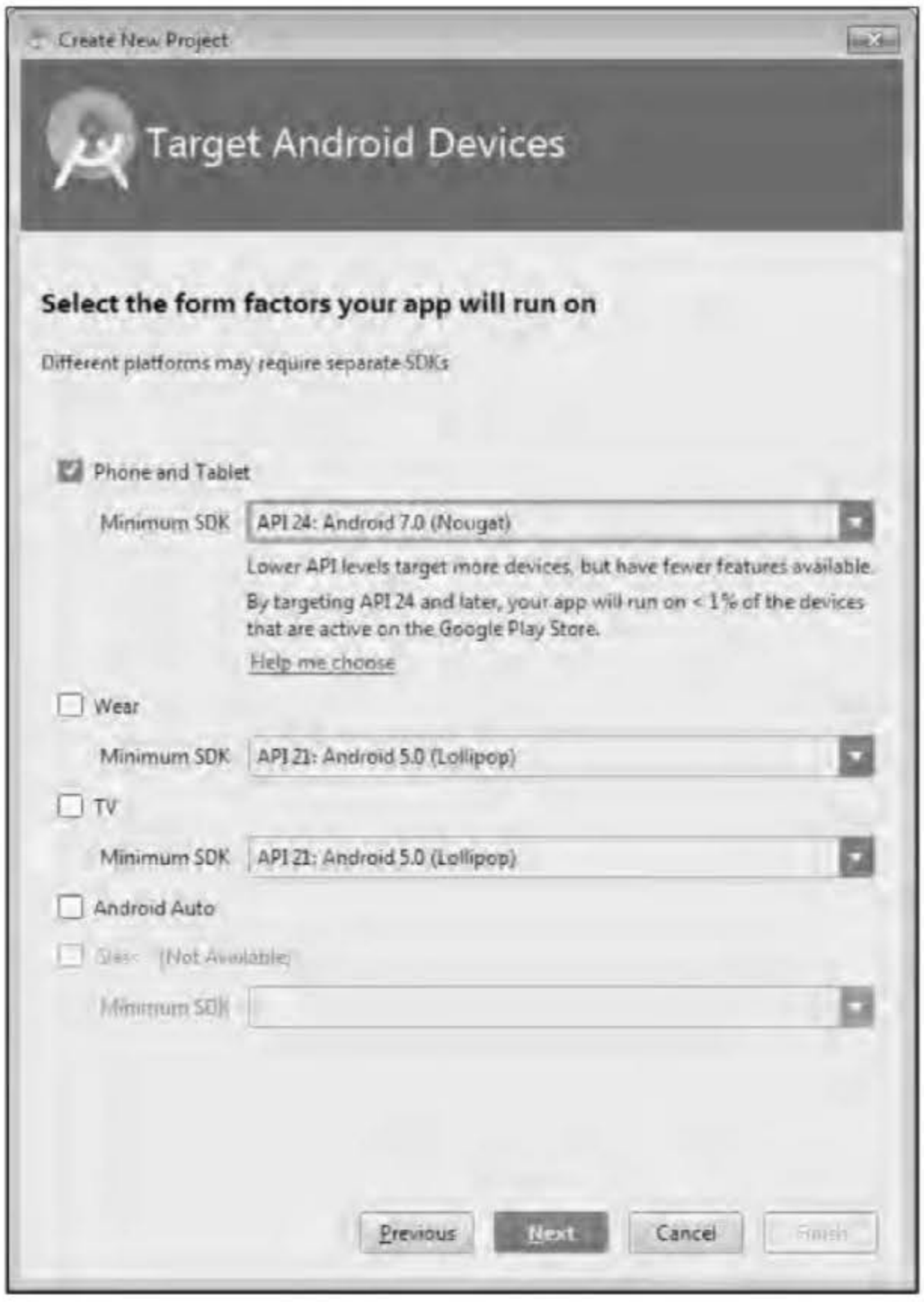


Paso 2: Asigne un nombre a la aplicación, por ejemplo, AppAbout. Se recomenda iniciar con una letra mayúscula el nombre de la aplicación.

Paso 3: Adicionalmente, puede asignar un nombre de dominio de la empresa donde se está desarrollando la aplicación, aquí podría dejarlo con el valor por defecto o colocar el nombre de su organización.

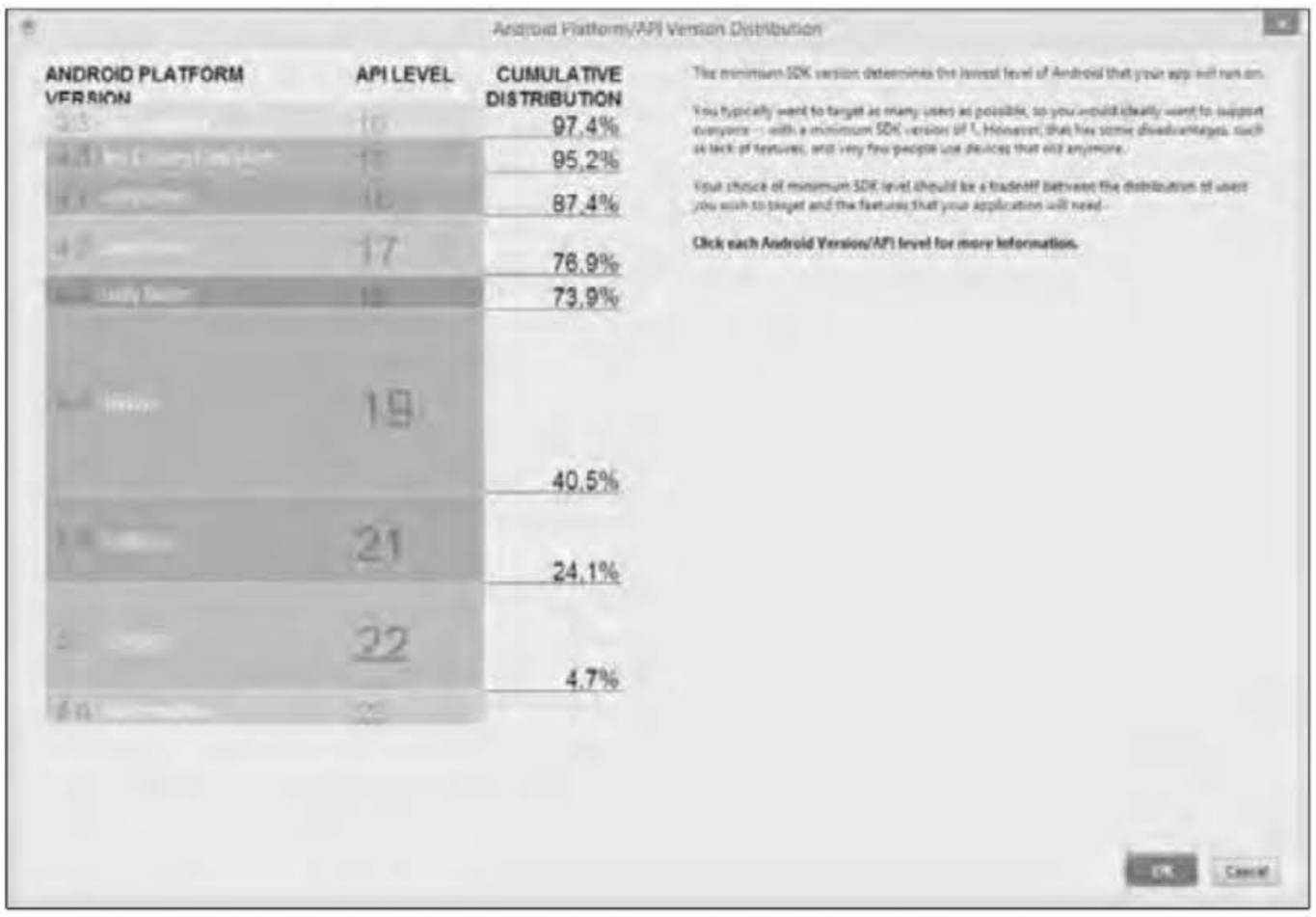
Paso 4: Asigne una ubicación al proyecto, observe que de forma predeterminada aparece la carpeta AndroidStudioProjects y es aquí donde se guardarán todos los proyectos realizados en Android Studio si no asigna una ubicación diferente.

Paso 5: Una vez ingresados todos los datos solicitados por la ventana de creación del nuevo proyecto, presione el botón Next para seleccionar un tipo de dispositivo de salida. Esto con el fin de emular las aplicaciones móviles.

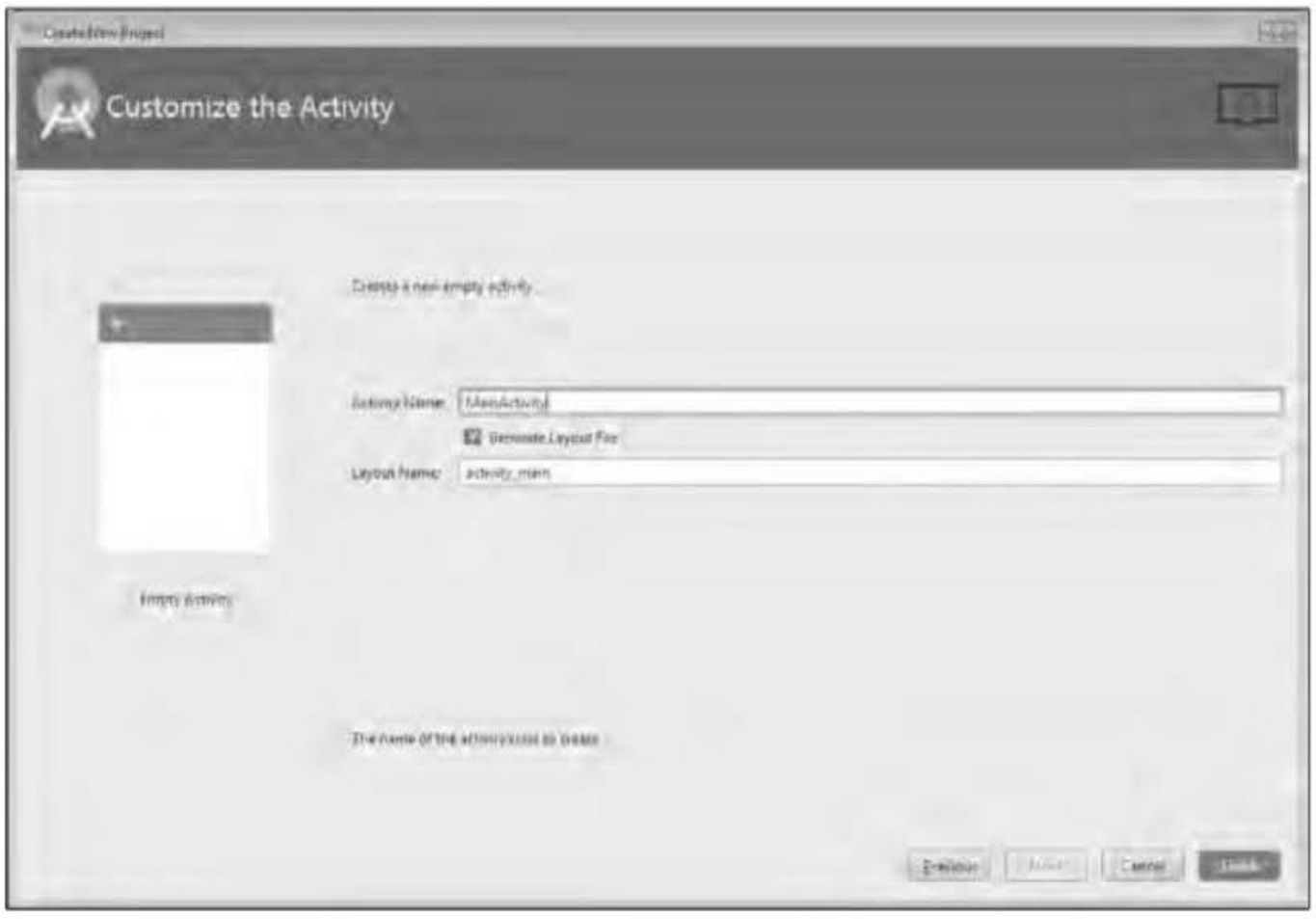


Paso 6: De la ventana anterior, debe seleccionar una plataforma de trabajo para las aplicaciones móviles. En este caso seleccione Phone and Tablet ya que las aplicaciones que implementará estarán basadas en dispositivos móviles como celulares o tabletas. Allí mismo, debe seleccionar el mínimo requerimiento de SDK que presentará las aplicaciones, para esto seleccione la última versión del sistema operativo Android llamado Android Nougat. Luego, presione Next para continuar con el nuevo proyecto de aplicación.

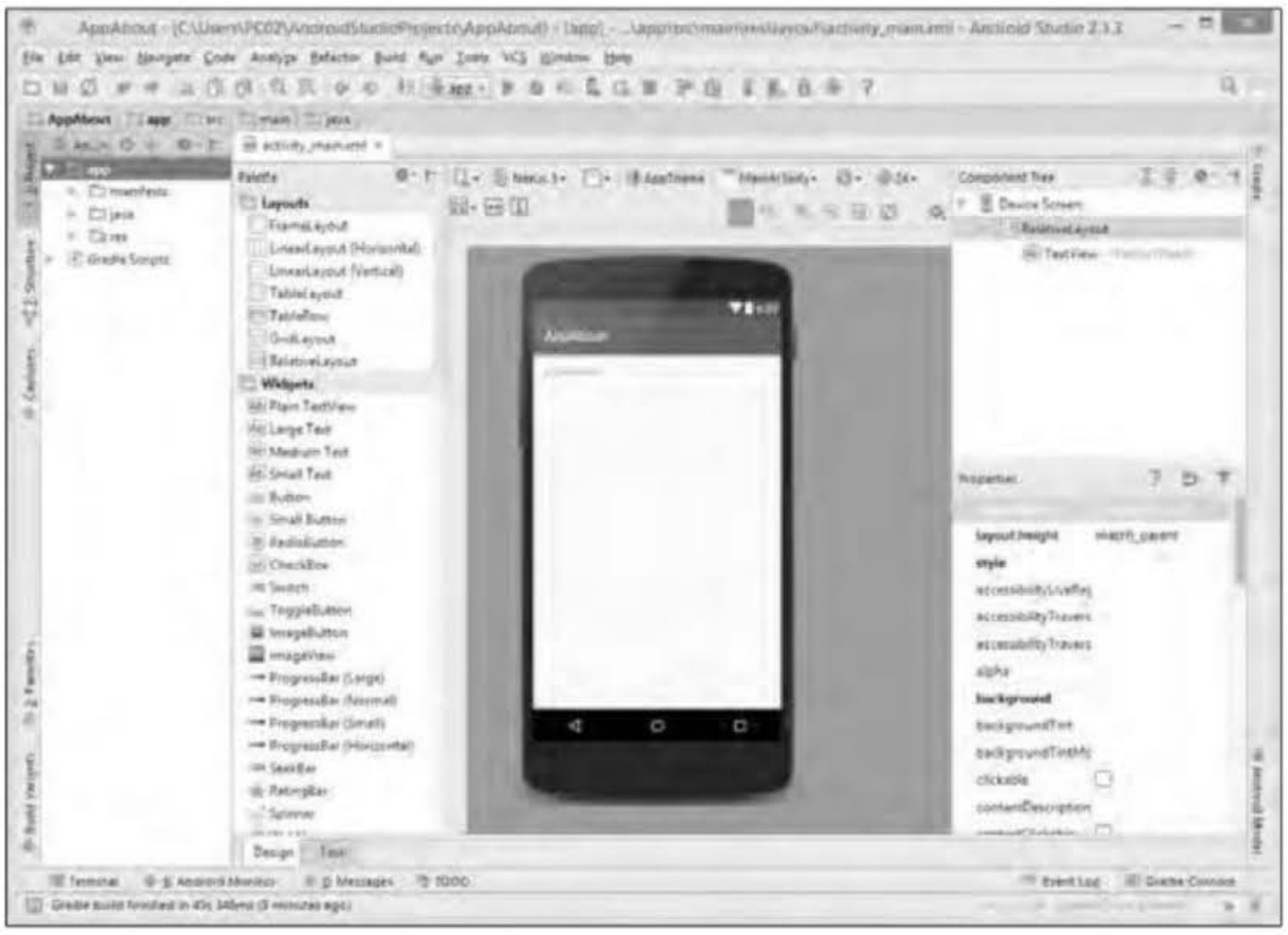
En la ventana también se presenta un link, Help me choose, que permite visualizar una estadística de distribución de uso de las diferentes plataformas Android. Como verá, la selección que realiza no aparece en la estadística porque a la fecha recién se ha incorporado la plataforma Nougat de Android.



Paso 7: Ahora seleccionará un modelo de actividad; para esto Android Studio muestra un conjunto de plantillas. Para este caso seleccionará una actividad vacía, Empty Activity, para iniciar una aplicación desde cero. Luego, debe personalizar dicha actividad agregándole un nombre. En este caso se va a mantener el nombre de la actividad como se visualiza en la siguiente ventana:



Paso 8: Finalmente, visualizará el entorno inicial de la aplicación móvil. Debe tener en cuenta que la aplicación ya ha creado una aplicación móvil, Hola mundo, y es a partir de aquí que debe empezar a trabajar la aplicación, tal como se muestra en la siguiente imagen:



3.10 Descripción de la pantalla principal Android Studio

Se describirán algunas características básicas para el manejo del entorno que presenta Android Studio ante una aplicación móvil.

3.10.1 Paleta de proyectos (Project)

Esta paleta presenta la estructura de una aplicación en diferentes formas; la que se maneja ahora es la de Android y presenta la siguiente estructura:

Acceso directo: <Alt+1>

Menú de opciones: *View > Tool Windows > Projects*



<table>
<tr><th>Componente</th><th>Descripción</th></tr>
<tr><td>AndroidManifest.xml</td><td>Contiene información referente a la aplicación Android como el nombre del paquete de la aplicación, datos propios de la aplicación (íconos, estilos, etc.), así como los datos de la actividad (Activity). A continuación, el código que compone dicho archivo:

<pre>
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/
android"
    package="com.miempresa.appabout">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.
            LAUNCHER" />
        </intent-filter>
    </activity>
 </application>

</manifest>
</pre>
</td></tr>
<tr><td>MainActivity</td><td>Es una clase Java que contiene información propia de la actividad. El código que presenta inicialmente se muestra a continuación:

<pre>
package com.miempresa.appabout;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
</pre>
Inicialmente, no muestra mucho código por el hecho de haber seleccionado una actividad vacía al momento de crear la aplicación, conforme se añada más actividad al proyecto verá que este archivo crecerá en líneas de código.</td></tr>
</table>

COMPONENTE	DESCRIPCIÓN

ApplicationTest	Es una clase Java pensada para insertar código testeador de la aplicación. Inicialmente, el código que se presenta es el siguiente:
	`package com.miempresa.appabout;`
	`import android.app.Application;`
	`import android.test.ApplicationTestCase;`
	`public class ApplicationTest extends ApplicationTestCase<Application> {`
	`  public ApplicationTest() {`
	`  super(Application.class);`
	`  }`
	`}`

drawable	Es una carpeta en la cual se almacenan archivos de tipo imágenes como JPG, PNG, etc.
layout	Es una carpeta en la cual se almacenan archivos de tipo XML con vistas de la aplicación; esto permitirá configurar las diferentes pantallas que presentará la aplicación. Por ejemplo, para visualizar el contenido de una actividad bastará hacer doble clic sobre una actividad y el resultado será como sigue: 
mipmap	Es una carpeta en la cual se recomienda almacenar todos los íconos que se usarán en la aplicación. Es así que tiene la misma finalidad que la carpeta res y drawable.

COMPONENTE	DESCRIPCIÓN

values	Es una carpeta en la cual se almacenan archivos con valores que se usarán en la aplicación. Por ejemplo, la definición de los colores que usará se registran dentro del archivo colors.xml, tal como se muestra en el siguiente código:
	`<?xml version="1.0" encoding="utf-8"?>`
	`<resources>`
	`<color name="colorPrimary">#3F51B5</color>`
	`<color name="colorPrimaryDark">#303F9F</color>`
	`<color name="colorAccent">#FF4081</color>`
	`</resources>`
	También, la definición de las dimensiones, como se muestra en el siguiente código:
	`<?xml version="1.0" encoding="utf-8"?>`
	`<resources>`
	`<color name="colorPrimary">#3F51B5</color>`
	`<color name="colorPrimaryDark">#303F9F</color>`
	`<color name="colorAccent">#FF4081</color>`
	`</resources>`
	Y, por último, la definición del estilo o temas que puede tener la aplicación se muestra en el siguiente código:
	`<resources>`
	`<!-- Default screen margins, per the Android Design guide-lines. -->`
	`<dimen name="activity_horizontal_margin">16dp</dimen>`
	`<dimen name="activity_vertical_margin">16dp</dimen>`
	`</resources>`
	También, la definición de las cadenas usadas en la aplicación:
	`<resources>`
	`<string name="app_name">AppAbout</string>`
	`</resources>`
	
	`<resources>`
	`<style name="AppTheme" parent="Theme.AppCompat.Light.Dark-ActionBar">`
	`<!– Customize your theme here. –>`
	`<item name="colorPrimary">@color/colorPrimary</item>`
	`<item name="colorPrimaryDark">@color/colorPrimaryDark</item>`
	`<item name="colorAccent">@color/colorAccent</item>`
	`</style>`
	`</resources>`

COMPONENTE	DESCRIPCIÓN
Gradle Scripts	Es una carpeta que almacena una serie de archivo de tipo Gradle cuya función es compilar y construir una aplicación. El archivo más importante es build.gradle(Module:app), ya que aquí se muestran los valores de la compilación de la aplicación:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.2"

    defaultConfig {
        applicationId "com.miempresa.appabout"
        minSdkVersion 24
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguard-
File('proguard-android.txt'),
            'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:24.2.0'
}
```

3.10.2 AndroidManifest

Cuando se crea un proyecto nuevo se realiza una serie de pasos para mostrar la aplicación básica antes de agregarle código. En estos pasos se realizan ciertas configuraciones que son registradas en el archivo AndroidManifest.xml, cuya composición de código se puede ver a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mtorres.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Entonces, un archivo AndroidManifest.xml contiene información elemental de la aplicación, como los activitys, intents, librerías, nombre de la aplicación, dispositivos hardware tanto para el programador como para el usuario, así como los permisos para un buen funcionamiento de la aplicación.

3.10.3 Características del AndroidManifest

Todas las características hacen referencia al funcionamiento de la aplicación móvil, por lo tanto, se debe tener claro que dicho archivo es administrarle por el desarrollador. A continuación, sus principales características:

- Son generados automáticamente al crear un proyecto Android.
- Se puede administrar por medio de código nativo de la aplicación.
- Se puede administrar de manera gráfica dependiendo del IDE que utilice; en este caso se usa Android Studio que sí permite la administración gráfica mediante ventanas de configuración.
- Describe todos los componentes que tendrá la aplicación; esto permite que el sistema operativo determine cómo se ejecutarán las aplicaciones.
- Especifica los permisos para el uso de las API en una aplicación.

3.10.4 Elementos que componen un AndroidManifest

Es importante conocer algunos atributos que tiene el archivo androidManifest, ya que estos determinan la configuración inicial de su aplicación. Aquí, una explicación fragmentada del archivo androidManifest.xml

Etiqueta manifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.mtorres.myapplication">
```

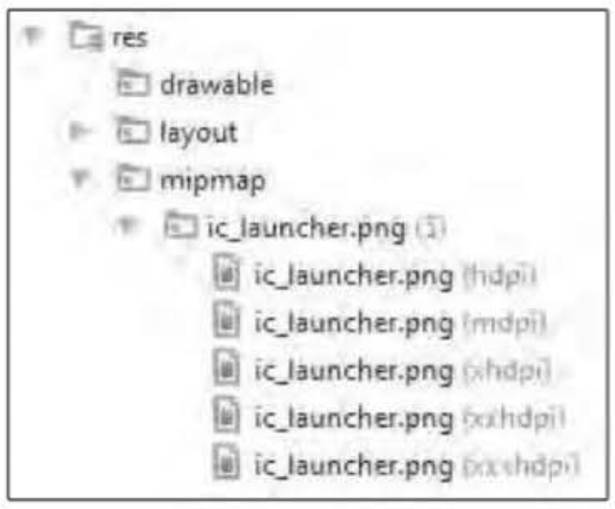
Al iniciar la etiqueta <manifest> se hace referencia al xmlns:android, el cual define el espacio de nombres de Android, que se obtiene desde la url mostrada en el código; normalmente este no debe ser modificado. También hace referencia al package, el cual especifica la ubicación de la aplicación y, asimismo, actúa como identificador de la aplicación, el cual no puede ser modificado una vez que se ha creado. Como opción, se pueden especificar las versiones de Android para desarrollador y usuario con los siguientes parámetros:

android:versionCode="1"
android:versionName="1.0"

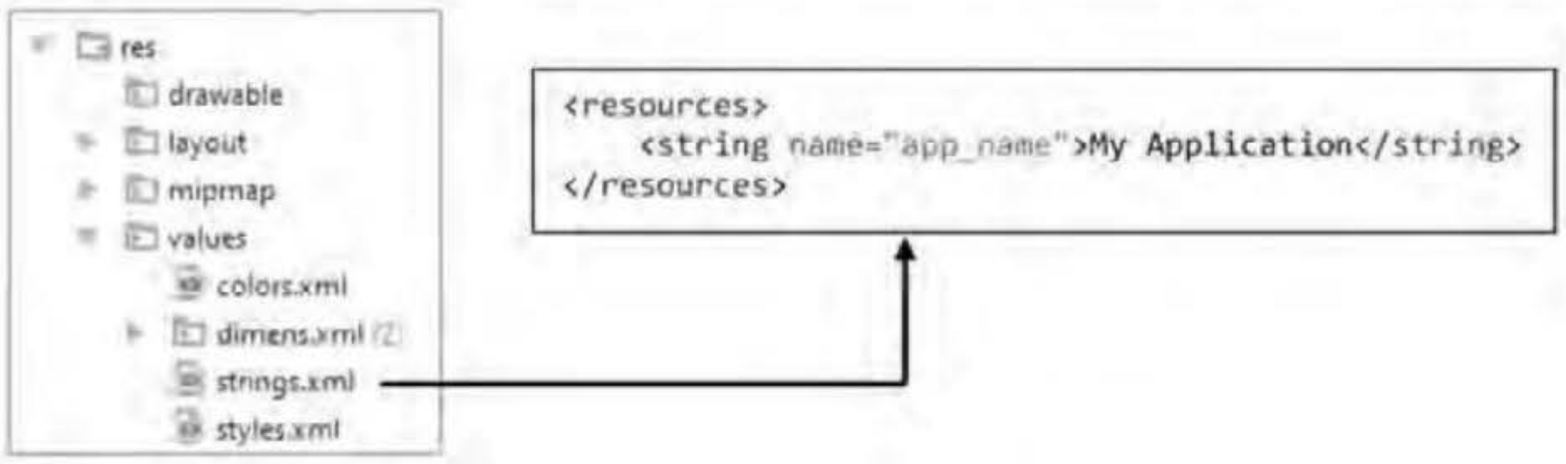
Etiqueta application

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
```

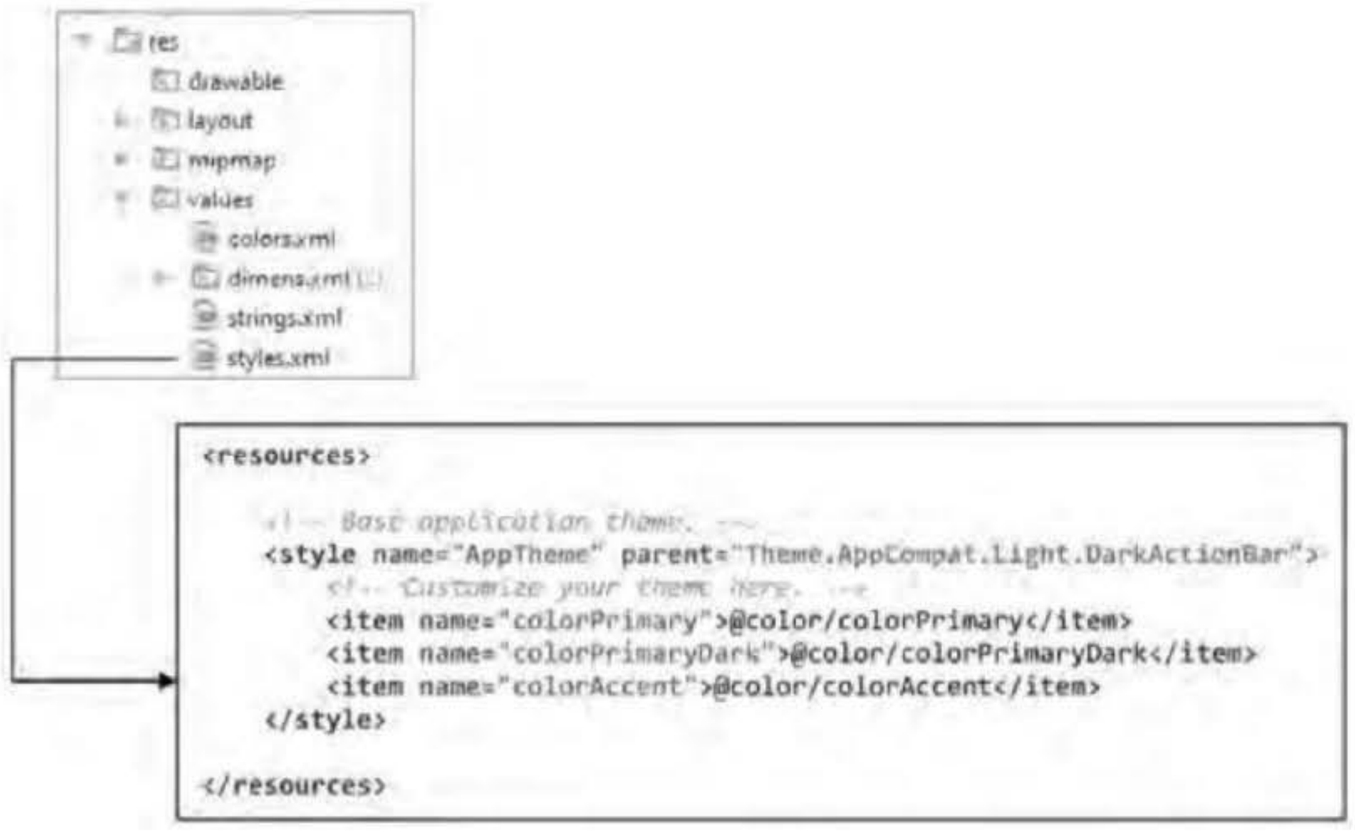
Aquí se realizan algunas especificaciones propias de la aplicación, el atributo android:allowBackup con valor true indica que si se realiza una copia de seguridad del sistema esta aplicación se adjuntará a dicha copia. El atributo android:icon hace referencia a la imagen ic_launcher.png, el cual será el ícono de la aplicación; esta se encuentra ubicada dentro del paquete res/mipmap. Cabe mencionar que se puede usar otros íconos pegando las imágenes directamente sobre este paquete dentro del Android Studio.



El atributo android:label muestra el título que tendrá la aplicación; para este caso se está haciendo referencia a la cadena app_name especificada dentro del paquete res/values/strings.xml, como se muestra en la siguiente imagen:



El atributo android:suportsRtl hace referencia al soporte de los lenguajes usados en la aplicación móvil. Finalmente, el atributo android:theme hace referencia a la apariencia de la aplicación, la cual se encuentra configurada en el paquete res/values/styles.xml como se muestra en la siguiente imagen:



Etiqueta activity

```
<activity android:name=".MainActivity">

</activity>
```

Es un subelemento de la aplicación dentro del cual se definen las características de un controlador de tipo actividad. Una actividad representa una interfaz grafica dentro de un proyecto de aplicación móvil, por tanto, por cada actividad que se use en la aplicación deberá declararse dentro del archivo manifestAndroid. Presenta el atributo android:name, el cual define el nombre de la clase que la implementa; también puede usar el atributo android:screenOrientation para definir la forma en que se presentará la actividad: puede ser vertical (Portrait) u horizontal (landscape). Si no se especifica la orientación, el dispositivo la determinará.

Etiqueta intent-filter

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Es un subelemento de la actividad dentro del cual se definen las acciones que realizará la actividad de forma agrupada. Los intents tienen la finalidad de limitar las acciones que puede realizar la actividad; dentro de ellos se encuentra la etiqueta <action>, la cual define la acción que tiene como soporte al android:name, que especifica que dicha actividad es la principal y, por lo tanto, desde ella se controlan las demás actividades del proyecto. El atributo category android:name define que el punto de entrada debe aparecer en el lanzador de aplicaciones; luego, estas se enumeran en el lanzador de aplicaciones de nivel superior.

3.10.5 Cambiando características de la aplicación usando AndroidManifest

Modificar el título de la aplicación:

Paso 1: Ingrese al archivo strings.xml ubicado en res/values/.

Paso 2: Agregue una cadena de nombre miTitulo, que muestre el mensaje Mi aplicación ANDROID. El código debe mostrarse de la siguiente manera:

```
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
```

```
<resources>
    <string name="app_name">My Application</string>
    <string name="miTitulo">Mi aplicación ANDROID</string>
</resources>
```

Paso 3: Ahora vaya al AndroidManifest.xml que se encuentra ubicado en app/manifest/ y modifique la siguiente línea de código:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mtorres.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/miTitulo"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Paso 4: Finalmente, el título de la aplicación se muestra de la siguiente manera:



Modificar el ícono de la aplicación

Paso 1: Si tiene la siguiente imagen:

Imagen	Archivo	Ruta
	libro.png	app/res/mipmap/

Dicha imagen debe encontrarse en el paquete mipmap de la aplicación; para esto, seleccione el archivo de la imagen (recomendado con extensión png) y presione <Ctrl+C>. Luego en Android Studio haga clic derecho sobre el paquete mipmap del proyecto, como se muestra en la siguiente imagen:



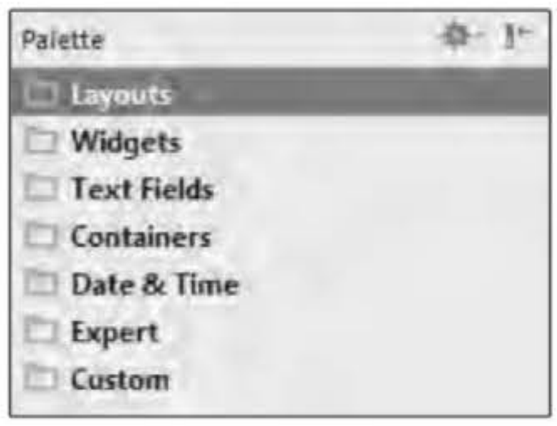
Paso 2: Ahora, modifique el archivo AndroidManifest para especificar el nuevo ícono de la aplicación:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.miempresa.personal.applayout_1">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/libro"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

3.11 Paleta de herramientas de actividad

Presenta un conjunto de herramientas que podrán insertarse dentro de la aplicación móvil. Se debe considerar que esta paleta solo se encuentra presente cuando selecciona un archivo desde la carpeta layout.

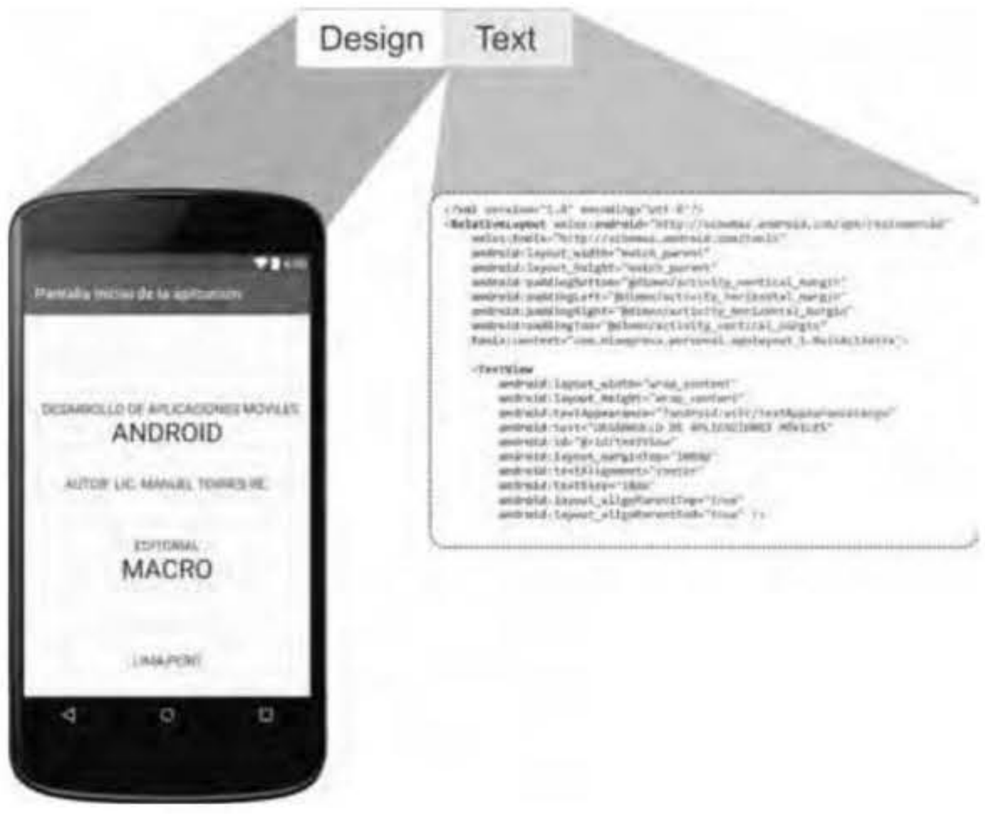
Menú de opciones: *View > Tools Windows > Pallete*



La paleta de herramientas presenta un completo listado de objetos visuales (controles) que podrá usar para implementar una aplicación móvil. Aquí, un ejemplo del uso de los controles sobre un activity_main.xml:



Al implementar una aplicación móvil en Android Studio se contará con dos archivos, uno con una extensión, .java, que contiene la lógica de la aplicación; y un archivo .xml, que presenta la parte gráfica de la aplicación. La parte gráfica presenta dos opciones que se muestran en la parte inferior de la pantalla principal:



- *Design*: Permite mostrar el diseño de la aplicación móvil con la idea principal de crear la aplicación a través de los controles que presenta Android Studio.
- *Text:* Permite mostrar el código que genera la parte gráfica de la aplicación móvil, aquí se puede crear la aplicación en forma de código. Aquí, el código generado de la aplicación inicial:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.miempresa.personal.applayout_1.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="DESARROLLO DE APLICACIONES MÓVILES"
        android:id="@+id/textView"
        android:layout_marginTop="100dp"
        android:textAlignment="center"
        android:textSize="18dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="ANDROID"
        android:id="@+id/textView2"
        android:textAlignment="center"
        android:textSize="35dp"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="AUTOR: LIC. MANUEL TORRES RE."
        android:id="@+id/textView3"
        android:layout_marginTop="37dp"
        android:textAlignment="center"
        android:textSize="18dp"
        android:layout_below="@+id/textView2"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="MACRO"
        android:id="@+id/textView4"
        android:textAlignment="center"
        android:textSize="35dp"
        android:layout_below="@+id/textView5"
        android:layout_centerHorizontal="true" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="EDITORIAL"
        android:id="@+id/textView5"
        android:layout_marginTop="61dp"
        android:textAlignment="center"
        android:textSize="18dp"
        android:layout_below="@+id/textView3"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="LIMA-PERÚ"
        android:id="@+id/textView6"
        android:textAlignment="center"
        android:textSize="18dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="21dp" />
</RelativeLayout>
```

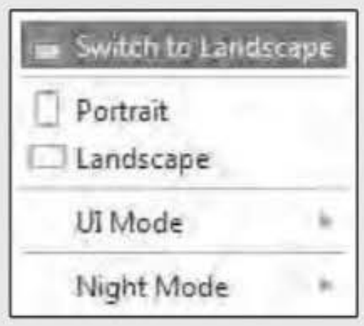
3.12 Controles del emulador

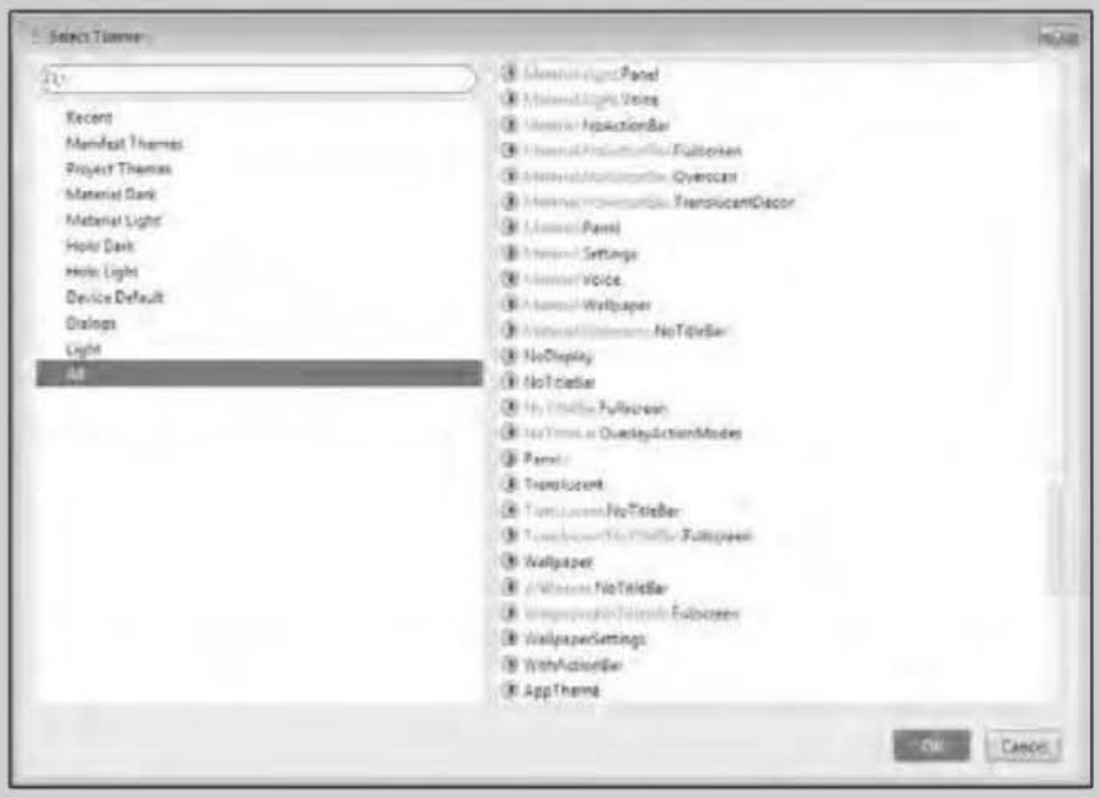
Son controles que permiten modificar algunas características del emulador en tiempo real.



Aquí, algunas características que presentan las opciones de la barra:

Botón	Descripción
	Permite configurar en qué emulador se mostrará la aplicación.

Botón	Descripción
Nexus 5	Permite modificar el tipo de emulador en tiempo real. Al seleccionar uno de la lista ya se visualiza cómo se mostrará la aplicación sin necesidad de ejecutarlo. Aquí, una lista de dispositivos que presenta la lista desplegable.  Si desea visualizar la aplicación en diferentes dispositivos al mismo tiempo, debe usar la opción Preview All Sreen Sizes, y se muestra la siguiente imagen:  Para retornar a su estado original seleccione la opción Remove previews que se encuentra en la misma lista de los dispositivos.
	Permite modificar la orientación del dispositivo mientras presenta los controles en tiempo de diseño. Por defecto, cualquier dispositivo se presentará de forma vertical (Portrait), pero se puede modificar a horizontal usando la opción Landscape que presenta la lista desplegable. 

Botón	Descripción
AppTheme	Presenta una lista de máscaras que modifican el aspecto que presenta la aplicación móvil. El formato estándar es *Project Themes > appTheme*. 

3.13 Árbol de componentes

Presenta una lista de controles que se han agregado al dispositivo móvil, asimismo, muestra su contenido textual y, especialmente, el nombre de cada control.



Al seleccionar un control del árbol de componentes se verá que dicho control se seleccionará en el emulador; asimismo, se muestran las propiedades del control según lo seleccionado.

Hay que tener en cuenta de que los controles no necesariamente se pueden agregar al contenedor del dispositivo móvil; también puede arrastrar el control desde el cuadro de herramientas y soltarlos en el árbol de componentes. Esto se realizará cuando haya varios tipos de Layout por un tema de organización.

3.14 Propiedades de los controles

Presenta una lista de características que presenta un control seleccionado desde la aplicación. Se debe tener en cuenta que todos los controles tienen propiedades y que estos se pueden administrar como sea conveniente.



3.15 Ejecución de la aplicación en el emulador

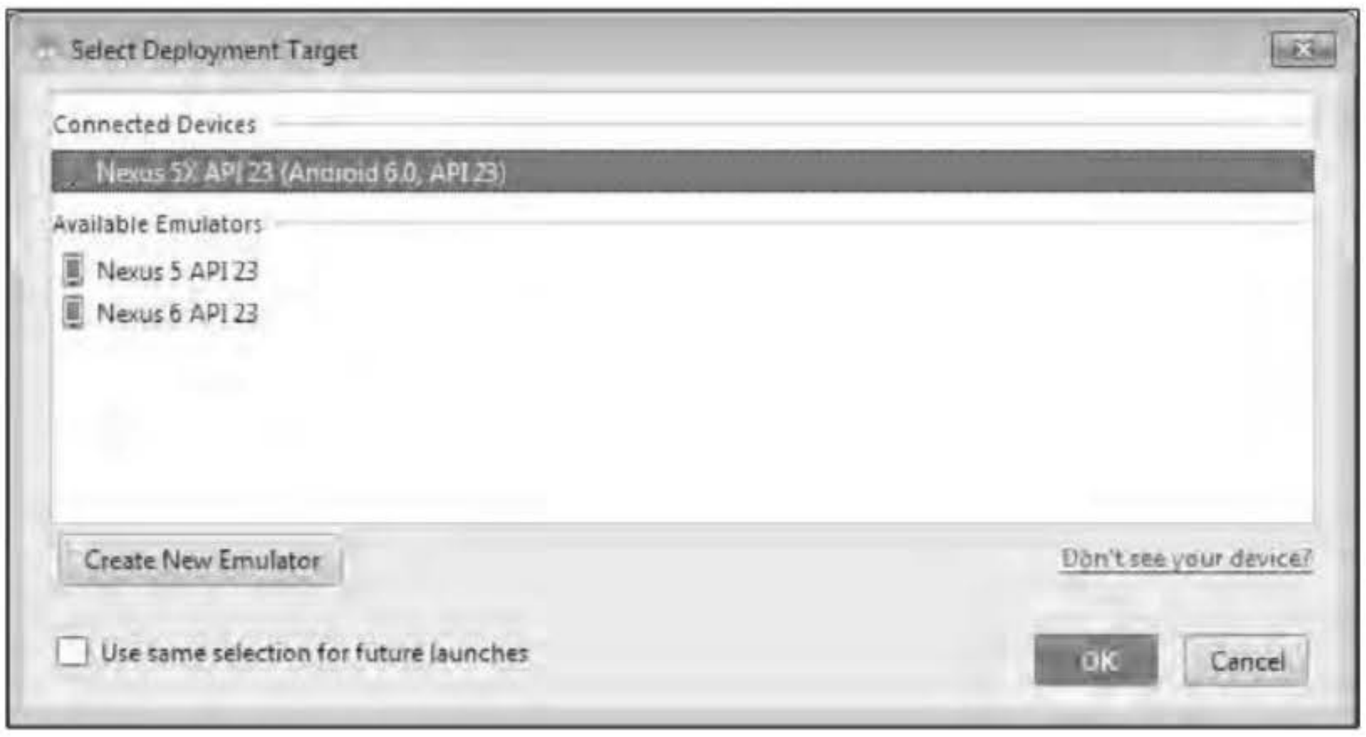
Cuando se colocan controles a la aplicación y luego se le asigna el código se le llama tiempo de diseño y codificación, respectivamente; pero, cuando se quiere visualizar el resultado de la aplicación en el dispositivo móvil de prueba, esto último se llama tiempo de ejecución.

Acceso directo: <Shift + F10>
Menú de opciones: Run > Run
Ícono: ▶

En cualquiera de los casos, se mostrará una pantalla adicional antes del resultado de la aplicación, el cual solicita seleccionar un dispositivo de salida. Si es la primera vez que se ejecuta una aplicación entonces se deberá seleccionar el botón Create New Emulator, el cual muestra la siguiente ventana:



Es importante asegurarse de que la categoría sea Phone o Tablet y seleccione de la lista un tipo de dispositivo de salida; adicionalmente, se muestran algunas características propias del dispositivo. Seleccione el botón Next para realizar algunas instalaciones propias del emulador y, finalmente, presione el botón Finish.



Se pueden crear muchos emuladores, luego estos se mostrarán en la ventana Select Deploymente Target cuando realice la ejecución de la aplicación móvil. Para ver el resultado final de la aplicación deberá seleccionar un emulador de la lista Available Emulators y presionar el botón Ok.

3.16 Descripción del emulador estándar

El emulador siempre presentará las mismas opciones a pesar de los distintos modelos que usted pueda seleccionar. Aquí, una lista de sus principales controles:



- Salir del emulador es similar a apagar el dispositivo móvil; para poder visualizar nuevamente la aplicación se deberá correr (*Run>Run* o <Shift + F10>) desde Android Studio.
- El control de volumen solo funcionará cuando se encuentre ejecutando una aplicación que tenga audio.
- El control de rotación permite al emulador rotar a la izquierda o derecha tal como lo haría en un dispositivo móvil real:



- El control de retorno permite devolver a la pantalla anterior; en caso no hubiera se muestra la pantalla inicial del dispositivo móvil. Se debe tener en cuenta que cuando aparece el teclado virtual en pantalla este mismo botón lo oculta.
- El control a la principal retorna directamente a la pantalla inicial del dispositivo móvil.
- El control de aplicaciones muestra una lista de aplicaciones precargadas en el dispositivo.

3.17 Unidades de medida

El desarrollo de aplicaciones con Android conlleva al trabajo en pantalla de un tipo de emulador que simule ser un dispositivo móvil; por lo tanto, se debe conocer ciertas unidades que serán muy comunes al implementar una aplicación con código.

Unidad de medida	Descripción
dp	**Density Pixel** Es una unidad que se basa principalmente en la densidad física que puede tener la pantalla de su dispositivo móvil. Esto se debe a los diferentes tamaños que presentan los dispositivos móviles donde se ejecutarán las aplicaciones. Su aplicación se da como si se estuviera tratando a un pixel, con la diferencia de que si su dispositivo presenta poca densidad se usarán menos números de píxeles y sucede al revés si el dispositivo presenta una densidad alta. Es por esta razón que las aplicaciones móviles no se distorsionan cuando se visualizan en diferentes dispositivos.
sp	**Scale Pixel** Su forma de trabajo es similar a la unidad dp; pero adiciona una escala automática según el tamaño de la fuente que asigne en una aplicación móvil.
pt	**Points** Representa solo el 72% de una pulgada y la resolución dependerá mucho del tamaño físico que presente el dispositivo móvil
px	**Pixel** Representa a un pixel en una resolución real de la pantalla del dispositivo. Se recomienda no hacer uso de dicha medida en aplicaciones móviles porque estos presentan una variedad de pixeles por pulgada.

3.18 Recursos Android

Toda aplicación móvil Android tiene un paquete de recursos que permite tener administrados los elementos que componen una aplicación. En Android Studio existe un paquete llamado res, el cual contiene todos los recursos, como se ve en la siguiente imagen:



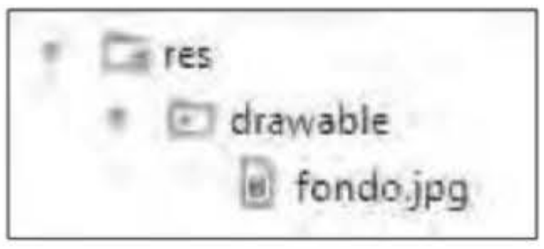
3.18.1 Drawable

Son los recursos gráficos que puede tener la aplicación móvil para sus diferentes procesos; inicialmente, en la aplicación este paquete se muestra vacío para lo cual se debecopiar desde el exterior una imagen de cualquier tipo permitido. Claro está que la recomendación sería jpg, png o gif.

Pasos para agregar una imagen al paquete drawable:

Paso 1: Seleccione el archivo de la imagen. Haga clic derecho en la opción Copiar.

Paso 2: En Android Studio haga clic derecho sobre el paquete drawable y seleccione la opción Pegar. El paquete debe mostrarse como sigue:



Una vez agregado la imagen al paquete drawable se puede usar en el archivo main_activity.xml colocando el siguiente código:

```
android:background="@drawable/fondo"
```

Para hacer referencia a la imagen se debe anteponer el término @drawable, que representa al paquete drawable. Finalmente, el código del archivo main_activity.xml quedaría de la siguiente manera:

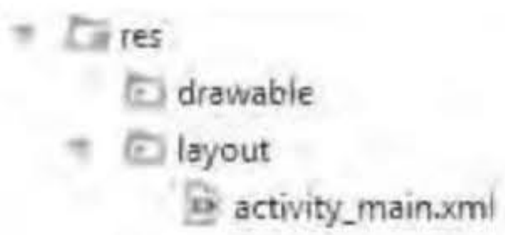
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="@drawable/fondo"
    tools:context="com.miempresa.personal.appdrawable.MainActivity">
</RelativeLayout>
```

El resultado sería el siguiente:

FONDO.JPG	MAIN_ACTIVITY.XML
	

3.18.2 Layout

Son archivos de extensión xml que contienen toda la definición de una interfaz de usuario; dicho de otra forma, es una presentación en el dispositivo móvil. En la ventana de proyectos del Android Studio se visualiza de la siguiente manera:



Aquí, algunos pasos que se pueden realizar para administrar los activitys dentro de un proyecto móvil:

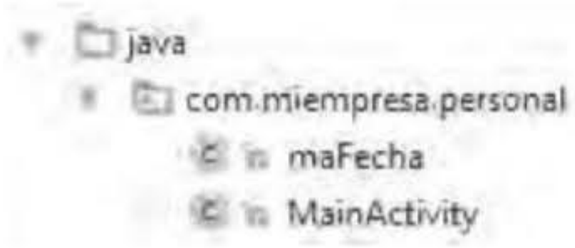
Agregar un activity

Paso 1: Haga clic derecho sobre el paquete layout ubicado en los recursos (res).

Paso 2: Seleccione *New > Activity > Empty activity*



Asigne un nombre a la actividad; en este caso se inicia con ma por ser un Main Activity; luego, asegúrese de que el check del Generate Layout File se encuentre activo. Finalmente, defina correctamente el nombre del paquete principal donde se creará la actividad, el cual debe ser el mismo donde se encuentra la primera actividad llamada MainActivity, dentro del paquete java de la aplicación, como se visualiza en la siguiente ventana:



Archivos que componen una Activity

Se debe tener en cuenta que al agregar un Activity dentro de la carpeta Layout se debe generar dos archivos, como se muestra a continuación:



El archivo que se encuentra dentro del paquete Layout representa la forma visual de la actividad en la cual se pueden añadir los controles para una presentación al usuario; mientras que, la que se encuentra dentro del paquete java, representa su lógica basada en código. Aquí, los contenidos de cada uno de los archivos:

PAQUETE: RES/LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.miempresa.personal.maPresentacion">
</RelativeLayout>
```

PAQUETE: APP/JAVA

```
package com.miempresa.personal;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import com.miempresa.personal.appactivity.R;

public class maPresentacion extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ma_presentacion);
    }
}
```

ELIMINAR UN ACTIVITY

Paso 1: Haga clic derecho sobre la actividad a eliminar, la cual se encuentra en el paquete layout de los recursos (res).

Paso 2: Seleccione la opción Delete y se muestra la siguiente ventana:



Paso 3: Mantenga seleccionados los check para la eliminación y presione Ok.

Paso 4: Del mensaje siguiente seleccione el botón Delete Anyway.

Paso 5: Ahora con los mismos pasos elimine el Activity generado en el paquete res/java; en el mensaje que se muestra solo presione Ok.

Tenga en cuenta que para eliminar una actividad debe realizar dos eliminaciones: la actividad mostrada en el Layout y Java.

Se realizará una aplicación que permita manejar dos activitys, de tal forma que, al iniciar la aplicación se muestre el archivo activitymain.xml y que al presionar el botón Mostrar fecha actual se muestre la segunda actividad desde el archivo activity_ma_fecha. Finalmente, dentro de esta última Activity adicione un botón Salir para cerrar la actividad y retornar a la pantalla inicial:

ACTIVITYMAIN.XML	ACTIVITY_MA_FECHA.XML
	

PASOS PARA LA SOLUCIÓN:

Paso 1: Cree un nuevo proyecto (*File > New > New project…)*

- Application name: AppActivity
- Company Domain: personal.miEmpresa.com

Paso 2: Añada los siguientes componentes al proyecto:

ARCHIVO	RUTA DENTRO DE LA APLICACIÓN
 libro.png	res/mipmap
 logo.png	res/drawable

Paso 3: Haga clic derecho sobre el paquete *layout > New > Activity > Empty Activity*

- Activity name: ma_fecha
- Package name: com.miempresa.personal

Paso 4: Presione el botón Finish para agregar la actividad al proyecto:

La distribución de los archivos en el explorador de proyectos.



Aquí, el código de cada uno de los archivos necesarios para la aplicación:

STRINGS.XML	PAQUETE: APP/RES/VALUES

```
<resources>
    <string name="titulo">Manejo de la clase Activity</string>
</resources>
```

ANDROIDMANIFEST.XML | **PAQUETE: APP/MANIFEST**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.miempresa.personal.appactivity">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/libro"
        android:label="@string/titulo"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.miempresa.personal.MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.miempresa.personal.maFecha" />
    </application>
</manifest>
```

ACTIVITY_MAIN.XML | **PAQUETE: APP/RES/LAYOUT**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.miempresa.personal.MainActivity">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@drawable/logo"
        android:layout_above="@+id/btnMostrar"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="171dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="MOSTRAR FECHA ACTUAL"
        android:id="@+id/btnMostrar"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="192dp"
        android:onClick="muestraVentana"/>
</RelativeLayout>
```

ACTIVITY_MA_FECHA.XML	PAQUETE: APP/RES/LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.miempresa.personal.maFecha">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="tvFecha"
        android:id="@+id/tvFecha"
        android:textSize="60dp"
        android:layout_above="@+id/btnSalir"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="44dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SALIR"
        android:id="@+id/btnSalir"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="178dp"
        android:onClick="salir" />
</RelativeLayout>
```

MainActivity	Paquete: app/java

```
package com.miempresa.personal;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import com.miempresa.personal.appactivity.R;

public class MainActivity extends AppCompatActivity {

    //GLOBALES
    Button btnSiguiente;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Instanciar al boton
        btnSiguiente = (Button) findViewById(R.id.btnMostrar);
    }

    //Metodo que muestra la ventana siguiente
    public void muestraVentana(View view){
        Intent objI = new Intent(MainActivity.this,maFecha.class);
        startActivity(objI);
    }
}
```

MAFECHA	PAQUETE: APP/JAVA

```
package com.miempresa.personal;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.PixelCopy;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import com.miempresa.personal.appactivity.R;
import java.util.Calendar;
public class maFecha extends AppCompatActivity {

    //GLOBAL
    private int dia,mes,año;
    TextView tvFecha;
    Button btnSalir;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ma_fecha);

        //Instanciando a los controles
        tvFecha = (TextView) findViewById(R.id.tvFecha);
        btnSalir = (Button) findViewById(R.id.btnSalir);

        //Invocando al metodo que muestra la fecha actual
        muestraFecha();
    }

    //Metodo que muestra la fecha actual
    public void muestraFecha() {
        Calendar fecha = Calendar.getInstance();
        dia = fecha.get(Calendar.DAY_OF_MONTH);
        mes = fecha.get(Calendar.MONTH);
        año = fecha.get(Calendar.YEAR);
        tvFecha.setText(dia+"/"+mes+"/"+año);
    }

    //Metodo que permite salir del activity
    public void salir(View view){
        finish();
    }
}
```

3.18.3 MIPMAP

Es un paquete que contiene el ícono de la aplicación móvil. Se presentan en varias carpetas por la necesidad de los tamaños de variados dispositivos móviles que puede ejecutarse la aplicación. En versiones pasadas de Android los íconos se encontraban dentro de la carpeta drawable. Entonces, dentro de mipmap, estará la distribución de carpetas para cada uno de los tamaños del ícono, como se puede ver en la siguiente imagen:



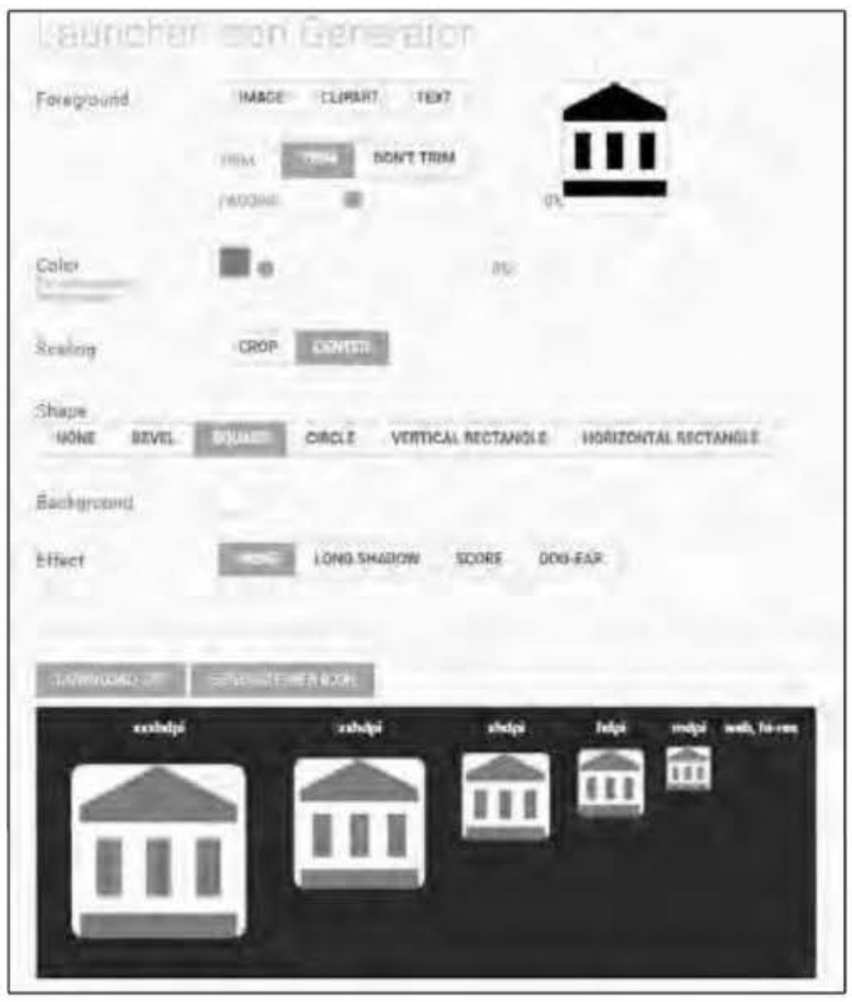
La imagen anterior muestra los archivos de la aplicación, para esto se debe cambiar a Project Files en el Explorador de Proyectos, ya que inicialmente se mostraba Android. Ahora, la distribución de tamaños por cada carpeta será:

PAQUETE DE ORIGEN	TAMAÑO DE ÍCONO
res/mipmap-mdpi/	48x48 píxeles
res/mipmap-hdpi/	72x72 píxeles
res/mipmap-xhdpi/	96x96 píxeles
res/mipmap-xxhdpi/	144x144 píxeles
res/mipmap-xxxhdpi/	192x192 píxeles

Si decide cambiar por otro ícono se recomienda crear el archivo de imagen con extensión .png, además de implementarlo en diferentes tamaños. Para esto se podría usar Photoshop como una alternativa, pero a continuación se presenta una solución web, para esto realice los siguientes pasos:

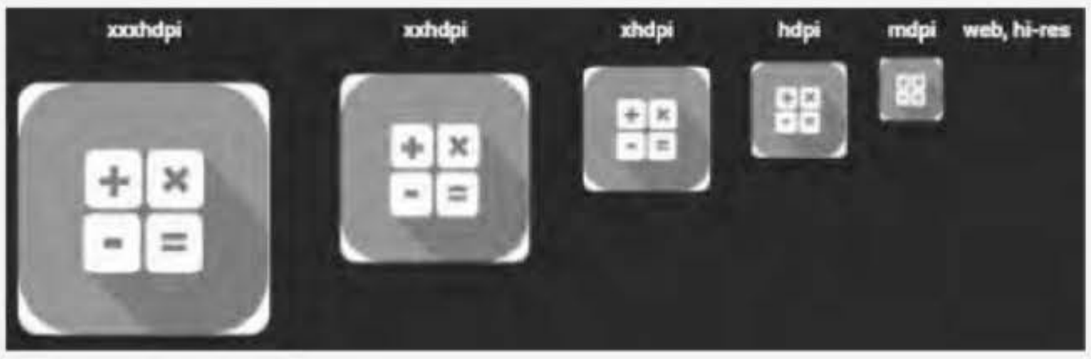
URL: https://romannurik.github.io/AndroidAssetStudio/index.html

Paso 1: De la url anterior seleccione Launcher icons y se mostrará la siguiente ventana:



Desde la web Launcher Icon Generator se pueden realizar las siguientes actividades:

- *Image*: Permite crear íconos a partir de la selección de una imagen.



- *ClipArt*: Permite seleccionar íconos prediseñados, este sería un medio muy rápido de encontrar íconos.



- *Text*: Permite crear íconos a partir de un texto; para esto debe ingresar un texto y asignar un tipo de letra.



- *Trim | Don't Trim*: Permite eliminar los espacios en blanco que pueda tener una imagen o clipart con respecto a su marco; es decir, trata de mostrar la imagen en todo el marco. Mientras que Don't Trim proporciona la imagen de forma que se visualiza centrado y con un espacio interior.



Trim



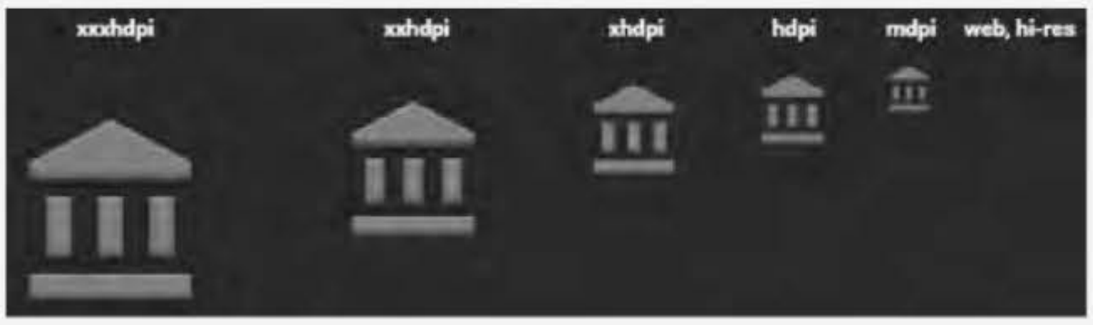
Don't Trim

- *Padding:* Permite ampliar el espacio interno y reduce el tamaño de la imagen. Por ejemplo, si asigna 25% el resultado sería:



- *Shape*: Permite especificar la forma que tendrá la exposición de las imágenes:

- *None*: No especifica la forma.
- *Bevel*: Aplica un biselado a la imagen.



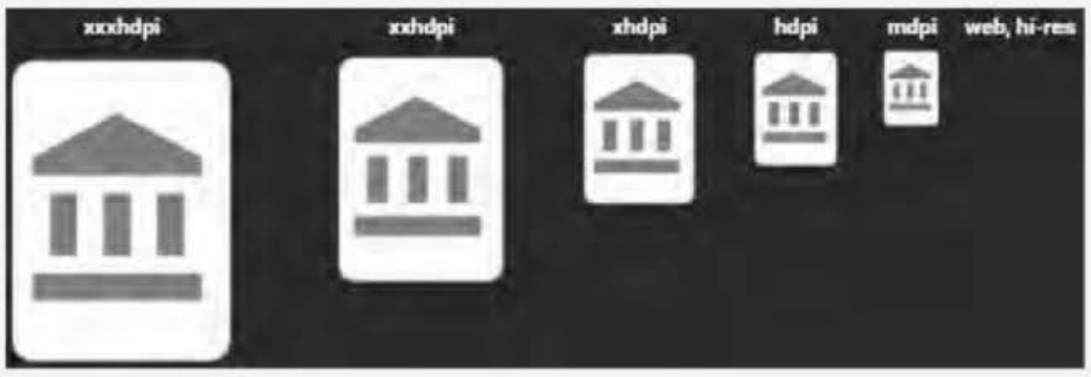
- *Square*: aplica un marco de cuadro a la imagen.



- *Circle*: Aplica un marco redondo a la imagen.



- *Vertical Rectangle*: Asigna un marco vertical a la imagen.

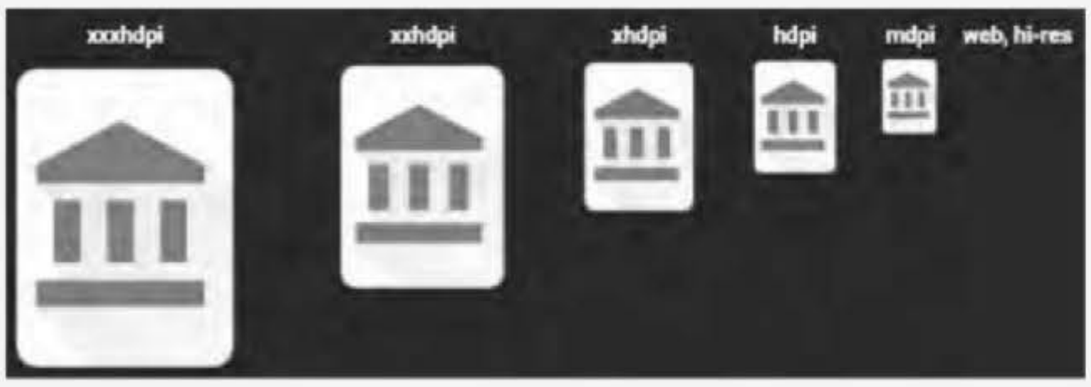


- *Horizontal Rectangle*: Asigna un marco horizontal a la imagen.



- *Background:* Permite definir un color de fondo dentro del marco de la imagen.

- *Effect:* Define algunos efectos sobre la imagen:
  - *None*: No aplica ningún efecto.
  - *Long Shadow*: Asigna una sombra a la imagen.



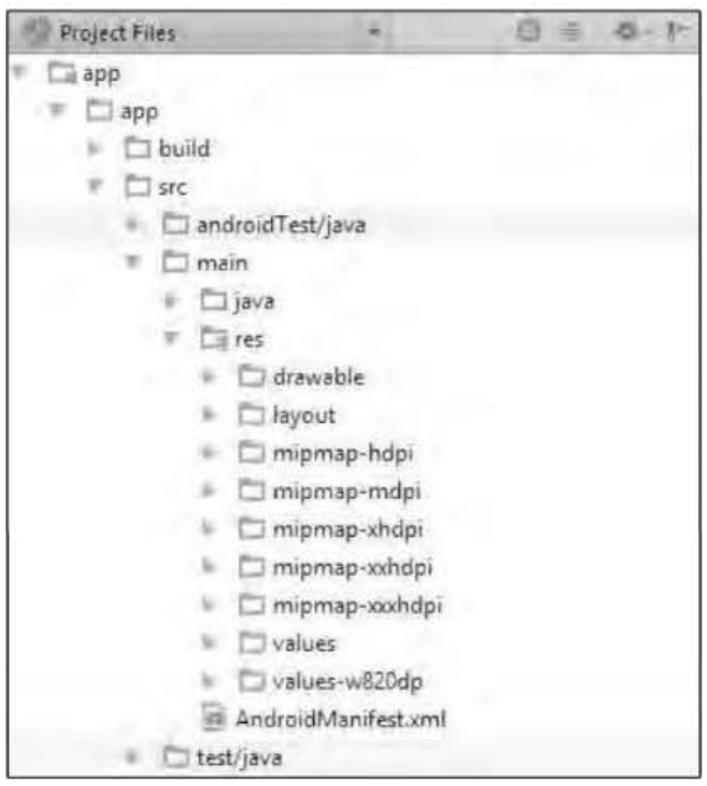
- *Score*: Asigna una sombra superior cuadrada a la imagen



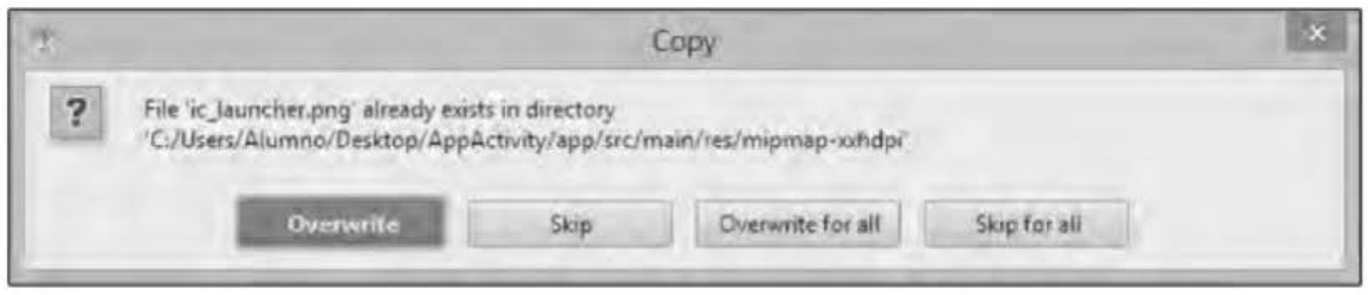
- *Dog-Ear.*



Paso 2: Para crear los íconos solo debe asegurarse de que la muestra sea la apropiada. Presione el botón DONWLOAD .ZIP. Luego, descomprima el archivo y copie todas las carpetas con nombre mipmap al paquete res de la aplicación en Android Studio (no se olvide que el Explorador de Proyectos debe encontrarse en vista Projects Files), como se muestra en la siguiente imagen:



Cuando termine de realizar el copiado en la carpeta res, Android Studio le mostrará la siguiente imagen:



Paso 3: En vista de que ya se cuenta en la aplicación con imágenes solo debe seleccionar el botón Overwrite para cada una de las imágenes. Finalmente, ahora puede usar dichas imágenes invocando en el archivo AndroidManifest.xml, como se muestra en el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.miempresa.personal.appactivity">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher.png"
        android:label="@string/titulo"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.miempresa.personal.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.miempresa.personal.maFecha" />
    </application>
</manifest>
```

3.18.4 Values

Contiene archivos XML que registran valores que se pueden usar en la aplicación móvil como números, cadenas, valores boléanos, dimensiones de la pantalla, colores, etc.



Aquí, la manera de trabajar cada uno de los archivos iniciales del paquete values:

Archivo: colors.xml
Ubicación: app/res/values

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Defina un color dentro del archivo colors.xml y aplíquelo al layout, para esto realice los siguientes pasos:

Paso 1: En el archivo colors.xml añada los colores rojo, verde y azul, respectivamente.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="colorRojo">#ff0000</color>
    <color name="colorVerde">#10a400</color>
    <color name="colorAzul">#0015ff</color>
</resources>
```

Paso 2: En el archivo activity_main.xml agregar el código android:background="@color/" seleccione el color especificado en el archivo colors.xml.



Archivo: dimens.xml
Ubicación: app/res/values/dimens.xml

```
<resources>
    <!– Default screen margins, per the Android Design guidelines. –>
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

Modifique el margen interno de los elementos que se colocarán dentro de un layout, para esto realice los siguientes pasos:

Paso 1: En el archivo dimens.xml modifique el margen horizontal y vertical por 36dp:

```
<resources>
    <!– Default screen margins, per the Android Design guidelines. –>
    <dimen name="activity_horizontal_margin">36dp</dimen>
    <dimen name="activity_vertical_margin">36dp</dimen>
</resources>
```

Paso 2: En el archivo activity_main.xml se observa que ya se encuentran los espacios usando el recurso activity_horizontal_margin.xml y activity_vertical_margin.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.alumno.myapplication.MainActivity">
</RelativeLayout>
```



Paso 3: Finalmente, observe que el botón añadido al Layout se posiciona en el margen 36dp con respecto a la especificación en el padding.

Archivo: strings.xml
Ubicación: app/res/values

```
<resources>
    <string name="app_name">My Application</string>
</resources>
```

Modifique el título de la aplicación usando una definición de cadena en el archivo strings.xml. Luego haga referencia a dicha cadena dentro del AndroidManifest:

Paso 1: En el archivo strings.xml agregue el siguiente código:

```
<string name="miTitulo">Manejo de valores strings</string>
```

El código queda de la siguiente manera:

```
<resources>
    <string name="app_name">My Application</string>
    <string name="miTitulo">Manejo de valores strings</string>
</resources>
```

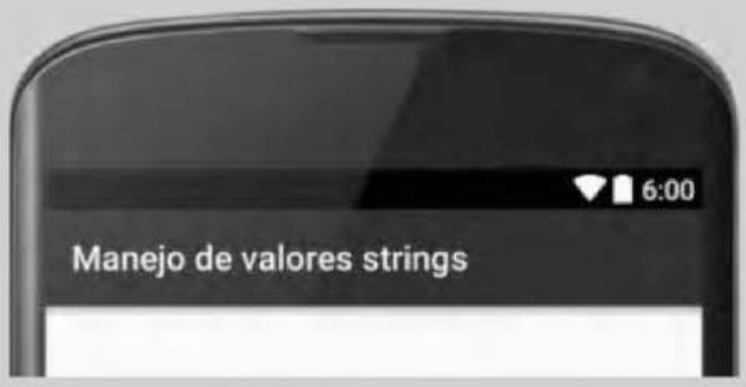
Paso 2: En el archivo AndroidManifest.xml haga referencia a la cadena dentro del atributo label, como se muestra en el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mtorres.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/miTitulo"
        android:supportsRtl="true"
        android:theme="@style/miTema">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```



En el siguiente ejemplo asigne un título a un botón de salida:

Paso1: En el archivo strings.xml agregue el siguiente código:

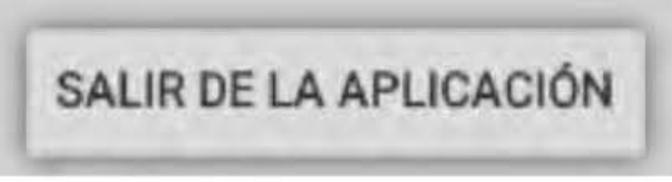
```
<string name="salida">SALIR DE LA APLICACIÓN</string>
```

El código queda de la siguiente manera:

```
<resources>
    <string name="app_name">My Application</string>
    <string name="salida">SALIR DE LA APLICACIÓN</string>
</resources>
```

Paso 2: En el archivo AndroidManifest.xml haga referencia a la cadena dentro del atributo label, como se muestra en el siguiente código:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/salida"
    android:id="@+id/btnSalir"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="94dp"
/>
```





CAPÍTULO

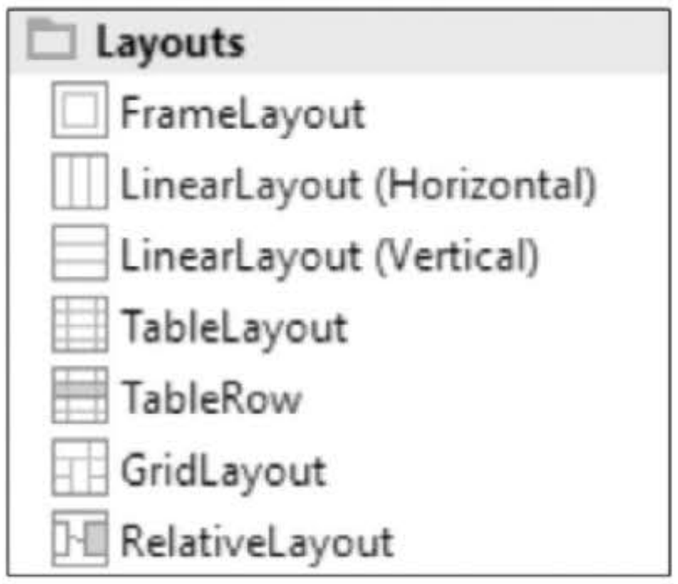
4

Controles de una aplicación Android

4.1 Controles layout

Android permite organizar los controles dentro de una aplicación para tener una mejor administración en pantalla, esto se puede realizar mediante los contenedores llamados Layout.

Para lograr una mejor organización existen diferentes tipos de layout como LinearLayout, FrameLayout, TableLayout, RelativeLayout.



4.1.1 LinearLayout

Es un contenedor de controles que permite alinearlos de forma horizontal o vertical dentro del diseño de la aplicación. El término Linear permite alinear los controles según el tipo; es decir, los controles se mostrarán en una misma fila uno a continuacion del otro o en columna una debajo del otro pero jamás combinados.

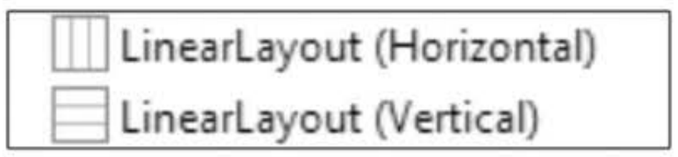
El código inicial de la implementacion de una LinearLayout es:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="match_parent">
</LinearLayout>
```

Es posible darse cuenta que no existe alineación, además, se sabe que de forma predeterminada los controles que se agreguen dentro de este Layout se alinearán de forma horizontal. Para modificar la alineación se agrega la propiedad orientation de la siguiente forma:

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="match_parent">
</LinearLayout>
```

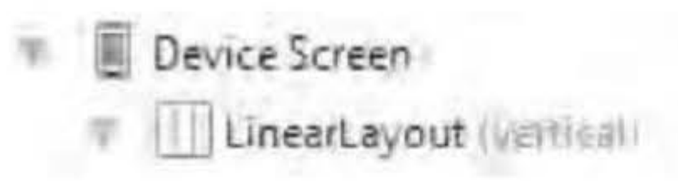
Así es que los controles LinearLayout (Horizontal) y LinearLayout (Vertical) que se muestran en el cuadro de herramientas del Android Studio solo se diferencian en la forma con la que especifican su orientación.



Véase ahora las principales propiedades del LinearLayout:

PROPIEDADES	DESCRIPCIÓN
android:orientation	Determina la orientación que tendrán los controles dentro del LinearLayout las cuales pueden ser horizontal y vertical: • Horizontal: Los elementos se acomodan de izquierda a derecha uno a continuación del otro. android:orientation="horizontal" • Vertical: Los elementos se acomodan en forma apilada, es decir, uno debajo de otro. android:orientation="vertical"
android:gravity	Determina la posición flotante de los controles que se encuentran dentro del Layout, tiene las siguientes opciones: bottom, center, center_horizontal, center_vertical, left, right, top.
android:paddingBottom android:paddingLeft android:paddingRight android:paddingTop	Permite definir el espacio interior entre los componentes y el mismo Layout. De forma predeterminada el margen es de 16dp para todos los lados del Layout.

Para añadir un LinearLayout, se debe arrastrar este control desde el cuadro de herramientas hacia el dispositivo móvil de tal forma que la ventana Component Tree se muestre de la siguiente forma:



Si se quiere agregar controles dentro del LinearLayout arrastre el control desde el cuadro de herramientas y suéltelo dentro del LinearLayout ubicado en la ventana Componente Tree. Considere que puede agregar LinearLayout dentro de otro LinearLayout y generar así una jerarquía de elementos organizados.

EJEMPLO LINEARLAYOUT 1

LinearLayout - Interfaz de una calculadora básica

Objetivo	Implementar una interfaz de una calculadora básica.
GUI	
Observaciones	• Usar LinearLayout Vertical. • Usar LinearLayout Horizontal.

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*...).

- Application name: AppCalculadora
- Company Domain: personal.miEmpresa.com

Paso 2: Añada los siguientes componentes al proyecto:

Archivo	Ruta dentro de la aplicación
 logo.png	res/drawable

Paso 3: En el nombre de la actividad coloque maCalculadora.

Paso 4: Asigne la siguiente distribución de los archivos en el explorador de proyectos:

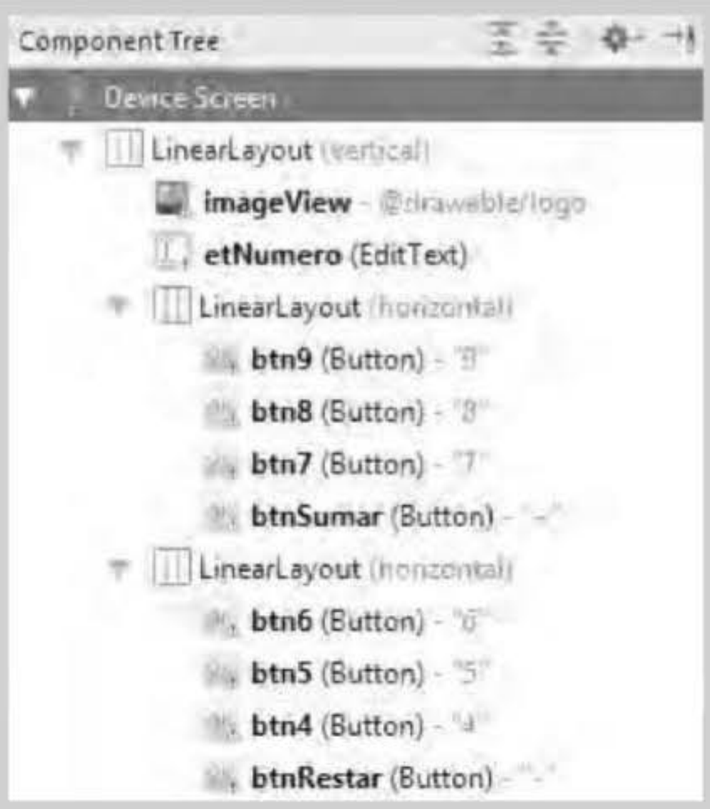


Paso 5: La ventana Component Tree debe mostrarse de la siguiente manera:

Al iniciar la aplicación por defecto se observa un RelativeLayout. Desde el código del archivo activity_ma_calculadora.xml modifique la etiqueta RelativeLayout por LinearLayout. Asigne la orientación vertical y la gravedad con alineación centrada para que los controles contenidos se mantengan en el centro.

Dentro del LinearLayout (vertical) agregue un control ImageView, EditText y cinco LinearLayout tipo horizontal como se muestra en la imagen.

Asimismo, agregue botones en cada LinearLayout, para asignar un título y un nombre (solo debe presionar doble clic sobre el botón). En el caso de del ImageView agregue una imagen desde la propiedad src y seleccione la imagen logo.jpg previamente copiada al paquete drawable.



Ahora bien, véase el código de cada uno de los archivos necesarios para la aplicación:

STRINGS.XML | **PAQUETE: APP/RES/VALUES**

```
<resources>
    <string name="app_name">Manejo de LayoutLinear</string>
</resources>
```

ANDROIDMANIFEST.XML | **PAQUETE: APP/MANIFEST**

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

ACTIVITY_MA_CALCULADORA.XML | **PAQUETE: APP/RES/LAYOUT**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context="com.miempresa.personal.appcalculadora.maCalculadora">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="100dp"
        android:id="@+id/imageView"
        android:src="@drawable/logo" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/etNumero"
        android:textSize="120dp"
        android:background="@color/foreground_material_light"
        android:textColor="@color/abc_primary_text_material_dark" />
```

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="9"
            android:id="@+id/btn9"
            android:textSize="20dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="8"
            android:id="@+id/btn8"
            android:textSize="20dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="7"
            android:id="@+id/btnm7"
            android:textSize="20dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="+"
            android:id="@+id/btnSumar"
            android:textSize="20dp" />
    </LinearLayout>
    <LinearLayout
        android:gravity="center"
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="6"
            android:id="@+id/btn6"
            android:textSize="20dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="5"
            android:id="@+id/btn5"
            android:textSize="20dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="4"
            android:id="@+id/btn4"
            android:textSize="20dp" />
```

```
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="-"
            android:id="@+id/btnRestar"
            android:textSize="20dp" />
    </LinearLayout>
    <LinearLayout
        android:gravity="center"
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="3"
            android:id="@+id/btn3"
            android:textSize="20dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="2"
            android:id="@+id/btn2"
            android:textSize="20dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="1"
            android:id="@+id/btn1"
            android:textSize="20dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="*"
            android:id="@+id/btnProducto"
            android:textSize="20dp" />
    </LinearLayout>
    <LinearLayout
        android:gravity="center"
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="0"
            android:id="@+id/btn0"
            android:textSize="20dp" />
```

```
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="RAIZ"
            android:id="@+id/btnRaiz"
            android:textSize="20dp" />

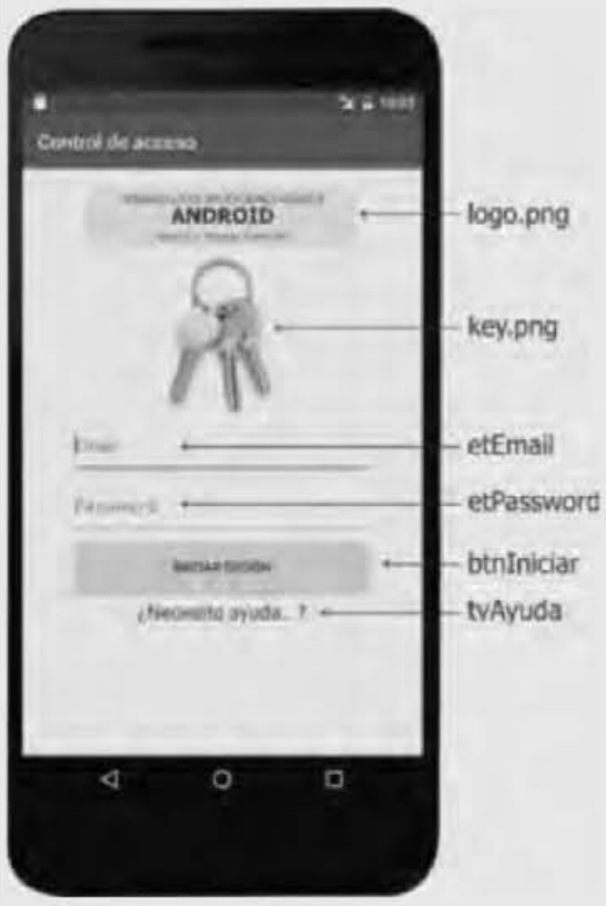
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="^"
            android:id="@+id/btnPotencia"
            android:textSize="20dp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="/"
            android:id="@+id/btnDividir"
            android:textSize="20dp" />
    </LinearLayout>
    <LinearLayout
        android:gravity="center"
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <Button
            android:layout_width="349dp"
            android:layout_height="wrap_content"
            android:text="="
            android:id="@+id/btnigual"
            android:textSize="20dp" />
    </LinearLayout>
</LinearLayout>
```

EJEMPLO LINEARLAYOUT 2

LinearLayout - Interfaz de acceso

Objetivo	Implementar una interfaz de acceso por medio de un correo electrónico.
GUI	
Observaciones	• Usar LinearLayout Vertical

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

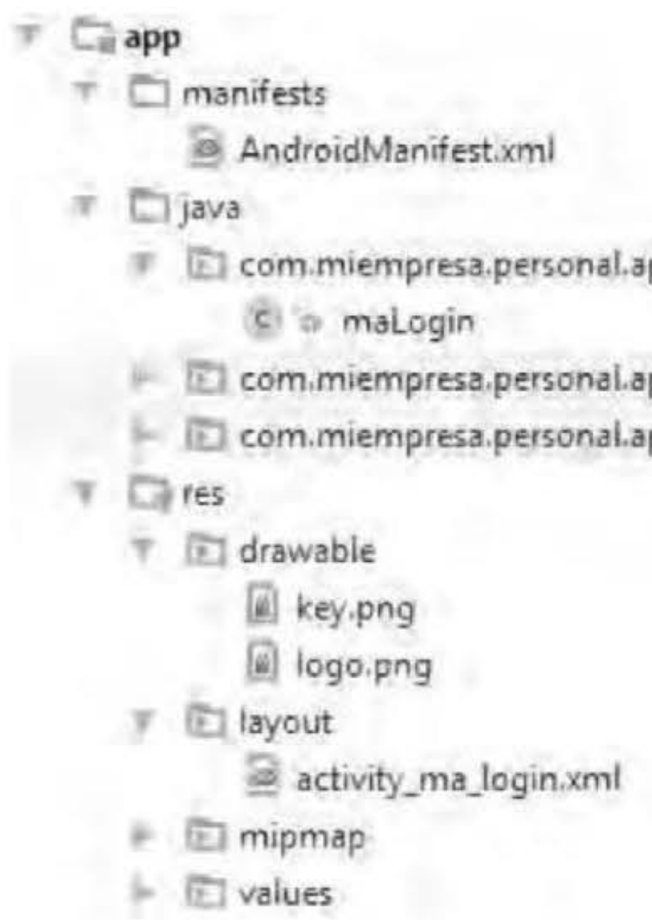
- Application name: AppAcceso
- Company Domain: personal.miEmpresa.com

Paso 2: Añada los siguientes componentes al proyecto:

ARCHIVO	RUTA DENTRO DE LA APLICACIÓN
 logo.png	res/drawable
 key.png	res/drawable

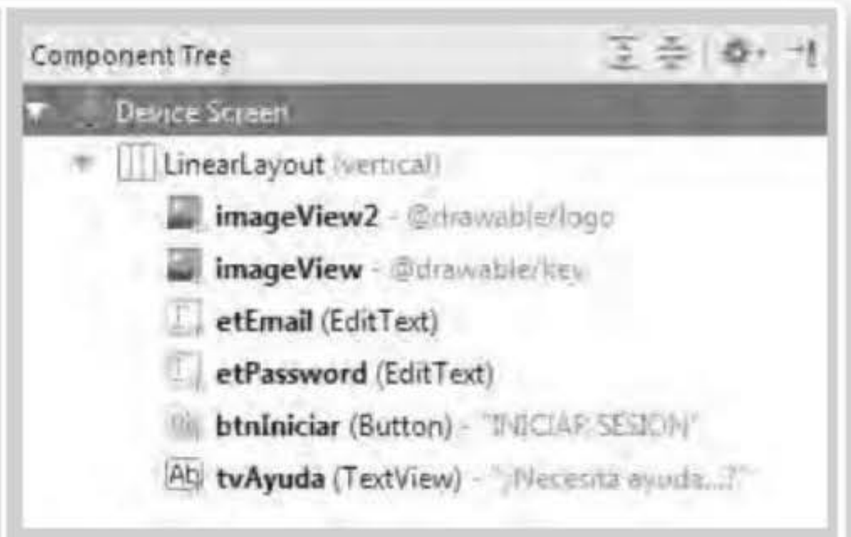
Paso 3: En el nombre de la actividad coloque maLogin.

Paso 4: Asigne la siguiente distribución de los archivos en el explorador de proyectos:



Paso 5: La ventana Component Tree debe mostrarse de la siguiente manera:

Al iniciar la aplicación por defecto se observa un RelativeLayout. Desde el código del archivo activity_ma_login.xml modifique la etiqueta RelativeLayout por LinearLayout. Asigne la orientación vertical.
Agregue dos controles de tipo imageView una para logo.png y el otro para key.png. Luego, dos controles de tipo EditText uno para el correo electrónico y otro para el password. No olvide colocar la propiedad nombre y hint, este último para mostrar un mensaje al usuario.
Agregue un botón que muestre el mensaje Iniciar sesion y asígnele el nombre btnIniciar.
Finalmente, agregue un control TextView que se encargue de mostrar el texto "¿Necesita ayuda?".



Ahora bien, véase el código de cada uno de los archivos necesarios para la aplicación:

STRINGS.XML — **PAQUETE: APP/RES/VALUES**

```
<resources>
    <string name="app_name">Control de acceso</string>
</resources>
```

ANDROIDMANIFEST.XML | **PAQUETE: APP/MANIFEST**

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

ACTIVITY_MA_LOGIN.XML | **PAQUETE: APP/RES/LAYOUT**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingTop="20dp"
    android:paddingBottom="50dp"
    android:paddingLeft="50dp"
    android:paddingRight="50dp"
    tools:context="com.miempresa.personal.appacceso.maLogin"
    android:weightSum="1">

    <ImageView
        android:layout_width="fill_parent"
        android:layout_height="60dp"
        android:id="@+id/imageView2"
        android:layout_gravity="center_horizontal"
        android:src="@drawable/logo" />

    <ImageView
        android:layout_width="312dp"
        android:layout_height="158dp"
        android:id="@+id/imageView"
        android:layout_gravity="center_horizontal"
        android:src="@drawable/key"
        android:layout_weight="0.11" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress"
        android:ems="10"
        android:id="@+id/etEmail"
        android:layout_gravity="center_horizontal"
        android:layout_weight="0.11"
        android:hint="Email" />
```

```
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:id="@+id/etPassword"
        android:layout_gravity="right"
        android:layout_weight="0.11"
        android:hint="Password" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="INICIAR SESIÓN"
        android:id="@+id/btnIniciar"
        android:layout_gravity="center_horizontal"
        android:layout_weight="0.11" />

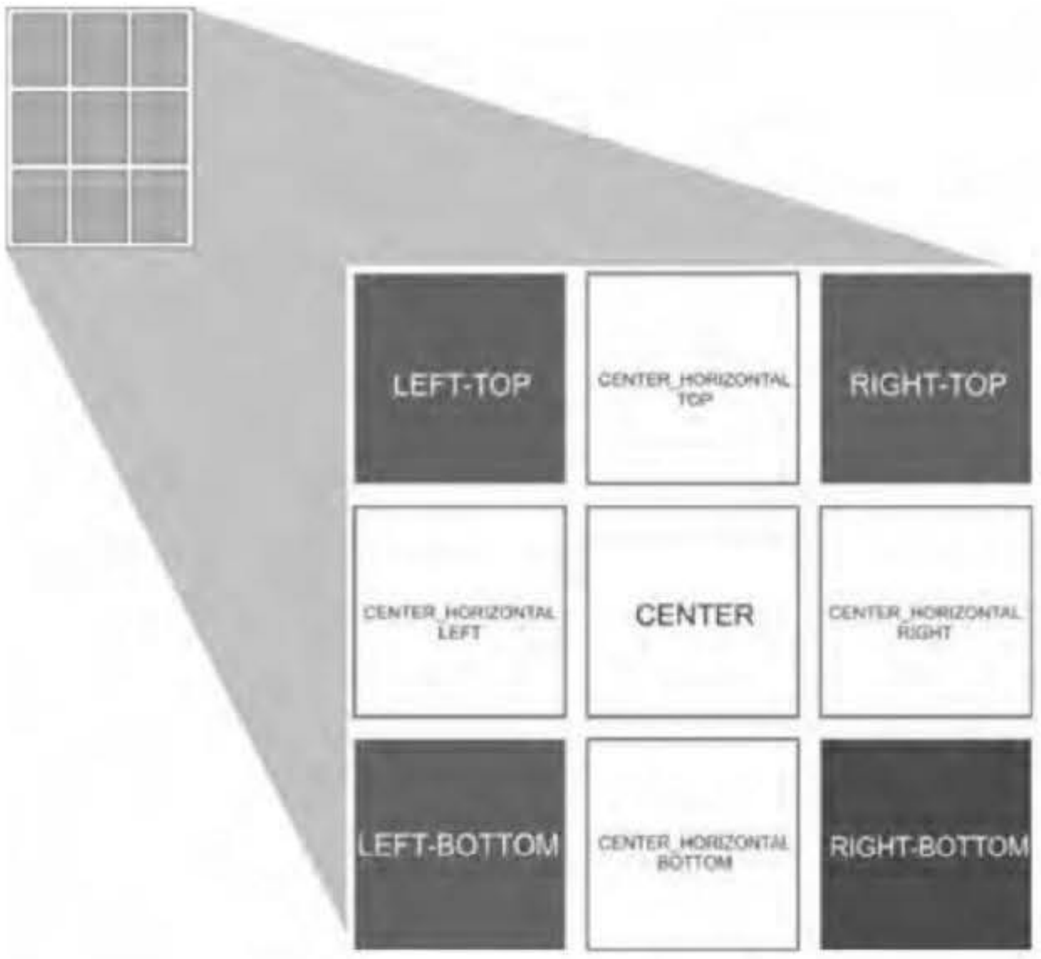
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="¿Necesita ayuda...?"
        android:id="@+id/tvAyuda"
        android:layout_gravity="center_horizontal"
        android:textSize="20dp"
        android:textColor="#0015ff" />
</LinearLayout>
```

4.1.2 FrameLayout

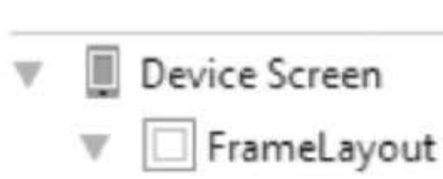
Un FrameLayout permite colocar todos los controles alineados al centro superior, dichos controles se apilarán uno tras otro; por lo tanto, solo se observa el último control agregado. Se recomienda su uso cuando el layout contiene un solo control o también cuando se implemente un medio de ocultamiento de los controles según la condición. El código inicial de la implementacion de un FrameLayout es el siguiente:

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal|top">
</FrameLayout>
```

Un FrameLayout muestra bloques segmentados en posiciones especificadas dentro de una actividad como se observa en la siguiente imagen:



Para añadir un FrameLayout debe arrastrarlo desde el cuadro de herramientas hacia el dispositivo móvil de tal forma que la ventana Component Tree se muestre de la siguiente forma:



Si se quiere agregar controles dentro del FrameLayout, arrastre el control desde el cuadro de herramientas y suéltelo dentro del FrameLayout ubicado en la ventana Componente Tree.

EJEMPLO FRAMELAYOUT 1

FrameLayout - Intercambio de imágenes

Objetivo	Implementar una aplicación que muestre inicialmente una imagen. Luego de hacer clic sobre la misma debe aparecer una segunda imagen.
GUI	 
Observaciones	• Usar FrameLayout

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

- Application name: AppFrameLayout
- Company Domain: personal.miEmpresa.com

Paso 2: En el nombre de la actividad coloque maImagenes.

Paso 3: Añada los siguientes componentes al proyecto:

Archivo	Ruta dentro de la aplicación
	res/drawable
	res/drawable

Paso 4: Asigne la siguiente distribución de los archivos en el explorador de proyectos:



Paso 5: La ventana Component Tree debe mostrarse de la siguiente manera:



Ahora bien, véase el código de cada uno de los archivos necesarios para la aplicación:

STRINGS.XML | **PAQUETE: APP/RES/VALUES**

```
<resources>
    <string name="app_name">Manejo de FrameLayout</string>
</resources>
```

ANDROIDMANIFEST.XML | **PAQUETE: APP/MANIFEST**

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

ACTIVITY_MA_IMAGENES.XML | **PAQUETE: APP/RES/LAYOUT**

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.miempresa.personal.appframelayout.maImagenes">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/ivFondo1"
        android:src="@drawable/fondo1"
        android:scaleType="centerCrop"
        android:onClick="muestraImagen"/>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/ivFondo2"
        android:src="@drawable/fondo2"
        android:scaleType="centerCrop"
        android:onClick="muestraImagen"/>
</FrameLayout>
```

MAİMAGENES.JAVA | **PAQUETE: APP/JAVA**

```
package com.miempresa.personal.appframelayout;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;

public class maImagenes extends AppCompatActivity {

    //GLOBAL
    ImageView ivFondo1,ivFondo2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ma_imagenes);

        //Instanciando a las imagenes
        ivFondo1 = (ImageView) findViewById(R.id.ivFondo1);
        ivFondo2 = (ImageView) findViewById(R.id.ivFondo2);
    }

//Método que determina que imagen se muestra
    public void muestraImagen(View view){
        if (view.getId() == R.id.ivFondo1){
            ivFondo1.setVisibility(View.GONE);
            ivFondo2.setVisibility(View.VISIBLE);
        }else{
            ivFondo2.setVisibility(View.GONE);
            ivFondo1.setVisibility(View.VISIBLE);
        }
    }
}
```

EJEMPLO FRAMELAYOUT 2

FrameLayout - Interfaz de una tienda virtual

Objetivo	Implementar una aplicación que muestre el aspecto de una tienda virtual usando FrameLayout.
GUI	
Observaciones	• Usar FrameLayout

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

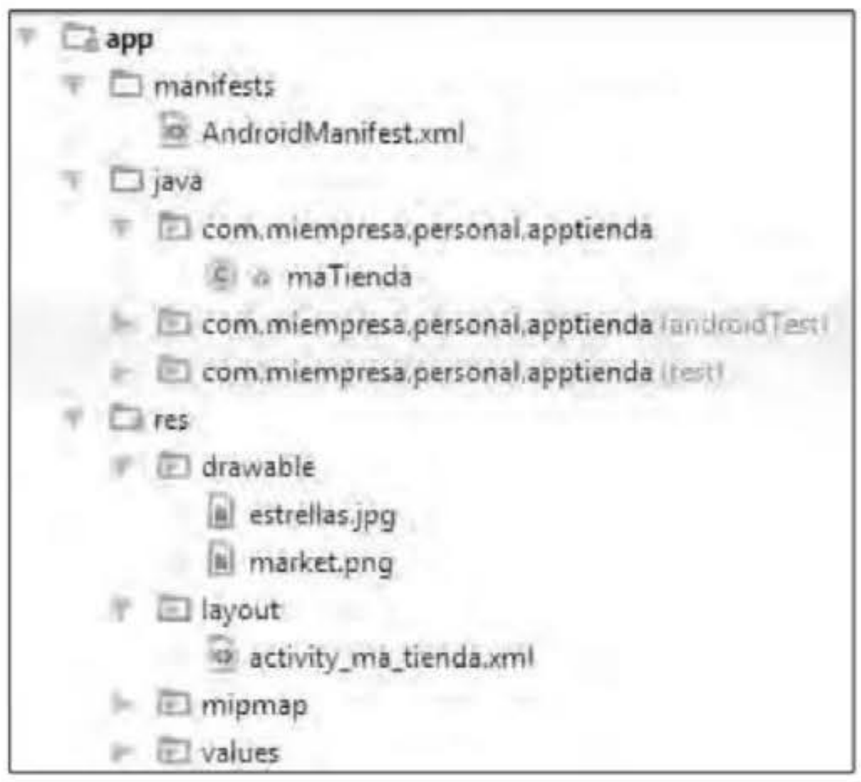
- Application name: AppTienda
- Company Domain: personal.miEmpresa.com

Paso 2: En el nombre de la actividad coloque maTienda.

Paso 3: Añada los siguientes componentes al proyecto:

ARCHIVO	RUTA DENTRO DE LA APLICACIÓN
 market.png	res/drawable
 estrellas.jpg	res/drawable

Paso 4: Asigne la siguiente distribución de los archivos en el explorador de proyectos:



Paso 5: La ventana Component Tree debe mostrarse de la siguiente manera:



Ahora bien, véase el código de cada uno de los archivos necesarios para la aplicación:

STRINGS.XML	PAQUETE: APP/RES/VALUES

```
<resources>
    <string name="app_name">Tienda Virtual con FrameLayout</string>
</resources>
```

AndroidManifest.xml | **Paquete: app/manifest**

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

activity_ma_tienda.xml | **Paquete: app/res/layout**

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.miempresa.personal.apptienda.maTienda">

    <ImageView
        android:layout_width="112dp"
        android:layout_height="112dp"
        android:id="@+id/imageView"
        android:layout_gravity="left|center_vertical"
        android:src="@drawable/market" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="MI TIENDA VIRTUAL"
        android:id="@+id/tvTitulo"
        android:layout_gravity="center"
        android:layout_marginBottom="30dp"
        android:textSize="16dp"
        android:textColor="@color/abc_input_method_navigation_guard" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:text="10,000 - 8MB"
        android:id="@+id/textView"
        android:layout_marginTop="25dp"
        android:layout_gravity="center"
        android:textColor="@color/abc_background_cache_hint_selector_material_dark" />
```

```
    <ImageView
        android:layout_width="121dp"
        android:layout_height="23dp"
        android:id="@+id/imageView2"
        android:layout_gravity="center"
        android:src="@drawable/estrellas" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="instalar"
        android:id="@+id/btnInstalar"
        android:background="@color/colorPrimary"
        android:textColor="#ffffff"
        android:layout_marginRight="5dp"
        android:layout_gravity="right|center_vertical" />

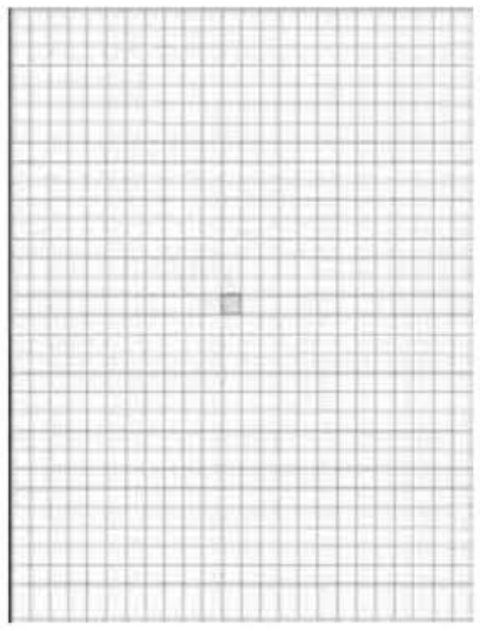
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Libre"
        android:id="@+id/textView2"
        android:layout_marginBottom="35dp"
        android:layout_marginRight="5dp"
        android:layout_gravity="right|center_vertical"
        android:textColor="@color/abc_background_cache_hint_selector_material_dark" />
</FrameLayout>
```

4.1.3 TableLayout

El Layout de tipo TableLayout agrupa los controles en filas y columnas. Para lograr esto el Layout usa componentes de tipo TableRow que permiten reunir controles por cada fila.

La característica de los TableLayout es que, al agregar un control, este crea un TableRow. Esto permitirá tener un mejor control de los elementos en pantalla, pero el lado negativo de esta particularidad es que se genera más código. Hay que tener en cuenta que todos los elementos que se agreguen deben habilitar su propiedad LayoutSpan y especificar la cantidad de columnas que ocupará como espacio.

android:stretchColumns	Indica el número de columnas en las que se puede expandir los objetos para optimizar el espacio libre dejado por las demás columnas que se encuentran a la derecha. Para este caso, el valor a utilizar será uno.



EJEMPLO TABLELAYOUT

FrameLayout - Interfaz de una boleta de pago

Objetivo	Implementar una aplicación que muestre el aspecto de una boleta de pago de un empleado en el que los valores mostrados serán predeterminados. Además, el costo por hora de trabajo será de $20.00, y las horas trabajadas, 40, todos los demás valores deben ser calculados.
GUI	
Observaciones	• Usar TableLayout

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

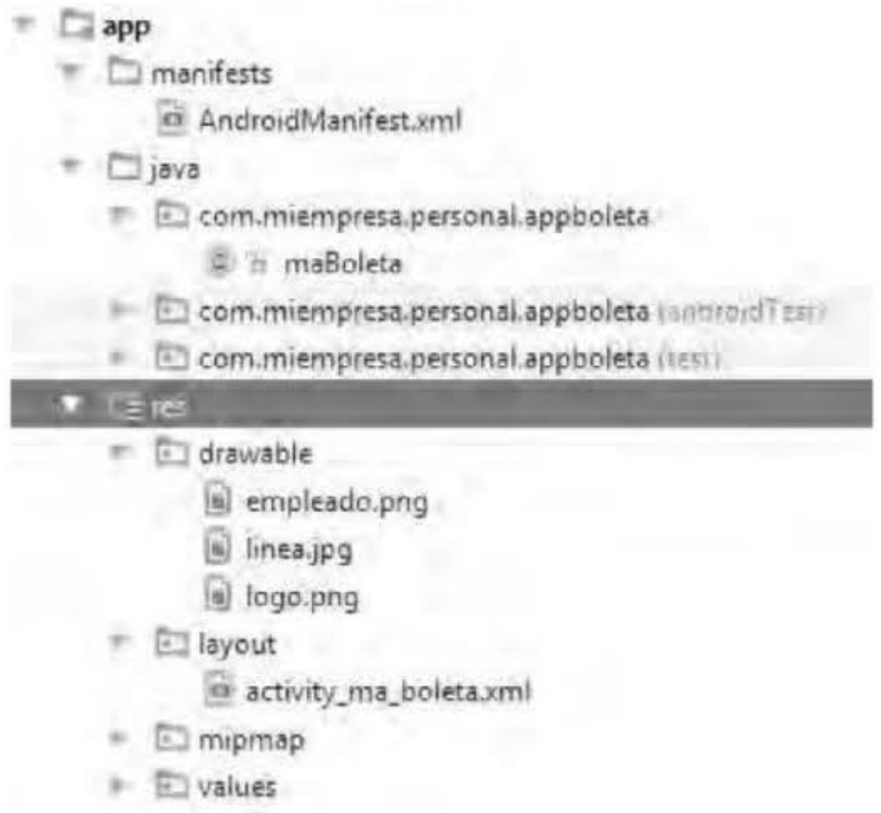
- Application name: AppTienda
- Company Domain: personal.miEmpresa.com

Paso 2: En el nombre de la actividad coloque maTienda.

Paso 3: Añada los siguientes componentes al proyecto:

ARCHIVO	RUTA DENTRO DE LA APLICACIÓN
 logo.png	res/drawable
 empleado.png	res/drawable
línea.jpg	res/drawable

Paso 4: Asigne la siguiente distribución de los archivos en el explorador de proyectos:



Paso 5: La ventana Component Tree debe mostrarse de la siguiente manera:

Modifique el valor inicial del Layout por TableLayout, para mantener la información adecuada en el Layout asigne la propiedad android:stretchColumns="1".

Asimismo, a los imageView asígneles un tamaño adecuado en alto(height) y ancho(width), y añada la propiedad android:layout_span=2 para usar dos columnas y así poder colocar los demás controles de texto en un lugar adecuado.

Finalmente, cuando añada un control al Layout, arrastre sobre la columna uno o dos, respectivamente. Así podrá obtener el aspecto que se observa en el diseño.



Ahora bien, véase el código de cada uno de los archivos necesarios para la aplicación:

STRINGS.XML	PAQUETE: APP/RES/VALUES

```
<resources>
    <string name="app_name">Boleta de pago - Empleado</string>
</resources>
```

ANDROIDMANIFEST.XML | **PAQUETE: APP/MANIFEST**

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

ACTIVITY_MA_BOLETA.XML | **PAQUETE: APP/RES/LAYOUT**

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:stretchColumns="1"
    tools:context="com.miempresa.personal.appboleta.maBoleta">

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >

        <ImageView android:id="@+id/imageView"
            android:layout_column="1"
            android:layout_height="80dp"
            android:layout_width="350dp"
            android:src="@drawable/logo"
            android:layout_span="2"/>
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView android:id="@+id/imageView2"
            android:layout_column="1"
            android:layout_height="80dp"
            android:layout_width="350dp"
            android:src="@drawable/empleado"
            android:layout_span="2"/>

    </TableRow>
```

```
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="BOLETA DE PAGO JUNIO 2018"
            android:id="@+id/tvTitulo"
            android:layout_column="1"
            android:layout_span="3"
            android:layout_marginBottom="5dp"
            android:layout_gravity="center"
            android:textColor="@color/abc_search_url_text_pressed" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:text="COSTO HORA"
            android:id="@+id/tvCosto"
            android:layout_column="1"
            android:textColor="@color/abc_search_url_text_pressed" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:id="@+id/tvCostoHora"
            android:layout_column="2" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:text="HORAS TRABAJADAS"
            android:id="@+id/tvHoras"
            android:layout_column="1"
            android:textColor="@color/abc_search_url_text_pressed" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:id="@+id/tvHorasTrabajadas"
            android:layout_column="2" />
    </TableRow>
```

```
        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <ImageView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/imageView3"
                android:layout_column="1"
                android:src="@drawable/linea"
                android:layout_span="2"
            />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="SALARIO"
                android:id="@+id/tvSalario"
                android:layout_column="1"
                android:textColor="@color/abc_search_url_text_pressed" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:id="@+id/tvMontoSalario"
                android:textColor="@color/abc_search_url_text_pressed"
                android:layout_column="2" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="DESCUENTO"
                android:id="@+id/tvDescuento"
                android:layout_column="1"
                android:textColor="@color/abc_search_url_text_pressed" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:id="@+id/tvMontoDescuento"
                android:textColor="@color/abc_search_url_text_pressed"
                android:layout_column="2" />
        </TableRow>
    <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">
```

```
        <TextView
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:text="NETO"
            android:id="@+id/tvNeto"
            android:layout_column="1"
            android:textColor="@color/abc_search_url_text_pressed" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:id="@+id/tvMontoNeto"
            android:textColor="@color/abc_search_url_text_pressed"
            android:layout_column="2" />
    </TableRow>
</TableLayout>
```

ACTIVITY_MA_BOLETA.XML	PAQUETE: APP/RES/LAYOUT

```
package com.miempresa.personal.appboleta;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class maBoleta extends AppCompatActivity {

    TextView tvCosto,tvHoras;
    TextView tvSalario,tvDescuento,tvNeto;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ma_boleta);
        //Instanciando a los controles
        tvCosto = (TextView) findViewById(R.id.tvCostoHora);
        tvHoras = (TextView) findViewById(R.id.tvHorasTrabajadas);

        tvSalario = (TextView) findViewById(R.id.tvMontoSalario);
        tvDescuento = (TextView) findViewById(R.id.tvMontoDescuento);
        tvNeto = (TextView) findViewById(R.id.tvMontoNeto);

        muestraBoleta();
    }

//Método que muestra la información de la boleta
    public void muestraBoleta(){
        double costo = 20;
        int horas = 40;

        double salario = costo*horas;
        double descuento = salario*0.05;
        double neto = salario-descuento;

        tvCosto.setText("$ "+String.format("%.2f",costo));
        tvHoras.setText(String.valueOf(horas));

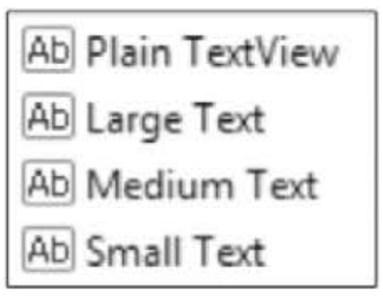
        tvSalario.setText("$ "+String.format("%.2f",salario));
        tvDescuento.setText("$ "+String.format("%.2f",descuento));
        tvNeto.setText("$ "+String.format("%.2f",neto));
    }
}
```

4.2 Controles de texto

Estos controles permiten diseñar un entorno adecuado para el usuario dentro de una aplicación móvil. Entre los principales controles se tienen:

4.2.1 TextView

Es una etiqueta de texto que permite añadir a la plicación información que le resulta visible al usuario. Son textos no editables en tiempo de ejecución, algunos lo conocen como textos estáticos, ya que solo se definen una vez y permanecen con la misma información en toda la aplicación. En Android Studio se presenta una variedad de TextView:



Principales propiedades:

Véase una lista básica de las principales propiedades que presenta el control TextView:

Propiedad	Descripción
android:id	Asigna un nombre identificativo al control, se recomienda asignar un identificador solo si el control TextView muestra un resultado de un determinado proceso; caso contrario, mantenerlo con su nombre inicial. Por ejemplo, se puede asignar el nombre tvFecha para mostrar la fecha actual.
android:text	Asigna un valor textual que se mostrará al usuario. También se puede realizar de manera visual haciendo doble clic sobre el control textView. 
android:layout_height android:layout_width	Permite asignar la dimensión que tiene el control TextView con respecto al layout. De forma estándar aparece el valor wrap_content el cual indica que las dimensiones del control se ajustarán al contenido.
android:gravity	Permite asignar una alineación del texto dentro del marco del control.
android:textSize	Modifica el tamaño del texto.
android:background	Asigna un color de fondo al control TextView.
android:textColor	Determina el color al texto al control TextView.
android:typeface	Asigna el tipo de letra al texto en el control TextView.

EJEMPLO TEXVIEW

TextView - Pantalla de presentación

Objetivo	Implementar una aplicación que muestre el aspecto de una pantalla de presentación usando controles de tipo TextView.
GUI	
Observaciones	- Usar TextView

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

- Application name: AppPresentacion
- Company Domain: personal.miEmpresa.com

Paso 2: En el nombre de la actividad coloque maPresentacion.

Ahora bien, véase el código de cada uno de los archivos necesarios para la aplicación:

STRINGS.XML — **PAQUETE: APP/RES/VALUES**

```
<resources>
    <string name="app_name">Manejo de controles-TextView</string>
</resources>
```

AndroidManifest.xml | **Paquete: app/manifest**

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

activity_ma_presentacion.xml | **Paquete: app/res/layout**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.miempresa.personal.aplicacionejemplo2016.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="EDITORIAL"
        android:id="@+id/textView"
        android:layout_marginTop="45dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="MACRO"
        android:id="@+id/textView2"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/textView"
        android:layout_alignParentLeft="true"
        android:textColor="@color/abc_search_url_text_pressed" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:text="AUTOR: LIC. MANUEL TORRES RE."
        android:id="@+id/textView3"
        android:layout_below="@+id/textView5"
        android:layout_alignParentStart="true"
        android:textColor="@color/abc_search_url_text_pressed" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="DESARROLLO DE APLICACIONES MÓVILES"
        android:id="@+id/textView4"
        android:layout_below="@+id/textView2"
        android:layout_alignParentStart="true"
        android:layout_marginTop="25dp"
        android:textSize="28dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="ANDROID"
        android:id="@+id/textView5"
        android:layout_below="@+id/textView4"
        android:layout_alignParentStart="true"
        android:textColor="@color/abc_search_url_text_pressed" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android es considerado el sistema operativo más vendido en el
        planeta."
        android:id="@+id/textView6"
        android:layout_below="@+id/textView3"
        android:layout_marginTop="32dp"
        android:textSize="16dp" />

</RelativeLayout>
```

Como se puede observar, todos los textos han sido creados bajo la misma etiqueta <TextView>, la diferencia entre los controles es la apariencia Android:textAppearance. Esto se presenta como un atributo dentro del código el cual se puede modificar. Considere la siguiente imagen que indica el inicio y el fin del control textView en el código.



4.2.2 EditText

Es el control que permite la interacción de la aplicación con el usuario, ya que este podrá ingresar valores que probablemente se necesiten para un determinado proceso. Este control permite el ingreso y modificacion de valores de todo tipo, en ese sentido, los controles de tipo EditText tienen la posibilidad de mostrarse a los usuarios de formas distintas como, por ejemplo, para ingresar *password*, correos electrónicos, télefono, fecha, hora, números, etc. En el cuadro de herramientas se muestran los siguientes tipos de entrada:

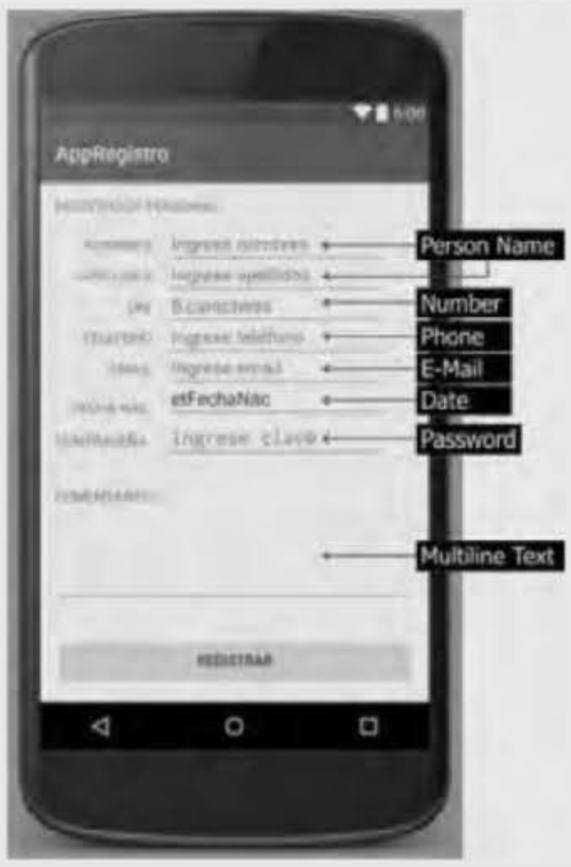


Véase una lista básica de las principales propiedades que presenta el control EditText:

Propiedad	Descripción
android:id	Asigna un nombre identificativo al control. Se recomienda colocar un nombre a todos los controles de tipo EditText porque forman una parte importante dentro de una aplicación móvil. Por ejemplo, si se coloca un EditText para el ingreso del nombre del usuario, podría denominársele a ese control etUsuario.
android:text	No es recomendable llenar esta propiedad, pues es el usuario es quien ingresará algún valor al control. Pero se podría dar la situación que se le envíe información mediante el control como, por ejemplo, algún resultado que el usuario puede modificar.
android:hint	Asigna un texto descriptivo al control EditText, el cual permite al usuario visualizar información referente a cómo debe llenarse correctamente el control.
android:inputType	Configura el tipo de valor que se ingresará por el control EditText esto permite al usuario ver un teclado en su dispositivo de acuerdo al valor que ingresará.
android:maxLength	Define una cantidad límite de caracteres que puede tener el control EditText.
android:singleLine	Permite ingresar un texto en una sola línea bloqueando el botón de cambio de línea.
android:enabled	Determina si es o no editable los datos mostrados en un EditText.
android:focusable	Asigna el modo de cursor activo del control EditText.

EJEMPLO EDITTEXT

Registro de información de personal

Objetivo	Implemente una aplicación que muestre la interfaz del registro de personal de una empresa.
GUI	
Observaciones	- Usar el control EditText

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

- Application name: AppRegistro
- Company Domain: personal.miEmpresa.com

Paso 2: En el nombre de la actividad coloque MainActivity.

Paso 3: El código que presenta la aplicación es:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.miempresa.personal.appregistro.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="REGISTRO DE PERSONAL"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="NOMBRES"
            android:id="@+id/textView2"
            android:layout_marginStart="30dp"
            android:layout_gravity="right"
            android:layout_alignBaseline="@+id/etNombres"
            android:layout_alignBottom="@+id/etNombres"
            android:layout_alignParentStart="true" />
    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="APELLIDOS"
            android:id="@+id/textView3"
            android:layout_gravity="right"
            android:layout_alignBaseline="@+id/etApellidos"
            android:layout_alignBottom="@+id/etApellidos"
            android:layout_alignEnd="@+id/textView2" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="CONTRASEÑA"
            android:id="@+id/textView4"
            android:layout_centerVertical="true"
            android:layout_alignStart="@+id/textView" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="DNI"
            android:id="@+id/textView5"
            android:layout_gravity="right"
            android:layout_alignBaseline="@+id/etDNI"
            android:layout_alignBottom="@+id/etDNI"
            android:layout_alignEnd="@+id/textView3" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TELEFONO"
            android:id="@+id/textView6"
            android:layout_gravity="right"
            android:layout_alignBaseline="@+id/etFono"
            android:layout_alignBottom="@+id/etFono"
            android:layout_alignEnd="@+id/textView5" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="EMAIL"
            android:id="@+id/textView7"
            android:layout_gravity="right"
            android:layout_alignBaseline="@+id/etCorreo"
            android:layout_alignBottom="@+id/etCorreo"
            android:layout_toStartOf="@+id/etNombres" />
```

```
<EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/etNombres"
        android:layout_marginTop="8dp"
        android:hint="Ingrese nombres"
        android:layout_below="@+id/textView"
        android:layout_toEndOf="@+id/textView2"
        android:layout_marginStart="15dp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/etApellidos"
        android:hint="Ingrese apellidos"
        android:layout_below="@+id/textView2"
        android:layout_alignStart="@+id/etNombres" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/etDNI"
        android:layout_below="@+id/textView3"
        android:layout_alignStart="@+id/etApellidos"
        android:hint="8 caracteres"
        android:maxLength="8" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="phone"
        android:ems="10"
        android:id="@+id/etFono"
        android:layout_below="@+id/textView5"
        android:layout_alignStart="@+id/etDNI"
        android:hint="Ingrese teléfono" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress"
        android:ems="10"
        android:id="@+id/etCorreo"
        android:layout_below="@+id/textView6"
        android:layout_alignStart="@+id/etFono"
        android:hint="Ingrese email" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="FECHA NAC."
        android:id="@+id/textView8"
        android:layout_below="@+id/etCorreo"
        android:layout_toStartOf="@+id/etNombres"
        android:layout_marginTop="10dp" />
```

```
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="date"
        android:ems="10"
        android:id="@+id/editText6"
        android:layout_below="@+id/textView7"
        android:layout_alignStart="@+id/etCorreo"
        android:hint="Fecha de nacimiento"
        android:text="etFechaNac" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:id="@+id/etClave"
        android:layout_below="@+id/textView8"
        android:layout_alignStart="@+id/editText6"
        android:hint="Ingrese clave" />

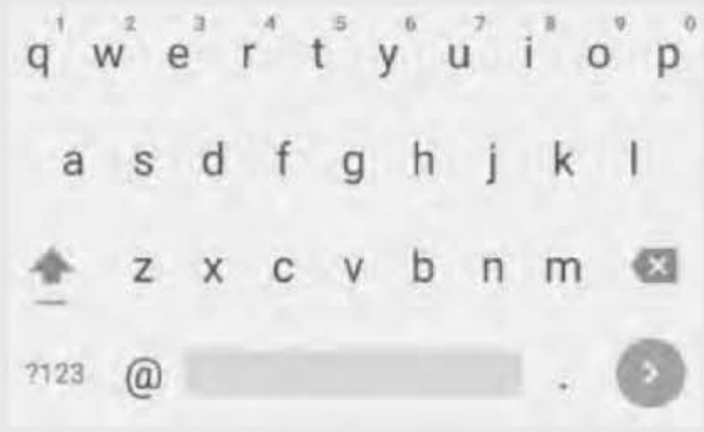
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textMultiLine"
        android:ems="10"
        android:id="@+id/etComentario"
        android:layout_marginTop="56dp"
        android:layout_below="@+id/etClave"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:lines="3"
        android:singleLine="false" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="COMENTARIOS"
        android:id="@+id/textView9"
        android:layout_below="@+id/etClave"
        android:layout_alignParentStart="true"
        android:layout_marginTop="27dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="REGISTRAR"
        android:id="@+id/btnRegistrar"
        android:layout_alignParentBottom="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true" />

</RelativeLayout>
```

Todos los controles usados en la aplicación corresponden al EditText lo que les diferencia es el tipo de entrada (inputType). Esto hace que al ejecutar la aplicación el dispositivo móvil muestre un teclado virtual adecuado para el ingreso de dicho valor. Véase una lista de estos tipos de teclados:

TIPO DE ENTRADA	TECLADO
Plain Text Person Name Password Multiline Text	 Presenta letras (mayúsculas y minúsculas), números y caracteres especiales.
Number Number (Signed) Number (Decimal)	 Presenta números entre 0 y 9.
Phone	 Presenta números entre 0 y 9, además de caracteres propios de un número telefónico.
e-mail	 Presenta letras, números, caracteres especiales y el simbolo de @.
Date	 Presenta números y caracteres propios de un registro de fecha.

4.3 CONTROLES DE ACCIÓN

Estos controles permiten tener el control de las actividades que puede realizar una aplicación móvil. Véase los diferentes botones que presenta Android para el diseño de aplicaciones:

4.3.1 BUTTON

Es un control que permite realizar una determinada acción dentro de una aplicación móvil. Esto solo sucederá cuando el usuario presione dicho botón.

Véase una lista básica de las principales propiedades que presenta el control Button:

PROPIEDAD	DESCRIPCIÓN
android:id	Asigna un nombre identificativo al control. Es recomendable colocar un nombre a todos los controles de tipo Button, ya que forman una parte importante dentro de una aplicación móvil. Por ejemplo, si se coloca un botón que registre los datos de los empleados podría denominársele btnRegistrar.
android:text	Permite mostrar un texto representativo al control button para que el usuario, al seleccionar un botón, asocie la actividad al título del botón.
android:enable	Permite administrar la forma de uso del botón, por defecto se encuentra habilitado (true), pero el desarrollador podría inhabilitarlo por algún criterio.
android:onClick	Es el evento que ocurre cuando se presiona el botón. Para esto se debe colocar el nombre del método que se ejecutará.
android:textColor	Asigna un color al texto contenido en el botón.

EJEMPLO BUTTON

Login de usuario

Objetivo	Implementar una aplicación que permita mostrar la interfaz que pueda tener un usuario para que realice su logueo al ingresar el nombre de usuario y su clave; además, se debe mostrar qué valores fueron registrados.
GUI	
Observaciones	- Usar el control Button

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

- Application name: AppLogin
- Company Domain: personal.miEmpresa.com

Paso 2: En el nombre de la actividad coloque MainActivity.

Paso 3: Asigne el siguiente código en el archivo activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.miempresa.personal.applogeo.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="LOGIN DE USUARIO"
        android:id="@+id/textView2"
        android:textColor="#000000"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="USUARIO"
        android:id="@+id/textView"
        android:textColor="#000000"
        android:layout_below="@+id/textView2"
        android:layout_alignParentStart="true"
        android:layout_marginTop="20dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CLAVE"
        android:id="@+id/textView4"
        android:textColor="#000000"
        android:layout_alignTop="@+id/textView3"
        android:layout_alignEnd="@+id/etClave"
        android:layout_marginEnd="63dp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/etUsuario"
        android:layout_alignBaseline="@+id/textView"
        android:layout_alignBottom="@+id/textView"
        android:layout_toEndOf="@+id/textView2"
        android:hint="Ingrese nombre" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:id="@+id/etClave"
        android:layout_below="@+id/etUsuario"
        android:layout_alignStart="@+id/etUsuario"
        android:hint="Ingrese clave" />
```

```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="MOSTRAR"
        android:id="@+id/btnMostrar"
        android:onClick="muestraDatos"
        android:layout_below="@+id/etClave"
        android:layout_alignParentStart="true"
        android:nestedScrollingEnabled="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="USUARIO"
        android:id="@+id/textView3"
        android:textColor="#010101"
        android:typeface="normal"
        android:singleLine="false"
        android:layout_below="@+id/btnMostrar"
        android:layout_alignEnd="@+id/tvUsuario"
        android:layout_marginTop="23dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="tvUsuario"
        android:id="@+id/tvUsuario"
        android:textColor="#010101"
        android:layout_alignBaseline="@+id/tvClave"
        android:layout_alignBottom="@+id/tvClave"
        android:layout_alignParentStart="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CLAVE"
        android:id="@+id/textView6"
        android:textColor="#010101"
        android:typeface="normal"
        android:singleLine="false"
        android:layout_alignBaseline="@+id/etClave"
        android:layout_alignBottom="@+id/etClave"
        android:layout_alignParentStart="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="tvClave"
        android:id="@+id/tvClave"
        android:textColor="#010101"
        android:layout_below="@+id/textView4"
        android:layout_alignRight="@+id/etClave"
        android:layout_marginRight="55dp" />
</RelativeLayout>
```

Asegúrese que en los atributos del control Button se encuentre la siguiente sentencia: android:onClick="-muestraDatos", donde se especifica el nombre del método "muestraDatos", el cual se implementa en el archivo MainActivity.java que se muestra a continuación:

```
package com.miempresa.personal.applogeo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    //Declaracion GLOBAL
    private EditText etUsu, etCla;
    private TextView tvUsu, tvCla;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Instanciando a los controles
        etUsu = (EditText)findViewById(R.id.etUsuario);
        etCla = (EditText)findViewById(R.id.etClave);

        tvUsu = (TextView)findViewById(R.id.tvUsuario);
        tvCla = (TextView)findViewById(R.id.tvClave);
    }

    public void muestraDatos(View view){
        //Capturando los valores desde los controles
        String usuario = etUsu.getText().toString();
        String clave = etCla.getText().toString();

        //Mostrando los resultados
        tvUsu.setText(usuario);
        tvCla.setText(clave);
    }
}
```

En la declaración global de la aplicación se crean objetos de la clase EditText y TextView para poder usar los controles especificados en el Activity (pantalla de la aplicación móvil).

```
private EditText etUsu, etCla;
private TextView tvUsu, tvCla;
```

Luego, se instancian los objetos a los controles mediante el método findViewById. Dentro de su parámetro se hace referencia a los controles mediante la clase R. Todo esto se debe realizar dentro del método onCreate.

```
etUsu = (EditText)findViewById(R.id.etUsuario);
etCla = (EditText)findViewById(R.id.etClave);
tvUsu = (TextView)findViewById(R.id.tvUsuario);
tvCla = (TextView)findViewById(R.id.tvClave);
```

Finalmente, se implementa el método muestraDatos en el cual se comienza capturando los datos con el método getText() por cada control, luego son devueltos a los controles TextView usando el método setText; como se muestra en el siguiente código:

```
public void muestraDatos(View view){
    //Capturando los valores desde los controles
    String usuario = etUsu.getText().toString();
    String clave = etCla.getText().toString();

    //Mostrando los resultados
    tvUsu.setText(usuario);
    tvCla.setText(clave);
}
```

No se olvide que este método debe ser referenciado en el atributo onClick del botón en el archivo activity_main.xml.

4.3.2 ToggleButton

Es una etiqueta de texto que permite añadir a la aplicación información que le resulta visible al usuario. Son textos que no son editables en tiempo de ejecución, algunos lo conocen como textos estáticos, ya que solo se definen una vez y permanecen con la misma información en toda la aplicación. En Android Studio se presenta una variedad de TextView:

Véase la siguiente lista básica de las principales propiedades que presenta el control ToggleButton:

Propiedad	Descripción
android:id	Asigna un nombre identificativo al control. Es recomendable colocar un nombre a todos los controles de tipo ToggleButton, ya que forman una parte importante dentro de una aplicación móvil. Por ejemplo, para activar un proceso se le puede denominar tbActivar.
android:onClick	Especifica el proceso que se ejecutará al seleccionar el control ToggleButton.
android:textOff	Define el texto que se mostrará cuando el botón se encuentre en el estado apagado.
android:textOn	Especifica el texto que se mostrará cuando el botón se encuentre en el estado de encendido.
android:checked	El valor true especifica que el ToggleButton se encuentre en el estado activo.

EJEMPLO TOGGLEBUTTON

Cambio de color de fondo

Objetivo	Implementar una aplicación que permita cambiar de color el fondo de la aplicación. Esto debe realizarse por medio de un boton de tipo toggle.
GUI	
Observaciones	- Usar el control ToggleButton

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

- Application name: AppColorFondo
- Company Domain: personal.miEmpresa.com

Paso 2: En el nombre de la actividad coloque MainActivity.

Paso 3: Asigne el siguiente código en el archivo activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:id="@+id/rlColorFondo"
    tools:context="com.miempresa.personal.appcolorfondo.MainActivity">

    <ToggleButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New ToggleButton"
        android:id="@+id/tbColor"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="27dp"
```

```
        android:onClick="cambiaColor"
        android:textOff="Fondo Rojo"
        android:textOn="Fondo Azul"
        android:checked="false" />
</RelativeLayout>
```

Asegúrese que en los atributos del RelativeLayout se encuentre la siguiente sentencia: android:id ="rlColorFondo", donde se especifica el nombre del espacio de trabajo al cual se le modificará su color. Asimismo, asigne la sentencia en el ToggleButton android:onClick="cambiaColor", el cual atribuye el método al evento onClick del botón tal como se especifica en el archivo MainActivity.java y que se muestra en el siguiente código:

```
package com.miempresa.personal.appcolorfondo;

import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.RelativeLayout;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {

    //GLOBAL
    RelativeLayout rlColor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        rlColor = (RelativeLayout)findViewById(R.id.rlColorFondo);
    }

    public void cambiaColor(View view){
        boolean activado = ((ToggleButton) view).isChecked();
        if (activado){
            rlColor.setBackgroundColor(Color.RED);
        }else{
            rlColor.setBackgroundColor(Color.BLUE);
        }
    }
}
```

En la declaración global se menciona el objeto rlColor de la clase RelativeLayout que permitirá tener acceso al espacio de trabajo de la aplicación. Dentro del evento onCreate instanciamos dicho objeto con el espacio de trabajo real mediante la sentencia **rlColor**=(RelativeLayout)findViewById(R.id.**rlColorFondo**).

Finalmente, se implementa el método "cambiaColor" donde se captura el estado actual del botón con la sentencia **boolean** activado = ((ToggleButton) view).isChecked(); luego se verifica si se encuentra activado para asignar el color rojo; caso contrario, el color azul.

Una variación de la aplicación se puede tomar a partir de un control de tipo Switch para que realice la misma actividad:




```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:id="@+id/rlColorFondo"
    tools:context="com.miempresa.personal.appcolorfondo.MainActivity">

    <Switch
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textOff="Azul"
        android:textOn="Rojo"
        android:id="@+id/tbColores"
        android:layout_gravity="center_horizontal"
        android:text="Cambio de color"
        android:checked="false"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true" />

</RelativeLayout>
```

Asegúrese que en los atributos del RelativeLayout se encuentre la siguiente sentencia: android:id =”rlColorFondo”, donde se especifica el nombre del espacio de trabajo al cual se modificará su color. Asimismo, se asigna el nombre a la sentencia Switch android:id =”tbColores”. El archivo MainActivity.java se muestra con el siguiente código:

```
package com.miempresa.personal.appcolorfondo;

import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.RelativeLayout;
import android.widget.Switch;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {

    //GLOBAL
    RelativeLayout rlColor;
    Switch aSwitch;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        rlColor = (RelativeLayout)findViewById(R.id.rlColorFondo);
        aSwitch = (Switch) findViewById(R.id.tbColores);
        aSwitch.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener()
{
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
                if (isChecked){
                    rlColor.setBackgroundColor(Color.RED);
                }else{
                    rlColor.setBackgroundColor(Color.BLUE);
                }
            }
        });
    }
}
```

4.3.3 ImageButton

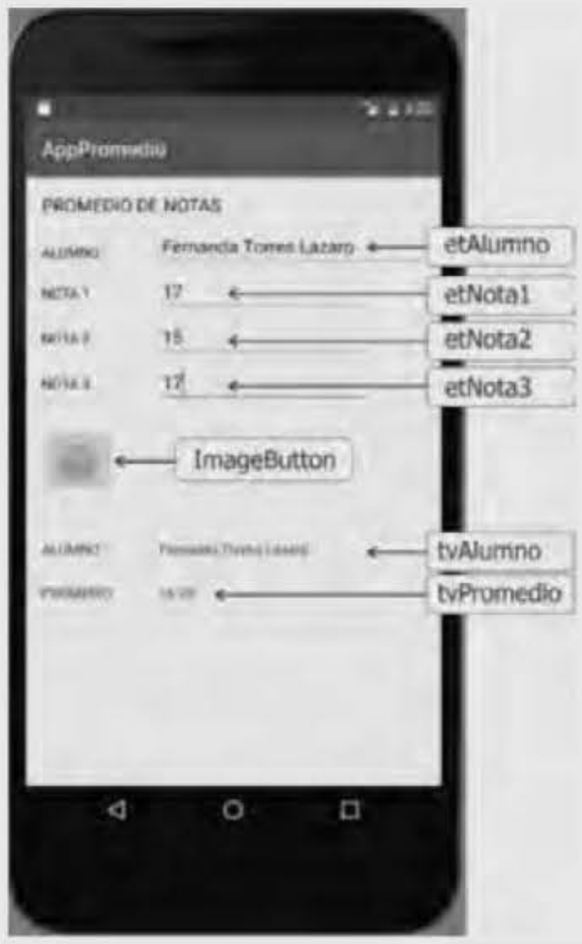
Tiene el mismo comportamiento que los controles de tipo botón con la diferencia que aquí no se podrán visualizar textos, ya que la imagen asignada ocupa todo el control.

Véase la siguiente lista básica de las principales propiedades que presenta el control ImageButton:

Propiedad	Descripción
android:id	Asigna un nombre identificativo al control ImageButton.
android:onClick	Asigna el método que se procesará cuando el botón de imagen sea seleccionado por el usuario.
android:src	Asigna una imagen al control ImageButton. Recuerde que este ocupará todo el espacio necesario para mostrarse y que no deja espacio a los textos.

EJEMPLO IMAGEBUTTON

Promedio de notas

Objetivo	Implementar una aplicación que permita calcular el promedio de notas de un alumno.
GUI	
Observaciones	- Usar el control ImageButton

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

- Application name: AppPromedio
- Company Domain: personal.miEmpresa.com

Paso 2: En el nombre de la actividad coloque MainActivity.

Paso 3: Asigne el siguiente código en el archivo activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.miempresa.personal.appromedio.MainActivity">
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="PROMEDIO DE NOTAS"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:textColorHighlight="#000000"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:textSize="18dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ALUMNO"
        android:id="@+id/textView2"
        android:layout_marginTop="25dp"
        android:textColorHighlight="#000000"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:layout_below="@+id/textView"
        android:layout_alignParentStart="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/etNota1"
        android:maxLength="2"
        android:textColorHighlight="#000000"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:layout_below="@+id/etNota2"
        android:layout_alignLeft="@+id/tvAlumno" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/etAlumno"
        android:textColorHighlight="#000000"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:layout_below="@+id/textView"
        android:layout_alignStart="@+id/etNota1"
        android:layout_marginTop="7dp"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="NOTA 1"
        android:id="@+id/textView3"
        android:textColorHighlight="#000000"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:layout_alignBaseline="@+id/etNota2"
        android:layout_alignBottom="@+id/etNota2"
        android:layout_alignParentStart="true" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="NOTA 2"
        android:id="@+id/textView4"
        android:textColorHighlight="#000000"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:layout_alignBaseline="@+id/etNota1"
        android:layout_alignBottom="@+id/etNota1"
        android:layout_alignParentStart="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/etNota2"
        android:maxLength="2"
        android:textColorHighlight="#000000"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:layout_below="@+id/etAlumno"
        android:layout_alignStart="@+id/tvAlumno" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="NOTA 3"
        android:id="@+id/textView5"
        android:textColorHighlight="#000000"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:layout_alignBaseline="@+id/etNota3"
        android:layout_alignBottom="@+id/etNota3"
        android:layout_alignParentStart="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/etNota3"
        android:maxLength="2"
        android:textColorHighlight="#000000"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:layout_below="@+id/etNota1"
        android:layout_alignLeft="@+id/tvAlumno" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton"
        android:src="@android:drawable/sym_def_app_icon"
        android:layout_below="@+id/etNota3"
        android:layout_alignParentStart="true"
        android:layout_marginTop="21dp"
        android:onClick="calculaPromedio"
    />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="ALUMNO"
        android:id="@+id/textView6"
        android:layout_marginTop="44dp"
        android:textSize="14dp"
        android:layout_below="@+id/imageButton"
        android:layout_alignParentStart="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="PROMEDIO"
        android:id="@+id/textView7"
        android:textSize="14dp"
        android:layout_below="@+id/textView6"
        android:layout_alignParentStart="true"
        android:layout_marginTop="24dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/tvAlumno"
        android:textSize="14dp"
        android:layout_marginStart="44dp"
        android:layout_above="@+id/textView7"
        android:layout_toEndOf="@+id/imageButton" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/tvPromedio"
        android:textSize="14dp"
        android:layout_marginRight="89dp"
        android:layout_alignTop="@+id/textView7"
        android:layout_alignStart="@+id/tvAlumno" />

</RelativeLayout>
```

Asegúrese que el atributo onClick del ImagenButton se direccione al método "calculaPromedio" que se implementará en el archivo MainActivity.java como se muestra en el siguiente código:

```
package com.miempresa.personal.apppromedio;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    //GLOBAL
    EditText etAlumno,etNota1,etNota2,etNota3;
    TextView tvAlumno,tvPromedio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Instancia de los controles
        etAlumno = (EditText) findViewById(R.id.etAlumno);
        etNota1 = (EditText) findViewById(R.id.etNota1);
        etNota2 = (EditText) findViewById(R.id.etNota2);
        etNota3 = (EditText) findViewById(R.id.etNota3);

        tvAlumno = (TextView) findViewById(R.id.tvAlumno);
        tvPromedio = (TextView) findViewById(R.id.tvPromedio);
    }

    public void calculaPromedio(View view){
        //Capturando los datos ingresados
        String alumno = etAlumno.getText().toString();
        int nota1= Integer.parseInt(etNota1.getText().toString());
        int nota2= Integer.parseInt(etNota2.getText().toString());
        int nota3= Integer.parseInt(etNota3.getText().toString());

        //Calculando el promedi
        double promedio = (nota1+nota2+nota3)/3.0;

        //Imprimir los resultados
        tvAlumno.setText(alumno);
        tvPromedio.setText(String.format("%.2f",promedio));
    }
}
```

4.4 Intent

Es una petición para hacer una determinada acción que podría realizarse dentro de la aplicación o fuera de esta. También podría describirse como una intención de realizar una tarea como, por ejemplo, mostrar otra actividad, enviar un mensaje de texto, etc.

Finalmente, se puede decir que un objeto de la clase Intent es un elemento básico de comunicación entre los distintos componentes de una aplicación Android. Su uso depende del tipo de aplicación que se desea implementar. Se pueden mencionar las siguientes actividades:

- **Inicio de una nueva actividad**

  Esto se realiza cuando una aplicación necesita navegar entre los componentes activity. Se debe considerar que aquí se presentan dos situaciones:

  - **Invocar una actividad sin paso de valor**

    Se crea un objeto de la clase Intent y por medio del método constructor se envía la información de la actividad origen seguidamente del destino. Este último usa la propiedad class para su invocación. Finalmente, se realiza la activación mediante la sentencia StartActivity.

```
//Objeto de la clase Intent
Intent objI= new Intent(actividadOrigen.this,actividadDestino.class);
//Activando el intent
startActivity(objI);
```

- **Invocar una actividad con paso de valor**

  Se crea un objeto de la clase Intent y, por medio del método constructor, se envía, como parámetro, información de la actividad de origen seguidamente del destino. Asimismo, los parámetros serán especificados mediante el método putExtra, el cual tiene como atributos el nombre del parámetro que será capturado desde otra actividad, luego el control que tiene la información será capturado mediante el método getText(). Finalmente, se realiza la activación mediante la sentencia StartActivity.

```
//Objeto de la clase Intent
Intent objI= new Intent(actividadOrigen.this,actividadDestino.class);
//Enviando parámetros
iInformacion.putExtra("parametro1",control.getText().toString());
iInformacion.putExtra("parametro2",control.getText().toString());
//Activando el intent
startActivity(objI);
```

EJEMPLO INTENT

Intent - Registro de información de empleados

Objetivo	Implementar una aplicación que registre los datos del personal, donde los datos registrados se muestren en una segunda ventana.
GUI	
	
Observaciones	- Usar la clase Intent. - Los valores ingresados deben ser mostrados en otro activity.

Pasos para la solución:

Paso 1: Cree un nuevo proyecto (*File > New > New project*).

- Application name: AppRegistroPersonal
- Company Domain: personal.miEmpresa.com

Paso 2: En el nombre de la actividad coloque maRegistro.

Paso 3: Agregue un activity al proyecto, para ello:

- Haga clic derecho sobre el paquete Layout del proyecto.
- Seleccione *New > activity > Empty Activity*.
- Asigne el nombre maInformacion.
- En el Package Name coloque com.miempresa.personal.

Paso 4: Añada los siguientes componentes al proyecto:

Archivo	Ruta dentro de la aplicación
 logo.png	res/drawable
 banner.jpg	res/drawable

Paso 5: Asigne la siguiente distribución de los archivos en el explorador de proyectos:



Paso 6: La ventana Component Tree debe mostrarse de la siguiente manera:

ACTIVITY_MA_REGISTRO.XML



ACTIVITY_MA_INFORMACION.XML



Ahora bien, véase el código de cada uno de los archivos necesarios para la aplicación:

STRINGS.XML — **PAQUETE: APP/RES/VALUES**

```
<resources>
    <string name="app_name">Registro de Personal</string>
</resources>
```

ANDROIDMANIFEST.XML — **PAQUETE: APP/MANIFEST**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.miempresa.personal.appregistropersonal">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".maRegistro">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".maInformacion"></activity>
    </application>
</manifest>
```

ACTIVITY_MA_REGISTRO.XML	PAQUETE: APP/RES/LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context="com.miempresa.personal.appregistropersonal.maRegistro"
    android:weightSum="1">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/ivLogo"
        android:src="@drawable/logo"
        android:adjustViewBounds="true"/>
        <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/ivBanner"
        android:src="@drawable/banner"
        android:adjustViewBounds="true" />

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stretchColumns="1" >

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:text="REGISTRO DE PERSONAL"
                android:id="@+id/tvTitulo"
                android:layout_column="1"
                android:gravity="center"
                android:layout_span="2"
                android:textColor="@color/abc_input_method_navigation_guard" />
        </TableRow>
        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="APELLIDOS"
                android:id="@+id/textView"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />
```

```
    <EditText
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:inputType="textPersonName"
                    android:ems="10"
                    android:id="@+id/etApellidos"
                    android:layout_column="2" />
            </TableRow>

            <TableRow
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="NOMBRES"
                    android:id="@+id/textView2"
                    android:layout_column="1"
                    android:textColor="@color/abc_input_method_navigation_guard" />

                <EditText
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:inputType="textPersonName"
                    android:ems="10"
                    android:id="@+id/etNombres"
                    android:layout_column="2" />
            </TableRow>

            <TableRow
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                <TextView
                    android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="TELEFONO"
                    android:id="@+id/textView3"
                    android:layout_column="1"
                    android:textColor="@color/abc_input_method_navigation_guard" />

                <EditText
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:inputType="phone"
                    android:ems="10"
                    android:id="@+id/etTelefono"
                    android:layout_column="2" />
            </TableRow>

            <TableRow
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                <TextView
                    android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="EMAIL"
                    android:id="@+id/textView4"
                    android:layout_column="1"
                    android:textColor="@color/abc_input_method_navigation_guard" />
```

```
    <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:inputType="textPersonName"
                android:ems="10"
                android:id="@+id/etApellidos"
                android:layout_column="2" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="NOMBRES"
                android:id="@+id/textView2"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />

            <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:inputType="textPersonName"
                android:ems="10"
                android:id="@+id/etNombres"
                android:layout_column="2" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="TELEFONO"
                android:id="@+id/textView3"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />

            <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:inputType="phone"
                android:ems="10"
                android:id="@+id/etTelefono"
                android:layout_column="2" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="EMAIL"
                android:id="@+id/textView4"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />
```

```
            <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:inputType="textEmailAddress"
                android:ems="10"
                android:id="@+id/etEmail"
                android:layout_column="2" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="FECHA NAC."
                android:id="@+id/textView5"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />

            <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:inputType="date"
                android:ems="10"
                android:id="@+id/etFechaNac"
                android:layout_column="2" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="REGISTRAR"
                android:id="@+id/btnRegistrar"
                android:layout_column="2"
                android:background="@android:color/holo_blue_dark"
                android:textColor="@color/abc_primary_text_material_dark"
                android:onClick="muestraInformacion"
                android:textSize="20dp" />
        </TableRow>
    </TableLayout>

</LinearLayout>
```

maRegistro.java	Paquete: app/java

```
package com.miempresa.personal.appregistropersonal;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class maRegistro extends AppCompatActivity {

    //GLOBAL
    EditText etApellidos, etNombres, etEmail, etTelefono, etFechaNac;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ma_registro);

        //Instanciando los controles
        etApellidos = (EditText) findViewById(R.id.etApellidos);
        etNombres = (EditText) findViewById(R.id.etNombres);
        etTelefono = (EditText) findViewById(R.id.etTelefono);
        etEmail = (EditText) findViewById(R.id.etEmail);
        etFechaNac = (EditText) findViewById(R.id.etFechaNac);
    }

    //Metodo que envia informacion capturada
    public void muestraInformacion(View view){
        //Objeto de la clase Intent
        Intent iInformacion= new Intent(maRegistro.this,maInformacion.class);

        //Enviando informacion
        iInformacion.putExtra("apellidos",etApellidos.getText().toString());
        iInformacion.putExtra("nombres",etNombres.getText().toString());
        iInformacion.putExtra("telefono",etTelefono.getText().toString());
        iInformacion.putExtra("email",etEmail.getText().toString());
        iInformacion.putExtra("fechaNac",etFechaNac.getText().toString());

        //Activando el intent
        startActivity(iInformacion);
    }
}
```

ACTIVITY_MA_INFORMACION.XML	PAQUETE: APP/RES/LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".maInformacion">
```

```
    <ImageView
    android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@drawable/logo"
        android:adjustViewBounds="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />

    <ImageView
    android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView2"
        android:src="@drawable/banner"
        android:adjustViewBounds="true"
        android:layout_below="@+id/imageView"
        android:layout_alignParentLeft="true" />

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/imageView2"
        android:layout_alignParentStart="true"
        android:stretchColumns="1"
        >

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:text="INFORMACIÓN DE REGISTRO"
                android:id="@+id/textView6"
                android:gravity="center"
                android:layout_span="2"
                android:layout_marginBottom="20dp"
                android:textColor="@color/abc_input_method_navigation_guard" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="APELLIDOS"
                android:id="@+id/textView8"
                android:layout_column="1"
                android:textColor="@color/abc_primary_text_disable_only_material_light" />
        </TableRow>
```

```
        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:id="@+id/tvApellidos"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="NOMBRES"
                android:id="@+id/textView9"
                android:layout_column="1"
                android:textColor="@color/abc_primary_text_disable_only_material_light" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:id="@+id/tvNombres"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="TELEFONO"
                android:id="@+id/textView11"
                android:layout_column="1"
                android:textColor="@color/abc_primary_text_disable_only_material_light" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:id="@+id/tvFono"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />
        </TableRow>
```

```
        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="CORREO ELECTRONICO"
                android:id="@+id/textView13"
                android:layout_column="1"
                android:textColor="@color/abc_primary_text_disable_only_material_light"
/>
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:id="@+id/tvCorreo"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="FECHA DE NACIMIENTO"
                android:id="@+id/textView15"
                android:layout_column="1"
                android:textColor="@color/abc_primary_text_disable_only_material_light"
/>
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:id="@+id/tvFechaNac"
                android:layout_column="1"
                android:textColor="@color/abc_input_method_navigation_guard" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:text="SALIR"
                android:id="@+id/btnSalir"
                android:onClick="salida"
                android:layout_column="1" />
        </TableRow>
    </TableLayout>
</RelativeLayout>
```

MAINFORMACION.JAVA	PAQUETE: APP/JAVA

```java
package com.miempresa.personal.appregistropersonal;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

import com.miempresa.personal.appregistropersonal.R;

public class maInformacion extends AppCompatActivity {

    //GLOBAL
    TextView tvApellidos,tvNombres,tvTelefono,tvCorreo,tvFechaNac;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ma_informacion);

        //Instanciando a los controles
        tvApellidos = (TextView)findViewById(R.id.tvApellidos);
        tvNombres = (TextView)findViewById(R.id.tvNombres);
        tvTelefono = (TextView)findViewById(R.id.tvFono);
        tvCorreo = (TextView)findViewById(R.id.tvCorreo);
        tvFechaNac = (TextView)findViewById(R.id.tvFechaNac);

        //Mostrando información
        muestraDatos();
    }

    //Metodo que muestra la informacion resultante
    public void muestraDatos(){
        //Capturando los datos del Activity principal
        Bundle datos = getIntent().getExtras();
        String apellidos = datos.getString("apellidos");
        String nombres = datos.getString("nombres");
        String telefono = datos.getString("telefono");
        String correo = datos.getString("email");
        String fechaNac = datos.getString("fechaNac");

        //Mostrando la informacion obtenida
        tvApellidos.setText(apellidos);
        tvNombres.setText(nombres);
        tvTelefono.setText(telefono);
        tvCorreo.setText(correo);
        tvFechaNac.setText(fechaNac);
    }

    //Metodo que permite salir del activity
    public void salida(View view){
        finish();
    }
}
```



CAPÍTULO

5

Juegos



5.1 Aspectos generales

Canvas

Canvas (lienzo) es una interfaz que permite pintar en un área rectangular, que refleja las acciones sobre un mapa de bits (Bitmap) asociado:

```
Bitmap b = Bitmap.createBitmap
    (100, 100, Bitmap.Config.ARGB_8888);
Canvas c = new Canvas(b);
```

Para escribir en la pantalla, no será necesario instanciar Canvas directamente.

Canvas-Métodos

Principales métodos de Canvas:

drawBitmap*(Bitmap bitmap, float left, float top, Paint paint). Dibuja un Bitmap con la esquina superior izquierda en la posición indicada del lienzo.*

drawBitmap*(Bitmap bitmap, Rect src, Rect dst, Paint paint). Escala/ajusta un Bitmap al área (sic) de destino.*

drawCircle(*float cx, float cy, float radius, Paint paint). Dibuja un círculo.*

drawColor*(int color). Llena el lienzo con un color.*

drawline*(float startX, float startY, float stopX, float stopY, Paint paint). Dibuja una línea de (startX, startY) a (stopX, stopY).*

drawRect*(float left, float top, float right, float bottom, Paint paint). Dibuja el rectángulo especificado.*

***drawText**(String text, float x, float y, Paint paint). Dibuja texto en (x, y) con el Paint especificado.*

***drawLine**(float startX, float startY, float stopX, float stopY, Paint paint). Dibuja una línea de (startX, startY) a (stopX, stopY).*

***drawRect(**float left, float top, float right, float bottom, Paint paint). Dibuja el rectángulo especificado.*

***drawText**(String text, float x, float y, Paint paint) Dibuja texto en (x, y) con el Paint especificado.*

*[**drawARGB/drawRGB/drawColor**. Llena el canvas con un solo color.].*

Rafael Morón Abad

Fuente: recursos.crfptic.es:9080

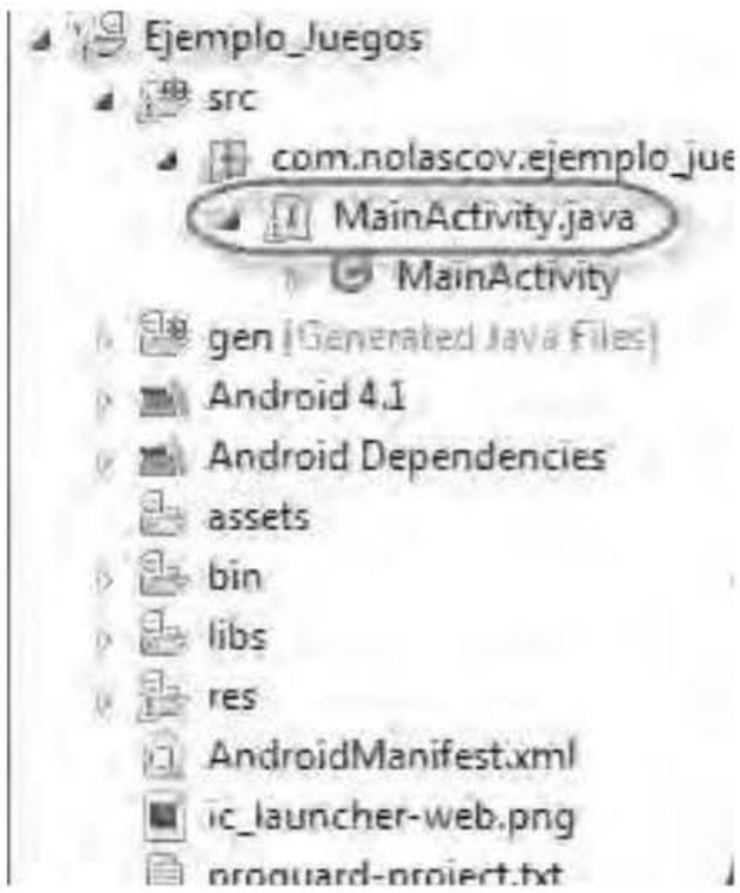
5.2 Aplicación de Canvas: Líneas, rectángulo y círculos

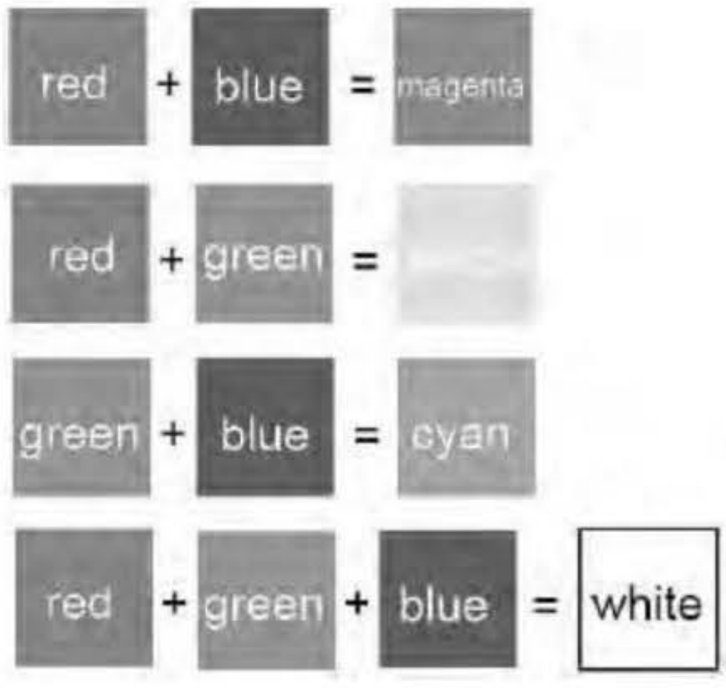
Cree una aplicación que permita demostrar el uso de canvas dibujando algunas figuras.

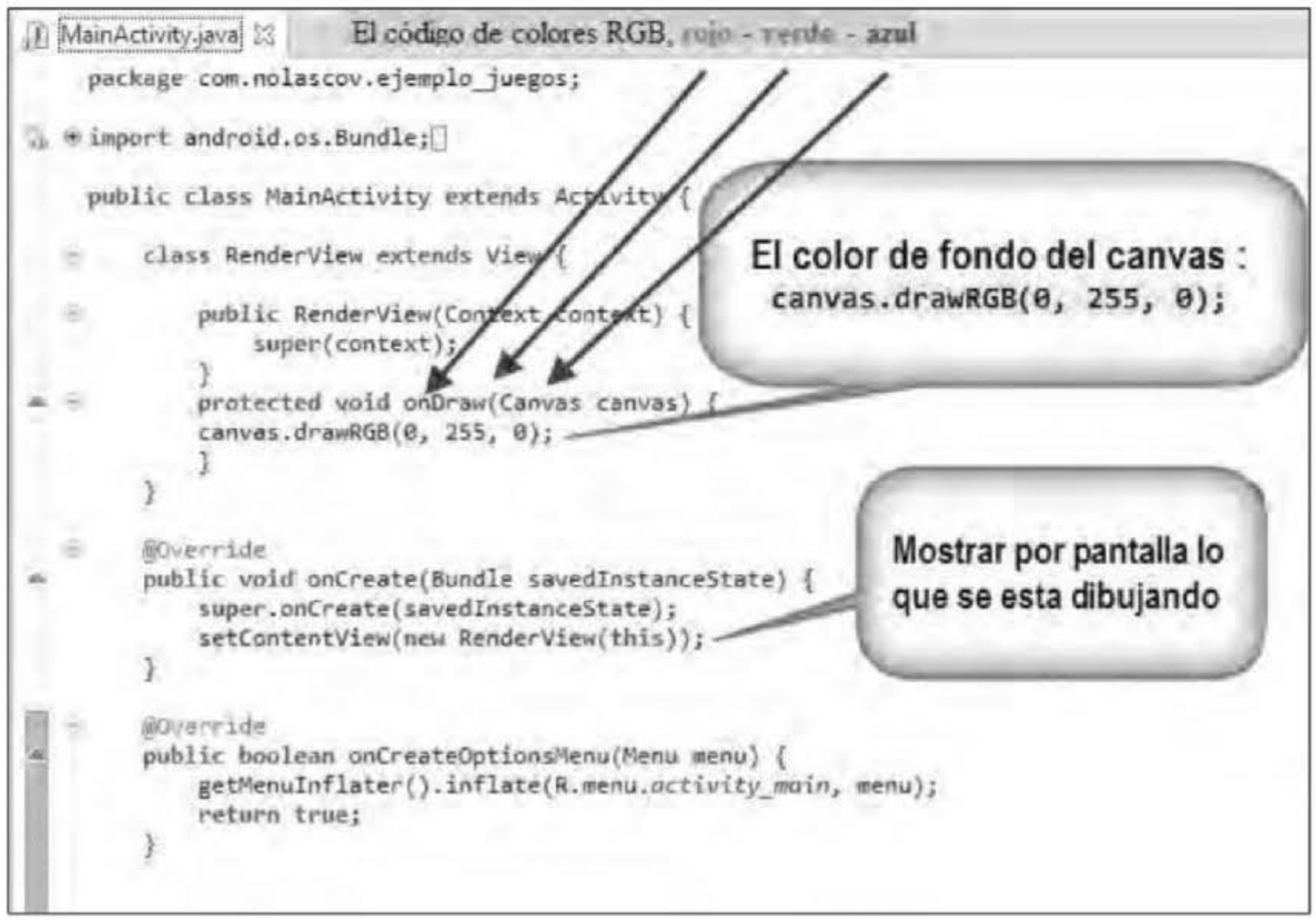
Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android proyecto*.

Paso 2: Ingrese el nombre del proyecto: Ejemplo_juegos (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

Paso 3: Modifique la actividad principal.








```
package com.nolascov.ejemplo_juegos;

import android.os.Bundle;

public class MainActivity extends Activity {

    class RenderView extends View {

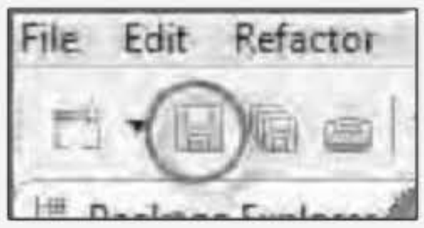
        public RenderView(Context context) {
            super(context);
        }
        protected void onDraw(Canvas canvas) {
        canvas.drawRGB(0, 255, 0);
        }
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new RenderView(this));
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
```

EJECUTANDO LA APLICACIÓN

Paso 1: *Guarde los cambios realizados, haciendo clic en el ícono*



o presionando la combinación <Ctrl+S>.

Paso 2: *Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:*



Paso 3: *Haga clic y mueva el ícono del candado en la zona superior para tocar un círculo que aparece al mover el ratón. Esto abre el emulador.*

La siguiente figura muestra la ventana principal de la pantalla de Android 4.1.



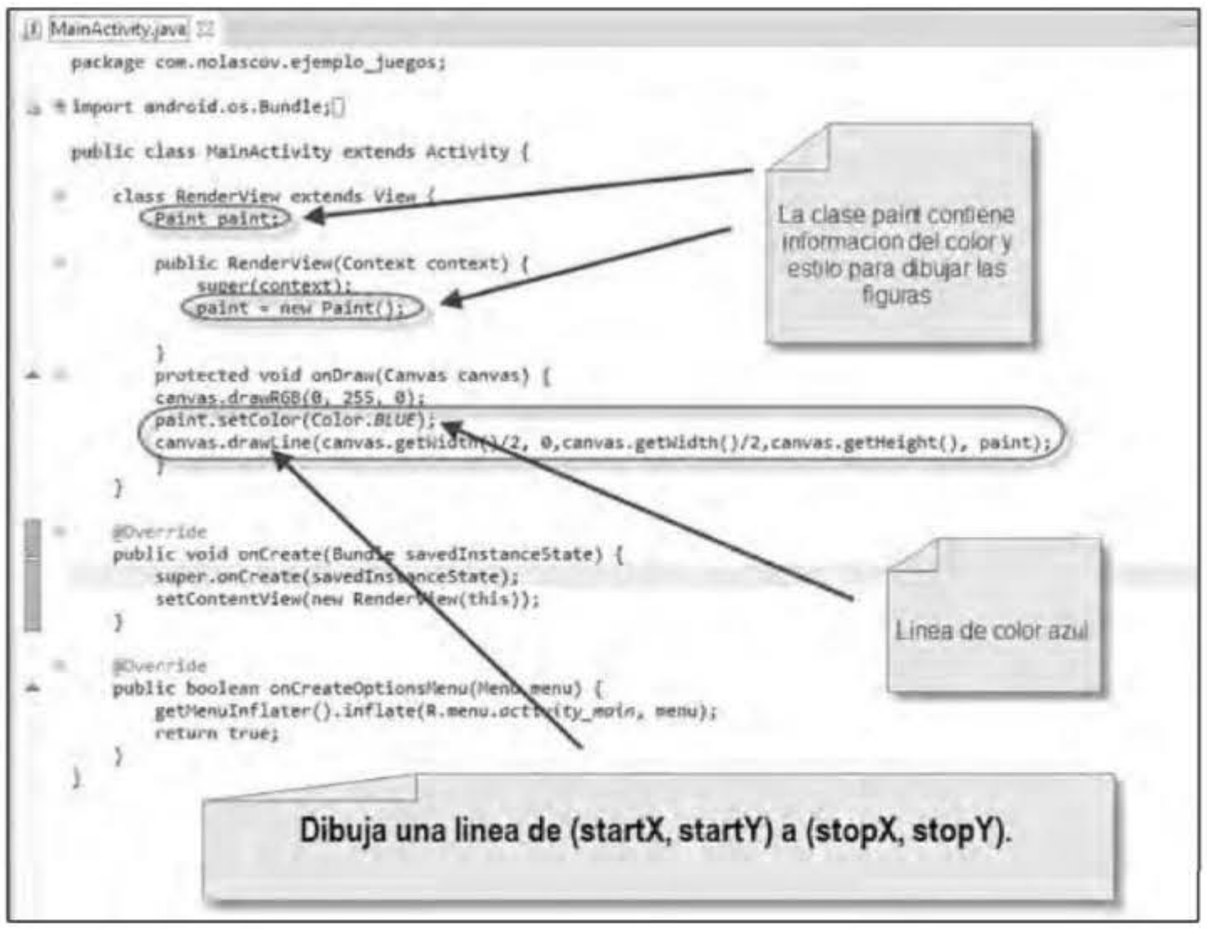
Paso 4: *Si el emulador se bloquea, deslice el botón de desbloqueo para contrarrestar su función.*

La figura muestra la aplicación que se ejecuta en el emulador de Android.



5.2.1 Dibujar una línea vertical

Para dibujar una línea vertical de color azul en medio de la pantalla, realice los siguientes pasos:





5.2.2 Dibujar rectángulos

Para dibujar algunos rectángulos, realice los siguientes pasos:

Paso 1:

```
*MainActivity.java
package com.nolascov.ejemplo_juegos;

+import android.os.Bundle;

public class MainActivity extends Activity {

    class RenderView extends View {
        Paint paint;

        public RenderView(Context context) {
            super(context);
            paint = new Paint();

        }
        protected void onDraw(Canvas canvas) {
        canvas.drawRGB(0, 255, 0);
        int ancho=canvas.getWidth();
        paint.setColor(Color.BLUE);
        canvas.drawLine(canvas.getWidth()/2, 0,canvas.getWidth()/2,canvas.getHeight(), paint

        paint.setARGB(255,255,0,0);
        canvas.drawRect(10,20,ancho-10,50,paint);
```

Rectángulo de color rojo

Calculamos el ancho de la pantalla

Dibujamos un rectángulo desde la coordenada columna:10 y fila 20 hasta la columna que coincide con el ancho de la pantalla menos 10 pixeles y la fila 50. Además le pasamos el pincel a utilizar



Paso 2:

```
MainActivity.java
package com.nolascov.ejemplo_juegos;

+import android.os.Bundle;

public class MainActivity extends Activity {

    class RenderView extends View {
        Paint paint;
        public RenderView(Context context) {
            super(context);
            paint = new Paint();
        }
        protected void onDraw(Canvas canvas) {
        canvas.drawRGB(0, 255, 0);
        int ancho=canvas.getWidth();
        paint.setColor(Color.BLUE);
        canvas.drawLine(canvas.getWidth()/2, 0,canvas.getWidth()/2,canvas.getHeight(),

        paint.setARGB(255,255,0,0);
        canvas.drawRect(10,20,ancho-10,50,paint);

        paint.setStyle(Style.STROKE);
        canvas.drawRect(10,80,ancho-10,100,paint);
```

El siguiente rectángulo configuramos el pincel para que solo pinte el perimetro llamando al método setStyle y pasando la constante STROKE

Dibujamos un rectángulo desde la coordenada columna:10 y fila 80 hasta la columna que coincide con el ancho de la pantalla menos 10 pixeles y la fila 100. Además le



5.2.3 Dibujar un círculo

Para dibujar un círculo de color blanco, realice los siguientes pasos:

MainActivity.java

```
class RenderView extends View {
    Paint paint;
    public RenderView(Context context) {
        super(context);
        paint = new Paint();
    }
    protected void onDraw(Canvas canvas) {
    canvas.drawRGB(0, 255, 0);
    int ancho=canvas.getWidth();

    paint.setColor(Color.BLUE);
    canvas.drawLine(canvas.getWidth()/2, 0,canvas.getWidth()
                                                    t(), paint);

    paint.setARGB(255,255,0,0);
    canvas.drawRect(10,20,ancho-10,50,paint);

    paint.setStyle(Style.STROKE);
    canvas.drawRect(10,80,ancho-10,100,paint);

    paint.setColor(Color.WHITE);
    canvas.drawCircle(ancho/2, 220, 100,paint);

    }
}
```

circulo de color blanco

dibuja un circulo , los dos primeros argumentos especifican las coordenadas del punto central el segundo el radio



5.3 Aplicación de Canvas: Texto

```
package com.nolascov.ejemplo_juegos;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.graphics.Path;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.support.v4.app.NavUtils;

public class MainActivity extends Activity {

    class RenderView extends View {
        Paint paint;
        public RenderView(Context context) {
            super(context);
            paint = new Paint();
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(0, 255, 0);
            int ancho=canvas.getWidth();

            paint.setColor(Color.BLUE);
            canvas.drawLine(canvas.getWidth()/2, 0,canvas.getWidth()/2,canvas.getHeight(),
            paint);

            paint.setARGB(255,255,0,0);
            canvas.drawRect(10,20,ancho-10,50,paint);

            paint.setStyle(Style.STROKE);
            canvas.drawRect(10,80,ancho-10,100,paint);

            paint.setColor(Color.WHITE);
            canvas.drawCircle(ancho/2, 220, 100,paint);

            paint.setColor(Color.RED);
            paint.setTextSize(38);
            paint.setShadowLayer(5, 0, 5, Color.BLACK);
            canvas.drawText("Este es un Texto", 60, 400, paint);
        }
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new RenderView(this));
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

EJECUTANDO LA APLICACIÓN



5.4 APLICACIÓN DE FUENTES EXTERNAS: TEXTO

Para confeccionar aplicaciones e incorporar fuentes True Type externas, implemente una aplicación que muestre un texto con una fuente externa.

Para crear una aplicación que permita demostrar el uso de fuentes externas, realice los siguientes pasos:

Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android proyecto*.

Paso 2: Ingrese el nombre del proyecto: Ejemplo_Fuentes (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

Paso 3: Descargue las fuentes a utilizar y cópielas en la carpeta assets (ver fig.).



Paso 4: Modifique la actividad principal.

```
package com.nolasco.ejemplo_fuentes;

import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.graphics.Typeface;
import android.graphics.Paint.Style;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.support.v4.app.NavUtils;

public class MainActivity extends Activity {

    class RenderView extends View {
        Paint paint;
        Typeface face;

        public RenderView(Context context) {
            super(context);
            paint = new Paint();
        }

        protected void onDraw(Canvas canvas) {
            canvas.drawRGB(255,255,255);
            paint.setARGB(255,0,0,0);
            paint.setTextSize(30);

            face = Typeface.createFromAsset(getAssets(),"A Drink For All Ages.TTF");
            paint.setTypeface(face);
            canvas.drawText("Tipo de Fuente : A Drink For All Ages", 0, 220, paint);

            face= Typeface.createFromAsset(getAssets(),"ABITE.ttf");
            paint.setTypeface(face);
            canvas.drawText("Tipo de Fuente : ABITE", 0, 320, paint);

            face= Typeface.createFromAsset(getAssets(),"ACAPPELLA.ttf");
            paint.setTypeface(face);
            canvas.drawText("Tipo de Fuente : ACAPPELLA", 0, 420, paint);

            face= Typeface.createFromAsset(getAssets(),"ADAMN.TTF");
            paint.setTypeface(face);
            canvas.drawText("Tipo de Fuente : ADAMN", 0, 520, paint);

            face= Typeface.createFromAsset(getAssets(),"adrip1.ttf");
            paint.setTypeface(face);
            canvas.drawText("Tipo de Fuente : adrip1", 0, 620, paint);

            face = Typeface.createFromAsset(getAssets(),"AGothiqueTime.ttf");
            paint.setTypeface(face);
            canvas.drawText("Tipo de Fuente : aGothiqueTime", 0, 720, paint);
```

```
            face= Typeface.createFromAsset(getAssets(),"alittlepot.ttf");
            paint.setTypeface(face);
            canvas.drawText("Tipo de Fuente : alittlepot", 0, 820, paint);

        }
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new RenderView(this));
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

}
```

5.5 Datos



Una manera muy sencilla y cómoda de guardar; por ejemplo, la información de un juego como es el usuario y la puntuación es por medio del uso de las preferencias.

5.5.1 Persistencia y preferencia

Son almacenes clave valor que permite crear varios almacenes de preferencias; sin embargo, normalmente bastará con usar un almacén por defecto para cada aplicación.

5.5.2 Aplicacion sobre el uso de Preferencias

Cree una aplicación que permita demostrar el uso de Preferencias.

Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android proyecto.*

Paso 2: Ingrese el nombre del proyecto: Ejemplo_Preferencias (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

Paso 3: Modifique el archivo: activity_main.xml.



activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="43dp"
        android:layout_marginTop="22dp"
        android:text="Ingrese un Valor" />

    <EditText
        android:id="@+id/Datos"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView1"
        android:layout_marginTop="39dp"
        android:ems="10" />

    <Spinner
        android:id="@+id/Tipo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/Datos"
        android:layout_below="@+id/Datos"
        android:layout_marginTop="24dp"
        android:entries="@array/etiquetas"
        android:entryValues="@array/valores"
        />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/Tipo"
        android:layout_below="@+id/Tipo"
        android:layout_marginTop="27dp"
        android:text="Acccion" />

    <TableLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView2"
        android:layout_centerVertical="true" >

        <TableRow
            android:id="@+id/tableRow1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >
```

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

        <Button
            android:id="@+id/btn_Salvar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Salvar" />

    </TableRow>
</TableLayout>

</RelativeLayout>
```

El Spinner utilizará los valores del archivo: strings.xml , desde un arreglo de cadenas.

strings.xml

```
<resources>

    <string name="app_name">Ejemplo_Preferencias </string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>

    <string-array name="etiquetas">
        <item >String</item>
        <item >Boolean</item>
        <item >Integer</item>
        <item >Float</item>
    </string-array>

      <array name="valores">
        <item >0</item>
        <item >1</item>
        <item >2</item>
        <item >3</item>

    </array>

</resources>
```

MainActivity.java

```
package com.nolascov.ejemplo_preferencias;

import android.os.Bundle;
import android.app.Activity;
import android.content.SharedPreferences;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
```

```
package com.nolascov.ejemplo_preferencias;

import android.os.Bundle;
import android.app.Activity;
import android.content.SharedPreferences;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;

public class MainActivity extends Activity {

    private EditText Datos;
    private Spinner Tipo;
    private Button btn_Leer;
    private Button btn_Salvar;
    private TextView Salida;

    // variable para las prefrencias
    private static String PREFERENCIA="MisPrefrencias";

    //modo de la preferencia privado solo acceso para la actividad
    private static int mode =Activity.MODE_PRIVATE;

    //variable de preferencia compartida
    private SharedPreferences MisPreferenciasCompartidas;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Datos = (EditText) findViewById(R.id.Datos);
        Tipo = (Spinner) findViewById(R.id.Tipo);
        btn_Leer = (Button) findViewById(R.id.btn_Leer);
        btn_Salvar = (Button) findViewById(R.id.btn_Salvar);
        Salida = (TextView) findViewById(R.id.Salida);

        //creando nuestra preferencia
        MisPreferenciasCompartidas= getSharedPreferences(PREFERENCIA, mode);

        this.btn_Leer.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {
                //datos leidos de la preferencia
                String cadena = "";
                //el tipo de la preferencia , el item seleccioando
                Integer i = Tipo.getSelectedItemPosition();
                switch(i)
                {
                case 0:
                      cadena =MisPreferenciasCompartidas.getString("cadena","Hola");
                      break;
                case 1:
                      cadena =new Boolean(MisPreferenciasCompartidas.getBoolean("Logi-
                      co",true)).toString();
                      break;
```

```
                case 2:
                      cadena =new Integer(MisPreferenciasCompartidas.getInt("Entero",11)).
                      toString();
                      break;
                case 3:
                      cadena =new Float(MisPreferenciasCompartidas.getFloat("flotante",(-
                      float)3.5)).toString();
                      break;
                default:
                      cadena ="";
                      break;
                }

                    Salida.setText(cadena);

            }
        });

        this.btn_Salvar.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {
                //datos leidos de la preferencia
                String Datos2 = Datos.getText().toString();
                //el tipo de la preferencia , el item seleccioando
                Integer i = Tipo.getSelectedItemPosition();

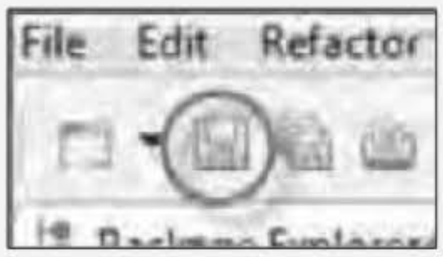
                SharedPreferences.Editor editor =MisPreferenciasCompartidas.edit();
                switch(i)
                {
                case 0:
                      editor.putString("cadena", Datos2);
                      break;
                case 1:
                      editor.putBoolean("Logico",Boolean.parseBoolean(Datos2));
                      break;
                case 2:
                      editor.putInt("Entero",Integer.parseInt(Datos2));
                      break;
                case 3:
                      editor.putFloat("Flotante",Float.parseFloat(Datos2));
                      break;
                default:
                      break;
                }

                editor.commit();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

EJECUTANDO LA APLICACIÓN

Paso 1: *Guarde los cambios realizados, haciendo clic en el ícono*



o presionando la combinación <Ctrl+S>.

Paso 2: *Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:*



5.6 DISPLAYMETRICS

Devuelve una estructura que describe la información general sobre la pantalla del dispositivo móvil, tal como :

- tamaño
- densidad
- escala de la fuente

Ejemplo:

```
DisplayMetrics metrics = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

Constantes

int	DENSITY_DEFAULT	La densidad de referencia utilizada en todo el sistema.
int	DENSITY_HIGH	Estándar cuantificado DPI para pantalla de alta resolución
int	DENSITY_LOW	Estándar cuantificado DPI para pantalla de baja resolución
int	DENSITY_MEDIUM	Estándar cuantificado DPI para pantalla de mediana resolución
int	DENSITY_TV	Densidad secundaria
int	DENSITY_XHIGH	Estándar cuantificado DPI para pantallas de extra alta densidad
int	DENSITY_XXHIGH	Estándar cuantificado DPI para pantallas de extra extra alta densidad

Propiedades

public float	density	La densidad de la pantalla.
public int	densityDpi	La densidad de la pantalla expresada en pulgadas
public int	**heightPixels**	**La altura de la pantalla en pixeles**
public float	scaledDensity	Un factor de escala para fuentes que se muestran en la pantalla.
public int	**widthPixels**	**El ancho de la pantalla en pixeles**
public float	xdpi	Los píxeles por pulgada de la pantalla en la dimensión X.
public float	ydpi	Los píxeles por pulgada de la pantalla en la dimensión Y.



Ahora diseñe una aplicación que permita realizar algunos trazos.

Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android proyecto*.

Paso 2: Ingrese el nombre del proyecto: EjemploTrazo (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

MainActivity.java

```
package com.nolascov.ejemplopintado;

import android.app.Activity;
import android.content.Context;
import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.Display;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.Window;
import android.view.WindowManager;
import java.io.InputStream;

public class MainActivity extends Activity {

    protected Display  display;
    protected MainView view;
    protected Paint  paint;
    protected Path   path;
    protected Canvas scr_canvas;
    protected Canvas dib_canvas;
    protected Bitmap scr_bitmap;
    protected Bitmap dib_bitmap;
    protected Bitmap icon_accept;
    protected Bitmap icon_accept_sel;
    protected Bitmap icon_cancel;
    protected Bitmap icon_cancel_sel;

    protected int width;
    protected int height;
    protected int color;
    protected int size;

    protected float brush_radius;
    protected boolean anti_alias;
    protected int[] palette;

    protected MainScreen mainScreen;
    protected PaletteScreen paletteScreen;
    protected Screen currScreen;
```

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
    //pantalla sin titulo
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.
    LayoutParams.FLAG_FULLSCREEN);
    // dm objetos que posee ancho , alto , densidad de la pantalla
    DisplayMetrics dm = new DisplayMetrics();
    // rellena el objetos con los datos
    getWindowManager().getDefaultDisplay().getMetrics(dm);

    //obtienes las imagenes desde los recursos
    icon_accept = BitmapFactory.decodeResource(getResources(), R.drawable.button_
    accept);
    icon_accept_sel = BitmapFactory.decodeResource(getResources(), R.drawable.button_
    accept_sel);
    icon_cancel = BitmapFactory.decodeResource(getResources(), R.drawable.button_can-
    cel);
    icon_cancel_sel = BitmapFactory.decodeResource(getResources(), R.drawable.button_
    cancel_sel);


    //obteniendo datos para el dm
    //ancho de la pantalla en pixeles
    width = dm.widthPixels;
    //alto de la pantalla en pixeles
    height = dm.heightPixels;
    //radio de la paleta
    brush_radius = 5;
    view = new MainView(this);
    //color negro
    color = Color.BLACK;
    //brocha para pintar
    paint = new Paint();
    //color a la brocha
    paint.setColor(color);
    //guardar puntos
    path = new Path();
    //crear una imagen en 32 bits del ancho y alto de la pantalla , pantalla de fondo
    scr_bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
    //es el objetos para pintar en el bitmap
    scr_canvas = new Canvas(scr_bitmap);
    //carga una paleta de 256 colores
    palette = loadPalette(this, "HEXEN.PAL");
    // instanciar la clase mainscreen
    mainScreen = new MainScreen(this);
    //no tiene uso
    paletteScreen = new PaletteScreen(this);
    // hace referencia a la pantalla actual
    currScreen = mainScreen;
    // muestra la pantalla
    mainScreen.show();

    setContentView(view);
}

@Override
    public boolean onTouchEvent(MotionEvent event) {
    return currScreen.onTouchEvent(event);
}
```

```
    @Override
        public final boolean onKeyDown(final int keyCode, final KeyEvent event) {
        return currScreen.onKeyDown(keyCode, event);
    }


    //manejo de la paleta de 256 colores
    public static int[] loadPalette(Context context, String name) {
        int palette[] = new int[256];
        //lista de byte ( flujo de entrada )
        InputStream is = null;
        //tener acceso a la carpeta assets
        AssetManager am = context.getAssets();
        //leer la paleta de colores
        try {
            //apertura
            is = am.open(name);
            for(int idx=0; idx<256; idx++) {
                //formato RGB 4 byte , alfa - Rojo - Verde - Azul
                palette[idx] = 0xFF000000 | (is.read() << 16) | (is.read() << 8) | is.read();
            }
        } catch(Exception ex) {
        } finally {
            if(is != null) {
                try {
                      is.close();
                } catch(Exception ex) {}
            }
            return palette;
        }

    }
```

MainScreen.java

```
package com.nolascov.ejemplopintado;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.net.Uri;
import android.provider.MediaStore;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.widget.Toast;

public class MainScreen extends Screen {

    private AlertDialog clearDialog;
    private AlertDialog brushDialog;
    private AlertDialog colorDialog;
    private AlertDialog exitDialog;
```

```
    protected Button btClear;
    protected Button btColor;
    protected Button btBrush;
    protected Button btShare;
    protected Button btExit;

    protected Bitmap icon_clear;
    protected Bitmap icon_clear_sel;
    protected Bitmap icon_brush;
    protected Bitmap icon_brush_sel;
    protected Bitmap icon_color;
    protected Bitmap icon_color_sel;
    protected Bitmap icon_share;
    protected Bitmap icon_share_sel;
    protected Bitmap icon_exit;
    protected Bitmap icon_exit_sel;

    protected int toolbar_height;
    protected int icon_width;
    protected int icon_height;
    protected int icon_x;
    protected int icon_y;

    protected float lastX;
    protected float lastY;
    protected float currX;
    protected float currY;

    public MainScreen(Context context) {
        super(context);
        //carga los iconos en la barra de herramientas
        loadBitmaps();
        //inicializa la barra herramientas
        initToolBox();
        //canvas donde se dibujara
        main.dib_bitmap = Bitmap.createBitmap(main.width, main.height-toolbar_height,
        Bitmap.Config.ARGB_8888);
        //objeto para dibujar
        main.dib_canvas = new Canvas(main.dib_bitmap);
        //borra el canvas con el color blanco
        main.dib_canvas.drawColor(Color.WHITE);
        //la brocha a color negro
        main.paint.setColor(0xFF000000);

    }

    @Override
    protected void show() {
        //pinta color gris
        main.scr_canvas.drawColor(Color.GRAY);
        //dibuja la barra de herramientas
        drawToolBar();
        main.scr_canvas.drawBitmap(main.dib_bitmap, 0, toolbar_height, null);
        main.view.invalidate();

    }

    @Override
    protected boolean onTouchEvent(MotionEvent event) {
        boolean res = false;
        int action = event.getActionMasked();
        if(action == MotionEvent.ACTION_DOWN || action == MotionEvent.ACTION_MOVE || action
        == MotionEvent.ACTION_UP) {
```

```
            res = true;
            float x = event.getX();
            float y = event.getY();
            if(y > toolbar_height ) {
                y -= toolbar_height;
                switch(action) {
                    case MotionEvent.ACTION_DOWN:
                            currX = x;
                            currY = y;
                            lastX = currX;
                            lastY = currY;
                            //dibuja un punto o circulo
                            drawPoint(currX, currY);
                            //actualizar la pantalla
                            main.scr_canvas.drawBitmap(main.dib_bitmap, 0, toolbar_height, null);
                            //para repinte la pantalla
                            main.view.invalidate();
                            break;
                        case MotionEvent.ACTION_MOVE:
                            currX = x;
                            currY = y;
                            //dibujo una línea desde el punto actual , al punto anterior
                            drawLine();
                            //actualizar la pantalla
                            main.scr_canvas.drawBitmap(main.dib_bitmap, 0, toolbar_height, null);
                            //para repinte la pantlla
                            main.view.invalidate();
                            lastX = currX;
                            lastY = currY;
                            break;
                    }
                } else {
                    // los botones presionado
                    if( btClear.isTouched(x,y) ) {
                            btClear.onTouchEvent(event);
                    } else
                            if( btBrush.isTouched(x,y) ) {
                                    btBrush.onTouchEvent(event);
                    } else
                    if( btColor.isTouched(x,y) ) {
                            btColor.onTouchEvent(event);
                    } else
                            if( btShare.isTouched(x,y) ) {
                                    btShare.onTouchEvent(event);
                            } else
                            if( btExit.isTouched(x,y) ) {
                                    btExit.onTouchEvent(event);
                            }
                    main.view.invalidate();
                }
            }
            return res;
        }

        @Override
        protected boolean onKeyDown(int keyCode, KeyEvent event) {
            boolean res = false;
            if(event.getAction() == KeyEvent.ACTION_DOWN) {
                if(keyCode == KeyEvent.KEYCODE_BACK) {
                    res = true;
                    getExitDialog();
                }
            }
```

```
        return res;
    }

    private void loadBitmaps() {
        icon_clear = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_menu);
        icon_clear_sel = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_
        menu_sel);
        icon_brush = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_brush);
        icon_brush_sel = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_
        brush_sel);
        icon_color = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_color);
        icon_color_sel = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_
        color_sel);
        icon_share = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_share);
        icon_share_sel = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_
        share_sel);
        icon_exit = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_tool);
        icon_exit_sel = BitmapFactory.decodeResource(main.getResources(), R.drawable.icon_
        tool_sel);
    }

    private void initToolBox() {
        icon_width = icon_clear.getWidth();
        icon_height = icon_clear.getHeight();
        main.size = icon_height;
        icon_y = 0;
        toolbar_height = icon_height + (icon_height >> 3);
        int sp = (main.width - (icon_width*5)) / 6;
        icon_x = sp;
        sp += icon_width;

        btClear = new Button(main, icon_x , icon_y, icon_clear, icon_clear_sel) {
            @Override
            protected void action() {
                getClearDialog().show();
            }
        };
        icon_x += sp;
        btBrush = new Button(main, icon_x , icon_y, icon_brush, icon_brush_sel) {
            @Override
            protected void action() {
                getBrushDialog().show();
            }
        };
        icon_x += sp;
        btColor = new Button(main, icon_x , icon_y, icon_color, icon_color_sel) {
            @Override
            protected void action() {
                getColorDialog().show();
            }
        };
        icon_x += sp;
        btShare = new Button(main, icon_x , icon_y, icon_share, icon_share_sel) {
            @Override
            protected void action() {
                shareBitmap();
            }
        };
        icon_x += sp;
```

```
        btExit = new Button(main, icon_x , icon_y, icon_exit, icon_exit_sel) {
            @Override
            protected void action() {
                getExitDialog().show();
            }
        };
    }

    public AlertDialog getClearDialog() {
        if(!(clearDialog instanceof AlertDialog)) {
            clearDialog = new AlertDialog.Builder(main).create();
            clearDialog.setTitle("Clear Canvas?");
            clearDialog.setButton(AlertDialog.BUTTON_POSITIVE, "Yes", new DialogInterface.
            OnClickListener() {
                public void onClick(DialogInterface paramDialogInterface, int paramInt) {
                main.dib_canvas.drawColor(Color.WHITE);
                main.scr_canvas.drawBitmap(main.dib_bitmap, 0, toolbar_height, null);
                main.view.invalidate();
            }
        });
            clearDialog.setButton(AlertDialog.BUTTON_NEGATIVE, "No", (DialogInterface.
            OnClickListener)null);
        }
            return clearDialog;
    }

    public AlertDialog getBrushDialog() {
        if(!(brushDialog instanceof AlertDialog)) {
            brushDialog = new BrushDialog(main);
        }
        return brushDialog;
    }

    public AlertDialog getColorDialog() {
        if(!(colorDialog instanceof AlertDialog)) {
            colorDialog = new ColorDialog(main);
        }
        return colorDialog;
     }

     public AlertDialog getExitDialog() {
         if(!(exitDialog instanceof AlertDialog)) {
             exitDialog = new AlertDialog.Builder(main).create();
             exitDialog.setTitle("Exit MicroPaint?");
             exitDialog.setButton(AlertDialog.BUTTON_POSITIVE, "Yes", new DialogInterface.
             OnClickListener() {
                 public void onClick(DialogInterface paramDialogInterface, int paramInt) {
                 SplashActivity.killed = true;
                 main.finish();
                 }
             });
             exitDialog.setButton(AlertDialog.BUTTON_NEGATIVE, "No", (DialogInterface.OnClick-
             Listener)null);
        }
        return exitDialog;
    }

    private void drawToolBar() {
        main.paint.setColor(0xFF202020);
        main.scr_canvas.drawRect(0, 0, main.width, main.size, main.paint);
        main.paint.setColor(main.color);
        btClear.paint(false);
        btColor.paint(false);
```

```
        btBrush.paint(false);
        btShare.paint(false);
        btExit.paint(false);
    }

        private void shareBitmap() {
            try {
                String fpath = MediaStore.Images.Media.insertImage(main.getContentResolver(),
                main.dib_bitmap, "dib_bitmap", null);
                    Uri screenshotUri = Uri.parse(fpath);
                    final Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
                    emailIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    emailIntent.putExtra(Intent.EXTRA_STREAM, screenshotUri);
                    emailIntent.setType("image/png");
                    main.startActivity(Intent.createChooser(emailIntent, "Share image..."));
            } catch(Exception ex) {
                Toast.makeText(main, ex.getMessage(), Toast.LENGTH_LONG).show();
            }
        }

        private void drawLine() {
            float lenX = currX - lastX;
            float lenY = currY - lastY;
            float vx = 0;
            if(lenX > 0) {
                vx = 1;
            } else if(lenX < 0) {
                vx = -1;
            }
            float vy = 0;
            if(lenY > 0) {
                vy = 1;
            } else if(lenY < 0) {
                vy = -1;
            }
            float len;
            lenX = Math.abs(lenX);
            lenY = Math.abs(lenY);
            if(lenX > lenY) {
                vy = (lenY / lenX) * vy;
                len = lenX;
            } else {
                vx = (lenX / lenY) * vx;
                len = lenY;
            }
            float x = lastX;
            float y = lastY;
            for(int i=0; i < len; i++) {
                drawPoint((int)x, (int)y);
                x += vx;
                y += vy;
            }
        }

        private void drawPoint(final float x, final float y) {
            main.dib_canvas.drawCircle(x, y, main.brush_radius, main.paint);
        }

    }
```

BrushDialog.java

```
package com.nolascov.ejemplopintado;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.view.View;
import android.widget.CheckBox;
import android.widget.LinearLayout;
import android.widget.LinearLayout.LayoutParams;
import android.widget.SeekBar;
import android.widget.TextView;

public class BrushDialog extends AlertDialog implements SeekBar.OnSeekBarChangeListener,
DialogInterface.OnClickListener, View.OnClickListener {

    private MainActivity main;
    private TextView text;
    private CheckBox check;
    private int brush_radius;

    public BrushDialog(Context context) {
        super(context);
        main = (MainActivity)context;
        brush_radius = (int)main.brush_radius;
        LayoutParams params = new LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.
        MATCH_PARENT);
        LinearLayout layout = new LinearLayout(main);
        layout.setOrientation(LinearLayout.VERTICAL);
        layout.setPadding(8,8,8,8);

        text = new TextView(main);
        text.setText("size: "+(brush_radius << 1));
        layout.addView(text);
        SeekBar seekbar = new SeekBar(main);
        seekbar.setLayoutParams(params);
        seekbar.setOnSeekBarChangeListener(this);
        seekbar.setMax(100);
        seekbar.setProgress(brush_radius << 1);
        layout.addView(seekbar);

        check = new CheckBox(main);
        check.setText("Antialias");
        check.setChecked(main.anti_alias);
        check.setOnClickListener(this);
        layout.addView(check);

        setView(layout);
        setTitle("Brush Settings");
        setIcon(0);
        setButton(DialogInterface.BUTTON_POSITIVE, "Accept", this);
        setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel", this);
     }

    @Override
    protected void onStart() {
        super.onStart();
        brush_radius = (int)main.brush_radius;
```

```
        LayoutParams params = new LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.
        MATCH_PARENT);
        LinearLayout layout = new LinearLayout(main);
        layout.setOrientation(LinearLayout.VERTICAL);
        layout.setPadding(8,8,8,8);

        text = new TextView(main);
        text.setText("size: "+(brush_radius << 1));
        layout.addView(text);
        SeekBar seekbar = new SeekBar(main);
        seekbar.setLayoutParams(params);
        seekbar.setOnSeekBarChangeListener(this);
        seekbar.setMax(100);
        seekbar.setProgress(brush_radius << 1);
        layout.addView(seekbar);
    }

    public void onProgressChanged(SeekBar seekbar, int progress, boolean user) {
        brush_radius = progress >> 1;
        text.setText("size: "+progress);
    }

    public void onStartTrackingTouch(SeekBar seekbar) {}

    public void onStopTrackingTouch(SeekBar seekbar) {}

    public void onClick(DialogInterface dialog, int button) {
        if( button == DialogInterface.BUTTON_POSITIVE ) {
            main.brush_radius = brush_radius;
        }
    }

    public void onClick(View arg0) {
        if(main.anti_alias) {
            main.anti_alias = false;
        } else {
            main.anti_alias = true;
        }
        main.paint.setAntiAlias(main.anti_alias);
    }

}
```

Button.java

```
package com.nolascov.ejemplopintado;

import android.graphics.Bitmap;
import android.view.MotionEvent;

public class Button {

    protected float x;
    protected float y;
    protected float xx;
    protected float yy;
    protected int width;
    protected int height;
    protected Bitmap icon;
    protected Bitmap icon_sel;
```

```
    protected float x1;
    protected float y1;
    protected float x2;
    protected float y2;
    private MainActivity main;


    public Button(final MainActivity main,final float x, final float y, final Bitmap icon, final
    Bitmap icon_sel) {
        this.main = main;
        this.x = x;
        this.y = y;
        width = icon.getWidth();
        height = icon.getHeight();
        xx = (x+width)-1;
        yy = (y+height)-1;
        this.icon = icon;
        this.icon_sel = icon_sel;
        x1 = x+2;
        y1 = y+2;
        x2 = xx-2;
        y2 = yy-2;
    }

    protected final boolean isTouched(final MotionEvent event) {
        return isTouched(event.getX(), event.getY());
    }

    protected final boolean isTouched(final float x, final float y) {
        return (x1 < x) && (x < x2) && (y1 < y) && (y < y2);
    }

    protected final boolean onTouchEvent(final MotionEvent event) {
        boolean res = false;
        int action = event.getActionMasked();
        if(action == MotionEvent.ACTION_DOWN) {
            paint(true);
            res = true;
        } else if (action == MotionEvent.ACTION_UP) {
            paint(false);
            res = true;
            action();
        }
        return res;
    }

    protected final void paint(final boolean hover) {
        if(hover) {
            main.scr_canvas.drawBitmap(icon_sel, x, y, null);
        } else {
            main.scr_canvas.drawBitmap(icon, x, y, null);
        }
        main.view.invalidate();
    }

    protected void action() {}
```

ColorDialog.java

```
package com.nolascov.ejemplopintado;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.graphics.Color;
import android.widget.LinearLayout;
import android.widget.LinearLayout.LayoutParams;
import android.widget.SeekBar;
import android.widget.TextView;

public class ColorDialog extends AlertDialog implements SeekBar.OnSeekBarChangeListener,
DialogInterface.OnClickListener {

    private MainActivity main;
    private TextView text_color;
    private TextView text_red;
    private TextView text_green;
    private TextView text_blue;
    private TextView text_alpha;
    private SeekBar bar_red;
    private SeekBar bar_green;
    private SeekBar bar_blue;
    private SeekBar bar_alpha;
    private int color;
    private int red;
    private int green;
    private int blue;
    private int alpha;

    public ColorDialog(Context context) {
        super(context);
        main = (MainActivity)context;
        LayoutParams params = new LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.
MATCH_PARENT);
        LinearLayout layout = new LinearLayout(main);
        layout.setOrientation(LinearLayout.VERTICAL);
        layout.setPadding(8,8,8,8);

        text_color = new TextView(main);
        text_color.setText(" ");
        layout.addView(text_color);

        text_red = new TextView(main);
        layout.addView(text_red);
        bar_red = new SeekBar(main);
        bar_red.setId(Color.RED);
        bar_red.setLayoutParams(params);
        bar_red.setOnSeekBarChangeListener(this);
        bar_red.setMax(255);
        layout.addView(bar_red);

        text_green = new TextView(main);
        layout.addView(text_green);
        bar_green = new SeekBar(main);
        bar_green.setId(Color.GREEN);
        bar_green.setLayoutParams(params);
        bar_green.setOnSeekBarChangeListener(this);
        bar_green.setMax(255);
```

```
        layout.addView(bar_green);

        text_blue = new TextView(main);
        layout.addView(text_blue);
        bar_blue = new SeekBar(main);
        bar_blue.setId(Color.BLUE);
        bar_blue.setLayoutParams(params);
        bar_blue.setOnSeekBarChangeListener(this);
        bar_blue.setMax(255);
        layout.addView(bar_blue);

        text_alpha = new TextView(main);
        layout.addView(text_alpha);
        bar_alpha = new SeekBar(main);
        bar_alpha.setId(Color.GRAY);
        bar_alpha.setLayoutParams(params);
        bar_alpha.setOnSeekBarChangeListener(this);
        bar_alpha.setMax(255);
        layout.addView(bar_alpha);

        setView(layout);
        setTitle("Color Settings");
        setIcon(0);
        setButton(DialogInterface.BUTTON_POSITIVE, "Accept", this);
        setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel", this);
    }

    @Override
    protected void onStart() {
        super.onStart();
        color = main.color;
        red = Color.red(color);
        green = Color.green(color);
        blue = Color.blue(color);
        alpha = Color.alpha(color);

        text_color.setBackgroundColor(color);

        text_red.setText("red: "+red);
        bar_red.setBackgroundColor(Color.rgb(red,0,0));
        bar_red.setProgress(red);

        text_green.setText("green: "+green);
        bar_green.setBackgroundColor(Color.rgb(0,green,0));
        bar_green.setProgress(green);

        text_blue.setText("blue: "+blue);
        bar_blue.setBackgroundColor(Color.rgb(0,0,blue));
        bar_blue.setProgress(blue);

        text_alpha.setText("alpha: "+alpha);
        bar_alpha.setBackgroundColor(Color.rgb(alpha,alpha,alpha));
        bar_alpha.setProgress(alpha);
    }

    public void onProgressChanged(SeekBar seekbar, int progress, boolean user) {
        switch(seekbar.getId()) {
        case Color.RED:
            red = progress;
            bar_red.setBackgroundColor(Color.rgb(red,0,0));
            text_red.setText("red: "+red);
```

```
            break;
        case Color.GREEN:
            green = progress;
            bar_green.setBackgroundColor(Color.rgb(0,green,0));
            text_green.setText("green: "+green);
            break;
        case Color.BLUE:
            blue = progress;
            bar_blue.setBackgroundColor(Color.rgb(0,0,blue));
            text_blue.setText("blue: "+blue);
            break;
        case Color.GRAY:
            alpha = progress;
            bar_alpha.setBackgroundColor(Color.rgb(alpha,alpha,alpha));
            text_alpha.setText("alpha: "+alpha);
            break;
        }
        color = Color.argb(alpha, red, green, blue);
        text_color.setBackgroundColor(color);
    }

    public void onStartTrackingTouch(SeekBar seekbar) {
    }

    public void onStopTrackingTouch(SeekBar seekbar) {
    }

    public void onClick(DialogInterface dialog, int button) {
        if( button == DialogInterface.BUTTON_POSITIVE ) {
            main.color = color;
            main.paint.setColor(main.color);
        }
    }

}
```

MainView.java

```
package com.nolascov.ejemplopintado;

import android.content.Context;
import android.graphics.Canvas;
import android.view.View;

public class MainView extends View {
    private MainActivity main;

    public MainView(Context context) {
        super(context);
        main = (MainActivity)context;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawBitmap(main.scr_bitmap, 0, 0, null);
    }

}
```

PaletteScreen.java

```
package com.nolascov.ejemplopintado;

import android.content.Context;
import android.graphics.Color;
import android.view.KeyEvent;
import android.view.MotionEvent;

public class PaletteScreen extends Screen {
    private Button btAccept;
    private Button btCancel;

    private int box;
    private int palX1;
    private int palY1;
    private int palX2;
    private int palY2;
    private int palWdt;
    private int palHgt;
    private int color;

    public PaletteScreen(Context context) {
        super(context);
        box = main.size >> 2;
        palWdt = box << 4;
        palHgt = palWdt;
        palX1 = (main.width - palWdt) >> 1;
        palY1 = main.size+box;
        palX2 = palX1+palWdt;
        palY2 = palY1+palHgt;
        color = main.color;

        btAccept = new Button(main, 0 , main.height-main.icon_accept.getHeight(), main.
        icon_accept, main.icon_accept_sel) {
            @Override
            protected void action() {
                main.color = color;
                goback();
            }
        };
        btCancel = new Button(main, main.width-main.icon_cancel.getWidth() , main.height-
        main.icon_cancel.getHeight(), main.icon_cancel, main.icon_cancel_sel) {
            @Override
            protected void action() {
                goback();
            }
        };
    }

    @Override
    protected void show() {
        main.scr_canvas.drawColor(Color.DKGRAY);
        drawColorBox();
        drawPalette();
        btAccept.paint(false);
        btCancel.paint(false);
        main.view.invalidate();
    }
```

```
    private void drawColorBox() {
        main.paint.setColor(color);
        main.scr_canvas.drawRect(0, 0, main.width, main.size, main.paint);
    }

    private void drawPalette() {
        int col = 0;
        for(int i=0; i<16; i++) {
            for(int j=0; j<16; j++) {
                main.paint.setColor(main.palette[col]);
                col++;
                main.scr_canvas.drawRect((j*box)+palX1, (i*box)+palY1, (j*box)+box+palX1,
                (i*box)+box+palY1, main.paint);
            }
        }
    }

    @Override
    protected final boolean onTouchEvent(final MotionEvent event) {
        boolean res = false;
        int action = event.getActionMasked();
        if(action == MotionEvent.ACTION_DOWN || action == MotionEvent.ACTION_MOVE ||
        action == MotionEvent.ACTION_UP) {
            res = true;
            float x = event.getX();
            float y = event.getY();
            if((x > palX1) && (y > palY1) && (x < palX2) && (y < palY2)) {
                int px = (int)((x-palX1)/box);
                int py = (int)((y-palY1)/box);
                color = main.palette[(py << 4) | px];
                drawColorBox();
                 main.view.invalidate();
            } else {
            if( btAccept.isTouched(x,y) ) {
                btAccept.onTouchEvent(event);
            } else
                if( btCancel.isTouched(x,y) ) {
                      btCancel.onTouchEvent(event);
                }
            }
        }
        return res;
    }

    @Override
    protected boolean onKeyDown(int keyCode, KeyEvent event) {
        boolean res = false;
        if(event.getAction() == KeyEvent.ACTION_DOWN) {
            if(keyCode == KeyEvent.KEYCODE_BACK) {
                res = true;
                goback();
            }
        }
        return res;
    }

    private void goback() {
         main.paint.setColor(main.color);
        main.currScreen = main.mainScreen;
        main.mainScreen.show();
    }
}
```

Screen.java

```
package com.nolascov.ejemplopintado;

import android.content.Context;
import android.view.KeyEvent;
import android.view.MotionEvent;

public class Screen {
    protected MainActivity main;

    public Screen(Context context) {
        main = (MainActivity)context;
    }

    protected boolean onTouchEvent(final MotionEvent event) {
        return false;
    }

    protected boolean onKeyDown(final int keyCode, final KeyEvent event) {
        return false;
    }

    protected void show() {}

}
```

SplashActivity.java

```
package com.nolascov.ejemplopintado;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;
import java.util.Timer;
import java.util.TimerTask;

public class SplashActivity extends Activity {
    protected static boolean killed;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //desactivar título
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        //mostrar en toda la pantalla
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.
        LayoutParams.FLAG_FULLSCREEN);
        //la imagen a mostrar inicialmente
        ImageView image = new ImageView(this);
        //código imagen
```

```
        image.setImageResource(R.drawable.logo);
        //colocar la imagen por defecto
        setContentView(image);
    }

    @Override
    protected void onStart() {
        super.onStart();
        //preguntar salida completa de la app
        if(killed) {
            killed = false;
            //cerrar la actividad
            finish();
        } else {
            //timer de 2 segundos
            new Timer().schedule(new TimerTask() {
                @Override
                public void run() {
                    Intent intent = new Intent(SplashActivity.this, MainActivity.class);
                    // no borrar la acterior actividad
                    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    startActivity(intent);
                }
            }, 4000);
        }
    }

}
```

EJECUTANDO LA APLICACIÓN

Paso 1: *Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:*



Puede descargar la aplicación desde el siguiente link:

https://play.google.com/store/search?q=Inkadroid&c=apps



CAPÍTULO

6

Introducción a la inteligencia artificial: Agentes Android y dispositivos móviles

6.1 Introducción

Los agentes surgieron de la investigación en inteligencia artificial (IA) y específicamente de la inteligencia artificial distribuida (DIA). Al principio, la IA hablaba de los agentes como programas especiales cuya naturaleza y construcción no se llegaba a detallar. Recientemente, se ha escrito acerca de este tema, explicando qué se puede entender por un agente y qué no, en general, atendiendo a características que debe tener un agente: autonomía, reactividad, iniciativa, habilidad social, etc.

En este libro se toma la definición de Newell, la misma se encuentra influenciada por la IA. Newell asume que existen diferentes niveles de abstracción en el diseño de sistemas, entre ellos el nivel de símbolos y el nivel de conocimiento. Del mismo modo que un programa convencional maneja símbolos (e.g. expresiones, variables) en el nivel de símbolos, un agente, que equivale a un programa en el nivel del conocimiento, maneja únicamente conocimiento y se comporta de acuerdo con el principio de racionalidad. Este principio estipula que un agente emprende acciones porque está buscando satisfacer un objetivo.

Se ha tomado esta definición porque establece que el concepto de agente es simplemente una forma de ver un sistema, como también indica Shoham, que luego puede reflejarse a nivel de símbolos, con un programa tradicional. Esta concepción es compatible con el ciclo de vida del software. De forma general, se pensaría en el agente como un ente que se comporta de acuerdo con el principio de racionalidad. Cuando hubiese que desarrollar físicamente el agente, se atendería a su arquitectura y componentes, lo cual entraría dentro del nivel simbólico.

El diseño de agentes aislados ha sido estudiado en profundidad en la IA. Como resultado, se tienen las arquitecturas de subsunción, arquitecturas de pizarra, arquitecturas BDI y arquitecturas para la resolución genérica de problemas.

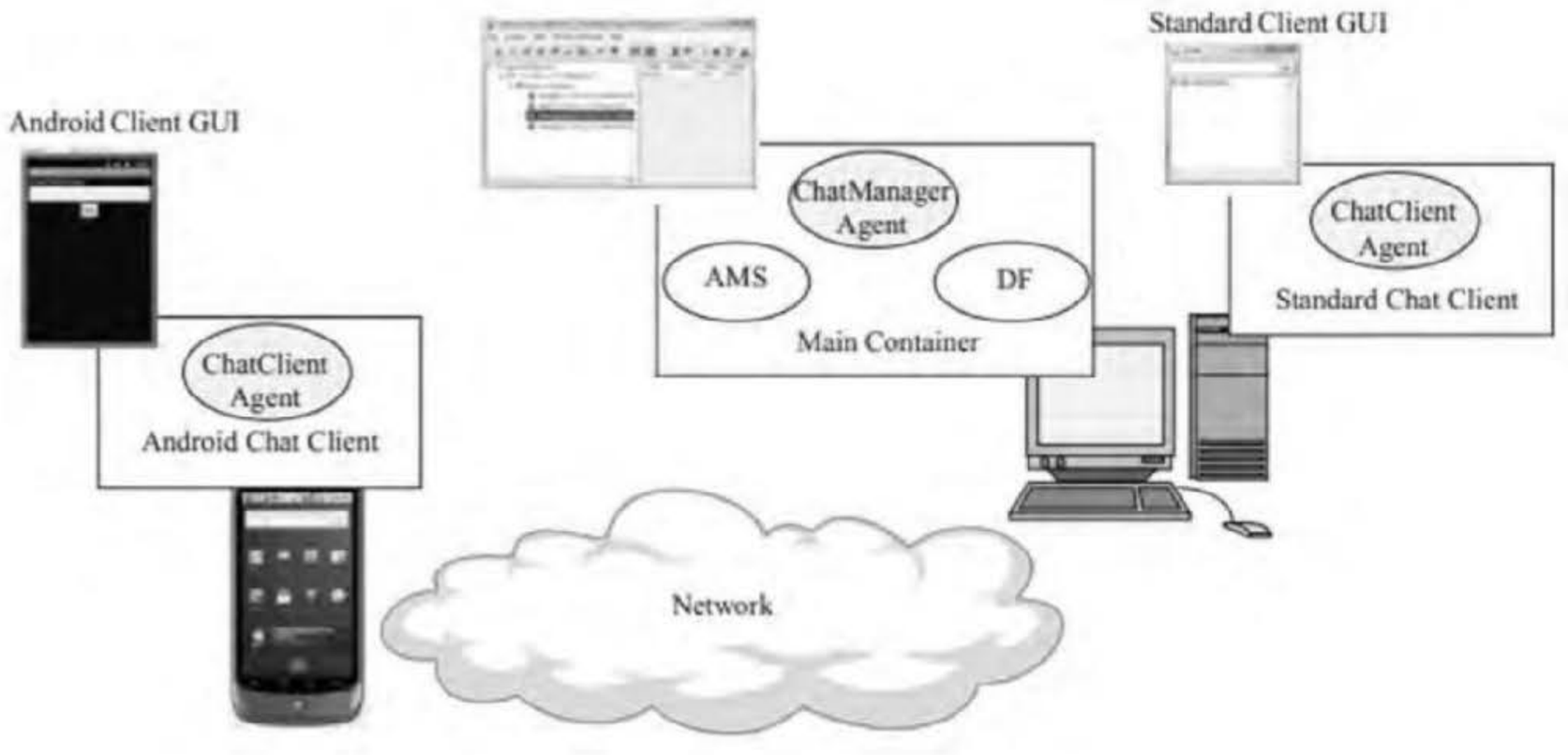
Teniendo en cuenta el carácter distribuido de los entornos de aplicación de agentes (especialmente sobre tratamiento de información en internet) es normal concebir la colaboración de varios agentes. En tales casos, la atención se centra en cómo construir Sistema Multi-Agentes (SMA) que pueden entenderse como grupos de agentes que interaccionan entre sí para conseguir objetivos comunes. La SMA es un sistema que reúne los elementos de la ilustración, el entorno y el conjunto de objetos. Estos objetos se encuentran integrados con el entorno, es posible en un momento dado asociar uno de estos objetos con un lugar en el entorno. Estos objetos son pasivos, pueden ser percibidos, creados, destruidos y modificados por agentes.

Un conjunto de agentes que se consideran como objetos especiales que representan las entidades activas del sistema.

Así pues, los agentes conformarán un nuevo paradigma de la Ingeniería de Software con el fin de diseñar e implementar sistemas complejos distribuidos, así también servirán como una herramienta para entender las sociedades humanas, pues permitirán una interesante forma de simular sociedades.

Los sistemas multiagentes (MAS) se describen así:

- Los agentes deben compartir conocimientos sobre el problema y las posibles soluciones.
- Los agentes en un MAS pueden compartir una meta o tener metas independientes.
- El “conocimiento global” puede incluir control global, consistencia global, metas globales, etc.
- La coordinación puede ser muy compleja.



6.2 Descripción de la aplicación

Como se representa en la figura anterior, esta aplicación se compone de tres módulos:

- *Android Chat Client (Android Chat Client):* Esta es una aplicación para Android dirigida a los participantes a la charla desde un dispositivo móvil. Incluye una interfaz gráfica de usuario de Android y un agente de la gestión de las interacciones con otros componentes.
- *Standard Chat Client (estándar cliente chat):* Se trata de una aplicación normal de Java enfocada a los participantes de la charla desde una PC o cualquier otro dispositivo que soporte JVM estándar. Incluye los siguientes elementos:
  - Una interfaz gráfica SWING y
  - un agente de la gestión de las interacciones con otros componentes.
- *Chat Server (chatear server):* Esta es la principal plataforma de contenedores con un agente (llamado ChatManagerAgent). Realiza un seguimiento de todos los agentes que están conectados al chat.

6.3 JADE programación para Android

JADE (Java Framework de desarrollo de agente) es un software marco totalmente implementado en lenguaje Java. Simplifica la implementación de sistemas multiagentes a través de un middle-ware que cumple con las especificaciones de la FIPA (Foundation for Intelligent Physical Agents) y también mediante de un conjunto de herramientas gráficas que soporta las fases de implementación y depuración. La plataforma del agente puede distribuirse a través de máquinas (que tampoco deben compartir el mismo sistema operativo), la configuración se controla a través de una GUI remota. La configuración se cambia en tiempo de ejecución incluso moviendo agentes de una máquina a otra cuando sea necesario. JADE está completamente implementado en lenguaje Java y el requisito de sistema mínimo es la versión 1.4 de JAVA (el entorno de tiempo de ejecución o JDK).

La sinergia entre la plataforma JADE y las bibliotecas de salto permite obtener una plataforma de agente FIPA compatible con menos espacio físico y compatibilidad con entornos Java móviles hasta CLDC J2ME MIDP 1.0. Las bibliotecas de salto se han desarrollado con la colaboración del proyecto y pueden descargarse como un complemento de JADE de este mismo sitio web.

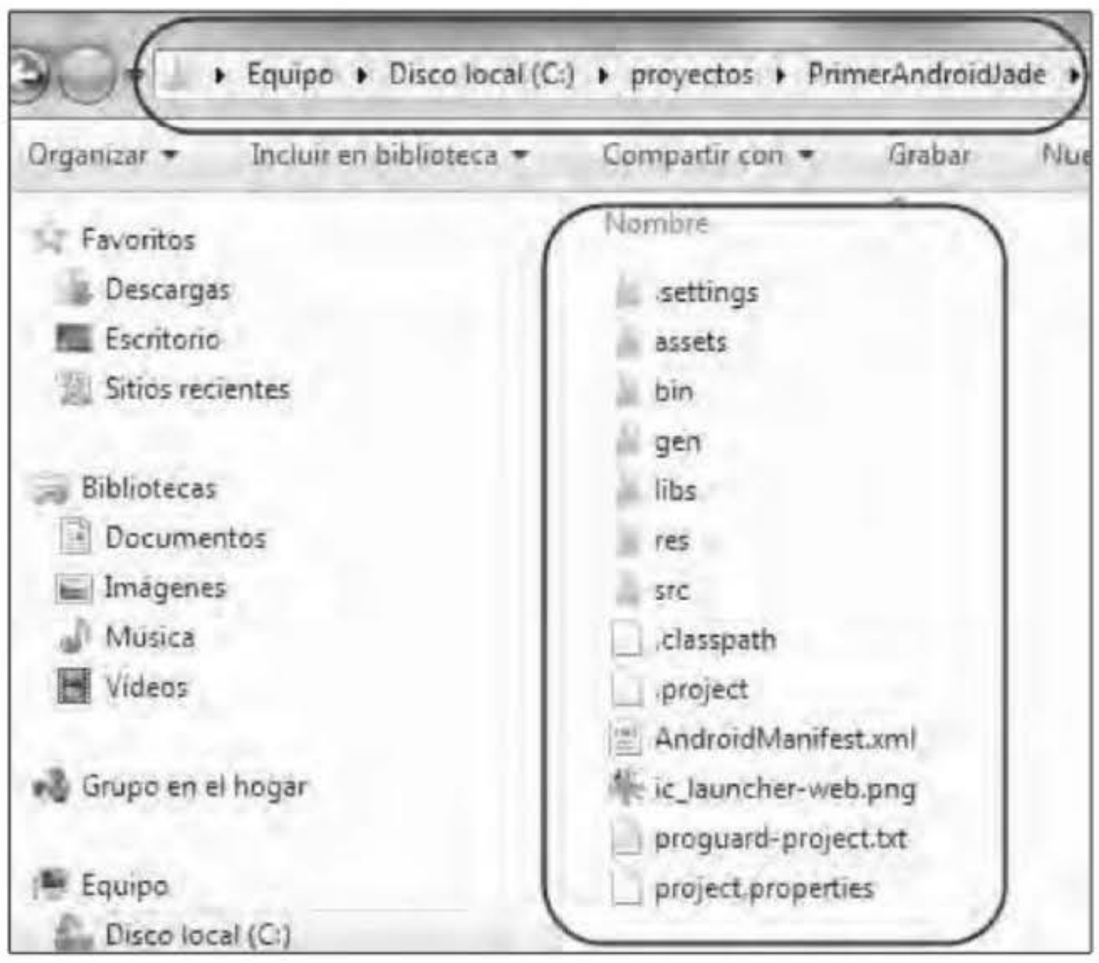
JADE es software libre y se distribuye por Telecom Italia, el titular del copyright, en software open source bajo los términos de la licencia LGPL (menor General público licencia versión 2). Desde mayo de 2003, una placa de JADE ha sido creada bajo la supervisión de la gestión del proyecto JADE. Actualmente, la junta JADE muestra cinco miembros: Telecom Italia, Motorola, Whitestein Technologies AG, Profactor GmbH y France Telecom R & D.

Paso 1: Descargue el archivo:

LeapAddOn-4.2.0.zip

Pasos 2: Descomprimirlo de:

LeapAddOn-4.2.0.zip



Pasos 3: Descargue el archivo:

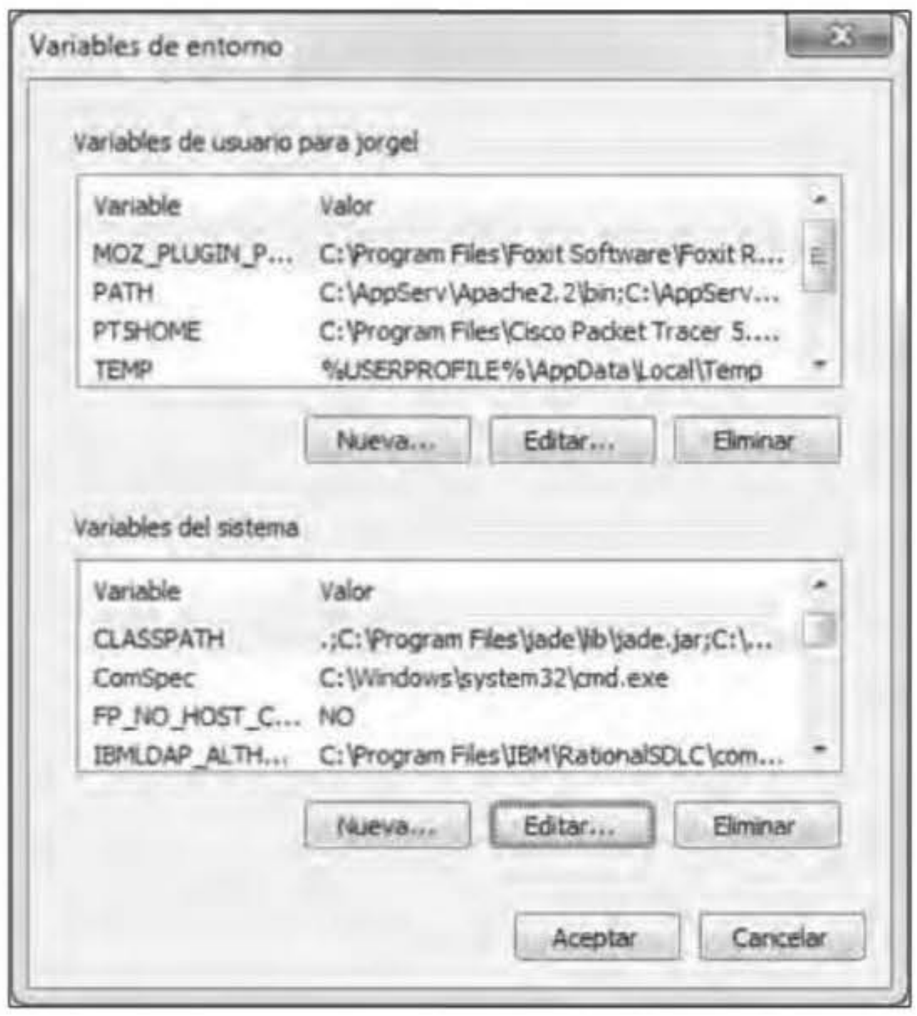
chatAndroid.zip

Pasos 4: Descomprimirlo de:

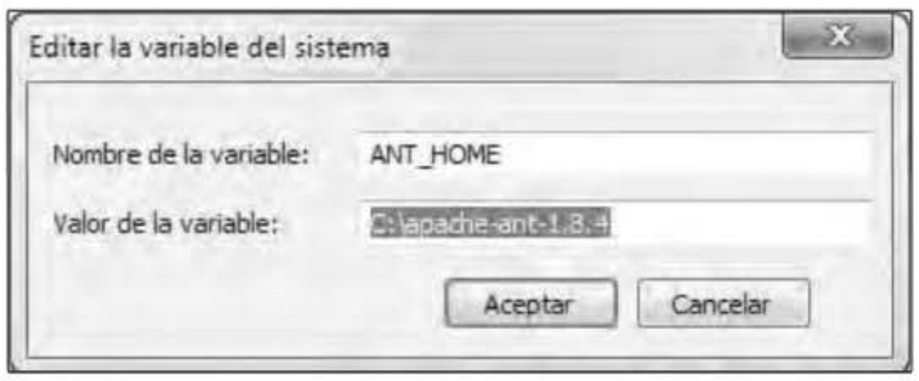
chatAndroid.zip



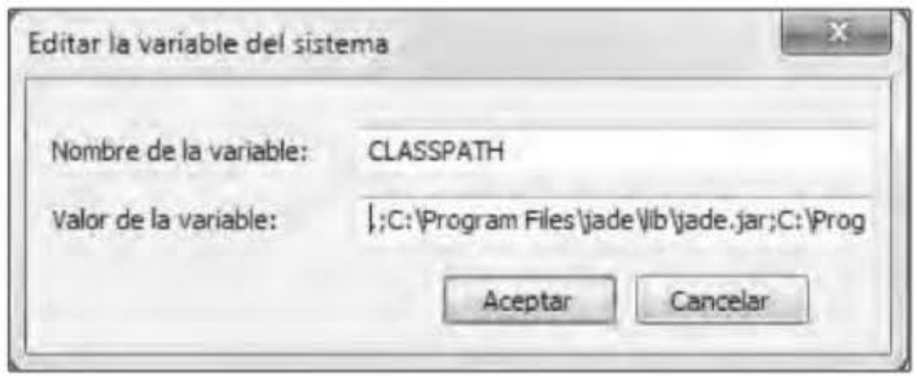
Pasos 5: Configuración del entorno.



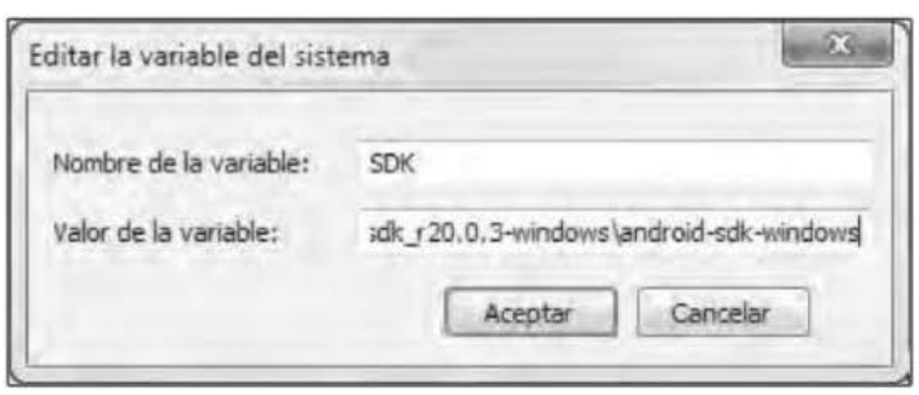
- Ant



- Jade

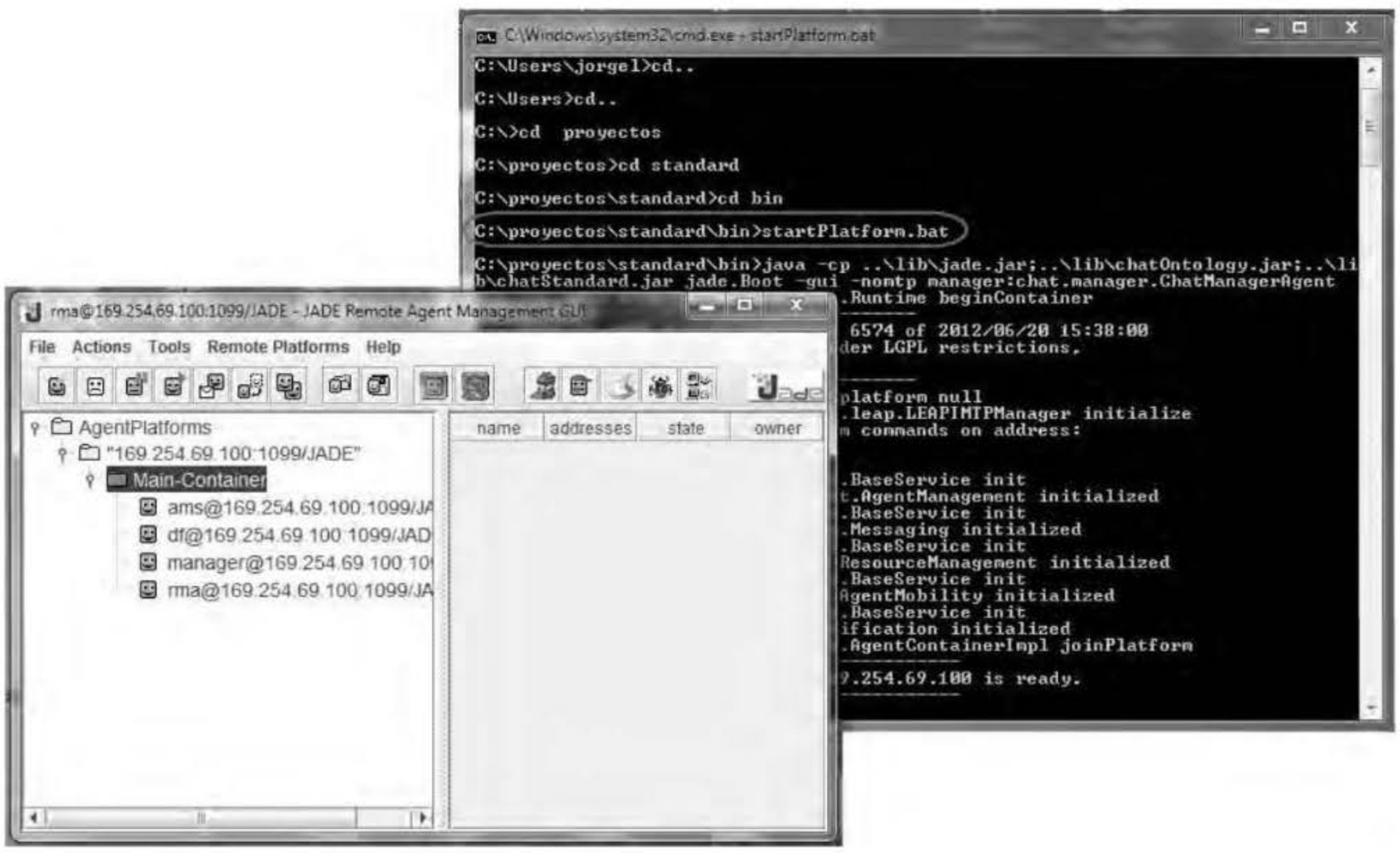


- SDK

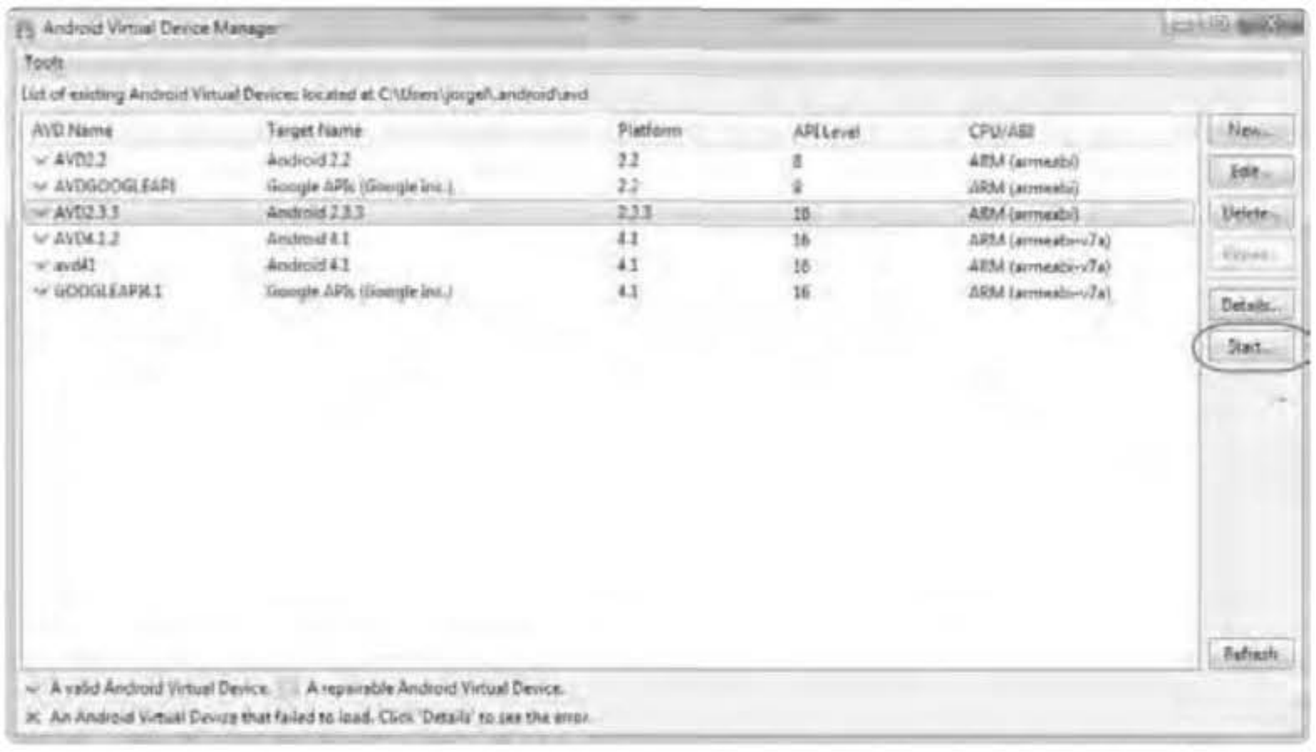


Paso 6: Arranque la plataforma Jade.





Paso 7: Inicie el emulador en la versión 2.33.



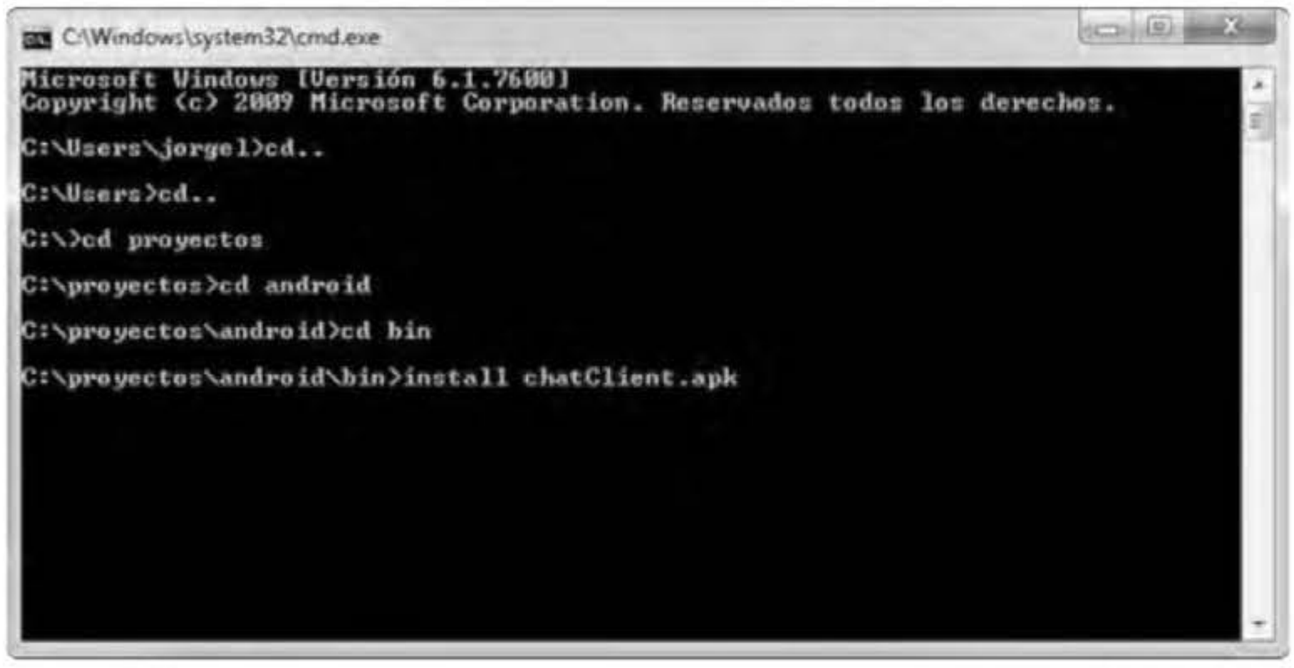
Se observa lo siguiente:



Paso 8: Instale el APK necesario para el chat-cliente.



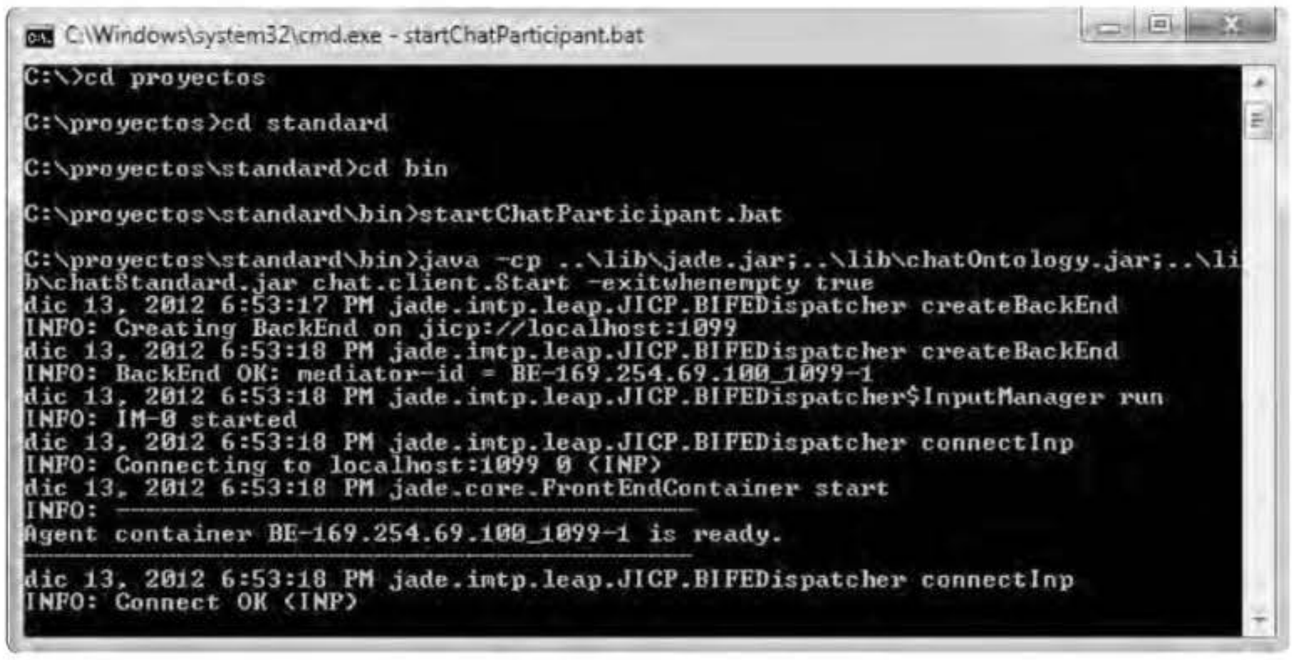
Instalación completa.



Paso 9: Ubique la aplicación Chat.



Paso 10: Inicie el chat Estándar.



Paso 11: Establezca comunicación.



6.4 Conceptos básicos

6.4.1 Web semántica

Razonando sobre la web semántica en el contexto de la historia clínica

Actualmente los datos que se encuentran en la World Wide Web (WWW) carecen de significado y son un conjunto de datos aislados que se encuentran agrupados por medio de etiquetas (tags) de HTML (HiperText Markup Language) las que solo representan instrucciones de formato como: H1, TD, TR, etc. En ese sentido, la semántica, el significado de los datos que nos muestra la Web, es proveído por el ser humano cuando lee la página web. Sin embargo, el lenguaje HTML tiene la etiqueta <META> con la cual se intenta darle algo de semántica a los datos en la Web, ésta se utiliza para añadir información sobre la página y es la que los buscadores como Google consultan para encontrar coincidencias exactas con las palabras que el usuarios pretende encontrar, por lo tanto, su semántica es primitiva (Hebeler et al, 2009).

La web semántica propuesta por Tim Berners-Lee en el 2001 implementa la semántica a través de conexiones estandarizadas entre los datos para crear contexto [es mucho más que un conjunto de palabras claves; esto solo se logrará cuando se establezca relaciones entre ellas; es decir, relaciones entre los datos para crear contexto] y, por lo tanto, significado. En la siguiente construcción HTML:

```
<META name="author" content="Research Group">
<META name="keywords" content="paciente">,
```

¿A qué se refiere la palabra paciente? ¿Se refiere a construir una ontología que se enfoque en datos del paciente o síntomas del paciente o diagnósticos del paciente?, es decir, ¿qué se quiere decir con la palabra paciente? Como vemos, la semántica queda para que el lector humano la interprete. Sin embargo, si las palabras clave se relacionaran a otras palabras clave se formará una red de datos que revela significado. Así, paciente podría relacionarse a muchas otras palabras como médico, enfermera, medicina, etc. y se puede notar que estas relaciones expresan semántica, es decir, significado.

Fuente: Dr. Néstor Adolfo Mamani Macedo
(obtenido de http://www.usmp.edu.pe/ffia/gaceta/gaceta2-1.pdf)

Por ejemplo, lo anterior podría interpretarse como cuando "un médico atiende a un paciente e indica que la enfermera le brinde una medicina". Aunque esta semántica es aún imprecisa, la frase es solo una de muchas otras interpretaciones que pudieran surgir, otra puede ser "una enfermera recibe un paciente en emergencia quien tomó una cierta medicina y solicita que el médico lo atienda para que realice el diagnóstico".

Implementar relaciones dentro de la Web requiere que se formalice una estructura o arreglo de los términos en un estándar de reglas gramaticales específicas y en la creación de un nuevo estándar de términos o elementos de formación primitivos para la descripción de los datos y de los procesos, el vocabulario. Estos dos estándares juntos, la gramática y el vocabulario restricto forman el lenguaje y ayudan a incorporar significado, es decir, semántica. Por otro lado, en la medida que la gramática y el vocabulario del lenguaje se expandan a través de nuevas relaciones la semántica se irá enriqueciendo. Los componentes de una aplicación de web semántica se dividen en dos categorías principales:

- componentes y
- herramientas de la web semántica asociadas.

Al respecto, Timothy (Tim) Berners-Lee, el creador de la Web, afirma:

> "La web semántica no es una Web separada sino una extensión de la actual, en la que la información está dada con significado bien definido lo que significa que permite que los computadores y las personas trabajen en cooperación" (Berners-Lee et al, 2001).

En otras palabras, la web semántica es una red de datos descritos y vinculados de manera que se establezca semántica y que se adhiere a una gramática establecida y a constructos o términos de un lenguaje (o vocabulario).

> "Conceptualmente, es factible añadir semántica a la Web por medio un conjunto de instrucciones de programación; pero la programación en sí misma se complicaría innecesariamente en una inmensidad de comandos if, case, búsquedas en la base de datos y otras técnicas de programación" (Hebeler et al., 2009).

La web semántica propuesta por Berner-Lee et al. (2001) implementa la semántica a través de conexiones estandarizadas para relacionar información. Así, una persona o un programa puede decir fácilmente si este paciente es lo mismo que otra referencia paciente. Cada dato único se conecta a un contexto más amplio o Web. Esta Web ofrece precisión para las definiciones, relaciones a una jerarquía conceptual, relaciones a información asociada y relaciones a individuos específicos de paciente. La flexibilidad de la Web posibilita conexiones a toda la información necesaria.

Los términos que se relacionan y los caminos que se forman con todas las relaciones dan forma a un vocabulario de dominio, también conocido como *ontologiá*.

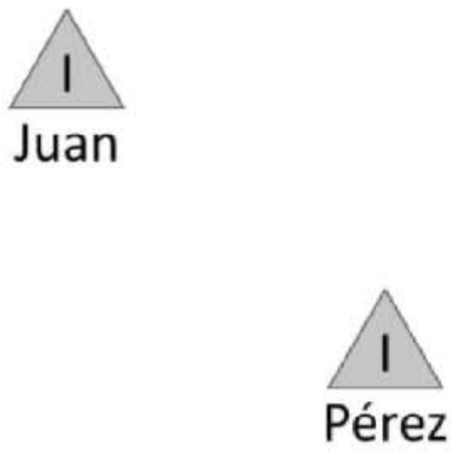


Figura 6.1 Las palabras claves Juan y Pérez en la WWW.

Las figuras 6.1 y 6.2 ilustran la diferencia con la que se almacena la información en la WWW, como palabras claves independientes y una web semántica de datos. En el ejemplo, se utilizan las palabras claves Juan, Pérez.

En la figura 6.1, se observa cómo en la WWW los datos están aislados y sin conexiones con otros datos limitando su semántica y dejando al lector humano que determine su significado. Un buscador retornará los datos de Juan y Pérez cuando están juntos en un único elemento (por ejemplo, una página web) obteniendo resultados limitados o brindando resultados de los dos datos por separado, en este caso la información no será coherente para el lector.

Leyenda: I →Instancia

En la figura 6.2, se ve una ontología en la web semántica, los datos existen relacionados entre sí y, además, interrelacionados con otros datos y definiciones por medio de relaciones; todo este conglomerado de relaciones lógicas dan contexto y significado a las palabras Juan y Pérez. De hecho, esto es una ontología de historia clínica dentro de la cual están las palabras claves. Aquí la web semántica permite que, al igual que lo hace un ser humano, una aplicación también pueda seguir estas relaciones y obtener información enriquecida. Un buscador en la web semántica retornará los datos Juan y Pérez; pero además dirá su DNI, dirección y fecha de nacimiento; adicionalmente, podrá decir el número de historia clínica y los síntomas que tiene: mareos, fiebre e ictericia. Por otro lado, si el buscador fuera capaz de inferir, también retornará información complementaria, en este caso, es obvio que Juan Pérez es un paciente de una institución sanitaria en particular.

Por supuesto, debe existir también un Juan Pérez, ingeniero, contador, etc., en cuyo caso, la web semántica retornará a todos los Juan Pérez con información acerca de cada uno. Es decir, la web semántica habrá navegado por varias ontologías diferentes y, todo esto, será transparente para el lector humano.

La web semántica alcanza a la semántica a través de conexiones estandarizadas e información relacionada de un dominio (ontología), cada elemento de dato se conecta a un contexto mayor en la red, la cual ofrece potenciales caminos para su definición: relaciones a una jerarquía conceptual, relaciones de información asociada y relaciones de instancias específicas. Por lo tanto, las aplicaciones de web semántica siempre usan ontologías, cada una especializada en un área de dominio específica.

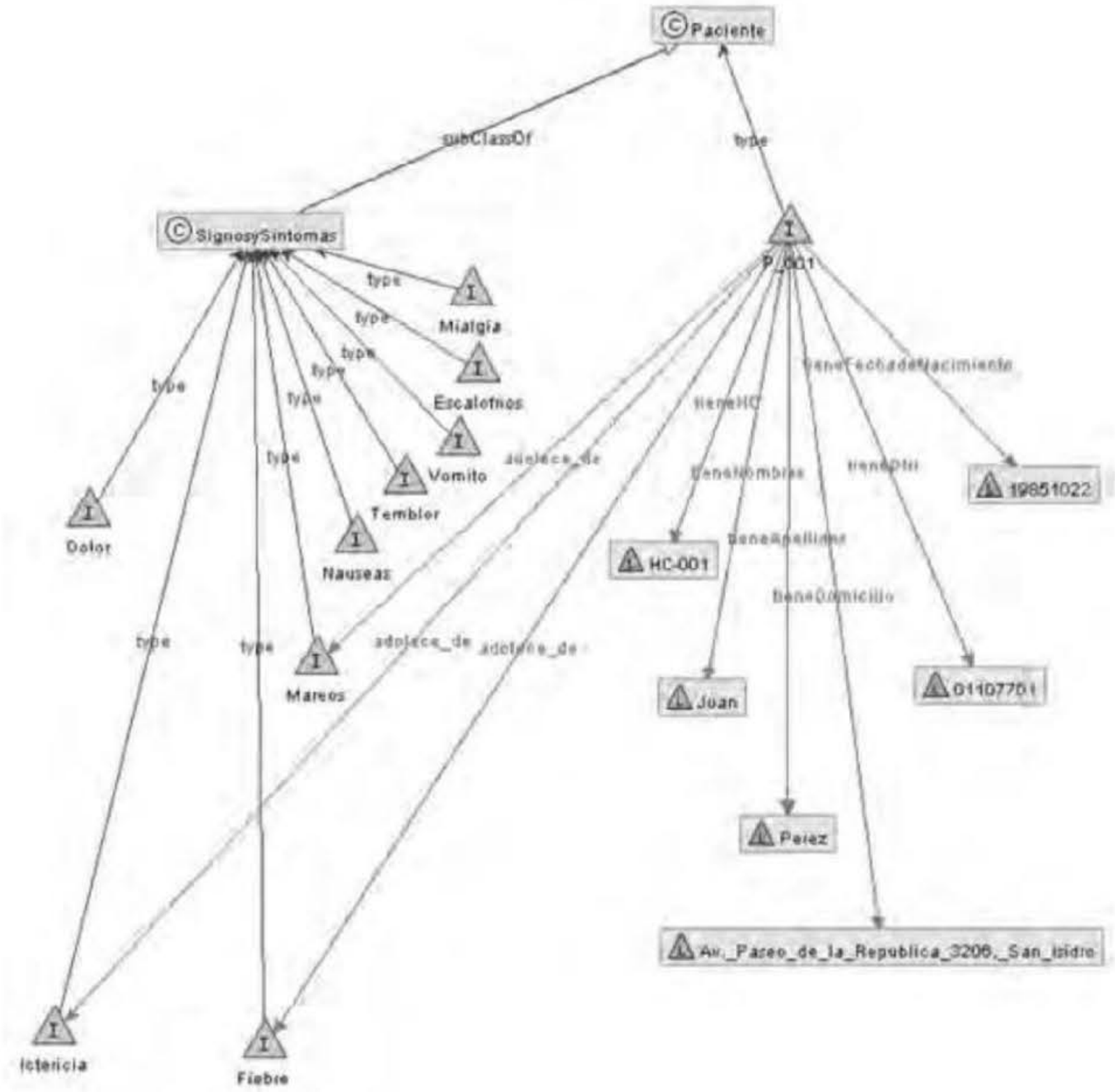


Figura 6.2 Las palabras claves Juan y Pérez en la web semántica.

6.4.2 Implementar la web semántica



La Web tal como se conoce, consiste principalmente de contenido estático dirigido al ser humano, quien debe interpretarlo y darle significado; es decir, el usuario infiere la semántica. El contenido de una página web se vincula a otro contenido vía el URL (Universal Resource Locator). El contenido de la WWW no tiene construcciones lógicas formales, en cambio, la web semántica consiste principalmente de declaraciones para consumo de aplicaciones de software; estas declaraciones se vinculan a través de construcciones que pueden formar semántica; así, la semántica del vínculo proporciona un camino significativo definido en vez de uno interpretado por el usuario, tome como ejemplo la declaración "Juan amigo de Carlos", es una declaración de asociación que menciona que Juan es un amigo de Carlos, no un enemigo, ni un desconocido; es más, esta declaración permite inferir que hay una relación entre Juan y Carlos y viceversa.

El elemento fundamental de la web semántica es la declaración. Los diferentes tipos de declaraciones comparten el mismo estándar y describen conceptos, lógica, restricciones e individuos (instancias). El utilizar un mismo estándar posibilita su compartición e integración. En vez de ser un sustituto de la WWW, la web semántica la extiende con significado mediante el uso de declaraciones estandarizadas

para darle semántica y lograr así compartir información dispersa en la Web.

Las relaciones semánticas forman la web semántica. Estas relaciones incluyen definiciones, asociaciones, agregaciones y restricciones. La figura 6.3 muestra un grafo de declaraciones. Las declaraciones y sus correspondientes relaciones establecen ambos conceptos (por ejemplo, una persona tiene Nombres y Apellidos) e instancias (por ejemplo, Código de la Persona P-001 con Nombres Juan y Apellidos Pérez). Las declaraciones que definen los conceptos y las relaciones entre estos forman una ontología. Las declaraciones que se refieren a individuos forman datos de instancia.

Las declaraciones pueden ser de dos tipos:

- asertivas
- inferidas

La primera requiere que la aplicación cree directamente la declaración de conceptos o de individuos, para hacerla valer en su momento (líneas sólidas). La última requiere de un razonador para inferir declaraciones adicionales desde las declaraciones asertivas (línea verde punteada). Aquí se puede inferir que Juan puede tener un cáncer pancreático debido a que tiene los síntomas.

Note que en este ejemplo, la web semántica, está utilizando una combinación de dos ontologías, la ontología de historia clínica y la ontología de enfermedades. Debido a que Juan Pérez tiene, entre sus síntomas, tres en particular: mareos, fiebre e ictericia, que también son los síntomas del cáncer pancreático, es que se puede inferir la enfermedad que afecta a Juan Pérez. En la web semántica, esta misma deducción podrá ser efectuada por un aplicativo.

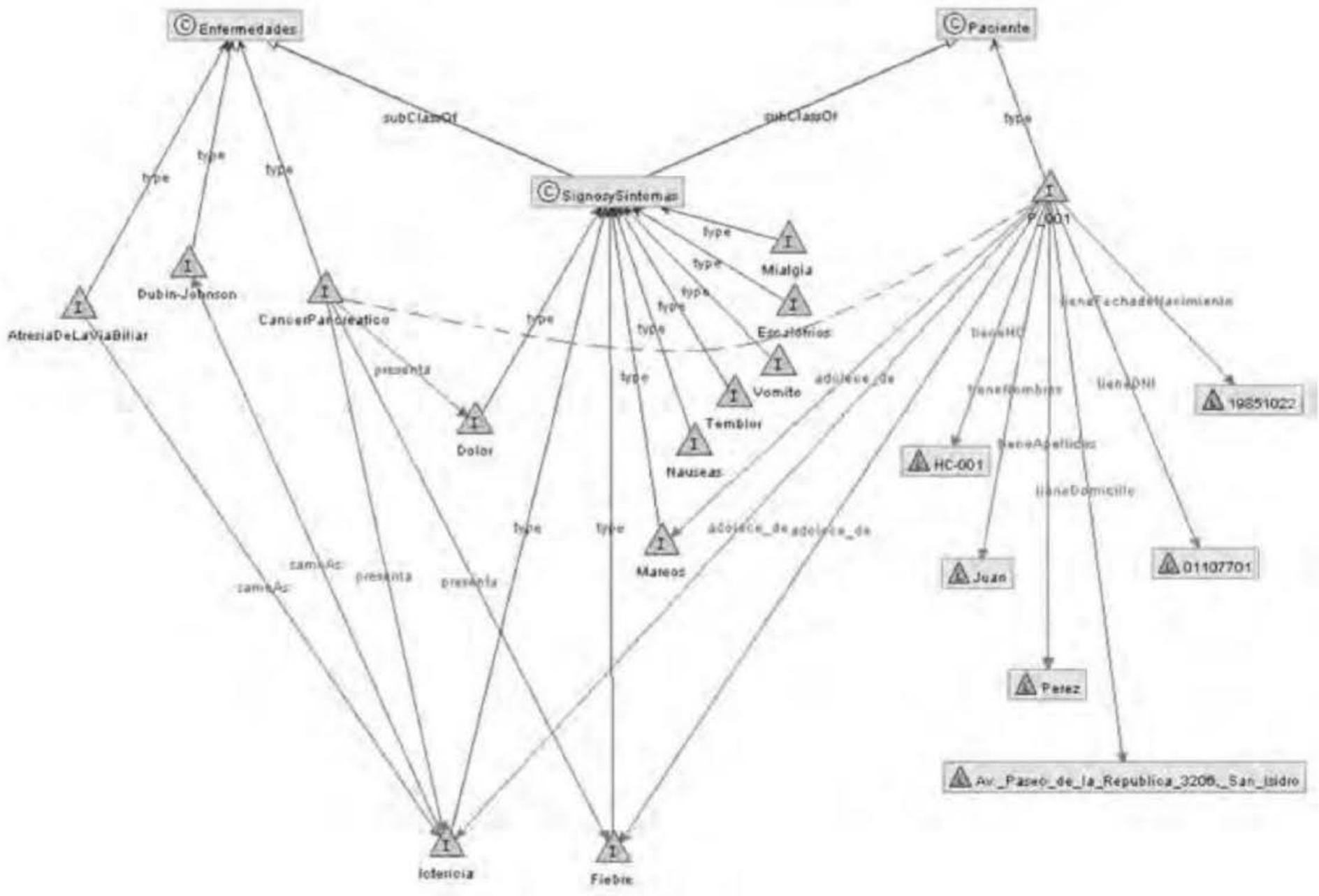


Figura 6.3 Inferencia en la web semántica.

Este es un ejemplo sencillo que solo quiere explicar el proceso interno de la web semántica. En la práctica, este proceso se repetirá constantemente y con muchas ontologías a la vez.

Las declaraciones de web semántica emplean un lenguaje de web semántica (vocabulario + reglas de gramática) para identificar los diversos tipos de declaraciones y relaciones. La web semántica tiene varios lenguajes, que van de básicos hasta complejos. La web semántica existe principalmente en dos formas:

- bases de conocimiento y
- archivos.

Las **bases de conocimiento** ofrecen almacenamiento dinámico y extensible similar a las bases de datos relacionales. Los archivos típicamente contienen declaraciones estáticas. La diferencia es que las bases de conocimiento utilizan herramientas (motor de almacenamiento), los archivos no. La tabla 6.1 compara las bases de datos relacionales y bases de conocimiento.

Tabla 6.1: Comparación de bases de datos relacionales y bases de conocimiento.

Característica	BD Relacional	Base de Conocimiento
Estructura	Esquema	Declaraciones de Ontología
Datos	Filas	Declaraciones de Instancia
Lenguaje de Administración	DDL	Declaraciones de Ontología
Lenguaje de Consulta	SQL	SPARQL
Relaciones	Claves Foráneas	Múltiples Dimensiones
Lógica	Externa a la BD/Triggers	Declaraciones de Lógica Formal
Singularidad	Clave de la tabla	URI

Las **bases de datos relacionales** dependen de un esquema por cada estructura. Una base de conocimiento en el contexto de la web semántica depende de declaraciones de ontología para establecer la estructura. Las bases de datos relacionales están limitadas a un tipo de relación, la *clave foránea*. La web semántica ofrece relaciones y restricciones multidimensionales tales como herencia, parteDe, asociadoCon y muchos otros tipos, incluyendo relaciones lógicas y restricciones.

En las bases del conocimiento el lenguaje utilizado para formar la estructura y las instancias es el mismo lenguaje; pero son diferentes en las bases de datos relacionales. Las bases de datos relacionales ofrecen un lenguaje diferente:

- DDL *(Data Description Language)* para crear el esquema y
- DML *(Data Manipulation Language)* para el mantenimiento de los datos.

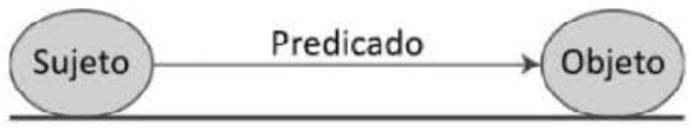
Es decir, en las bases de datos relacionales, añadir una tabla o columna es diferente a añadir una fila.

6.4.3 Componentes de programación

Según Hebeler et al. (2009), los componentes principales de la web semántica son los siguientes:

- *Declaración:* Es la base de la web semántica; se compone de varios elementos que suelen formar una triada o tripla (3-tupla). La triada consta de un:
  - sujeto,
  - predicado y
  - objeto (por ejemplo, Juan es un tipo de persona).

Las declaraciones definen la estructura de información, instancias específicas y los límites de esa estructura y se relacionan entre sí para formar la red de datos que constituye la web semántica.



- URI (identificador uniforme de recursos): Proporciona un nombre único para los elementos que figuran en una declaración a través de toda la Internet. Así, cada componente de una declaración (sujeto, predicado y objeto) contiene un URI para afirmar su identidad en toda la WWW. Esto elimina conflictos de nomenclatura, verifica si dos elementos son los mismos o no y brinda una ruta de acceso a información adicional. La estructura del URI es la siguiente:

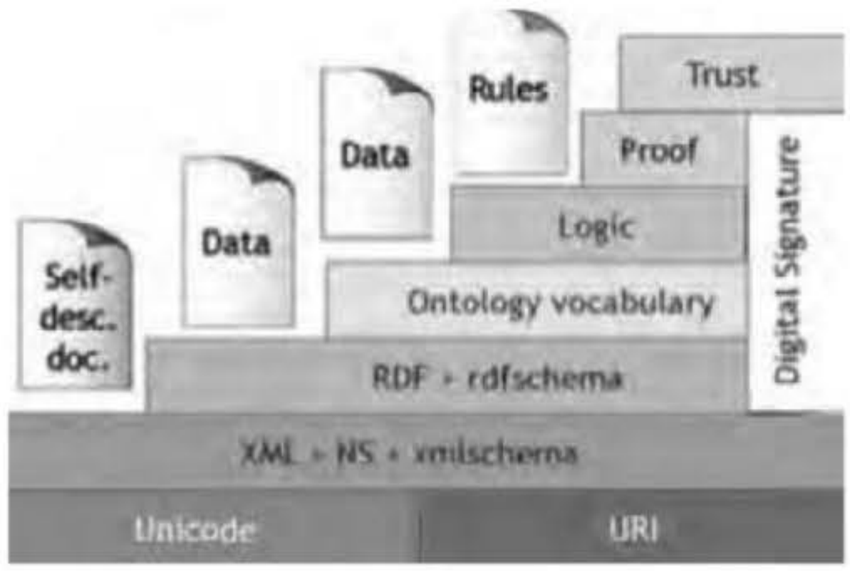
ESQUEMA: PARTE JERARQUICA? SOLICITUD # FRAGMENTO

A diferencia del URL, el URI permite incluir en la dirección una subdirección, determinada por el "fragmento".

- *Esquema:* Nombre que se refiere a una especificación para asignar los identificadores, e.g. urn:, tag:, cid:. En algunos casos también identifica el protocolo de acceso al recurso, por ejemplo "http:", "mailto:", "ftp:".
- *Autoridad:* Elemento jerárquico que identifica la autoridad de nombres (por ejemplo, "//es.wikipedia.org").
- *Ruta:* Información usualmente organizada en forma jerárquica que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres (por ejemplo, "/wiki/Uniform_Resource_Identifier").
- *Consulta:* Información con estructura no jerárquica (usualmente pares clave=valor) que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres. El comienzo de este componente se indica mediante el carácter "?".
- *Fragmento:* Permite identificar una parte del recurso principal o vista de una representación del mismo. El comienzo de este componente se indica mediante el carácter "#".

Aunque se acostumbra llamar URL a todas las direcciones web, URI es un identificador más completo y, por eso, es recomendable su uso en lugar de la expresión URL.

Un URI se diferencia de un URL en que permite incluir en la dirección una subdirección, determinada



por el "fragmento".

- *Lenguaje:* Las declaraciones se expresan de acuerdo con un lenguaje de web semántica. El lenguaje se compone de un conjunto de palabras claves que proporcionan instrucción a las diversas herramientas de la web semántica. Hay varios lenguajes para elegir que ofrecen diversos grados de complejidad y expresividad semántica.

- *Ontología:* Una ontología consiste de declaraciones que definen conceptos, relaciones y restricciones; es análoga a un esquema de base de datos. La ontología forma un modelo de dominio de información especializada en áreas como finanzas y medicina o generalizada para la descripción de objetos comunes. Actualmente, existen muchas ontologías enriquecidas para incorporar en las aplicaciones que pueden ser usadas directamente o adaptarlas a necesidades específicas.

- *Datos de instancia:* Son las declaraciones que contienen información sobre casos específicos en lugar de un concepto genérico. "Juan" es una instancia, mientras que "persona" es un concepto o clase. Esto es análogo a los objetos/instancias en un programa orientado a objetos. Los datos de instancia constituyen la mayor parte de la web semántica. Una ontología que contiene el concepto persona puede ser utilizada por millones de instancias de persona.

6.4.4 Herramientas

Las herramientas que construyen, manipulan, interrogan y enriquecen la Web Semántica son las siguientes: (Hebeler et al, 2009):

- *Herramientas de construcción:* Permiten construir e integrar una web semántica a través de la creación o importación de las declaraciones de la ontología y las instancias. Hay herramientas basadas en GUI que permiten ver y explorar los datos de la web semántica; por ejemplo, Protégé. Existen, también, varias interfaces de programación de aplicaciones (API) para integrar con los programas como, por ejemplo, Jena.

- *Herramientas de interrogación:* Estas herramientas navegan a través de la web semántica para devolver una respuesta solicitada. Existen varios métodos de interrogatorio que van desde la simple navegación gráfica para buscar, hasta un lenguaje de consulta completo; por ejemplo, SPARQL (SPARQL *Protocol and RDF Query Language*).

- *Razonadores:* Los razonadores añaden inferencia a su web semántica. La inferencia crea adiciones lógicas que permiten que los conceptos y las relaciones se relacionen adecuadamente con otros, tal como una "persona" es un *ser vivo*, el "papá" es un *padre*, "casado" es un *tipo de relación* o "casado" es una *relación simétrica*. Hay varios tipos de razonadores que ofrecen distintos niveles de razonamiento; por ejemplo, "Pellet" presenta un alto nivel de razonamiento. Los razonadores aprovechan las declaraciones afirmativas para crear declaraciones auxiliares lógicamente válidas.

- *Motores de reglas:* Los motores de reglas soportan inferencia normalmente más allá de lo que puede deducirse con la lógica descriptiva, como ejemplo está Jess. Las reglas permiten fusionar

ontologías y otras tareas de lógica más grandes, incluyendo métodos de programación, tales como búsquedas de conteo y de cadena. Los motores de reglas son manejados por reglas que se pueden considerar parte de toda la representación del conocimiento. Cada motor de reglas se adhiere a un lenguaje de reglas determinado.

- *Entornos semánticos:* Estos empaquetan las herramientas mencionadas anteriormente para trabajar como una unidad integrada. Existen alternativas de código abierto tanto para un entorno gráfico de desarrollo integrado (IDE) como para una interfaz de programación (API); un ejemplo es el Jena, aunque también hay varios entornos semánticos comerciales excelentes. Esto le permite empezar a programar de inmediato.

6.4.5 Impactos de la web semántica en la programación

Para que las aplicaciones tomen ventaja de la web semántica y de sus herramientas, estas deben adaptar sus expectativas e impactos en algunas de las siguientes categorías (Hebeler et al, 2009):

- *Centrada en la Web de datos:* La aplicación de web semántica debería estar centrada en los datos. Los datos son la clave, no las instrucciones de programación. En la web semántica los datos están enriquecidos a tal punto que se requiere de menos programación.
- *Datos semánticos:* Una aplicación de web semántica debería poner significado directamente dentro de los datos en vez de ponerlo dentro de las instrucciones de programación o dejarla a la interpretación del usuario. La web semántica ofrece métodos para definir la información, sus relaciones con otras informaciones, su herencia conceptual y su formación lógica.

Ejemplo:

Una historia clínica dentro de la web semántica podría tener las relaciones "paciente de" para asociar a un paciente con el(los) médico(s) con quien(es) se está atendiendo. Ya que las relaciones tienen independencia de los conceptos, una aplicación podría consultar por todos los datos "paciente de" Carlos Zapata. El resultado sería todos los pacientes del Dr. Carlos Zapata.

- *Integración/Compartición de datos:* Una aplicación de web semántica debería tratar de accesar y compartir recursos ricos en información a través de la WWW cuando sea apropiado, tomando ventaja de las diversas fuentes de datos ya existentes. La capacidad de intercambiar información aumenta el valor de la información, ya que permite a las aplicaciones utilizar la más actualizada en vez de formarla desde cero. Los artefactos de web semántica (ontologías identificadas con URI) son únicos en toda la Internet.
- *Datos dinámicos:* Una aplicación de web semántica debería permitir cambios dinámicos en tiempo de ejecución a la estructura y contenidos de su información. En la WWW, el cambio es dinámico y continuo. Las aplicaciones de web semántica pueden insertar nuevos datos en cualquier momento. Estos nuevos datos se obtienen a través de la inferencia.

Estos cuatro impactos potencialmente cambian la forma en que se diseña y programa una aplicación y guían su solución a un uso óptimo de la web semántica.

6.4.6 Razonador

Un razonador se guía por una regla de tres (porque, se sigue que, en consecuencia); este vive solo en el presente; el futuro es consecuencia de lo que se realice en el presente. Se encuentra siempre a la espera de las consecuencias que deban seguirse; esto implica que siempre estará ocupado controlando

las premisas para que ocurra lo que deba ocurrir. Es un tecnólogo de lo que realiza.
Un razonador orientado a la historia clínica sería aquel que, basado en el conocimiento, proponga un posible diagnóstico; para lograr este propósito se necesita una base de conocimiento (compuesto normalmente por hechos y reglas) y reglas que sean capaces de inferir un nuevo conocimiento.

Existen dos formas de conseguir información de una base de conocimiento:

- Mediante una consulta al modelo de la cual el razonador devolverá la información de las instancias existentes.
- Mediante la adición de reglas, donde el razonador; además, de devolver las instancias que resulten de la consulta al modelo, será capaz de generar nuevas instancias que se infieran de las existentes. Se le denomina a este tipo inferencia o razonamiento.

La base del conocimiento (KB) está basada en dos componentes:

- TBox y
- ABox

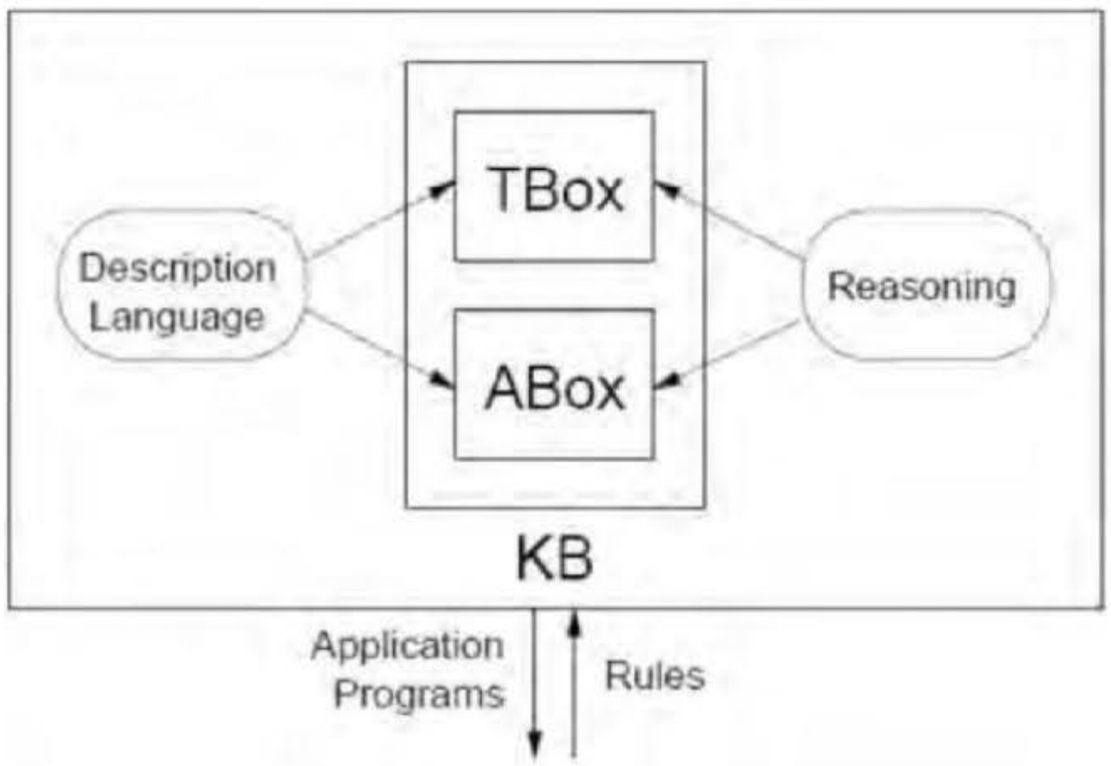


Figura 6.4 Componentes de una base del conocimiento.

El TBox define los dominios y las propiedades que pertenecen a esos dominios y la dependencia jerárquica entre dichos dominios. Los individuos se especifican sobre el componente ABox.
Así, por ejemplo, se definen los siguientes axiomas:

Médico es Persona (Médico $\subseteq$ Persona).

Carlos es Médico (Carlos $\in$ Médico).

Donde el primer axioma es de tipo TBox y el segundo es ABox. De esta información se infiere que Carlos $\in$ Persona.

Una base de conocimiento (TBox y ABox) es equivalente a un conjunto de axiomas de la LPO (lógica de primer orden) y, por tanto, se puede definir un cálculo o sistema de inferencia que permite derivar conocimiento implícito a partir del explícito de la base de conocimiento.

6.4.7 Lógica descriptiva

También llamada lógica de descripción (DL por *Description Logics*) son una familia de lenguajes de representación de conocimiento que describen al dominio en función de conceptos (clases), roles (relaciones entre clases) e individuos (instancias de las clases, pueden relacionarse con un rol).

El nombre de lógica descriptiva nació en la década del 1980. Anteriormente, se le llamaba de la siguiente manera, según un orden cronológico:

- Sistemas terminológicos
- Lenguajes de conceptos

Hoy en día, la lógica descriptiva es la piedra angular de la web semántica para su uso en la base de conocimiento o en el diseño de las ontologías.

El primer sistema experto basado en DL fue KL-ONE (por Brachman y Schmolze, 1985). Luego aparecieron otros sistemas como LOOM, BACK, KRIS, CLASSIC, FACT, RACER, CEL, KAON2, PELLET y otros.

A. Notaciones

- *Lógica ALC (attributive Language):* Conocida como sintaxis alemana debido a sus creadores. Soporta la lógica descriptiva de conceptos, roles e individuos.
- *Sistema SH:* Actualmente, se está implementando para el uso de las ontologías.
- $\mathcal{SHIQ}$ es $\mathcal{ALCQI}$ + roles transitivos + inclusión roles.
- $\mathcal{SHOIQ}$ es $\mathcal{SHIQ}$ + nominales.
- Se demuestra también que $\mathcal{SHOIQ}$ es $\mathcal{SHOIN}$ extendida con restricciones cualificadas.
- $\mathcal{SHOIN}(\mathcal{D})$ es $\mathcal{ALCIN}$ + nominales + dominios concretos ($\mathcal{D}$).

  Aunque extender una lógica con dominios concretos la dota de una expresividad valorada para representar ontologías, fácilmente puede llevar a la indecidibilidad. Verá, sin embargo, que $\mathcal{SHOIN}(\mathcal{D})$ es decidible y es base para el lenguaje de ontología actualmente más aceptado.

DL	PROPIEDAD EXPRESABLE EN LA LÓGICA	COMPLEJIDAD
$\mathcal{ALC}$	lógica de descripción básica	PSPACE
$\mathcal{ALCN}$	+ restricciones numéricas no calificadas $(\mathcal{N})$	PSPACE
$\mathcal{ALC}$ **reg**	+ expresiones regulares sobre roles (reg)	EXPTIME
$\mathcal{ALCI}$ **reg**	+ inverso de roles $(\mathcal{I})$	EXPTIME
$\mathcal{ALCFI}$ **reg**	+ restricciones funcionales sobre roles atómicos $(\mathcal{F})$	EXPTIME
$\mathcal{ALCQI}$ **reg**	+ restricciones numéricas calificadas $(\mathcal{Q})$	EXPTIME
$\mathcal{ALCQO}$ **reg**	+ un alfabeto para los objetos del dominio $(\mathcal{O})$	EXPTIME
$\mathcal{SHIQ}$	$\mathcal{ALCQI}$ + roles transitivos + inclusión roles	EXPTIME
$\mathcal{SHOIN}$	+ restricciones numéricas no calificadas $(\mathcal{N})$	EXPTIME
$\mathcal{SHOIQ}$	$\mathcal{SHIQ}$ + nominales	NEXPTIME
$\mathcal{SHOIN}(\mathcal{D})$	$\mathcal{SHOIN}$ + dominios concretos	EXPTIME

B. *Algunos razonadores para lógica descriptiva*

Pueden ser FaCT++, KAON2, Pellet, RacerPro, etc. Tómese en cuenta que Pellet, de Clark & Parsia son los que más se adecúan al tipo de proyecto. Debido a su condición de open source (facilita la integración y libre modificación del código fuente) así como por la calidad de su documentación, el soporte y las constantes actualizaciones que mantienen activo el proyecto y en continuo crecimiento.

6.4.8 LENGUAJES DE LA WEB SEMÁNTICA

Desde la década de los 90, se comenzó la investigación de cómo llevar una representación de conocimiento a la *World Wide Web*. En 1997, paralelamente al desarrollo de XML *(Extensible Markup Language)*, apareció RDF *(Resource Description Framework)* como una capa por encima de XML, proporcionando una base común para expresar la semántica, inspirado en PICS, Dublin Core, XML y MCF. Un año después, apareció RDFS *(Resource Description Framework Schema)* que se inspiró en los sistemas orientados a objetos y lenguajes como Java. En 1999, la primera especificación RDFS apareció OIL *(Ontology Inference Layer)*.

La *Web-Ontology* (WebOnt) del W3C tomó como punto de partida OIL que es básicamente una ampliación de la definición de RDFS y su evolución, DAM+OIL, para la construcción de un nuevo lenguaje con el nombre "OWL", el cual se convirtió en el estándar por la *W3C Recommendation* en febrero de 2004 y que tomó 19 años de investigación en el área de *Description Logics*.

A. *Componentes de la web semántica*

- *XML* aporta la sintaxis superficial para los documentos estructurados, pero sin dotarles de ninguna restricción sobre el significado.
- *XML* Schema es un lenguaje para definir la estructura de los documentos XML.
- *RDF* es un modelo de datos para los recursos y las relaciones que se puedan establecer entre ellos. Aporta una semántica básica para este modelo de datos que puede representarse mediante XML.

Estas expresiones se conocen como tripletas:

- Sujeto que identifica un recurso.
- Predicado o propiedades, estos definen las relaciones y tipos de valores a un sujeto.
- Objeto, también llamado "literal" o valor.

- *RDF Schema* describe las propiedades y relaciones de un RDF; además, proporciona la semántica para establecer jerarquías entre las relaciones.
- *OWL* aporta más al vocabulario para describir clases y propiedades: ofrece cardinalidad, tipos de relaciones (ej. disyunción), igualdad, tipología de propiedades más complejas, clases enumeradas y otras. OWL se divide en tres tipos:
  - Lite, el más sencillo.
  - DL, evolución del anterior, con mayor expresividad soporta razonadores.
  - Full.

Sobre el acrónimo OWL

Alguien podría pensar que el acrónimo correcto para Web Ontology Language debería ser WOL en lugar de OWL. Otros creen que el orden ha sido elegido en honor del personaje Owl de Winnie the Pooh, quien escribe su nombre WOL en lugar de OWL. Realmente, OWL fue propuesto como un acrónimo que fuese fácilmente pronunciable en inglés, facilitase buenos logos y se relacione con el prestigioso proyecto de representación del conocimiento de los años setenta de Bill Martin One World Language.

Fuente: es.wikipedia.org

B. Implementar un razonador en la web semántica orientada a historias clínicas

Como se ha visto, la **OWL** es una base de conocimiento derivado de un RDF al cual se le puede añadir razonadores para obtener una inferencia a partir de una consulta, reglas o ambas.

Para el proyecto de historias clínicas se optó por usar el razonador Pellet. Para lograr esta integración de un razonador con una base de conocimiento se necesita de algún framework que lo permita entre los cuales se tienen los siguientes:

FRAMEWORK	LENGUAJE INTERFAZ	NIVEL SEMÁNTICA	DESCRIPCIÓN
Jena	Java	RDF to OWL 2	Ampliamente utilizado marco de la Web Semántica para Java. Proporciona una interfaz de SPARQL, RDF y API OWL, y el apoyo de inferencia. Permite almacenar múltiples y mecanismos de razonamiento y permite la integración de los mecanismos de costumbre. http://jena.sourceforge.net.
Sesame	Java &RESTful web service	RDF	Proporciona una SPARQL interfaz y una interfaz de servidor HTTP. Se suministra con almacenamiento y múltiples mecanismos de razonamiento y también permite la integración de los mecanismos de medida. http://www.openrdf.org.

FRAMEWORK	LENGUAJE INTERFAZ	NIVEL SEMÁNTICA	DESCRIPCIÓN
OWL API	Java	OWL 2	La OWL API se basa en la sintaxis funcional de OWL 2 y contiene un interfaz común para muchos razonadores. http://owlapi.sourceforge.net.
RAP-RDF API	PHP	RDF	RAP es una fuente abierta RDF API y una suite de software para almacenar, consulta y manipulación de RDF en PHP. http://sourceforge.net/projects/dfapi-php.
Redland	C, Python, Ruby, Perl & PHP	RDF	Colección de las bibliotecas RDF para C, con enlaces para los demás idiomas. Proporciona RDF API, analizadores, y las interfaces de consulta. http://librdf.org.
linqToRDF	.NET	RDF	El marco de Semántica Web para él .NET creado en la Microsoft Consulta de Lenguaje-Integrado (LINQ) Marco (Consulta independiente del lenguaje y el sistema de procesamiento de datos). http://code.google.com/p/linqtordf.

Jena puede manejar OWL y OWL2 y permite añadirle un razonador para nuestro caso Pellet y obtener las inferencias; además se podrá introducir a un agente inteligente la base de conocimiento OWL y añadirle las inferencias obtenidas del modelo a este.

6.4.9 PELLET

Razonador basado en tableros ha sido programado en Java, es gratuito y open source. Originalmente desarrollado por el laboratorio Mindswap de Maryland. Se puede utilizar fácilmente desde las librerías de gestión de ontologías (como la API de Jena y OWL) y ofrece una interfaz DIG.

Soporta razonamiento con la expresividad total de OWL-DL (sintaxis SHOIN(D) y SHIN(D)) incluyendo razonamiento acerca de los nominales (clases enumeradas), soporta especificaciones OWL2(SROIQ(D)); también brinda razonamiento con las proporciones inversas de OWL-Full.

Pellet comprueba la consistencia incremental y la clasifica. Están basados en los algoritmos Tableaux desarrollados para lógicas expresivas potentes, la cual valida ontologías.

A. Arquitectura de Pellet

Comprueba la consistencia de una KB (base de conocimiento); es decir, un par de ABox y un TBox. El razonador se acopla con un datatype de oracle para poder chequear la coherencia de las conjunciones de un *XML Schema simple datatype*.

Las ontologías OWL se cargan en el razonador, la cual valida y repara la ontología. Este paso asegura que todos tengan un tipo apropiado de tripletas (un requerimiento de la OWL DL, pero no de la OWL FULL) y si faltan algunas declaraciones de tipo se añaden usando algunas heurísticas. Durante la fase de carga axiomas sobre las clases (subclase, la clase equivalente o axiomas disyunción) se ponen en el componente TBox y afirmaciones acerca de los individuos (tipo y afirmación propiedad) se almacenan en el componente ABox.Axiomas TBox para pasar por el proceso previo del estándar de razonadores DL antes de que se alimenten al razonador de tablas.

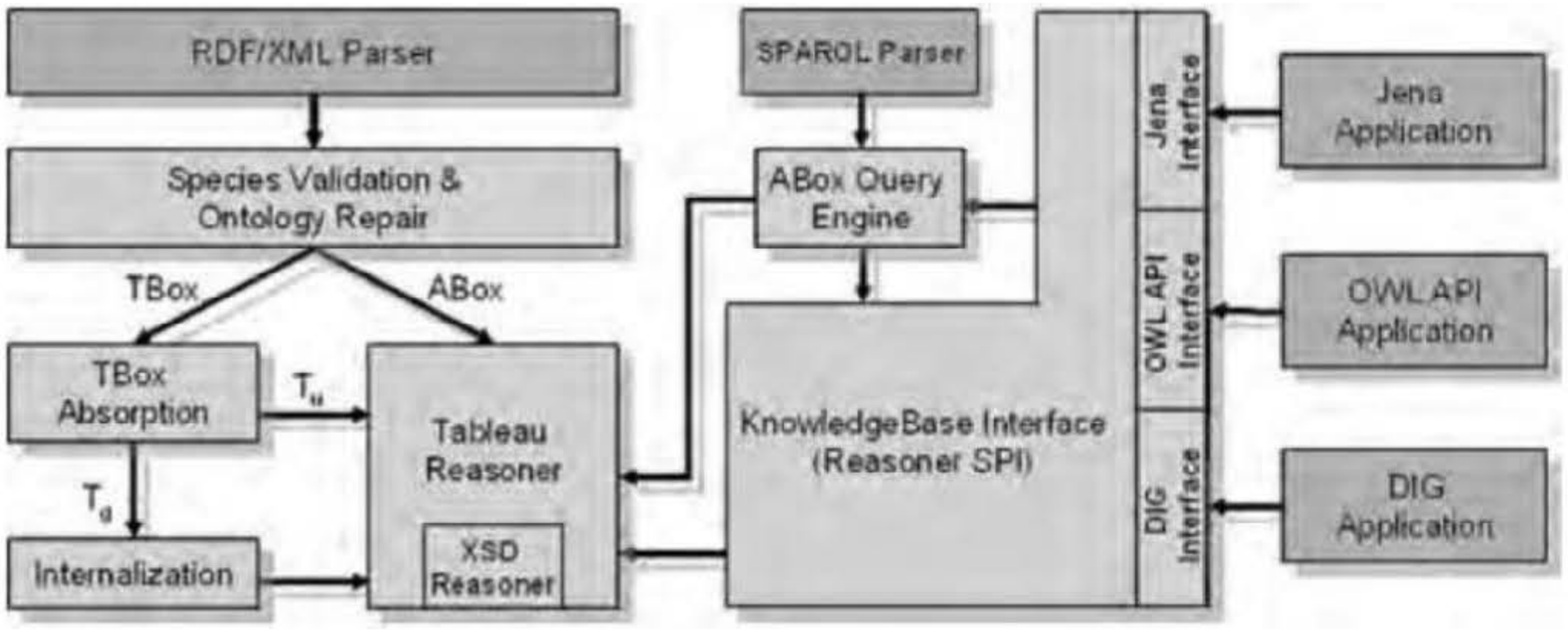


Figura 6.5 Arquitectura Pellet.

La función de un razonador de tablas es en esencia el control de la satisfacibilidad de una ABox xon respecto a un TBox. Las demás tareas de razonamiento se reducen a la consistencia KB, la cual es testeada con una adecuada transformación. El sistema de programación de interface (SPI) de Pellet ofrece funciones genéricas, las cuales administran tales transformaciones. La interfaz de KB se basa como el resto de los componentes internos en la librería Aterm (término de anotaciones a corto plazo). Este consiste en un tipo de datos abstractos creado para el intercambio de datos de estructuras en forma de árbol entre las aplicaciones distribuidas. Esta librería permite maximizar el subtermino sharing (compartir) y la recolección automática de basura por lo que es muy adecuado para la representación de las expresiones de la clase del OWL. Los términos de la función sharing reduce el consumo de memoria total dedicado para el almacenamiento de concepto de expresiones y facilita transformar los datos de Pellet SPI para una API externa.

6.5 JENA

Jena es un framework de Java para la construcción de aplicaciones en la web semántica. Es de código abierto y ha sido desarrollado a partir de trabajar con *HP Labs Semantic Web Programme*.

6.5.1 CARACTERÍSTICAS

Permite manipular ontologías (añadir hechos, borrarlos y editarlos), realizar consultas contra ellas y almacenar. Los recursos no están ligados estáticamente a una clase Java particular.
Los principales problemas que presentan son los siguientes:

- El recurso no cambió, pero la clase Java para modelarlo es otra.
- No se puede modificar la clase dinámicamente.

Jena considera que la abstracción Java del recurso es solo una vista del mismo. Se compone de:

- API de procesamiento RDF.
- API de procesamiento OWL, DAML y RDF Schema.
- Un motor de razonamiento basado en reglas.
- Un motor de consultas SPARQL, RDQL.
- El soporte es proporcionado por jena-users@incubator.apache.org.

6.5.2 Procesamiento de la API RDF y OWL

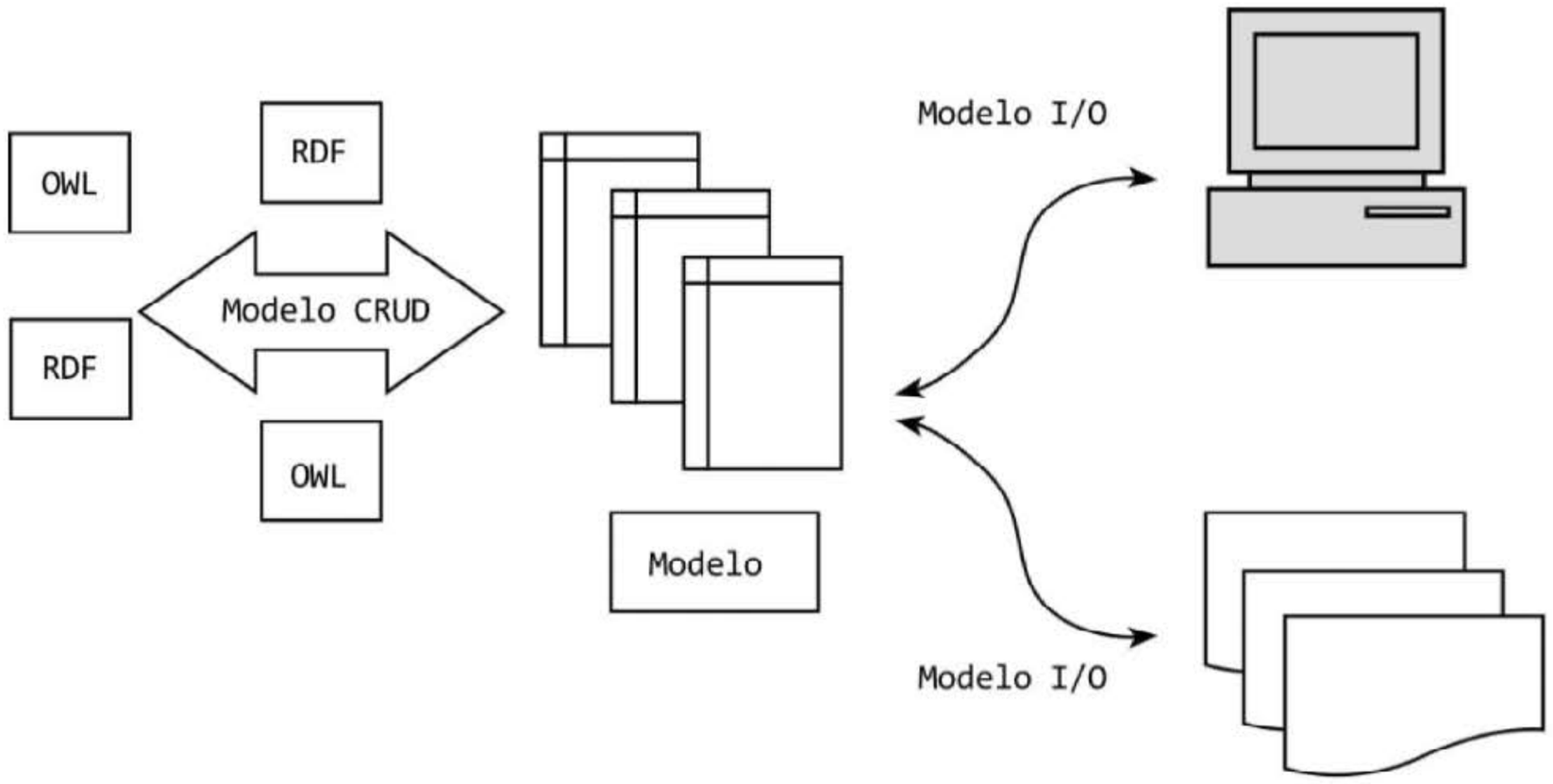


Figura 6.6 Jena API RDF y OWL.

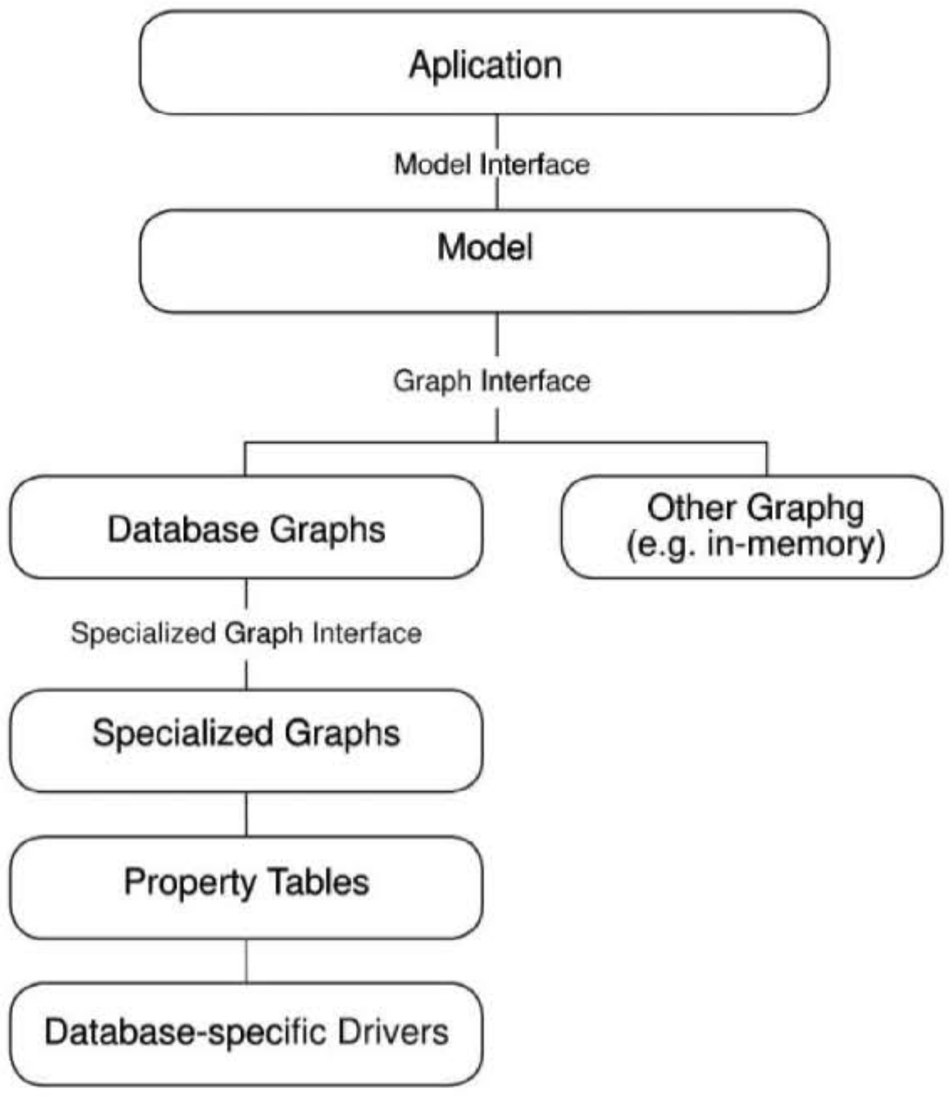


Figura 6.7 Descripción de la arquitectura Jena.

El gráfico detalla cómo es que Jena procesa una base de conocimiento:

Paso 1: Carga o crea una ontología.

Paso 2: Almacena la información en un modelo que se almacena en memoria (esta podrá ser manipulada).

Paso 3: Una vez realizada todas las acciones, alternativamente guarda un archivo físico.

6.5.3 Mecanismo de inferencia en Jena

La inferencia se da mediante un razonador que puede provenir del mismo Jena u otro como Pellet. Estos se deben ejecutar al momento de realizar algún cambio, inserción o consulta a la base de conocimiento.

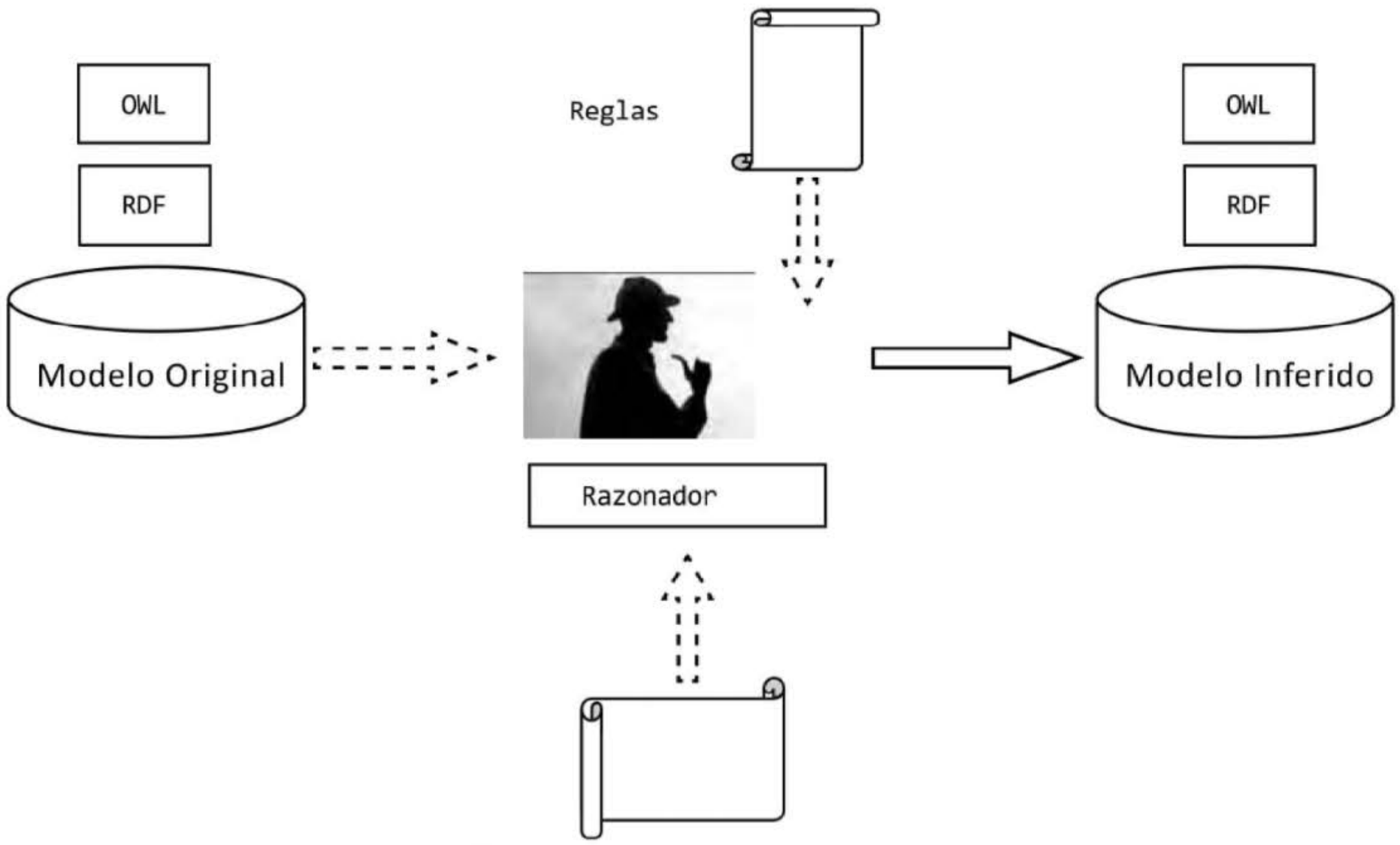


Figura 6.8 Jena y razonador

6.5.4 Consultas SPARQL (estándar de la W3C) en Jena

SPARQL se basa en anteriores lenguajes de consulta RDF como rdfDB, RDQL y SeRQL. Se trata de una herramienta que permite hacer consultas a una base de conocimiento como la OWL, el cual ha sido elaborado por la comunidad de la W3C. Técnicamente la consulta SPARQL se basa en un conjunto de tripletas (sujeto, predicado y objeto) o (ID, propiedad y valor). Al respecto, Tim Berners Lee señala lo siguiente:

> *"Pretender usar la web semántica sin SPARQL es como pretender usar una base de datos relacional sin SQL, SPARQL hace posible consultar información desde bases de datos y otros orígenes de datos en sus estados primitivos a través de la Web".*

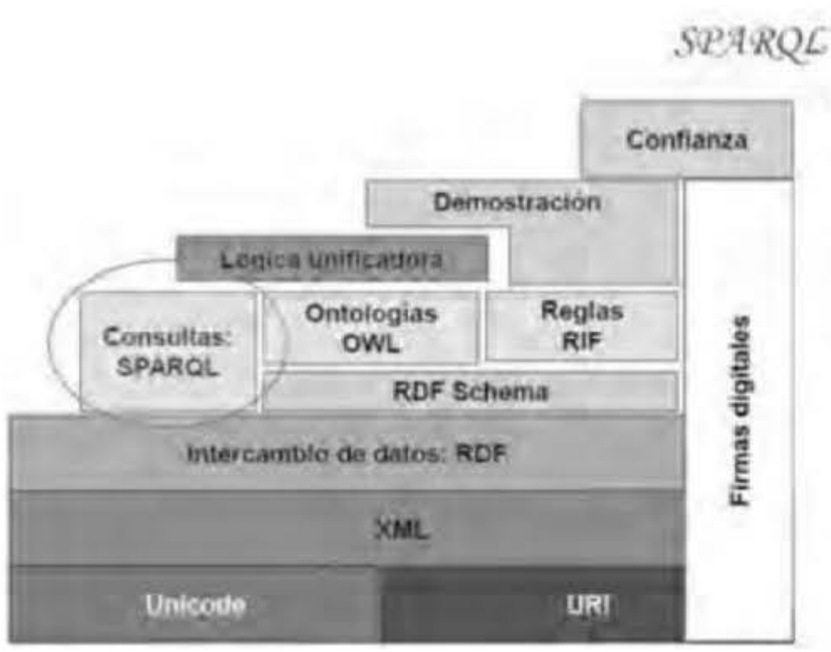


Figura 6.9 Esquema de SPARQL.

A. Sintaxis

- URIs entre < >

  <http://www.misitio.com>

- Prefijos para espacios de nombres (o Name Spaces)

  PREFIX x: <http://www.hca.pe/>
  x:doctor

- Nodos anónimos

  _:nombre ó [ ]

- Literales entre " "

  "Jesus fernandez"
  "234"^^xsd:integer

- Variables empiezan con "?" o "$"

  ?nombre
  $apellido

- Comentarios empiezan con #

  # esto es un comentario

B. Como lenguaje de consulta posee las siguientes cláusulas

- PREFIX: Inserta una etiqueta a un URI de la ontología.
- WHERE: Es un proceso secuencial del cual se obtienen los valores que contendrán esas variables.
- SELECT: Aquí se declaran las variables a mostrar las cuales siguen el patrón "sujeto, predicado y valor". En el siguiente ejemplo, se muestra cómo se listan todos los sujetos con sus propiedades y valores correspondientes.

```
select * where {
?s ?p ?o. }
```

- FILTER: Añade restricciones a los valores de las variables encontradas, utiliza funciones y operadores de Xpath 2.0.
- REGEX: Se utiliza para restringir valores de tipo texto o string.
  - regex(?Expresión, ?Patrón [, ?Flags])
  - ?Expresión: variable a tratar.
  - ?Patrón: patrón de la restricción.
  - ?Flags: opciones para el encaje.

```
PREFIX e: < http://www.misitio.com#>
SELECT ?n ?e WHERE {
    ?x e:nombre ?n .
    ?x e:edad ?e
    FILTER regex(?n,"A","i")
}
```

- CONSTRUCT: Permite construir un grafo RDF a partir de una consulta.
- DESCRIBE: Devuelve una descripción del grafo en RDF.
- ASK: Devuelve falso o verdad a una determinada búsqueda.

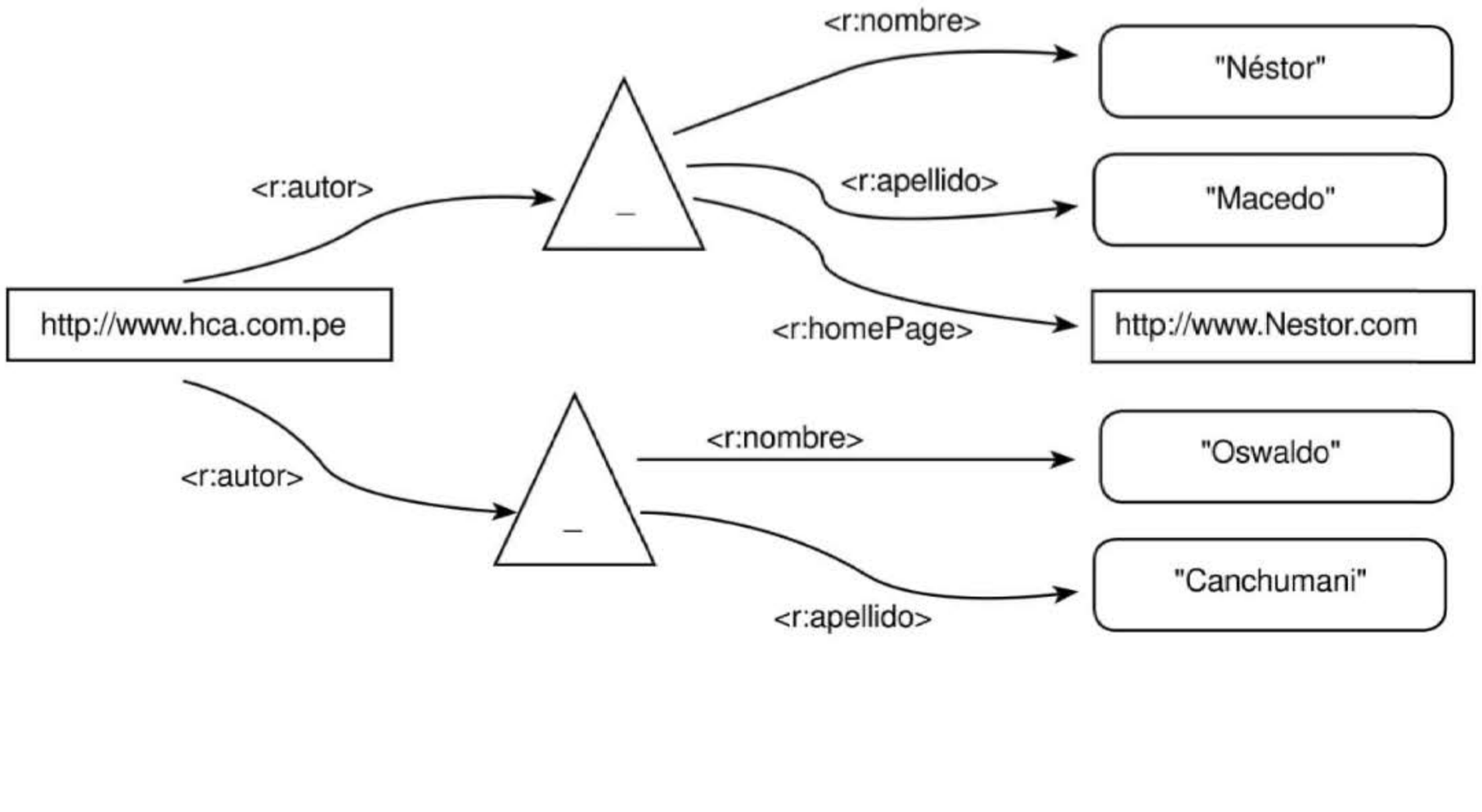
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX r: < http://www.misitio.com#>
CONSTRUCT { ?x r:nombre ?n }
WHERE { ?x foaf:name ?n }
```

- OPTIONAL: Muestra los valores de las tripletas sin fallar aun cuando estas no existan.
- UNION: Junta dos o más grafos.

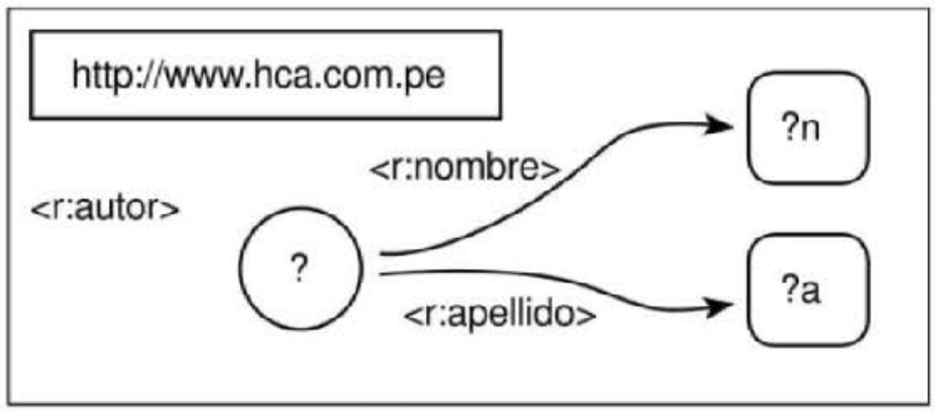
```
prefix e: < http://www.misitio.com #>.
prefix foaf: <http://xmlns.com/foaf/0.1/>.
SELECT ?n
WHERE {
{ ?x foaf:name ?n }
    UNION{ ?y p:nombre ?n }
}
```

- DISTINCT: Muestra solo los valores diferentes.
- ODER BY: Ordena la variable de forma descendente o ascendente.
- LIMIT: Limita la cantidad de valores a mostrar.

C. Se basa en encaje de patrones en el grafo



```
PREFIX r: <http://www.ministeriosalud.gob.pe>
SELECT ?n ?a
WHERE {http://www.hca.com.pe r:autor ?x.
?x r:nombre ?n.
?x r:apellidos ?a.}
```



D. *Los resultados se obtienen de la cláusula WHERE de la siguiente manera*

["?d" variable que toma los valores del dominio los cuales pueden ser cualquiera de la tripleta, está determinada por la siguiente instrucción. No siempre es una variable] ["prefijo"]:["propiedad/clase/instancia"] ["?r" variable que toma los valores del rango y al igual que el dominio puede ser cualquiera de la tripleta, pero está determinada por la instrucción anterior. No siempre es una variable] ["." El punto nos indica que sigue una instrucción en la línea siguiente y también sirve para terminar la cláusula WHERE].

- **RESULTADOS:**

?n	?a
"Néstor"	"Macedo"
"Oswaldo"	"Canchumani"

- **ALGUNOS EJEMPLOS**

- Lista todos los atributos DataType Property

```
prefix owl: <http://www.w3.org/2002/07/owl#>
select $dp
where {
    $dp a owl:DatatypeProperty
}
```

- Listar todas las clases RDF

```
select ?class
where {
    [] a ?class
}
```

- Listar todas las propiedades RDF

```
select ?prop
where {
    [] ?prop []
}
```

- Listar todas las clases OWL

```
prefix owl: <http://www.w3.org/2002/07/owl#>
select ?class
where {
 ?class a owl:Class
}
```

- Listar todas las clases con sus individuos (sujetos), propiedades y valores

```
SELECT DISTINCT ?clases ?id ?propiedad ?valor
where {?clases a owl:Class.
?id rdf:type ?clases.
?id ?propiedad ?valor.
?d a owl:ObjectProperty.
?s a owl:DatatypeProperty;
FILTER(?propiedad = ?d || ?propiedad = ?s)
}
```

EJECUTANDO LA APLICACIÓN

El siguiente ejemplo es una ejecución de una consulta en el framework JENA:

```
import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.query.* ;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.rdf.model.Model;

public class Main {
  private static OntModel model; //consultar capitulo 7.10
  private static String NS;
  public static void main(String[] args) {
```

```
String queryString =""
+ "PREFIX owl: <http://www.w3.org/2002/07/owl#>"
+ "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
+ "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"

+ "PREFIX base:<http://www.owl-ontologies.com/Ontology1282851039.owl#>"

+ "SELECT ?s"
+ " WHERE{?s rdf:type base:Profesor}";
```

```
        model = ModelFactory.createOntologyModel( OntModelSpec.OWL_DL_MEM );

        //Establecemos el NameSpace por defecto para nuestra ontología
        NS = "http://www.owl-ontologies.com/Ontology1282851039.owl#";

        //Si se desea se crea un prefijo en el modelo ontológico de la siguiente manera.
        //model.setNsPrefix(NS, "http://www.owl-ontologies.com/Ontology1282851039.owl#");

        //abrir el archivo con la ontología
        java.io.InputStream in = FileManager.get().open( "datos/profesor.owl" );
        if (in == null) {
            throw new IllegalArgumentException("Archivo no encontrado");
        }
        // Lee el archivo OWL y lo carga al modelo
        model.read(in, "");

        Query query = QueryFactory.create(queryString);
        // Ejecutar la consulta y obtener los resultados
        QueryExecution qe = QueryExecutionFactory.create(query, model);

        try {
            ResultSet results = qe.execSelect();
            ResultSetFormatter.out(System.out, results, query) ;
        } finally { qe.close() ; }
    }
}
```

El ejemplo anterior muestra todos los individuos de la clase profesor.

```
| s       |
====================
| base:Profesor_3 |
| base:Profesor_1 |
| base:Profesor_2 |
```

Nota:

Es conveniente crear los prefijos básicos para realizar cualquier consulta tales como:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
```

Luego cree el prefijo "base", por ejemplo, que constituye el URL de su página u ontologiá.

E. Grafo de RDF

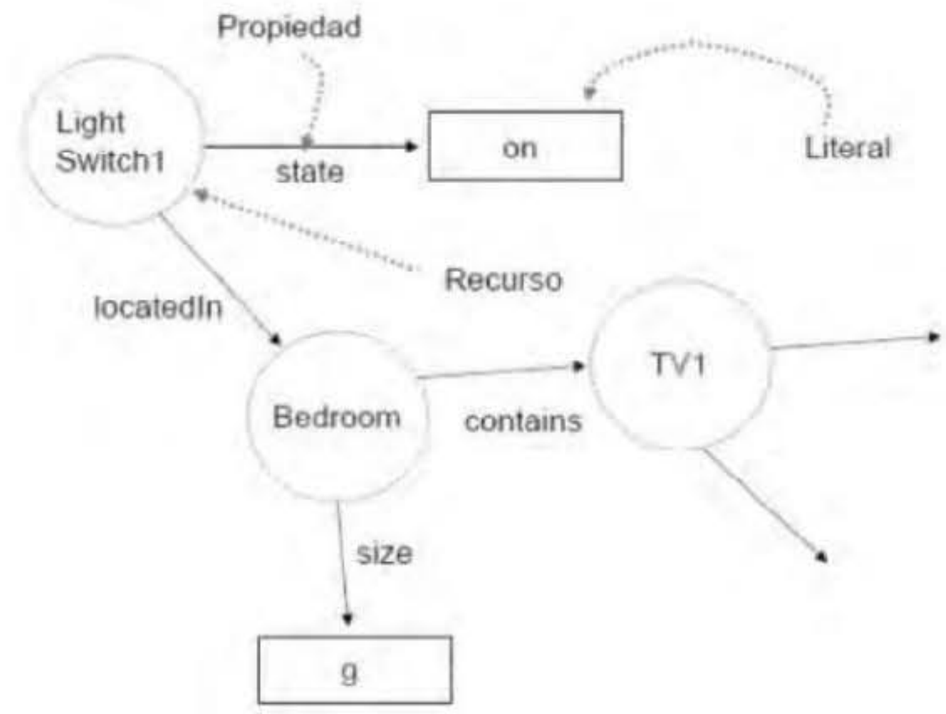


Figura 6.10 Jena API RDF.

Se basa en la tripleta ya mencionada (recurso, propiedad, valor/literal).

F. Abstracciones claves

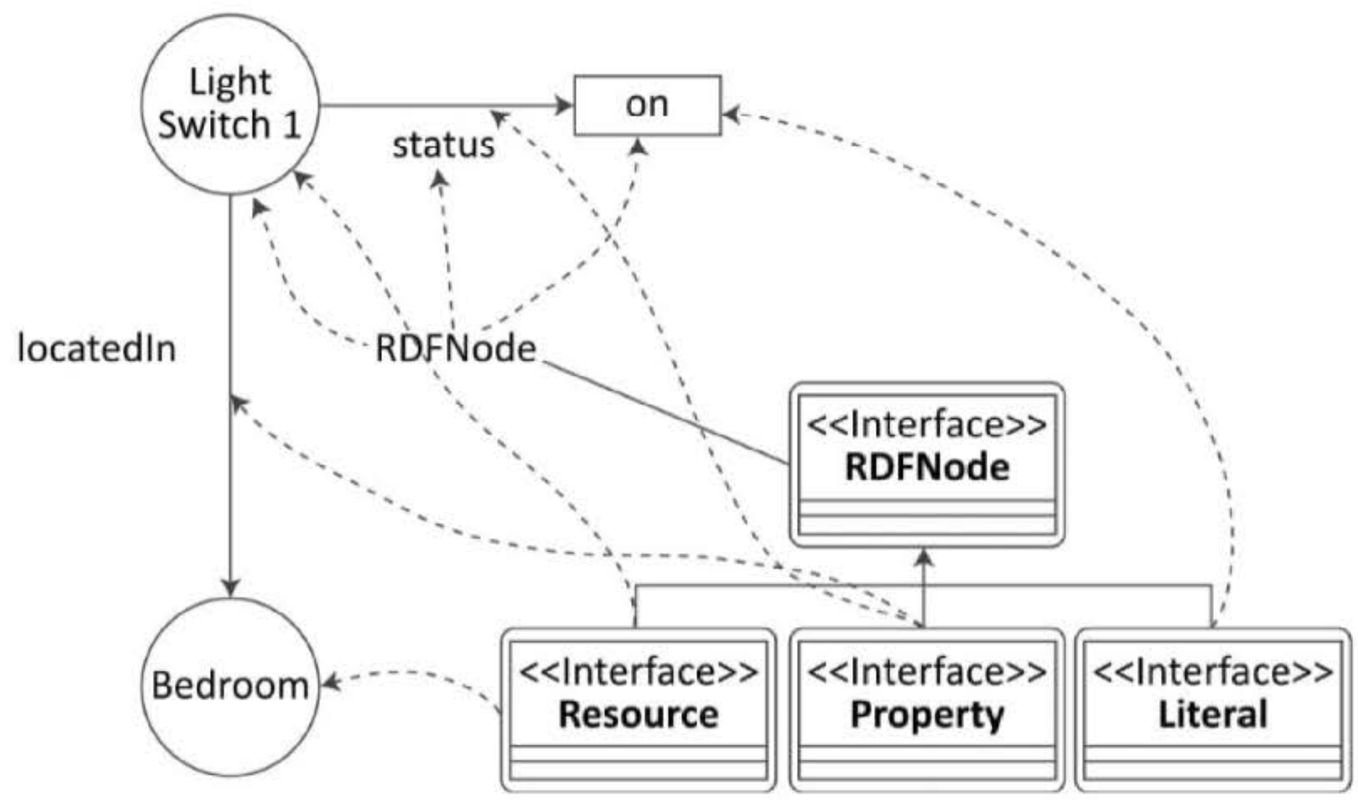


Figura 6.11 Jena API RDF, abstracciones claves.

G. Escenario típico de uso

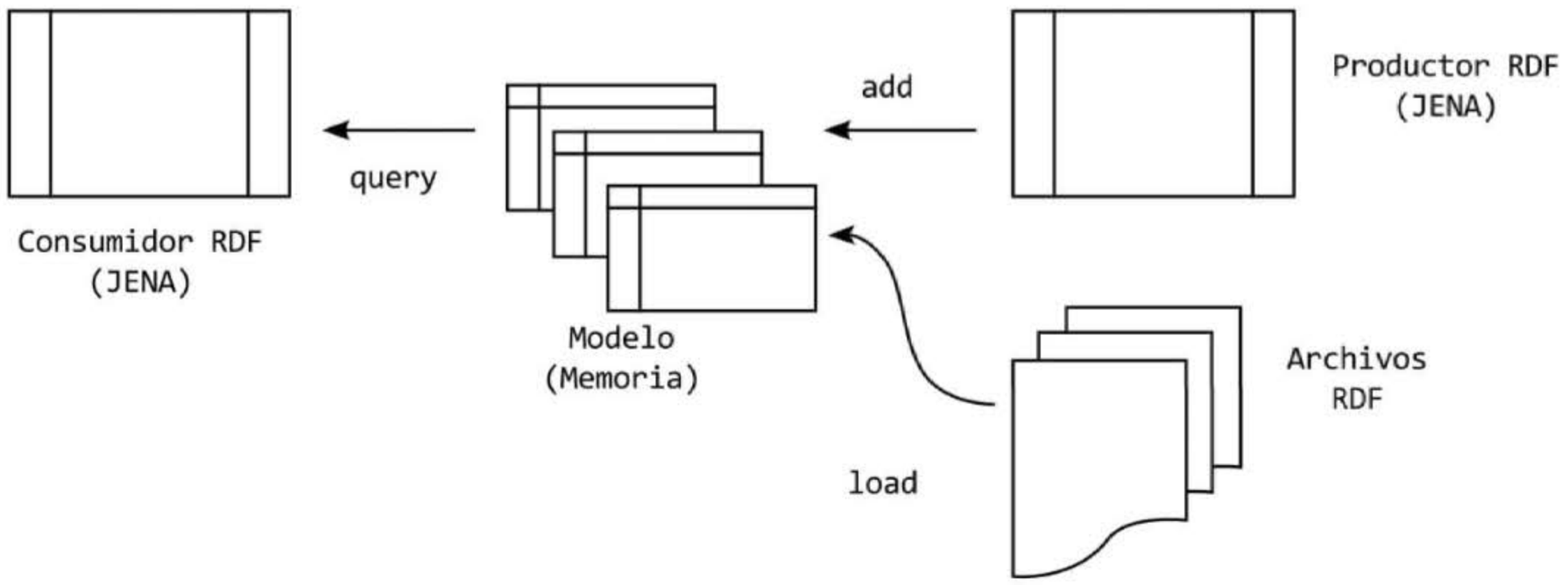


Figura 6.12 Jena, escenario típico de uso.

Se trata de un modelo en memoria; donde el usuario realiza consultas y modificaciones, en un entorno de escritorio sin uso de la Web.

H. Escenario avanzado de uso

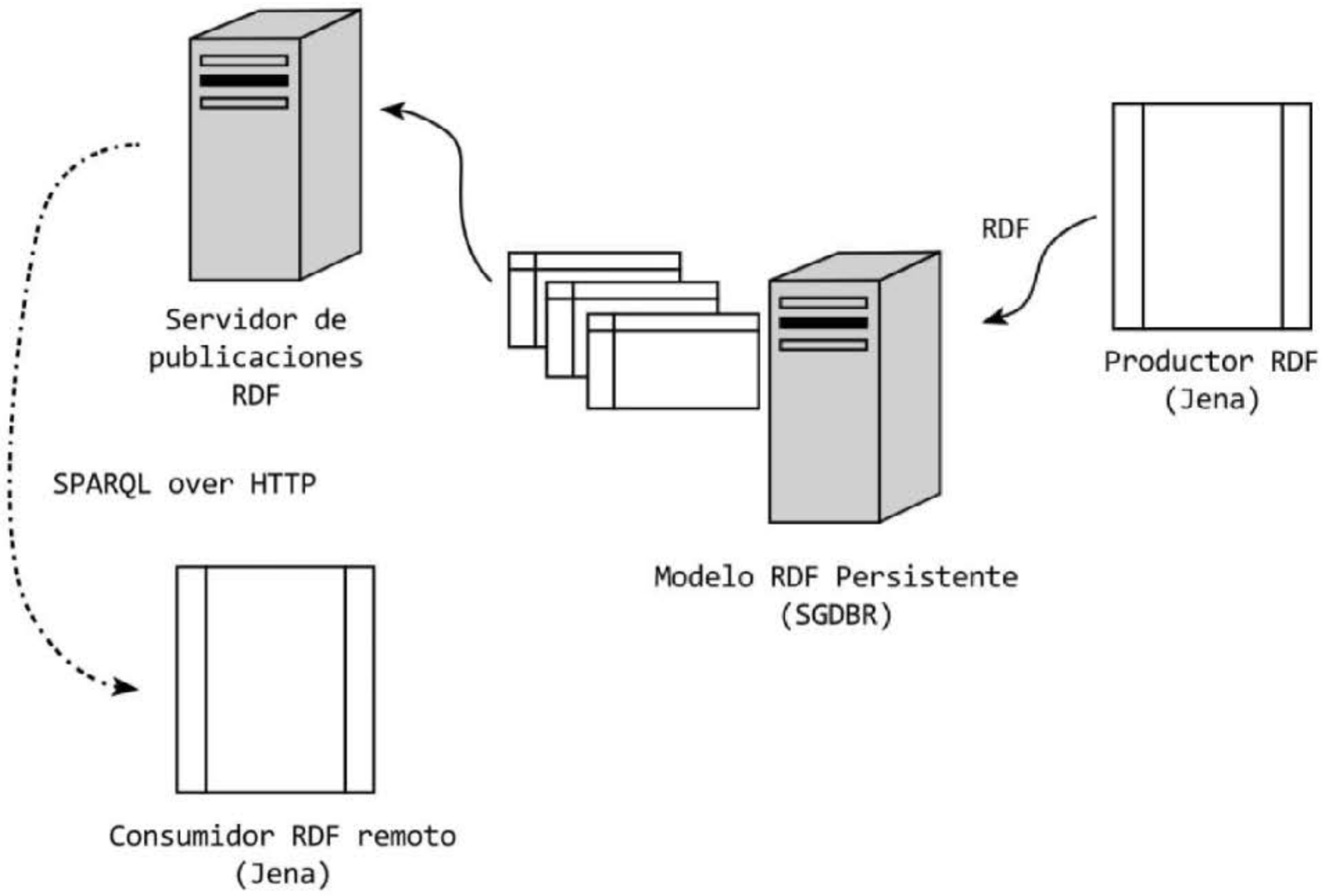


Figura 6.13 Jena, escenario avanzado de uso.

En este modelo, se genera una persistencia del modelo ontológico, el cual es publicado en un servidor que se accede de forma remota.

I. Creación de RDF

Existen 2 formas:

- Crearlo a través del API de Jena.
- Al cargar un modelo existente.
  - Desde un archivo.
  - Desde un modelo persistente.

API Jena RDF creará un Statement de RDF.

```
Model model = ModelFactory.createDefaultModel();
Resource id = model.createResource("http://www.pacis.org#Nestor");
Property edad = model.createProperty(Constantes.NATURALLY_OCCURRING_NS + "edad");
Literal id_edad= model.createLiteral("42");
yangtze.addProperty(edad, id_edad);
model.write(System.out,"N-TRIPLE");
```

Tendría la forma (Néstor, edad, 42) donde:

- Néstor sería el individuo, instancia, ID.
- Edad sería una propiedad, predicado.
- 42 sería valor, objeto.

Cargar un archivo RDF externo

```
model.read("file:ubicación/archivo.rdf","RDF/XML");
model.write(new FileWriter("ubicación/archivo.rdf"),"RDF/XML");
```

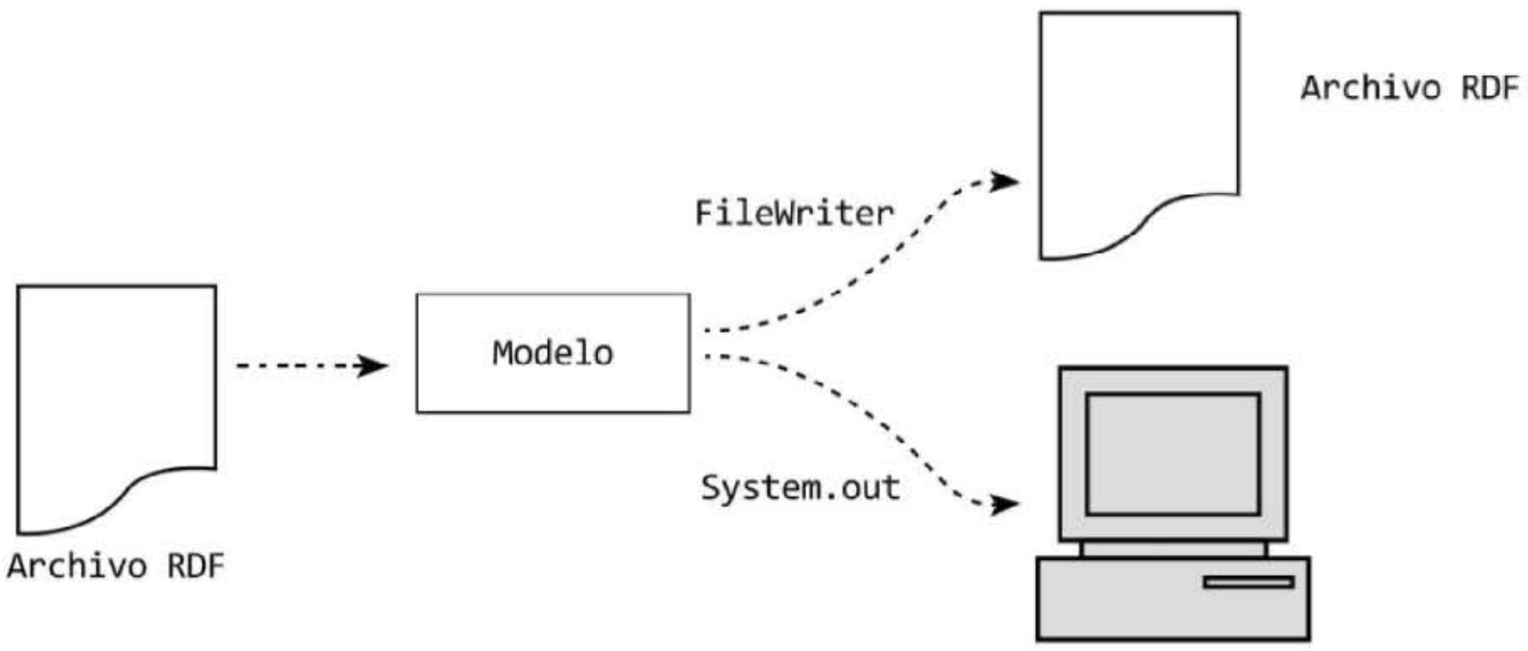


Figura 6.14 Jena, cargar y escribir modelo ontológico.

- Tipos de modelo
  - N-TRIPLE
  - N3
  - RDF/XML
  - RDF/XML-ABBREV
- Modelo CRUD
  - Create, crear.
  - Retrieve, recuperar.
  - Update, actualizar.
  - Delete, eliminar o borrar.

```
for(Iterator it = model.listStatements();it.hasNext();){
    Statement stmt = (Statement) it.next();
    System.out.println(stmt.getSubject().getLocalName());
}
```

◉ CONSULTA DEL MODELO: SELECTORES

```
" Select * from MODEL where SUBJECT='Nestor' "

Resource id = model.getResource("http://www.pacis.org#Nestor ");
Selector selector =new SimpleSelector(id, (Property)null, (Resource)null);
for(Iterator it = model.listStatements(selector);it.hasNext();){ …}
// o, equivalentemente…
for(Iterator it = model.listStatements(id, (Property)null,
(Resource)null) ;it.hasNext();){ …}
```

◉ ACTUALIZANDO EL MODELO

El método changeobject de la clase statement es usado para este propósito.

```
Resource id = model.getResource("http://www.pacis.org#Nestor ");
Resource chinaSea =model.getResource("http://www.china.gov#ChinaSea");
Property emptiesInto =model.getProperty(Constantes.NATURALLY_OCCURRING_NS + "emp-
tiesInto");
id.getProperty(emptiesInto).changeObject(chinaSea);
```

◉ BORRAR UN STATEMENT

El método remove() de la clase Statement es usado para este propósito.

```
Resource yangtze =model.getResource("http://www.china.org/rivers#Yangtze");
Property emptiesInto =model.getProperty(Constantes.NATURALLY_OCCURRING_NS +
"emptiesInto");
yangtze.getProperty(empriesInto).remove();
```

K. Inferencia básica a un RDF

```
Property teachesAtProp = model.createProperty(ns, "teachesAt");
Property worksAtProp = model.createProperty(ns, "worksAt");
model.add(teachesAtProp, RDFS.subPropertyOf, worksAtProp);

model.createResource(ns + "Carmen").addProperty(teachesAtProp, "ETSIT");

(Carmen, teachesAt, ETSIT)

Reasoner reasoner = ReasonerRegistry.getRDFSReasoner();
InfModel infModel = ModelFactory.createInfModel(reasoner, data);

(Carmen, worksAt, ETSIT)
```

El *namespace* es la dirección fuente o base del RDF (http://www.pacis.org#). La variable "ns" es un string que contiene al namespace.

En el ejemplo anterior se agrega al individuo "Carmen", el cual posee como propiedad teachesAt y esta tiene como subpropiedad "worksAt", por lo que al ejecutar el razonador se muestra que el individuo Carmen posee como propiedad worksAt.

L. Jena API Ontología

Cree un modelo ontológico con el API de Jena. Los modelos son conjuntos statements.

```
OntModel ontology =ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM);
ontology.createClass(ns + " Stream ");
ontology.createClass(ns + "Brook");
ontology.createClass(ns + "River");
ontology.createClass(ns + "Tributary");

FunctionalProperty emptiesInto = m.createObjectProperty(ns + "emptiesInto",true).con-
vertToFunctionalProperty();

ObjectProperty feedsFrom = m.createObjectProperty(ns + "feedsFrom");
contains.setInverseOf(locatedIn);

ontology.add(feedsFrom, RDF.type, OWL.InverseFunctionalProperty);

//Almacenamos la ontología en un archivo OWL (opcional)
File file = new File("/Miproyecto/Mi_archivo.owl");
//Hay que capturar las Excepciones
if (!file.exists()){
   file.createNewFile();
}
model.write(new PrintWriter(file));
```

- **MODELFACTORY**

  Permite crear el modelo a través de su método createOntologyModel(), el cual crea un modelo por defecto que es compatible con versiones anteriores a Jena. Estos valores predeterminados son los siguientes:

- OWl-Full lenguaje.
- In-memory de almacenamiento.
- RDFS para inferencia, que produce principalmente vínculos para las subclases y subproperty jerárquicas.

```
OntModel model = ModelFactory.createOntologyModel();
```

Lenguaje Ontológico	URI
RDFS	http://www.w3.org/2000/01/rdf-schema#
DAML+OIL	http://www.daml.org/2001/03/daml+oil#
OWL Full	http://www.w3.org/2002/07/owl#
OWL DL	http://www.w3.org/TR/owl-features/#term_OWLDL
OWL Lite	http://www.w3.org/TR/owl-features/#term_OWLLite

- **OntModelSpec**

Define el tipo de modelo ontológico.

```
OntModel model = ModelFactory.createOntologyModel(OntModelSpec.OWL_ MEM);
```

OntModelSpec	Language profile	Almacenamiento model	Razonador
OWL_MEM	OWL full	in-memory	Ninguno
OWL_MEM_TRANS_INF	OWL full	in-memory	transitive class-hierarchy inference
OWL_MEM_RULE_INF	OWL full	in-memory	rule-based reasoner with OWL rules
OWL_MEM_MICRO_RULE_INF	OWL full	in-memory	optimised rule-based reasoner with OWL rules
OWL_MEM_MINI_RULE_INF	OWL full	in-memory	rule-based reasoner with subset of OWL rules
OWL_DL_MEM	OWL DL	in-memory	Ninguno
OWL_DL_MEM_RDFS_INF	OWL DL	in-memory	rule reasoner with RDFS-level entailment-rules
OWL_DL_MEM_TRANS_INF	OWL DL	in-memory	transitive class-hierarchy inference
OWL_DL_MEM_RULE_INF	OWL DL	in-memory	rule-based reasoner with OWL rules
OWL_LITE_MEM	OWL Lite	in-memory	Ninguno
OWL_LITE_MEM_TRANS_INF	OWL Lite	in-memory	transitive class-hierarchy inference
OWL_LITE_MEM_RDFS_INF	OWL Lite	in-memory	rule reasoner with RDFS-level entailment-rules
OWL_LITE_MEM_RULES_INF	OWL Lite	in-memory	rule-based reasoner with OWL rules
DAML_MEM	DAML+OIL	in-memory	Ninguno
DAML_MEM_TRANS_INF	DAML+OIL	in-memory	transitive class-hierarchy inference
DAML_MEM_RDFS_INF	DAML+OIL	in-memory	rule reasoner with RDFS-level entailment-rules
DAML_MEM_RULE_INF	DAML+OIL	in-memory	rule-based reasoner with DAML rules
RDFS_MEM	RDFS	in-memory	Ninguno
RDFS_MEM_TRANS_INF	RDFS	in-memory	transitive class-hierarchy inference
RDFS_MEM_RDFS_INF	RDFS	in-memory	rule reasoner with RDFS-level entailment-rules

M. Personalizar un modelo específico

Se crea un *OntModelSpec* con un tipo definido, el cual se modificará con el documento ontológico y luego se creará el modelo con el tipo de ontología, tal como se muestra:

```
OntModelSpec s = new OntModelSpec( OntModelSpec.OWL_DL_MEM_RULE_INF);
    s.setDocumentManager( myDocMgr );
    OntModel m = ModelFactory.createOntologyModel( s );
```

6.5.5 Procesamiento de importaciones ontológicas

Una ontología puede importar a otra para la integración de los modelos. La figura muestra cómo un modelo de ontología construye una colección de modelos de importación.

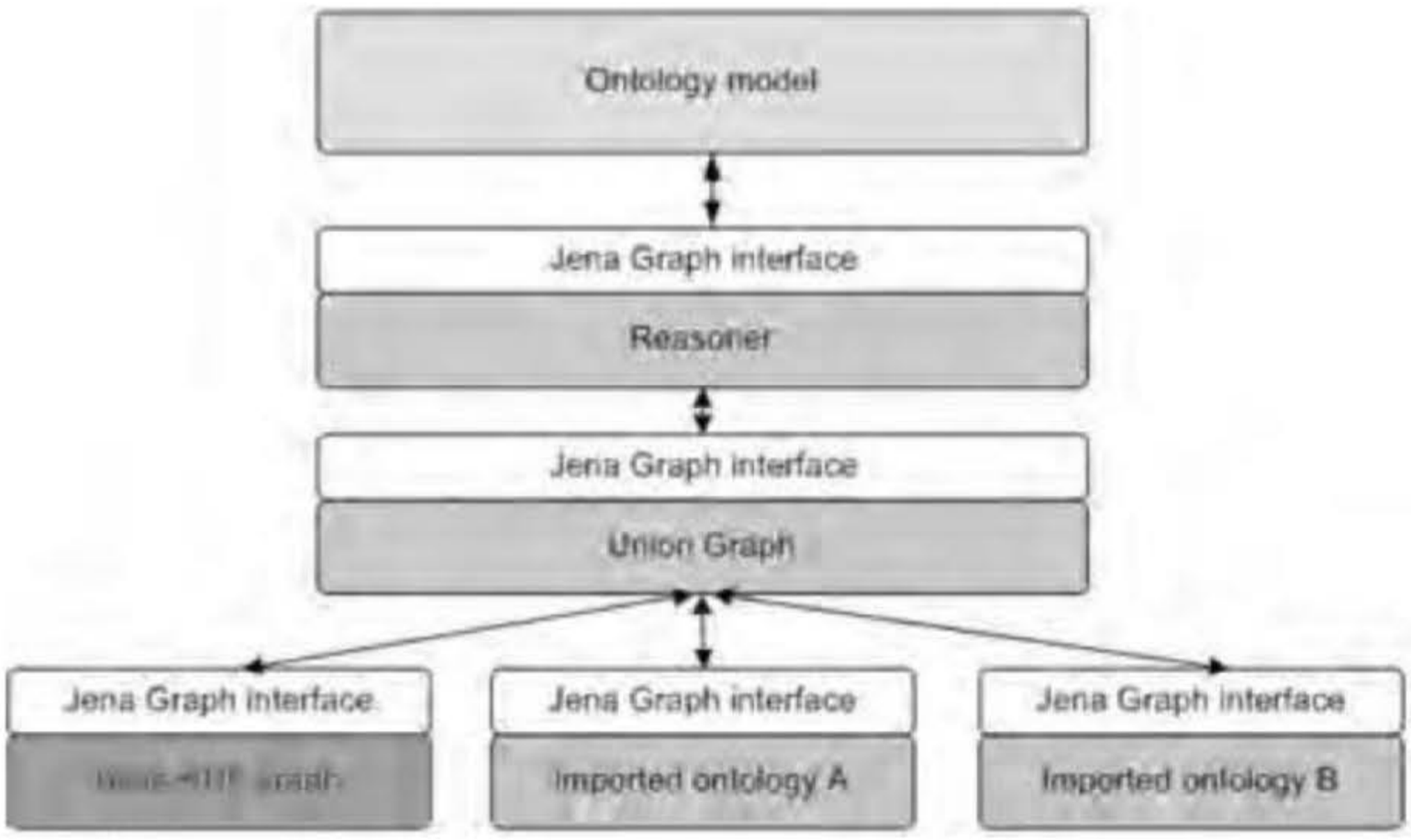


Figura 6.15 Modelo ontológico compuesto por una estructura del documento para la importación.

Se va a cargar un archivo o documento de ontología de la misma manera que un modelo de Jena, con el método read de la clase OntModel. Existen varias variantes de lectura.

```
OntModel m = ModelFactory.createOntologyModel(OntModelSpec. OWL_DL_MEM_RULE_INF);
//tipos de lectura
m.read( String url )
m.read( Reader reader, String base )
m.read( InputStream reader, String base )
m.read( String url, String lang )
m.read( Reader reader, String base, String Lang )
m.read( InputStream reader, String base, String Lang )
```

Al usar el método read() se debe tener cuidado, ya que pueden aceptar variantes del java.io.Reader de argumento al cargar documentos XML que contienen conjuntos de caracteres internacionalizados, puesto que el manejo de codificación de caracteres por el lector y por los analizadores XML no es compatible.

Al cargarse la ontología, esta también carga a los individuos o instancias que se encuentran en ella. Cada documento ontológico importado se mantiene en una estructura de grafo separada al de la base. Esto es importante, ya que solo es el modelo base del cual se actualizará sin afectar a las importaciones.

Las **importaciones** son un proceso recursivo; por lo que si un documento base importa a la ontología A y este, a su vez, importa a B, quedaría la forma que se muestra en la figura anterior. Tenga presente que las importaciones se han aplanado. Un **ciclo de verificación** se utiliza para prevenir que el alimentador de documentación se atasque; así por ejemplo, si A importa a B, ¿qué importa A?

A. Administrando documentos ontológicos

Cada modelo de la ontología tiene asociado un *gestor de documentos* que ayuda a la transformación y se ocupa de la manipulación de los documentos relacionados. Para mayor comodidad, hay un gestor de documentos global que se utiliza por defecto por los modelos de la ontología. Se puede obtener una referencia a esta instancia para compartir, a través de *OntDocumentManager.getInstance ()*. En muchos casos, será suficiente con solo cambiar la configuración en el gestor del documento global para satisfacer las necesidades de su aplicación. Sin embargo, para tener un control más fino, puede crear separadamente un documento administrador y pasarlo al modelo ontológico cuando se crea a través del modelo *factory*. Para ello, cree un objeto especificación de la ontología y establezca el gestor de documentos. Por ejemplo:

```
OntDocumentManager mgr = new OntDocumentManager();
 … Codificación …
// ahora utilizamos lo siguiente
OntModelSpec s = new OntModelSpec( OntModelSpec.RDFS_MEM );
s.setDocumentManager( mgr );
OntModel m = ModelFactory.createOntologyModel( s );
```

Tenga en cuenta que el modelo conserva una referencia al gestor de documentos que se creó. Así, si usted cambia las propiedades del administrador de documentos, serán afectados los documentos que previamente han sido construidos por el administrador de documentos.

B. Política sobre la administración de documentos

El administrador de documentos tiene un gran número de opciones de configuración y hay dos formas en que se puede cambiar los requisitos de la aplicación:

- Cuando se configuran los distintos parámetros del administrador de documentos por el código de Java.
- Cuando un administrador de documento dado puede cargar los valores de los diferentes parámetros a partir de una política del archivo, expresada en RDF.

El administrador de documentos tiene una lista de URL que se va a buscar desde una política de documento. Se detendrá en la primera entrada en la lista que se resuelve en el documento recuperable. La ruta de búsqueda por defecto de la política es así:

```
file::. /etc/ ont-policy.rdf; file:ont-policy.rdf . Usted puede encontrar la política por
defecto, que puede servir como plantilla para definir sus propias políticas, en el etc/ den-
tro del directorio de descarga de Jena.
Puede establecer las propiedades generales del administrador de documentos en la política
de la siguiente manera:
```

```
<DocumentManagerPolicy>
    <!-- policy for controlling the document manager's behaviour -->
    <processImports rdf:datatype="&xsd;boolean">true</processImports>
    <cacheModels rdf:datatype="&xsd;boolean">true</cacheModels>
</DocumentManagerPolicy>
```

Se puede encontrar el esquema simple que declara las distintas propiedades que se utilizan en este tipo de documento de política de ontología (en el directorio de los vocabularios de la descarga de Jena) llamada ont-manager.rdf. Para cambiar la ruta de búsqueda que el administrador de documentos utiliza para inicializar, se puede pasar nuevamente la ruta de búsqueda como una cadena al momento de crearse un objeto de documento nuevo gerente o llamando al método setMetadataSearchPath().

6.5.6 Insertar un razonador al modelo

```
Model data = ModelFactory.createDefaultModel();
data.read("file:./etc/inf/naturally-occurring.nt", "N-TRIPLE");

OntModel ontology=ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM);
ontology.read("file:./etc/inf/naturally-occurring.owl");

Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
InfModel infModel= ModelFactory.createInfModel(reasoner, ontology, data);

// O análogamente
Reasoner owlReasoner = ReasonerRegistry.getOWLReasoner();
Reasoner wnReasoner = owlReasoner.bindSchema(ontology);
InfModel infModel= ModelFactory.createInfModel(wnReasoner, data);
```

- **Combinación del razonador genérico y OWL**

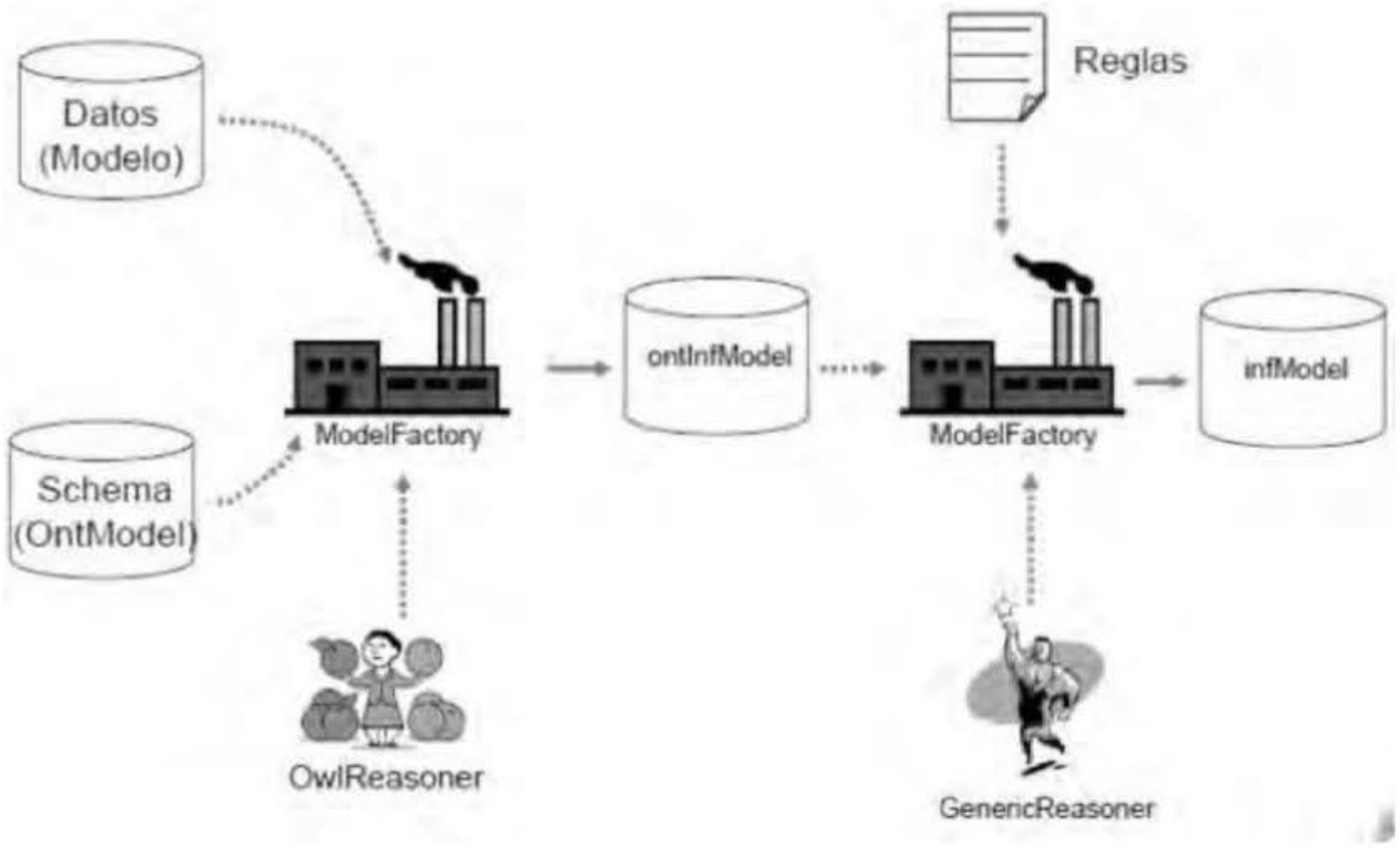


Figura 6.16 El razonador genérico y el OWL.

```
OntModel schema =
ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM_TRANS_INF);
schema.read("file:./etc/inf/schema-inf-owl.owl");
Model data = FileManager.get().loadModel("file:./etc/inf/data.rdf");
Reasoner owlReasoner = ReasonerRegistry.getOWLReasoner();
InfModel owlInfModel = ModelFactory.createInfModel(owlReasoner, schema,data);
GenericRuleReasoner reasoner =new GenericRuleReasoner(Rule.rulesFromURL("file:./etc/inf/
myrule.rule"));
reasoner.setDerivationLogging(true);
InfModel infModel = ModelFactory.createInfModel(reasoner, owlInfModel);
```

◉ Ejemplo del razonador Pellet

```
// creamos el razonador de tipo PELLET
Reasoner reasoner = PelletReasonerFactory.theInstance().create();
//Crear un modelo vacio.
Model emptyModel = ModelFactory.createDefaultModel( );

//Creamos un modelo inferenciado usando el razonador PELLET
InfModel model = ModelFactory.createInfModel(reasoner, emptyModel);

// Leer el archivo
 model.read(inFoafInstance, "");

// Imprime un reporte de validación del modelo
ValidityReport report = model.validate();
printIterator(report.getReports(), "Validation Results");

//ejecuta las reglas a todos los individuos
model.prepare();
```

```
public static void printIterator(Iterator<?> i, String header) {
    System.out.println(header);
    for(int c = 0; c < header.length(); c++)
        System.out.print("=");
    System.out.println();

    if(i.hasNext()) {
        while (i.hasNext())
            System.out.println( i.next() );
    }
    else
        System.out.println("<EMPTY>");

    System.out.println();
}
```

El método prepare de la clase InfModel() es el que ejecuta las reglas contenidas en el modelo ontológico. Esto da como resultado una inferencia, la cual es colocada en el modelo ya sea como instancia, propiedad o valor.

Ejemplo:

Si el modelo se tratara de pacientes y enfermedades, una de las reglas podría ser la de determinar qué enfermedad posee un determinado paciente, la cual sería agregada al momento de ejecutar este método en el modelo.

6.5.7 Resumen de razonadores

- *Razonador transitivo:* Básico, implementa propiedades transitivas y simétricas de *rdfs:subPropertyOf* y *rdfs:subClassOf.* consultas de jerarquías de clase y sus propiedades; además, da soporte de almacenamiento.
- *Razonador RDFS:* Implementa un subconjunto configurable de la funcionalidad ofrecida por RDFS.
- Razonadores OWL, OWL Mini, OWL Micro: Una implementación útil (aunque incompleta) de OWL/ Lite.
- *Razonador DAML:* Se utiliza internamente para proporcionar un soporte mínimo (a escala RDFS) a la inferencia mediante DAML.
- *Razonador genérico basado en reglas:* Un razonador basado en reglas.
- Jena permite incorporar nuevos razonadores externos (Pellet y otros).

6.5.8 Consultar un modelo

- Consultar directamente al modelo mediante list o selectores.
- Mediante consultas SPARQL o RDQL para RDF.
- Mediante SPARQL con JENA.

```
String queryString =""
+ "PREFIX owl: <http://www.w3.org/2002/07/owl#>"
+ "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
+ "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"

+ "PREFIX base:<http://www.owl-ontologies.com/Ontology1282851039.owl#>"

+ "SELECT ?s"
+ " WHERE{?s rdf:type base:Profesor}";

//Se establece la conexión a la base de datos por medio del archivo ttl
Store store = SDBFactory.connectStore( "archivo/miconexion.ttl");
if(store != null){
    System.out.println("Conexión al store SDB... Ok y consulta a la base de datos");
}
Dataset ds = DatasetStore.create(store) ;

// Ejecutar la consulta y obtener los resultados
QueryExecution qe = QueryExecutionFactory.create(queryString, ds);
try {
    ResultSet rs = qe.execSelect();
    ResultSetFormatter.out(rs);
} finally { qe.close(); }
    store.getConnection().close();
    store.close();
```

6.5.9 Jena persistencia

Jena tiene tres formas de dar persistencia a una base de conocimiento:

- Jena genera una persistencia de una OWL a una base relacional.
- Jena/SDB genera una persistencia a un RDF/OWL a una base relacional orientada a transacciones.
- Jena/TDB genera una persistencia a un RDF/OWL a una base relacional no transaccional.

6.5.10 Jena SDB

Permite el almacenamiento y consulta de datos RDF y OWL por medio de bases datos relacionales.

6.5.11 Instalación en Windows

Para el sistema Windows, se requiere del software Cygwin, el cual puede descargarlo de la siguiente dirección:

http://www.cygwin.com/

- Java jdk: **http://www.oracle.com/technetwork/java/javase/downloads/index.html**
- Jena SDB:

http://openjena.org/wiki/SDB link home
http://sourceforge.net/projects/jena/files/SDB/ link descarga

- Base de datos que soporta:

Oracle 10g	Including OracleXE
Microsoft SQL Server 2005	Including MS SQL Express
DB2 9	Including DB2 9 Express
PostgreSQL	v8.2
MySQL	v5.0.22
Apache Derby	v10.2.2.0
H2	1.0.71
HSQLDB	1.8.0

DATABASE ENGINE	JDBC DRIVER
HSQLDB 1.8.0	
MySQL 4.1.11 MySQL 5.0.18	JDBC driver versions: 3.0, 3.1, 5.0
PostgreSQL 7.3 PostgreSQL 8.0	JDBC driver 7.3 JDBC driver 8.0
Apache Derby 10.1	
Oracle 10 XE	Oracle ojdbc14 driver (thin driver) 10.2.0.2
Oracle 9i Release 2	Oracle ojdbc14 driver (thin driver) 10.2.0.2
Oracle 10g Release 2	Oracle ojdbc14 driver (thin driver) 10.2.0.2
Microsoft SQL Server 2005 Express SP1	Microsoft SQL Server 2005 JDBC Driver
Microsoft SQL Server 2000 Microsoft SQL Server Desktop Edition	Microsoft SQL Server 2005 JDBC Driver jTDS version 1.2

Antes de comenzar debe tener todo instalado como Java, el jdbc de MySQL (protege modelado de ontologias) y luego instale el cygwin para configurar el Cygwin; el cual es un aplicativo que permite ejecutar programas de la plataforma Linux en Windows. Este manual enseñará cómo navegar en la consola del "cygwin" que es lo mismo que navegar en Linux.

Cuando se instala el programa se creará una carpeta "../ cygwin/home/user" donde user es el usuario actual de Windows.

- *cd /cygdrive/[unidad de disco]/[nombre carpeta]* permite ir a una unidad del disco distinta dentro de una carpeta. Es necesario introducir una carpeta.
- *ls /cygdrive/[unidad de disco]/[nombre carpeta]* lista los archivos de esa carpeta.

EJECUTANDO LA APLICACIÓN

Paso 1: *Escriba el path.*

Existen diversas formas de escribir los path, cualquiera de estas es válida.

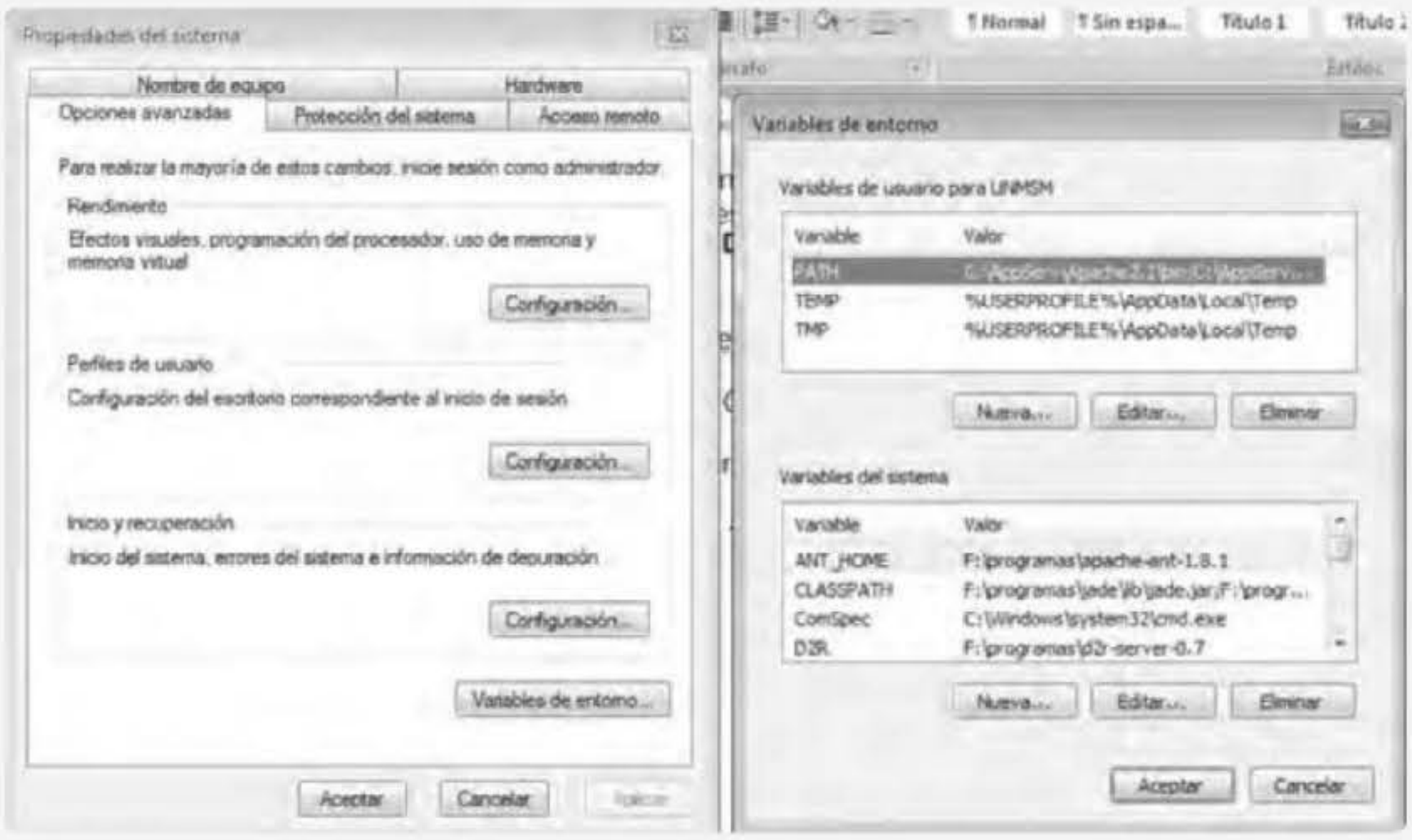
```
ls \c/window
ls c:/windows
ls c:\\windows
ls \c\\windows
ls /cygdrive/c/Windows
```

Paso 2: *Se va a exportar una aplicación de Linux para su cygwin, en este caso el SDB.*

- *SDBROOT: Se coloca la dirección de la carpeta raíz del SDB.*
- *SDB_JDBC: Se coloca la dirección del jdb (F:\programas\mysql-connector-java-5.1.13\mysql-connector-java-5.1.13-bin.jar).*

Donde SDBROOT será la variable a usar en el path de Windows.

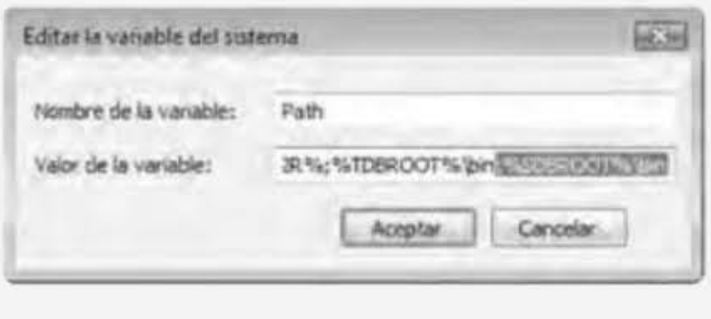
Paso 3: *Diríjase a Propiedades del sistema > Opciones avanzadas > Variables de entorno.*



Paso 4: *Se creará una nueva variable de sistema para que esté habilitada para todos los usuarios de ese ordenador.*



Paso 5: *Coloque en el path de Windows Propiedades de mi PC > Opciones avanzadas > Variables de entorno > variables de sistema > buscar Path (editar) > Valor de variable y escriba en la última línea ";%SDBROOT%\bin".*



Paso 6: *Para la nueva variable de entorno SDB_JDBC, se realiza de la misma manera con excepción del path y se coloca el jar del driver de su jdbc.*

Paso 7: *Se obtiene como nombre de variable la dirección del .jar. Para este ejemplo, se usó el de MySQL "F:\programas\mysql-connector-java-5.1.13\mysql-connector-java-5.1.13-bin.jar".*

6.5.12 Construcción del modelo persistente en la base datos Mysql y carga de los datos por la consola Cygwin

Una vez instalado, el primer paso es crear el almacén de tripletas, para lo cual creará un archivo con el nombre store.ttl, por ejemplo, con el siguiente contenido:

```
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
 @prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
 @prefix ja:    <http://jena.hpl.hp.com/2005/11/Assembler#> .
 @prefix sdb:   <http://jena.hpl.hp.com/2007/sdb#> .

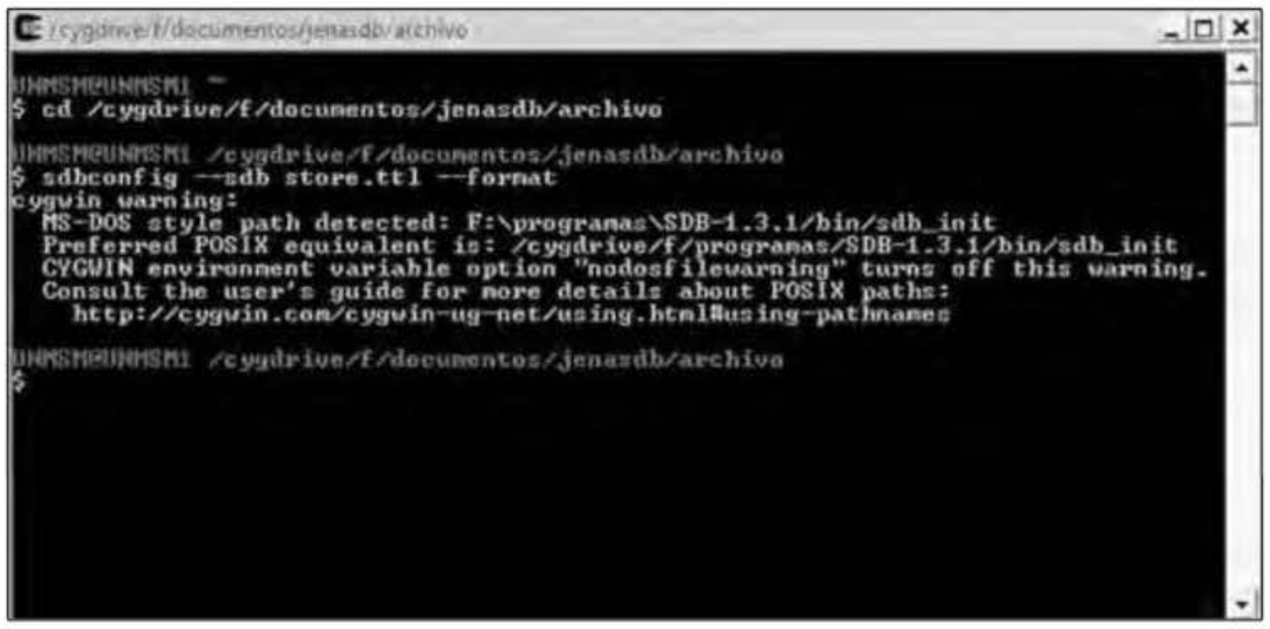
 _:c rdf:type sdb:SDBConnection ;
   sdb:sdbType    "mysql" ;
   sdb:sdbName    "rdf_testdb" ;
   sdb:sdbHost    "localhost" ;
   sdb:sdbUser  "userdb";
   sdb:sdbPassword  "pwddb";
   sdb:driver     "com.mysql.jdbc.Driver" ;
   .

 [] rdf:type sdb:Store ;
   S
db:layout     "layout2/index" ;
   sdb:sdbName    "rdf_testdb" ;
   sdb:connection   _:c ;
   .
```

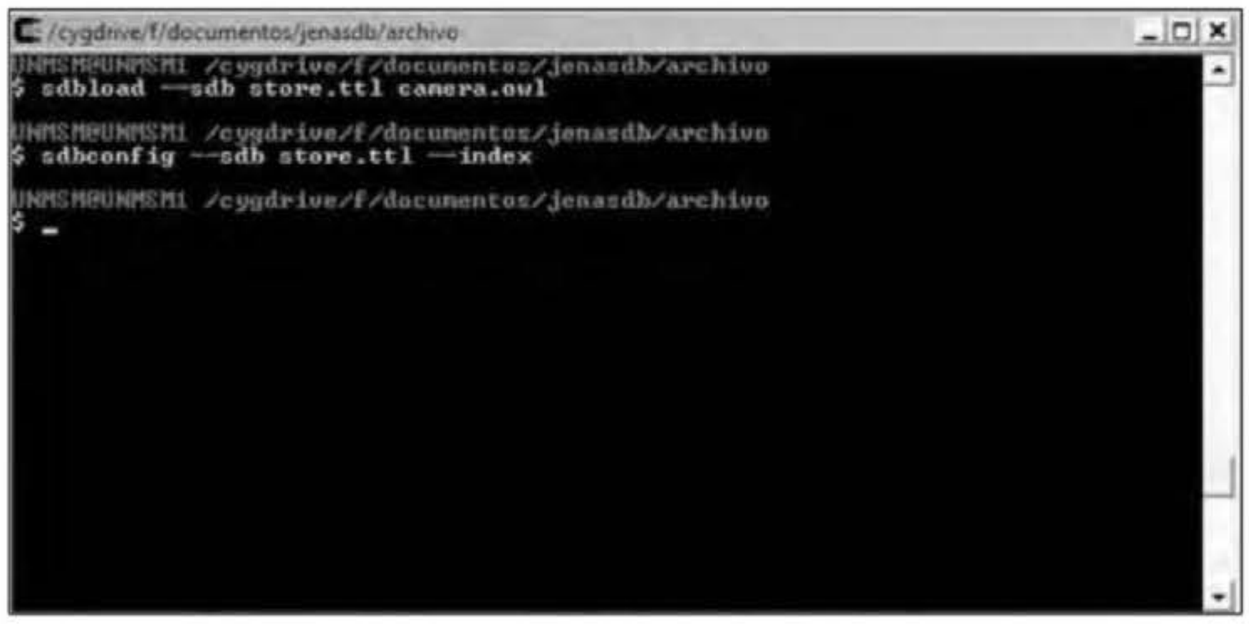
En él se indica que se utilizará MySQL como almacenamiento y cómo acceder a la base de datos que contendrá la información:

- servidor *(localhost)*,
- usuario *(userdb)*,
- contraseña *(pbddb)* y
- nombre de la base de datos *(rdf_testdb)*.

La base de datos (en la cual se están almacenando los valores de los archivos OWL o RDF) puede contener toda una base de datos relacionados. Una vez hecho esto se puede crear el almacén de tripletas RDF sobre la base de datos indicada. Ubique la carpeta que contenga el archivo *.ttl.



El paso siguiente consiste en cargar el archivo OWL y seguidamente cargar los índices. Es conveniente hacerlo en este orden sobre todo si se va a cargar muchos datos.



Una vez cargada la información, consulte en SPARQL.

Por ejemplo, se ha cargado una ontología con estilos de música y autores. Se puede solicitar obtener todos los individuos o instancias de la clase Window; para ello, se digita lo siguiente:

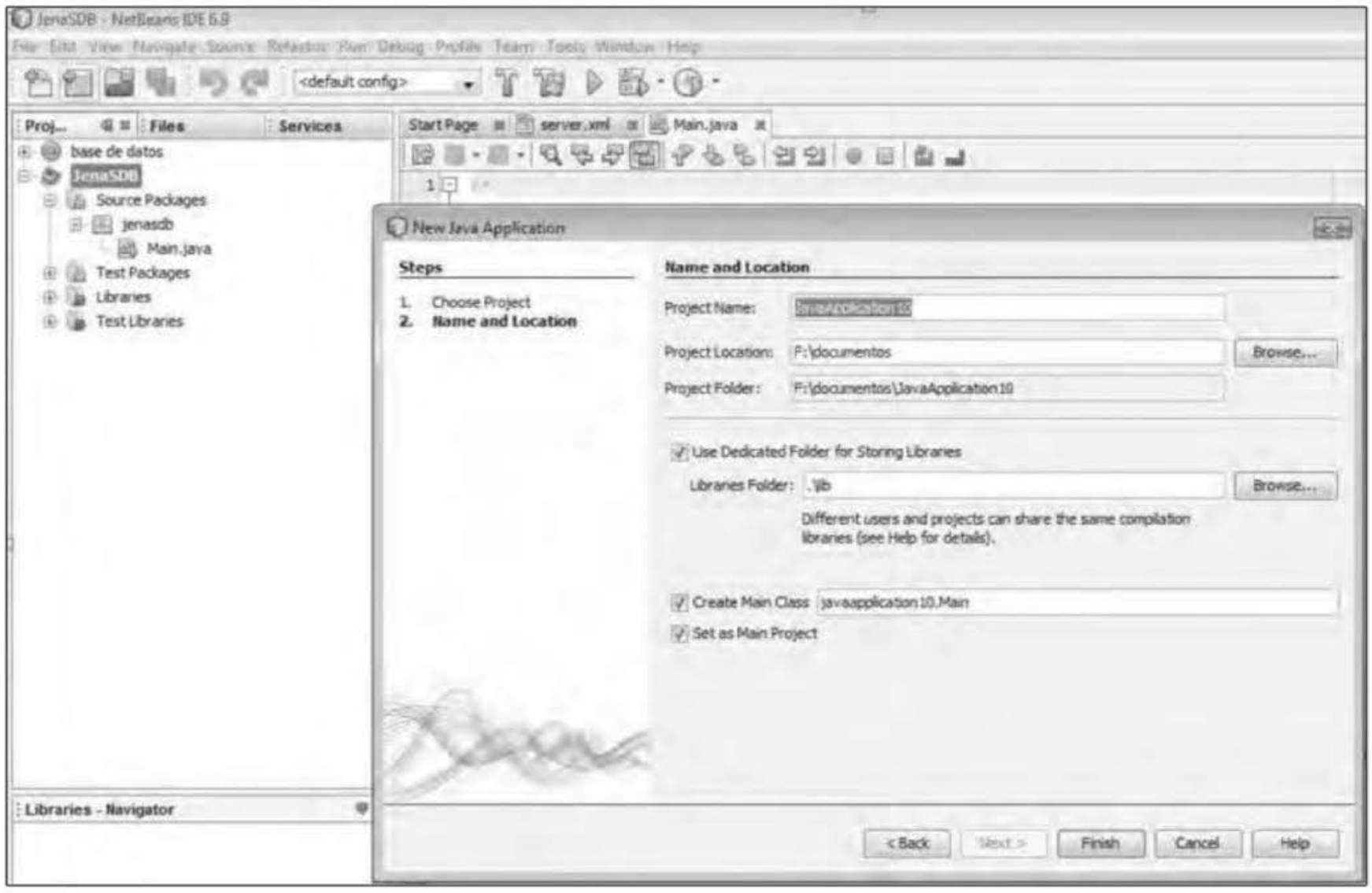
```
"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
 "PREFIX base: <http://www.xfront.com/owl/ontologies/camera/#>" +
 "SELECT ?subject "+
 "WHERE {"+
 "?subject rdf:type base:Window ."+
 "}";
```

6.5.13 Aplicación Java utilizando la API Jena/SDB

Cree un nuevo proyecto, por ejemplo con NetBeans.

Paso 1: Añada las librerías .jar de la carpeta lib de SDB. El siguiente código muestra cómo conectar con el almacén OWL y realizar la consulta recorriendo el recordset, registro a registro. Existe también un método específico que muestra por pantalla directamente los resultados en forma de tabla sin tener que recorrerlos.

Paso 2: Cree una nueva aplicación de java.



Paso 3: Cargue las librerías necesarias haciendo clic derecho en el *Proyecto > Propiedades*. En la parte de categorías, haga clic para selecionar *Librerías > Add JAR/Folder*. Luego, aparece:



Paso 4: Haga clic a *Cargar el archivo MySQL-connector-java-5.1.13-bin.jar*.

Paso 5: En el archivo Main del proyecto, copie el siguiente código:

```
/**
*
*/
public class Main {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        String queryString ="PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
        + "PREFIX base: <http://www.xfront.com/owl/ontologies/camera/#>"
        + "SELECT ?subject "+
        "WHERE {"+
        "?subject rdf:type base:Window ."+
        "}";
        QueryOWL a=new QueryOWL();
        a.query(queryString);
        System.out.println();
        QueryDataBase b= new QueryDataBase();
        b.query(queryString);
    }
}
```

Paso 6: Cree dos objetos uno que realiza la consulta a la OWL y el otro a la base de datos:

```
import com.hp.hpl.jena.query.*;
import com.hp.hpl.jena.sdb.SDBFactory;
import com.hp.hpl.jena.sdb.Store;
import com.hp.hpl.jena.sdb.store.DatasetStore;

/**
*
*/
public class QueryDataBase {
    public void query (String query){
        Store store = SDBFactory.connectStore( "archivo/store.ttl");
        if(store != null){
            System.out.println("Conexión al store SDB ... Ok");
        }
        Dataset ds = DatasetStore.create(store) ;
        QueryExecution qe = QueryExecutionFactory.create(query, ds);

        try {
            ResultSet rs = qe.execSelect();
        } finally { qe.close() ; }

        store.getConnection().close();
        store.close() ;
    }
}
```

Paso 7: Con la siguiente instrucción:

```
sdbconfig --sd [mi_archivo].ttl --format
```

Se agregan las tablas necesarias a la base de datos, las cuales contendrán el modelo de la OWL y las instancias que posea. Se hará lo mismo, pero con una clase de java, véase lo siguiente:

```
import com.hp.hpl.jena.sdb.SDBFactory;
import com.hp.hpl.jena.sdb.Store;
import com.hp.hpl.jena.sdb.StoreDesc;
import com.hp.hpl.jena.sdb.sql.JDBC;
import com.hp.hpl.jena.sdb.sql.SDBConnection;
import com.hp.hpl.jena.sdb.store.DatabaseType;
import com.hp.hpl.jena.sdb.store.LayoutType;
import java.sql.SQLException;
public class Create {
    public void Crear_store () throws SQLException{
        StoreDesc storeDesc =new StoreDesc (LayoutType.LayoutTripleNodesHash, DatabaseType.
        MySQL);

        storeDesc=StoreDesc.read("archivo/miconexion.ttl");
        Store store =SDBFactory.connectStore(storeDesc);

        //crea las tablas e indices que necesita el sdb
        store.getTableFormatter().create();
        //Vacia las tablas
        store.getTableFormatter().truncate();

        store.getConnection().close();
        store.close();
    }
}
```

Si no se contase con un archivo .ttl, el código sería de la siguiente manera:

```
import com.hp.hpl.jena.sdb.SDBFactory;
import com.hp.hpl.jena.sdb.Store;
import com.hp.hpl.jena.sdb.StoreDesc;
import com.hp.hpl.jena.sdb.sql.JDBC;
import com.hp.hpl.jena.sdb.sql.SDBConnection;
import com.hp.hpl.jena.sdb.store.DatabaseType;
import com.hp.hpl.jena.sdb.store.LayoutType;
import java.sql.SQLException;
public class Create {
    public void Crear_store () throws SQLException{
        StoreDesc storeDesc =new StoreDesc (LayoutType.LayoutTripleNodesHash, DatabaseType.
        MySQL);

        JDBC.loadDriverMySQL();
        String jdbcURL ="jdbc:mysql://localhost:3306/java_jenaSDB";
        SDBConnection conn= new SDBConnection(jdbcURL, "root", "root");
        Store store =SDBFactory.connectStore(conn,storeDesc);
```

```
            //crea las tablas e índices que necesita el sdb
            store.getTableFormatter().create();
            //Vacía las tablas
            store.getTableFormatter().truncate();

            store.getConnection().close() ;
            store.close();
        }
    }
```

Por último, en el main:

```
import CrearStore.Create;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Main {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Create sdbfile=new Create();
        try {
            sdbfile.Crear_store();
        } catch (SQLException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE,null,ex);
        }
    }
}

sdbload --sd [mi_archivo].ttl -[Mi_archivo].owl
sdbconfig --sd [mi_archivo].ttl --index
```

Para modificar el código anterior, realice lo siguiente:

```
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.sdb.SDBFactory;
import com.hp.hpl.jena.sdb.Store;
import com.hp.hpl.jena.sdb.StoreDesc;
import com.hp.hpl.jena.sdb.sql.JDBC;
import com.hp.hpl.jena.sdb.sql.SDBConnection;
import com.hp.hpl.jena.sdb.store.DatabaseType;
import com.hp.hpl.jena.sdb.store.LayoutType;
import com.hp.hpl.jena.util.FileManager;
import java.io.InputStream;
import java.sql.SQLException;
```

```
public class Create {
    public void Crear_store () throws SQLException{
        StoreDesc storeDesc=new StoreDesc (LayoutType.LayoutTripleNodesHash, DatabaseType.MySQL);
        storeDesc=StoreDesc.read("archivo/miconexion.ttl");
        Store store =SDBFactory.connectStore(conn, storeDesc);

        // Crea todas las tablas e índices necesarios para la sdb
        store.getTableFormatter().create();
        // Vacía las tablas
        store.getTableFormatter().truncate();

        /*
         * Las siguientes líneas se encargarán de realizar lo escrito en la consola
         * sdbload --sdb miconexion.ttl camera.owl
         * sdbconfig --sdb miconexion.ttl --index
         */
        Model base = SDBFactory.connectDefaultModel( store);
        OntModel ontmodel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM, base);
        InputStream in = FileManager.get().open("archivo/camera.owl");
        ontmodel.read( in, "");
        /*
         * Cierra la conexión a la base de datos y al store
         */
        store.getConnection().close() ;
        store.close() ;
    }
}
```

6.5.14 Jena/SDB diseños de base de datos

SDB no tiene un diseño de base de datos único. Esta página es un resumen informal de los dos tipos principales ("layout2/hash" y "layout2/index"). En SDB, una tienda es un conjunto de datos RDF y una base de datos SQL. Bases de datos de tipo Layout2 tiene una tabla Triples de la gráfica; por defecto, una tabla Quads para los nombres de gráficos. En la tabla Triples y en la Quads, las columnas son enteras referenciando a una tabla Nodes.

- En la forma de hash, los números enteros son los hashes de 8 bytes del nodo.
- En la forma de índice, los números enteros son identificadores de secuencia de 4 bytes en la tabla de nodo.

- **Triples**

```
+-----------+
| S | P | O |
+-----------+
```

Clave principal: SPO
Índices: PO, OS

- **Quads**

```
+---------------+
G | | S | P | S |
+---------------+
```

Clave principal: GSPO
Índices: GPO, GOS, SPO, OS, PO.

- **NODES**

El diseño de índice basado en la tabla es la siguiente:

```
+-----------------------------------------------+
| Id | Hash | lex | lang | datatype | value type |
+-----------------------------------------------+
Clave principal: Id.
Índice: Hash
```

- **HASH**

```
+---------------------------------------+
| Hash | lex | lang | datatype | value type |
+---------------------------------------+

Hash: Primaria clave
```

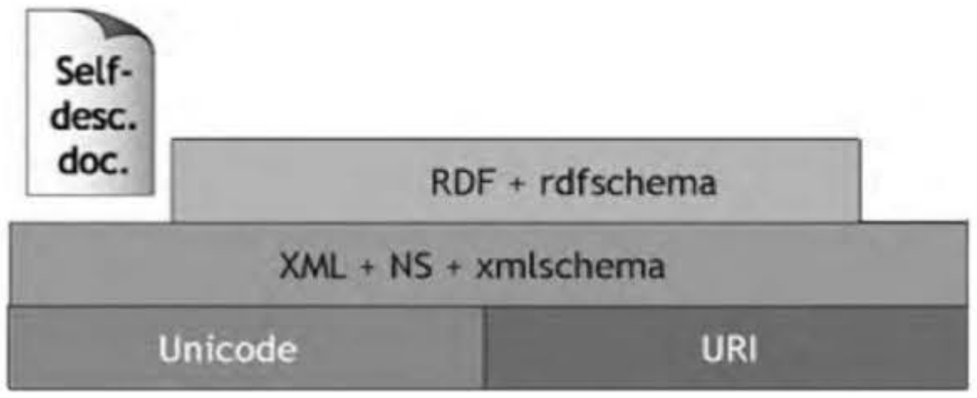
Todos los campos de caracteres son Unicode, soporta cualquier conjunto de caracteres, incluyendo el uso de lenguaje mixto.

SWRL: *Semantic Web Rule Language* sirve de puente para las reglas de negocios y la producción de sistemas de reglas lógicas más expresivas.

6.6 ANEXOS

6.6.1 RDF (RESOURCE DESCRIPTION FRAMEWORK)

Se construyó sobre tecnologías Unicode, URI y XML + NS + xmlschema.



- **UNICODE**

Estándar para representar y manipular texto en cualquier idioma del mundo. Brinda la capacidad de intercambiar información entre las máquinas.

- **URI (UNIFORM RESOURCE IDENTIFIERS)**

Identificadores de recursos únicos, sin posibilidad de ambigüedad. Los documentos en la web actual tienen una URL.

  - Localizador único del recurso: http://www.pacis.com

Cada objeto en la web semántica tiene una URI.

- Identificador único de un recurso.

Algunas URI pueden ser URL:

- El paciente posee una URI (http://www.w3.org/2000/01/rdf-schema#Paciente).

- **XML (eXtensible Markup Language)**

  Metalenguaje de etiquetas extensibles que se puede "acomodar" a las necesidades de cada usuario. Asimismo, describe una estructura de árbol capaz de ser procesada por las máquinas.

- **XML + NS (espacio de nombres, Name Space)**

  Etiqueta una determina URI para su utilización en una aplicación o consulta:

```
<Jesus:price xmlns:Jesus='http://www.pacis.com/schema' units='Soles'>
32.18
</Jesus:price>
```

- **XML + Schema**

  Permite definir documentos de tipo XML. Ejemplo obtenido de la ref[9]:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2000/08/XMLSchema">
<xsd:element name="persona" type="tipoPersona"/>
<xsd:element name="comentario" type="xsd:string"/>
<xsd:complexType name="tipoPersona">
<xsd:sequence>
<xsd:element name="nombre" type="xsd:string"/>
<xsd:element ref="comentario" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

- **RDF**

METADATOS



Se trata de una descripción de un modelo de datos; en lugar de la descripción de un vocabulario de datos específicos, le agrega semántica a un XML. Se trata de un modelo de metadatos que se basa en la sintaxis XML y otras como N3, N-Triple, Turtle, RxR, Trix.

No ofrece un significado formal, computable a las propiedades. Sigue el modelo de la tripleta:

```
sujeto-propiededad-objeto
```

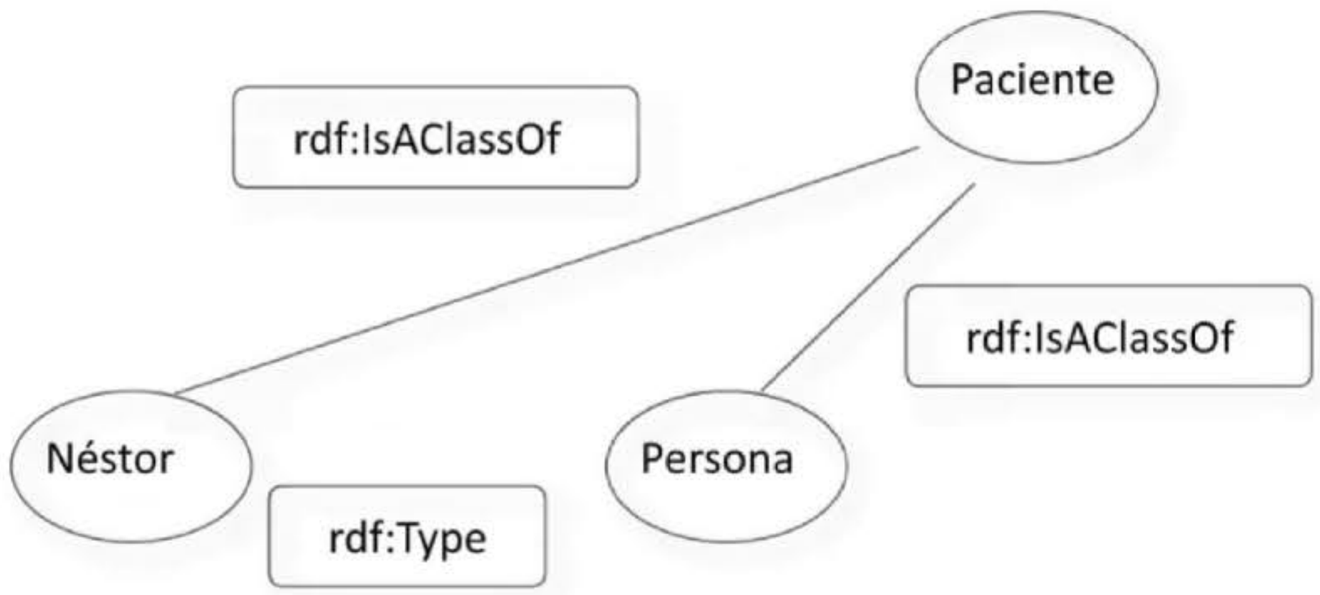
Sujeto: Se trata de un identificador de un recurso.
Propiedad: Define la relación o tipo de valor.
Objeto: Es el valor o literal de la propiedad.

6.6.2 RDFS (RDFSCHEMA)

Otorga un framework para describir clases y propiedades. Es una especie de type system para RDF (comparable a los type systems usados en OOP como Java).

Define vocabularios con términos y relaciones entre los mismos. Brinda cierta semántica a los recursos y propiedades; además, permite a las computadoras hacer ciertas tareas Class, SubClassof, Type, Propety.

Es posible inferir a partir de un RDFS, lo siguiente:



6.6.3 D2R SERVER

Publica la base de datos relacionales en la web semántica.

A. Descargar

d2rserver

B. Requerimientos

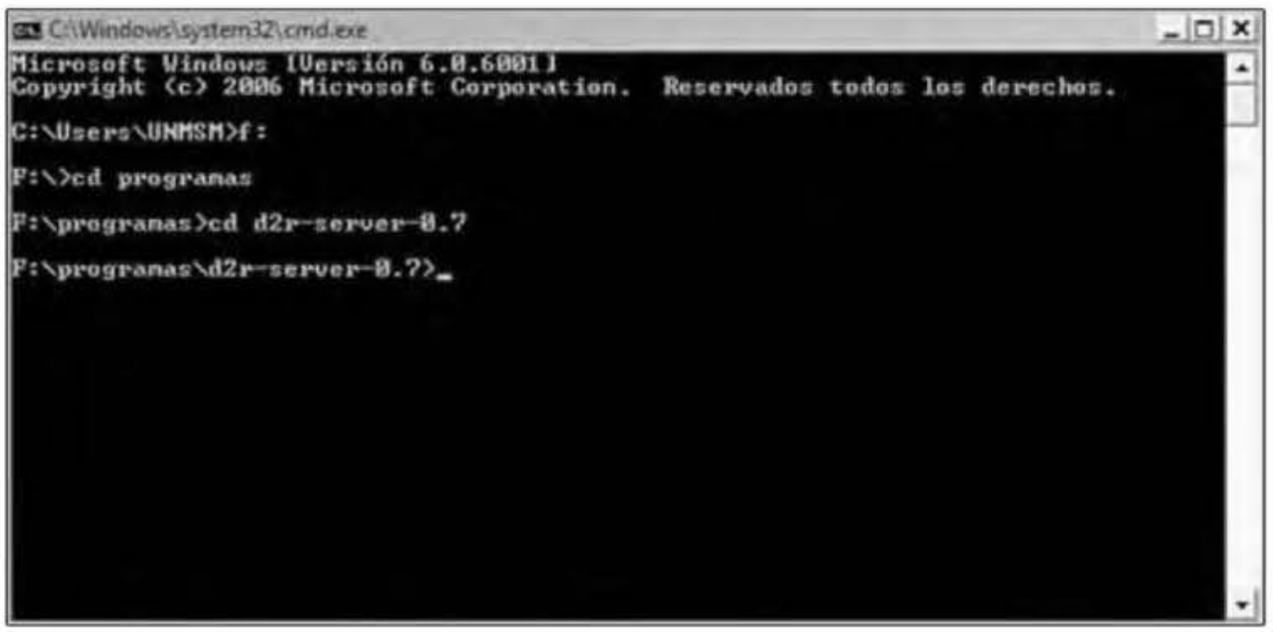
- Java,
- jdbc drivers,
- opcionalmente tener un servlet (como toncat, apache, J2EE, Glasfiship u otro).

C. Instalación

- Descomprima el archivo descargado.
- Genere un mapa de la base de datos, para este caso es una base creada en MySQL.
  - En Windows:

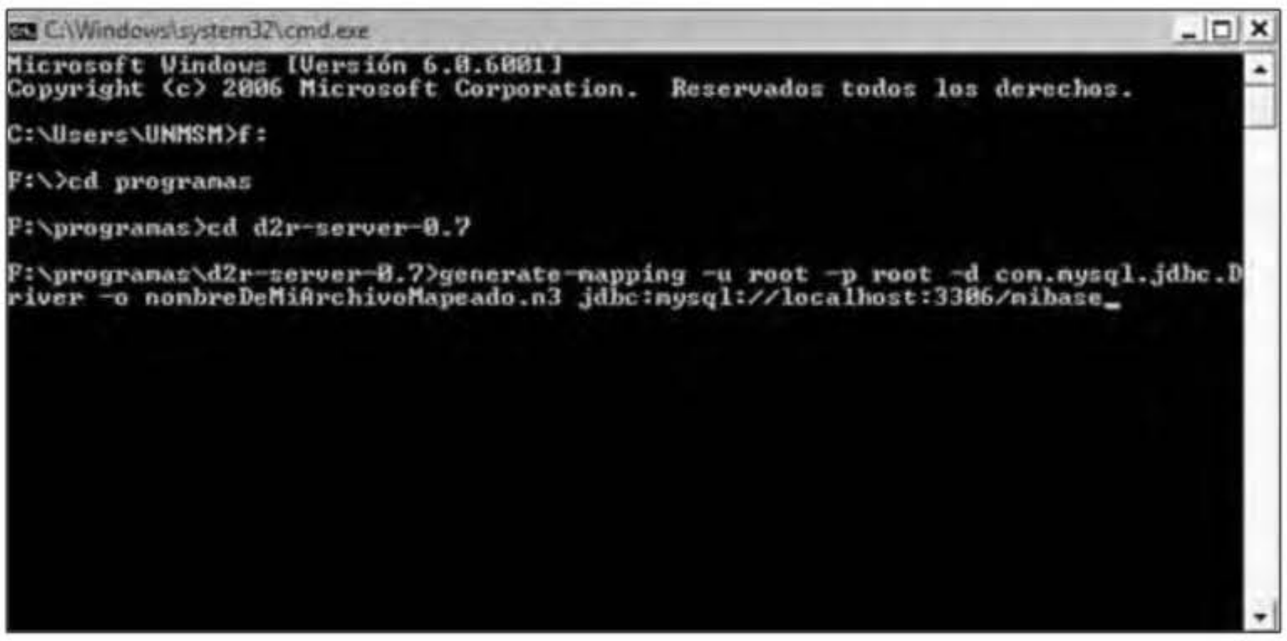
Paso 1: Abra cmd.

Paso 2: Diríjase a la carpeta donde se descomprimió el archivo.



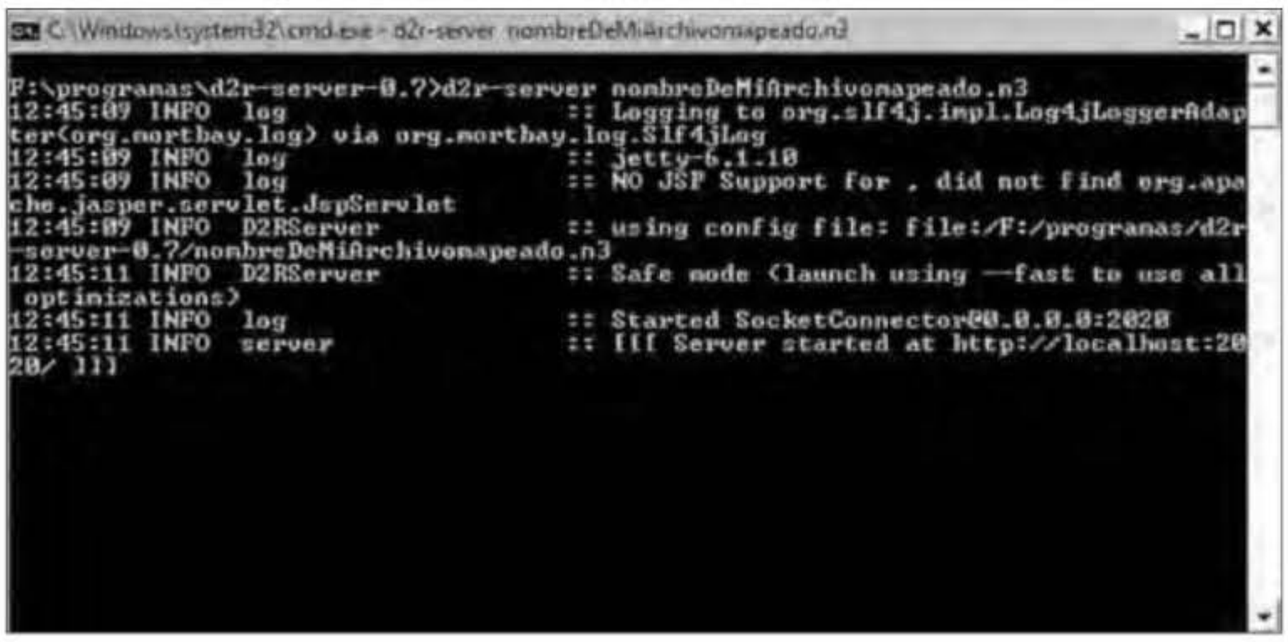
Paso 3: Escriba lo siguiente:

```
generate-mapping -u root -p root -d com.mysql.jdbc.Driver -o nombreMapeado.n3 jdbc:mysql://
localhost:3306/mibase
```



Paso 4: Corra el servidor con el archivo mapeado.

```
d2r-server nombreDeMiArchivomapeado.n3
```



Paso 5: Ejecute en un explorador web: http://localhost:2020



6.6.4 D2RQ con Jena

Para el tiempo en que se escribió este manual, se usó la versión d2rq-0.7.jar. Para el siguiente ejemplo, se creará un proyecto en Netbeans y se usará el archivo n3, el cual es generado por el d2r-server, donde se agregará todos los jar de la carpeta \d2r-server-0.7\lib. También es necesario el jdbc driver, para este proyecto se utilizó el MYSQL.

```
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.StmtIterator;
import com.hp.hpl.jena.sparql.vocabulary.FOAF;
import com.hp.hpl.jena.vocabulary.DC;
import com.hp.hpl.jena.vocabulary.RDF;
import de.fuberlin.wiwiss.d2rq.ModelD2RQ;
public class Main {
  public static void main(String[] args) {
    // Se crea un modelo en memoria
    Model m =new ModelD2RQ("doc/test.n3");

    StmtIterator paperIt=m.listLiteralStatements(null, RDF.type, true);
    while (paperIt.hasNext()){
      Resource paper =paperIt.nextStatement().getSubject();
      System.out.println("Paper: " + paper.getProperty(DC.title).getString());
      StmtIterator authorIt =paper.listProperties(DC.creator);
       while (authorIt.hasNext()) {
         Resource author = authorIt.nextStatement().getResource();
         System.out.println("Author: " + author.getProperty(FOAF.name).getString());
       }
         System.out.println();
      }
    }

  }
```



CAPÍTULO

7

HTML 5

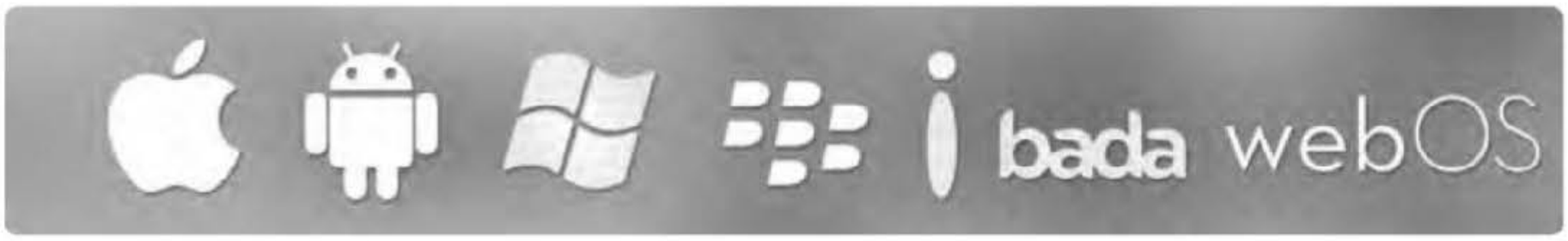
7.1 Introducción



Gartner predice que HTML5 aumentará su dominio en el mercado de aplicaciones web móviles; lo que no significa que las aplicaciones nativas desaparezcan; siempre ofrecerán la mejor experiencia al usuario.

HTML5 es una especificación preliminar para la próxima versión importante de HTML. Representa un cambio con sus predecesores, HTML4 y XHTML. Algunos elementos se han eliminado y ya no se basa en SGML, un estándar antiguo para marcas. Hay muchas adiciones notables a HTML, como el soporte nativo de dibujo y elementos audiovisuales.

HTML5 es la plataforma ideal para la Web Móvil. Tanto si eres un desarrollador de contenido web para móviles como una empresa con unas determinadas necesidades comerciales o un desarrollador de juegos profesional, que quiere explorar las posibilidades de la Web como nueva plataforma.



7.2 Requerimiento de Software

Una vez instalado el JDK, SDK e IDE Eclipse, solo queda descargarlo. Se necesitarán los siguientes requerimientos de software que se detallarán en los siguientes apartados.

7.2.1 PhoneGap

PhoneGap es un framework para el desarrollo de aplicaciones móviles producido por Nitobi y comprado posteriormente por Adobe Systems. Permite a los programadores desarrollar aplicaciones para dispositivos móviles utilizando herramientas genéricas tales como JavaScript, HTML5 y CSS3. Las aplicaciones resultantes son híbridas; es decir, que no son realmente aplicaciones nativas al dispositivo (ya que el renderizado es realizado mediante vistas web y no con interfaces gráficas específicas a cada sistema), pero no se trata tampoco de aplicaciones web (teniendo en cuenta que son aplicaciones empaquetadas, para poder ser desplegadas en el dispositivo incluso trabajando con el API del sistema nativo).



PhoneGap puede ser considerado como una distribución de Apache Cordova (software de código abierto). La aplicación fue primero llamada “PhoneGap” y luego “Apache Callback”.

A continuación deberá seguir los siguientes pasos:

Paso 1: Ingrese al link http://www.phonegap.com.

Paso 2: Haga doble clic en el ícono siguiente:

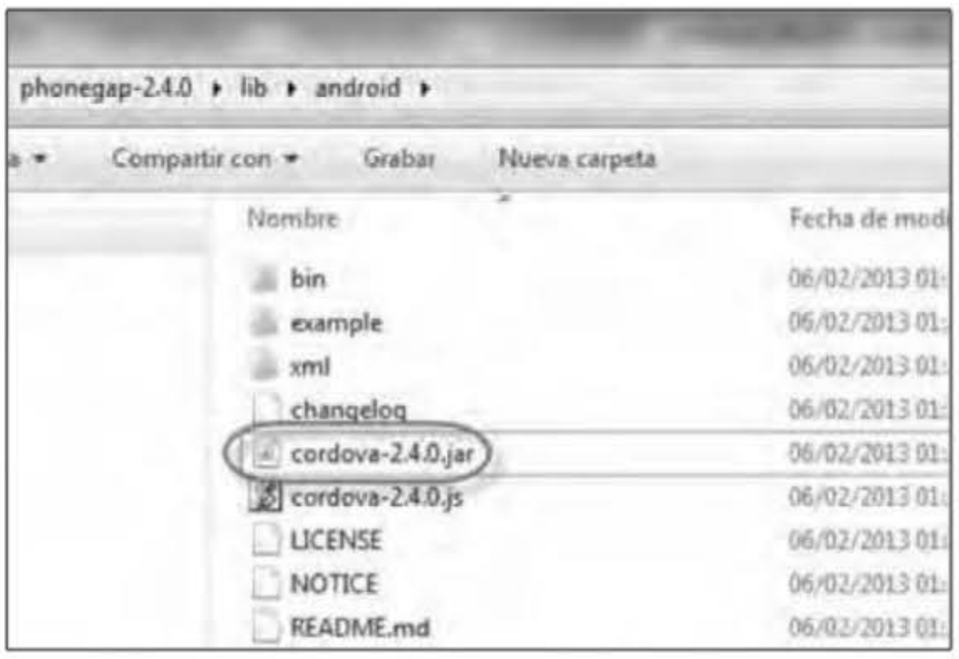


Paso 3: Haga clic en:



Paso 4: Guarde el archivo phonegap-2.4.0.zip, en alguna carpeta específica y luego descomprima.

Paso 5: Al descomprimir ubique la librería que más adelante utilizará.





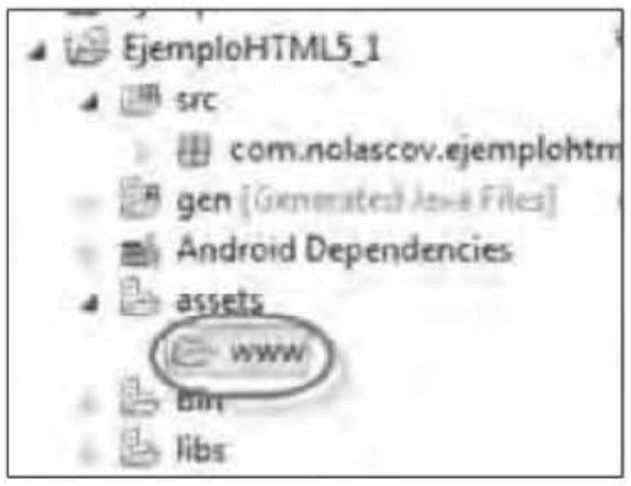
Paso 6: Diseñe una aplicación; la cual permita mostrar el texto "Hola mundo".

Paso 7: Después de haber creado la primera aplicación para Android, ahora cree un nuevo proyecto seleccionando Archivo > Nuevo > Android proyecto, el nombre de proyecto: EjemploHTML5_1 (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

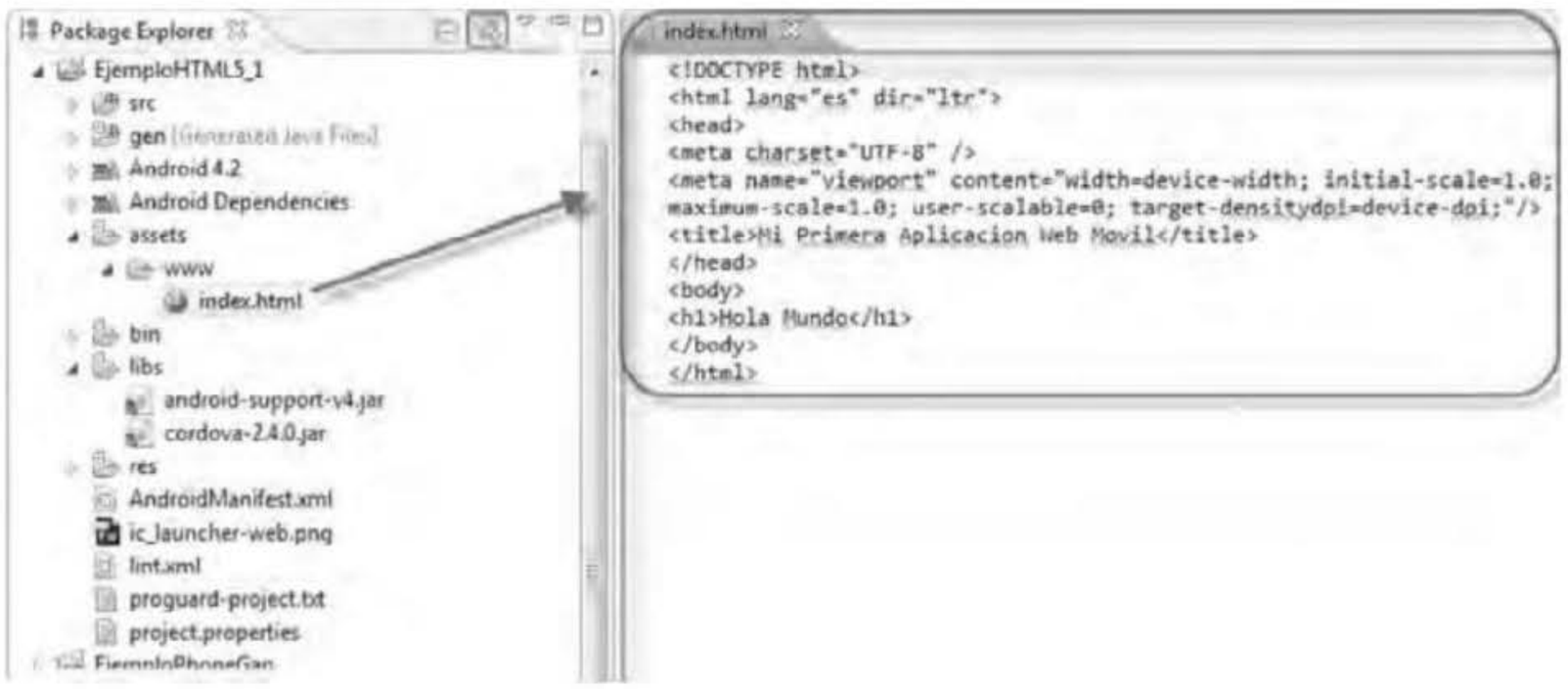
Paso 8: Almacene el fichero .jar en una carpeta y añádala a su proyecto como una librería externa (cordova-2.4.0.jar).



Paso 9: Cree dentro de la carpeta assets , la carpeta www.



Paso 10: El siguiente archivo se encontrará dentro de la carpeta www, index.html.



Index.html

```
<!DOCTYPE html>
<html lang="es" dir="ltr">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width; initial-scale=1.0;
        maximum-scale=1.0; user-scalable=0; target-densitydpi=device-dpi;"/>
        <title>Mi Primera Aplicacion Web Movil</title>
    </head>
    <body>
        <h1>Hola Mundo</h1>
    </body>
</html>
```

Paso 11: Codifique en la actividad MainActivity.java lo siguiente:

```
package com.nolascov.ejemplohtml5_1;

import android.os.Bundle;
import org.apache.cordova.*;

public class MainActivity extends DroidGap
{
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    super.loadUrl("file:///android_asset/www/index.html");
  }
}
```

EJECUTANDO LA APLICACIÓN

Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:



7.3 CANVAS



Canvas es un elemento HTML incorporado en HTML5 que permite la generación de gráficos, dinámicamente por medio del scripting. Permite generar gráficos estáticos y animaciones. Fue implementado por Apple para su navegador Safari. Más tarde, fue adoptado por otros navegadores, como Firefox a partir de su versión 1.5, Opera y es adoptado por WHATWG.

Posee dos atributos width (ancho) y height (alto). El tamaño por defecto es 160. Para acceder al objeto canvas, es a través de Javascript; permitiendo generar gráficos 2D, juegos, animaciones y composición de imágenes. Asimismo, existe otra etiqueta SVG que cumple con funciones similares.

7.3.1 CONCEPTOS BÁSICOS SOBRE CANVAS

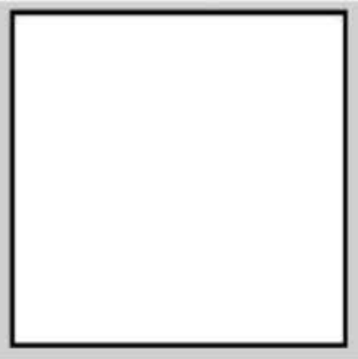
A. Crear un Canvas

```
<canvas id="primerCanvas" width="100" height="100"></canvas>
```

Como se mencionó anteriormente, Canvas posee dos atributos width (ancho) y height (alto), el tamaño por defecto es 150.

B. Definir bordes en un Canvas

```
<canvas id="primerCanvas" width="100" height="100"
    style="border:1px solid #0000FF;">
</canvas>
```

CÓDIGO FUENTE	VISTA PRELIMINAR
<html> <body> <canvas id="primerCanvas" width="100" height="100" style="border:1px solid #0000FF;"> </canvas> </body> </html>	

C. Dibujar con Canvas

CÓDIGO FUENTE	VISTA PRELIMINAR

<!DOCTYPE html>
<html>
 	
<body>
 	
<canvas id="segundoCanvas" width="100" height="100"
style="border:3px solid #FFFF00;">
 	
</canvas>
 	
<script>
 	
var c=document.getElementById("segundoCanvas");
var ctx=c.getContext("2d");
 	
//define el color de relleno
ctx.fillStyle="#FF0000";
 	
// Dibuja un rectángulo con relleno
ctx.fillRect(0,0,100,100);
 	
</script>
 	
</body>
 	
</html>
	

Nota:

- *El método document.getElementById permite obtener la referencia a un elemento de la página mediante el ID de dicho elemento (en este caso al Canvas).*

```
var c=document.getElementById("segundoCanvas");
```

- *El método getContext y el segundo permite acceder al lienzo del Canvas, sobre el cual se va a dibujar un rectángulo con HTML5.*

```
var ctx=c.getContext("2d");
```

D. Dibujar sombras con Canvas

CÓDIGO FUENTE	VISTA PRELIMINAR

```
<!DOCTYPE html>

<html>

<body>

<canvas id="tercerCanvas" width="270" height="120"
style="border:3px solid #d3d3d3;">
</canvas>

<script>

//referencia a un elemento de la pagina mediante
un id
var c=document.getElementById("tercerCanvas");

//acceder al lienzo del canvas
var ctx=c.getContext("2d");

//define el valor del tamaño de desenfoque
ctx.shadowBlur=30;

//define el color de relleno
ctx.fillStyle="red";

//define el color de sombra
ctx.shadowColor="#FFFF00";

//Dibuja un rectángulo con relleno
ctx.fillRect(20,20,100,80);

//define el color de sombra
ctx.shadowColor="#FFFF00";

// Dibuja un rectángulo con relleno
ctx.fillRect(140,20,100,80);

</script>

</body>

</html>
```

E. Dibujar texto gradiente con Canvas

CÓDIGO FUENTE	VISTA PRELIMINAR

```
<!DOCTYPE html>
<html>
<body>
<canvas id="cuartoCanvas" width="420" height="100"
style="border:1px solid #d3d3d3;">
</canvas>
<script>
var c=document.getElementById("cuartoCanvas");
var ctx=c.getContext("2d");

ctx.font="30px Verdana";
// Create gradient
var gradient=ctx.createLinearGradient(0,0,c.width,0);
gradient.addColorStop("0","magenta");
gradient.addColorStop("0.5","blue");
gradient.addColorStop("1.0","red");
// asigna el degradado al strokeStyle
ctx.strokeStyle=gradient;
ctx.strokeText("Texto con color gradiente",10,50);

</script>

</body>
</html>
```



Para degradados, utilice este objeto para asignar a las propiedades fillStyle o strokeStyle. Cree un objeto canvasGradient mediante uno de los métodos siguientes:

```
El método createLinearGradient toma cuatro argumentos que representan el punto de partida
o inicial (x1, y1) y el punto final (x2, y2) del gradiente.
var gradient=ctx.createLinearGradient(0,0,c.width,0);
```

```
El método AddColorStop(position, color) Este método toma dos argumentos. La posición debe
ser un número entre 0.0 y 1.0 y define la posición relativa de los colores en el degradado.
El argumento de color debe ser una cadena que representa un color CSS (es decir, # FFF,
rgba (0,0,0,1))

gradient.addColorStop("0","magenta");
gradient.addColorStop("0.5","blue");
gradient.addColorStop("1.0","red");
```

```
asigna el degradado al strokeStyle
ctx.strokeStyle=gradient;
```

F. Manejo de texto en Canvas

CÓDIGO FUENTE	VISTA PRELIMINAR

```
<!DOCTYPE html>

<html>
<body>

<canvas id="quintoCanvas" width="230" height="80"
style="border:3px solid #90EE90;">
</canvas>

<script>

var c=document.getElementById("quintoCanvas");
var ctx=c.getContext("2d");
ctx.font="40px Arial";
ctx.fillText("Hola Amigos",2,50);

</script>

</body>
</html>
```


◉ PROPIEDAD FONT

```
ctx.font="40px Arial";
```

TAMAÑO Y TIPO DE LETRAS	40px Arial (Valores por defecto 10px sans-serif)
SINTAXIS JAVASCRIPT	`context.font="italic small-caps bold 12px arial";`

◉ Algunos valores de la propiedad Font

Valor	Descripción
font-style	Especifica los estilos de Fuentes: • normal • italic • oblique
font-variant	Especifica la Variante de la fuente: • normal • small-caps
font-weight	Especifica el grosor de la fuente: • normal • bold • bolder • lighter • 100 • 200 • 300 • 400 • 500 • 600 • 700 • 800 • 900
font-size/line-height	Especifica el tamaño de fuente y la altura de la línea, en píxeles
font-family	Especifica la familia de fuentes
caption	Utilizado para los controles de fuente subtítulos (como botones, menús desplegables, etc)
icon	Utilice la fuente utilizada para etiquetar iconos
Menu	Utilice el tipo de letra utilizado en los menús (menús desplegables y las listas de menú)
message-box	Utilice la fuente utilizada en los cuadros de diálogo

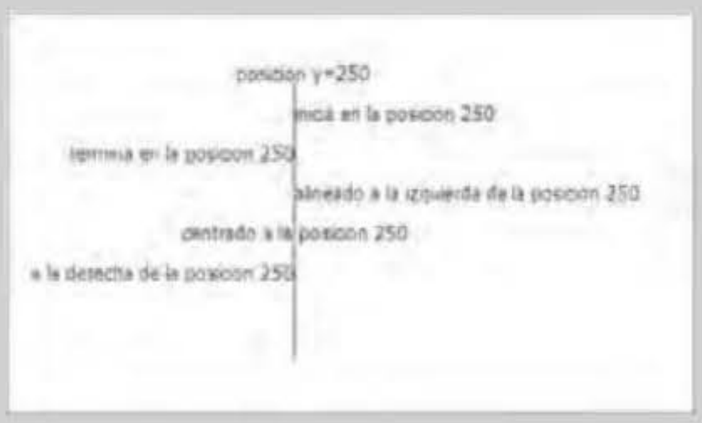
G. Alineación de texto

La propiedad textAlign establece o devuelve la alineación actual de contenido de texto, de acuerdo con el punto de anclaje. Normalmente, el texto comenzará en la posición especificada; sin embargo, si se establece textAlign = “right” y coloca el texto en la posición 200. Esto significa que el texto debe terminar en la posición 200.

ALINEACIÓN DE TEXTO EN CANVAS

CÓDIGO FUENTE	VISTA PRELIMINAR

```
<!DOCTYPE html8
<html>
<body>
<canvas id="sextoCanvas" width="600" height="300"
style="border:1px solid #00FF7F;">
</canvas>
<script>
var c=document.getElementById("sextoCanvas");
var ctx=c.getContext("2d");
//crea una linea en la posicion 250
ctx.strokeStyle="blue";
ctx.moveTo(250,50);
ctx.lineTo(250,260);
ctx.stroke();
ctx.font="15px Verdana";
ctx.fillText("posicion y=250",200,50);
//diferentes alineaciones
ctx.textAlign="start";
ctx.fillText("inicia en la posicion 250",250,80);
ctx.textAlign="end";
ctx.fillText("termina en la posicion
250",250,110);
ctx.textAlign="left";
ctx.fillText("alineado a la izquierda de la posi-
cion 250",250,140);
ctx.textAlign="center";
ctx.fillText("centrado a la posicion
250",250,170);
ctx.textAlign="right";
ctx.fillText("a la derecha de la posicion
250",250,200);
</script>
</body>
</html>
```



H. Escribir texto en Canvas

CÓDIGO FUENTE	VISTA PRELIMINAR

```
<!DOCTYPE html>
<html>
<body>

<canvas id="septimoCanvas" width="300" height="150"
style="border:3px solid #00BFFF;">
</canvas>

<script>

var c=document.getElementById("septimoCanvas");

var ctx=c.getContext("2d");

ctx.font="22px Verdana";

ctx.fillText("Texto Numero 1",11,55);

ctx.font="33px Verdana";

// Crear gradiente
var gradient=ctx.createLinearGradient(0,0,c.width,0);
gradient.addColorStop("0","red");
gradient.addColorStop("0.5","blue");
gradient.addColorStop("1.0","white");

// llenado con el gradiente
ctx.fillStyle=gradient;
ctx.fillText("Texto Numero 2",10,90);

</script>

</body>

</html>
```


7.3.2 DRAWIMAGE()

Método utilizado para dibujar imágenes. Para ello, se utiliza el método drawImage(), que pertenece al objeto contexto del Canvas, con la siguiente sintaxis:

```
drawImage(objeto_imagen, x, y)
```

Envíe tres parámetros: El primero, es el objeto Javascript de la imagen que se desea incluir en el Canvas. Los dos siguientes, son las coordenadas donde situar la imagen, siendo (X, Y) el punto donde se colocará la esquina superior izquierda de la imagen.

Cabe advertir que este método pertenece al objeto del Canvas; por lo que antes de poder invocarlo debe haber obtenido el contexto del Canvas.

Código fuente	Vista preliminar

```
<!DOCTYPE html>
<html>
<body>

<p>Imagen:</p>
<img id="inkadroid" src="inkadroid.jpg" alt="The
Scream" width="210" height="254">

<p>Canvas:</p>
<canvas id="octavoCanvas" width="260" height="300"
style="border:3px solid #EEE8AA;">
</canvas>

<script>

var c=document.getElementById("octavoCanvas");
var ctx=c.getContext("2d");
var img=document.getElementById("inkadroid");
ctx.drawImage(img,10,10);

</script>

</body>
</html>
```



7.4 Manejo de eventos en HTML5

Existen tres diferentes formas de registrar un evento para un elemento HTML:

- Agregar un nuevo atributo al el evento.
- Registrar un manejador de evento como una propiedad del evento.
- Usar el nuevo método estándar addEventListener().

7.4.1 Método addEventListener()

El método addEventListener() es la técnica ideal considerada como estándar por la especificación de HTML5. Por otro lado, este método tiene tres argumentos:

- Nombre del evento
- Función a ser ejecutada
- Valor lógico (falso o verdadero)

```
//escuchar eventos en el momento de cargar la pantalla
window.addEventListener('load', load, false);
```

La sintaxis del método addEventListener() es como sigue:

3. El primer atributo es el nombre del evento.
4. El segundo es la función a ser ejecutada; la cual puede ser una referencia a una función (como en este caso) o una función anónima.
5. El tercer atributo especificará usando true (verdadero) o false (falso), como múltiples eventos serán disparados.

7.5 Ejemplo HTML5 básico

Ahora se diseñará una aplicación que permita realizar algunos trazos.

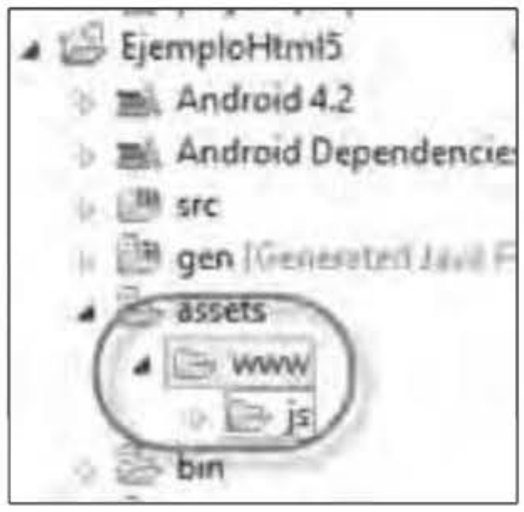
Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android* proyecto.

Paso 2: Ingrese el nombre del proyecto: EjemploHtml5 (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

Paso 3: Cree dentro de la carpeta assets, la carpeta www.



Asimismo, dentro de la carpeta www, almacenará los javascrip (js).



Paso 4: Dentro de esta carpeta tendrá los siguientes ficheros:

- **MAIN.JS**

```
/*** MAIN ***/
//es el programa principal
//escuchar eventos en el momento de cargar la pantalla
window.addEventListener('load', load, false);

var WIN_WDT = window.innerWidth;
var WIN_HGT = window.innerHeight;
var TOOLBAR_HGT = 40;
var ctxToolbar;
var ctxCanvas;
var ctxPalette;
var ctxBrushes;
var mousedown = false;
var colorIdx = 16;
var colorTmp = colorIdx;
var bgcolorIdx = 0;
var bgcolorTmp = bgcolorIdx;
var lineWidth = 16;
var lineTmp = lineWidth;
var radius = lineWidth >> 1;
var currX;
var currY;
var lastX;
var lastY;
var img_button;
var buttonIdx = 0;
var BUTTON_WDT = 80;
var BUTTON_HGT = 31;
var MAX_RES = 1;
var countRes = 0;
//carga todas los recursos, imagenes y botones
function load() {
    loadResources();
    //crear el objeto imagen
    img_button = new Image();
    img_button.onload = init;
    //ruta de la iamgen
    img_button.src = 'images/button.png';
}

function loadResources() {
    // console.log('loadResources()');
    img_button = new Image();
    img_button.onload = countResources;
    img_button.src = 'images/button.png';
}
//cargador de imagenes
function countResources() {
    //  console.log('countResources()');
    countRes++;
    if(countRes = MAX_RES) {
        init();
    }
}
```

```
//incializa la pantalla
function init() {
    BUTTON_WDT = img_button.width;
    BUTTON_HGT = img_button.height;
    WIN_WDT = window.innerWidth;
    WIN_HGT = window.innerHeight;
    initToolbar();
    initCanvas();
    initPalette();
    initBrushes();
}

// dibujo un boton
function drawButton(ctx, x, y, text, hover) {
    var px;
    if(hover) {
        ctx.fillStyle = '#FF8000';
    } else {
        ctx.fillStyle = '#404040';
    }
    ctx.fillRect(x-2, y-2, BUTTON_WDT+4, BUTTON_HGT+4);
    ctx.drawImage(img_button, x, y);
    px = x + ((BUTTON_WDT - ctx.measureText(text).width) >> 1)
    ctx.fillStyle = '#000000';
    ctx.fillText(text,px+1,y+8);
    ctx.fillStyle = '#FFFFFF';
    ctx.fillText(text,px,y+7);
}

// pregunta si x y esta dentro de un rectangulo ( boton)
function inside( px, py, x, y, wdt, hgt ) {
    if( (px >= x) && (px < (x+wdt)) && (py >= y) && (py < (y+hgt)) ) {
        return true;
    } else {
        return false;
    }
}
// rellena con color la brocha actual
function ctxSetColor(ctx) {
    imgData = ctx.getImageData(0, 0, ctx.canvas.width, ctx.canvas.height);
    size = imgData.width * imgData.height;
    pixel = imgData.data;
    pos = 0;
    r = paleta[colorIdx][0];
    g = paleta[colorIdx][1];
    b = paleta[colorIdx][2];
    for(i=0; i < size; i++ ) {
        pixel[pos + 0] = r;
        pixel[pos + 1] = g;
        pixel[pos + 2] = b;
        pos += 4;
    }
    ctx.putImageData(imgData, 0, 0);
}

// establece los niveles de alfa de la brocha
function ctxSetAlpha(ctx) {
    imgData = ctx.getImageData(0, 0, ctx.canvas.width, ctx.canvas.height);
    size = imgData.width * imgData.height;
    pixel = imgData.data;
```

```
        pos = 0;
        for(i=0; i < size; i++ ) {
            gray = (pixel[pos + 0] * 0.299) + (pixel[pos + 1] * 0.587) + (pixel[pos + 2] * 0.114);
            // gray = Math.floor((pixel[pos + 0] * 0.299) + (pixel[pos + 1] * 0.587) + (pixel[pos + 2] * 0.114));
            // if( gray > 255 ) {
            //   console.log(gray);
            // }
            if( pixel[pos + 3] > 0 ) {
                // pixel[pos + 3] = (255 - gray) >> 1;
                pixel[pos + 3] = 255 - gray;
                }
                pos += 4;
            }
            ctx.putImageData(imgData, 0, 0);
        }

        //paleta de colores
        var paleta =
            [[255,255,255], // white
            [245,181,107],
            [236,166,92],
            [227,151,78],
            [217,136,65],
            [208,123,53],
            [199,108,41],
            [192,95,30],
            [186,92,28],
            [179,87,26],
            [173,84,23],
            [167,77,22],
            [160,72,20],
            [154,69,18],
            [147,63,16],
            [141,59,14],
            [135,56,13],
            [128,52,11],
            [123,47,10],
            [116,44,9],
            [110,41,7],
            [104,38,6],
            [97,33,5],
            [91,31,4],
            [84,28,3],
            [78,24,3],
            [72,21,2],
            [65,19,1],
            [59,17,0],
            [52,14,0],
            [46,12,0],
            [0,0,0], // black
            [246,161,71],
            [237,162,82],
            [229,159,92],
            [220,159,101],
            [202,140,85],
            [186,126,71],
            [170,114,58],
            [154,99,46],
            [138,86,35],
            [123,73,26],
```

```
        [107,62,18],
        [91,50,11],
        [75,41,6],
        [59,31,2],
        [43,22,0],
        [255,214,173],
        [241,196,153],
        [228,181,134],
        [215,163,118],
        [202,148,102],
        [184,137,100],
        [168,128,97],
        [152,118,93],
        [136,108,88],
        [121,97,82],
        [106,87,75],
        [90,75,67],
        [74,63,58],
        [58,51,47],
        [42,37,35],
        [26,24,23],
        [16,16,16],
        [117,19,0],
        [82,7,0],
        [46,0,0],
        [218,255,255],
        [255,198,124],
        [245,181,107],
        [236,166,92],
        [227,151,78],
        [217,136,65],
        [208,123,53],
        [199,108,41],
        [192,95,30],
        [186,92,28],
        [179,87,26],
        [173,84,23],
        [167,77,22],
        [160,72,20],
        [154,69,18],
        [147,63,16],
        [141,59,14],
        [135,56,13],
        [128,52,11],
        [123,47,10],
        [116,44,9],
        [110,41,7],
        [104,38,6],
        [97,33,5],
        [91,31,4],
        [84,28,3],
        [78,24,3],
        [72,21,2],
        [65,19,1],
        [59,17,0],
        [52,14,0],
        [46,12,0],
        [255,165,59],
        [246,161,71],
        [237,162,82],
```

```
        [229,159,92],
        [220,159,101],
        [202,140,85],
        [186,126,71],
        [170,114,58],
        [154,99,46],
        [138,86,35],
        [123,73,26],
        [107,62,18],
        [91,50,11],
        [75,41,6],
        [59,31,2],
        [43,22,0],
        [255,214,173],
        [241,196,153],
        [228,181,134],
        [215,163,118],
        [202,148,102],
        [184,137,100],
        [168,128,97],
        [152,118,93],
        [136,108,88],
        [121,97,82],
        [106,87,75],
        [90,75,67],
        [74,63,58],
        [58,51,47],
        [42,37,35],
        [26,24,23],
        [152,39,0],
        [117,19,0],
        [82,7,0],
        [46,0,0],
        [218,255,255],
        [255,198,124],
        [245,181,107],
        [236,166,92],
        [227,151,78],
        [217,136,65],
        [208,123,53],
        [199,108,41],
        [192,95,30],
        [186,92,28],
        [179,87,26],
        [173,84,23],
        [167,77,22],
        [160,72,20],
        [154,69,18],
        [147,63,16],
        [141,59,14],
        [135,56,13],
        [128,52,11],
        [123,47,10],
        [116,44,9],
        [110,41,7],
        [104,38,6],
        [97,33,5],
        [91,31,4],
        [84,28,3],
        [78,24,3],
```

```
        [72,21,2],
        [65,19,1],
        [59,17,0],
        [52,14,0],
        [46,12,0],
        [255,165,59],
        [246,161,71],
        [237,162,82],
        [229,159,92],
        [220,159,101],
        [202,140,85],
        [186,126,71],
        [170,114,58],
        [154,99,46],
        [138,86,35],
        [123,73,26],
        [107,62,18],
        [91,50,11],
        [75,41,6],
        [59,31,2],
        [43,22,0],
        [255,214,173],
        [241,196,153],
        [228,181,134],
        [215,163,118],
        [202,148,102],
        [184,137,100],
        [168,128,97],
        [152,118,93],
        [136,108,88],
        [121,97,82],
        [106,87,75],
        [90,75,67],
        [74,63,58],
        [58,51,47],
        [42,37,35],
        [26,24,23],
        [152,39,0],
        [117,19,0],
        [82,7,0],
        [46,0,0],
        [218,255,255],
        [255,198,124],
        [245,181,107],
        [236,166,92],
        [227,151,78],
        [217,136,65],
        [208,123,53],
        [199,108,41],
        [192,95,30],
        [186,92,28],
        [179,87,26],
        [173,84,23],
        [167,77,22],
        [160,72,20],
        [154,69,18],
        [147,63,16],
        [141,59,14],
        [135,56,13],
        [128,52,11],
```

```
        [123,47,10],
        [116,44,9],
        [110,41,7],
        [104,38,6],
        [97,33,5],
        [91,31,4],
        [84,28,3],
        [78,24,3],
        [72,21,2],
        [65,19,1],
        [59,17,0],
        [52,14,0],
        [46,12,0],
        [255,165,59],
        [246,161,71],
        [237,162,82],
        [229,159,92],
        [220,159,101],
        [202,140,85],
        [186,126,71],
        [170,114,58],
        [154,99,46],
        [138,86,35],
        [123,73,26],
        [107,62,18],
        [91,50,11],
        [75,41,6],
        [59,31,2],
        [43,22,0],
        [255,214,173],
        [241,196,153],
        [228,181,134],
        [215,163,118],
        [202,148,102]];
```

- **BRUSHES.JS**

```
/*** BRUSHES DIALOG ***/
//brushes la pantalla para definir el tamaño de la brocha
// la posiciones y el numero maximo de botones
BRUSHES_BTN_Y = 32;
BRUSHES_BTN_MAX = 4;
BRUSHES_BTN_LEFT = 0;
BRUSHES_BTN_RIGHT = 1;
BRUSHES_BTN_OK = 2;
BRUSHES_BTN_CANCEL = 3;

var brushes_btn_pos = new Array(BRUSHES_BTN_MAX);
var brushes_btn_label = ['-','+','Ok','Cancel'];

//inicializa la pantalla
function initBrushes() {
    temp=document.getElementById('brushes');
    ctxBrushes=temp.getContext('2d');
    //crear un canvas de toda la pantalla
    ctxBrushes.canvas.width = WIN_WDT;
    ctxBrushes.canvas.height = WIN_HGT;
```

```
    //el tipo de letra para el canvas
    ctxBrushes.font = 'bold 16px Myriad Pro';
    //la linie base del texto
    ctxBrushes.textBaseline = 'top';
    //dibuja la brocha
    drawBrushes();
    //inicializar los botones
    initBrushesButtons();
    //los lintener para los diferentes eventos
    ctxBrushes.canvas.addEventListener('mousedown', brushes_mousedown, false);
    ctxBrushes.canvas.addEventListener('mousemove', brushes_mousemove, false);
    ctxBrushes.canvas.addEventListener('mouseup', brushes_mouseup, false);
    ctxBrushes.canvas.addEventListener('touchstart', brushes_touchstart, false);
    ctxBrushes.canvas.addEventListener('touchmove', brushes_touchmove, false);
    ctxBrushes.canvas.addEventListener('touchend', brushes_touchend, false);
    ctxBrushes.canvas.addEventListener('touchcancel', brushes_touchend, false);
}

//inicializa  los botones en las posiciones XY
function initBrushesButtons() {
    brushes_btn_pos[0] = [200,BRUSHES_BTN_Y];
    brushes_btn_pos[1] = [340,BRUSHES_BTN_Y];
    brushes_btn_pos[2] = [600,BRUSHES_BTN_Y];
    brushes_btn_pos[3] = [720,BRUSHES_BTN_Y];
    drawButton(ctxBrushes, brushes_btn_pos[BRUSHES_BTN_LEFT][0], brushes_btn_pos[BRUSHES_
    BTN_LEFT][1], brushes_btn_label[BRUSHES_BTN_LEFT], false);
    drawButton(ctxBrushes, brushes_btn_pos[BRUSHES_BTN_RIGHT][0], brushes_btn_pos[BRUSHES_
    BTN_LEFT][1], brushes_btn_label[BRUSHES_BTN_RIGHT], false);
    drawButton(ctxBrushes, brushes_btn_pos[BRUSHES_BTN_OK][0], brushes_btn_pos[BRUSHES_BTN_
    LEFT][1], brushes_btn_label[BRUSHES_BTN_OK], false);
    drawButton(ctxBrushes, brushes_btn_pos[BRUSHES_BTN_CANCEL][0], brushes_btn_pos[BRUSHES_
    BTN_LEFT][1], brushes_btn_label[BRUSHES_BTN_CANCEL], false);
}

//pinta el texto de color blanco
function drawBrushes() {
    ctxBrushes.fillStyle = '#FFFFFF';
    ctxBrushes.fillText('Brush Size', 120, 38);
    drawWidth();
}

//tamaño de la brocha
function drawWidth() {
    ctxBrushes.fillStyle = '#404040';
    ctxBrushes.fillRect(288, 38, 48, 20);
    ctxBrushes.fillStyle = '#FFFFFF';
    ctxBrushes.fillText(lineTmp+' px', 288, 38);
}

//evento mouse
function brushes_mousedown(event) {
  brushes_pressed(event.pageX, event.pageY);
}

// evento touch
function brushes_touchstart(event) {
    event.preventDefault();
    var touch = event.changedTouches[0];
    brushes_pressed(touch.pageX, touch.pageY);
}
```

```
//evento manejo de teclas
function brushes_pressed(x,y) {
    if(x || y) {
        mousedown = true;
        // if( inside(x, y, BRUSHES_X, BRUSHES_Y, BRUSHES_WDT, BRUSHES_HGT) ) {
        //  selectColor(x,y);
        // } else {
            buttonIdx = -1;
            // que botón ha sido presionado 0,1,2,3
            for(i=0; i < BRUSHES_BTN_MAX; i++) {
                if( inside(x, y, brushes_btn_pos[i][0], brushes_btn_pos[i][1], BUTTON_WDT,
                BUTTON_HGT) ) {
                    buttonIdx = i;
                    break;
                }
            }
            if( buttonIdx > -1 ) {
                drawButton(ctxBrushes, brushes_btn_pos[buttonIdx][0], brushes_btn_pos[but-
                tonIdx][1], brushes_btn_label[buttonIdx], true);
            }
        // }
    }
}

// evento mousemove
function brushes_mousemove(event) {
  brushes_dragged(event.pageX, event.pageY);
}

//evento touchmove
function brushes_touchmove(event) {
    event.preventDefault();
    var touch = event.changedTouches[0];
    brushes_dragged(touch.pageX, touch.pageY);
}

// evento dragged
function brushes_dragged(x,y) {
}

// evento mouseup
function brushes_mouseup(event) {
    brushes_released(0,0);
}

//Evento dejo de tocar la pantalla
function brushes_touchend(event) {
    brushes_released(0,0);
}

// Evento cuando existe inactividad
function brushes_released(x,y) {
    mousedown = false;
    if( buttonIdx > -1 ) {
        drawButton(ctxBrushes, brushes_btn_pos[buttonIdx][0], brushes_btn_pos[buttonIdx][1],
        brushes_btn_label[buttonIdx], false);
        brushes_action(buttonIdx);
    }
}
```

```
//Manejo de acciónes de cada boton
function brushes_action(idx) {
    switch(idx) {
        case BRUSHES_BTN_LEFT:
            if( lineTmp > 2 ) {
                lineTmp -= 2;
                drawWidth();
            }
            break;
        case BRUSHES_BTN_RIGHT:
            if( lineTmp < 32 ) {
                lineTmp += 2;
                drawWidth();
            }
            break;
        case BRUSHES_BTN_OK:
            lineWidth = lineTmp;
            radius = Math.floor(lineWidth /2);
            document.getElementById('brushes').style.display = 'none';
        break;
        case BRUSHES_BTN_CANCEL:
            document.getElementById('brushes').style.display = 'none';
        break;
    }
}
```

CANVAS.JS

```
/*** CANVAS ***/
//canvas es la pantalla de dibujo
var img_ball = new Image();
img_ball.onload = initBrush;
//img_ball.src = 'gfx/commodore64.png';
//img_ball.src = 'images/brushes/ball.png';
img_ball.src = 'images/brushes/brush2.png';
ofsX = 0;
ofsY = 0;
//inicializa la pantalla donde dibujar o area de dibujo
function initCanvas() {
    temp=document.getElementById('canvas');
    ctxCanvas=temp.getContext('2d');
    ctxCanvas.canvas.width = WIN_WDT;
    ctxCanvas.canvas.height = WIN_HGT - TOOLBAR_HGT;
    ctxCanvas.canvas.style.background = '#FFFFFF';
    ctxCanvas.canvas.addEventListener('mousedown', canvas_mousedown, false);
    ctxCanvas.canvas.addEventListener('mousemove', canvas_mousemove, false);
    ctxCanvas.canvas.addEventListener('mouseup', canvas_mouseup, false);
    ctxCanvas.canvas.addEventListener('touchstart', canvas_touchstart, false);
    ctxCanvas.canvas.addEventListener("touchmove", canvas_touchmove, false);
    ctxCanvas.canvas.addEventListener('touchend', canvas_touchend, false);
    ctxCanvas.canvas.addEventListener('touchcancel', canvas_touchend, false);
    ctxCanvas.textBaseline = 'top';
    ctxCanvas.fillStyle = '#000';


}
```

```
//inicializa la pantalla
function initBrush() {
    brush = document.createElement('canvas');
    ctxBrush = brush.getContext('2d');
    ctxBrush.canvas.width = img_ball.width;
    ctxBrush.canvas.height = img_ball.height;
    ctxBrush.drawImage(img_ball, 0, 0);
    ctxBrush.transform(2.0, 0, 0, 2.0, 0, 0);
    ctxSetAlpha(ctxBrush);
    ctxSetColor(ctxBrush);
    ofsX = -(ctxBrush.canvas.width >> 1);
    ofsY = -(ctxBrush.canvas.height >> 1);
}

function canvas_mousedown(event) {
  canvas_pressed(event.pageX, event.pageY);
}

function canvas_touchstart(event) {
    event.preventDefault();
    var touch = event.changedTouches[0];
    canvas_pressed(touch.pageX, touch.pageY);
}
// dibujar el punto incial
function canvas_pressed(x,y) {
    if(x || y) {
        mousedown = true;
        currX = x;
        currY = y - TOOLBAR_HGT;
        lastX = currX;
        lastY = currY;
        drawPixel(currX, currY);
    }
}

function canvas_mousemove(event) {
    if( mousedown ) {
      canvas_dragged(event.pageX, event.pageY);
    }
}
//evento arrastre del mouse
function canvas_touchmove(event) {
    if( mousedown ) {
        event.preventDefault();
      var touch = event.changedTouches[0];
      canvas_dragged(touch.pageX, touch.pageY);
    }
}

function canvas_dragged(x,y) {
    if(x || y) {
        currX = x;
        currY = y - TOOLBAR_HGT;
        drawLine();
        lastX = currX;
        lastY = currY;
    }
}
```

```
function canvas_mouseup(event) {
    canvas_release(0,0);
}

function canvas_touchend(event) {
    canvas_release(0,0);
}
function canvas_release(x,y) {
    mousedown = false;
}
function drawPixel(x,y) {
    ctxCanvas.drawImage(brush, x + ofsX, y + ofsY);
}
//dibuja una linea desde el punto actual hasta el incial
function drawLine() {
    lenX = currX - lastX;
    lenY = currY - lastY;
    vx = 0;
    if(lenX > 0) {
        vx = 1;
    } else if(lenX < 0) {
        vx = -1;
    }
    vy = 0;
    if(lenY > 0) {
        vy = 1;
    } else if(lenY < 0) {
        vy = -1;
    }
    len = 0;
    lenX = Math.abs(lenX);
    lenY = Math.abs(lenY);
    if(lenX > lenY) {
        vy = (lenY / lenX) * vy;
        len = lenX;
    } else {
        vx = (lenX / lenY) * vx;
        len = lenY;
    }
    x = lastX;
    y = lastY;
    for(i=0; i < len; i++) {
        drawPixel(Math.floor(x),Math.floor(y));
        x += vx;
        y += vy;
    }
}
```

- **PALETTE.JS**

```
/*** PALETTE DIALOG ***/

//paleta de colores

COLOR_WDT = 30;
COLOR_HGT = 40;
PALETTE_WDT = COLOR_WDT*32;
PALETTE_HGT = COLOR_HGT*8;
PALETTE_X = (WIN_WDT - PALETTE_WDT) >> 1;
PALETTE_Y = 248;
```

```
SLIDER_WDT = 254;
SLIDER_HGT = COLOR_HGT - 2;

PALETTE_BTN_Y = 32;
PALETTE_BTN_MAX = 2;
PALETTE_BTN_SPACING = 0;
PALETTE_BTN_OK = 0;
PALETTE_BTN_CANCEL = 1;

var palette_btn_pos = new Array(TOOLBAR_BTN_MAX);
var palette_btn_label = ['Ok','Cancel'];
var palX;
var palY;
var palLx;
var palLy;
// crear canvas o la pantalla de la paleta
function initPalette() {
    temp=document.getElementById('palette');
    ctxPalette=temp.getContext('2d');
    ctxPalette.canvas.width = WIN_WDT;
    ctxPalette.canvas.height = WIN_HGT;
    ctxPalette.font = 'bold 16px Myriad Pro';
    ctxPalette.textBaseline = 'top';
    drawPalette();
    initPaletteButtons();

    ctxPalette.canvas.addEventListener('mousedown', palette_mousedown, false);
    ctxPalette.canvas.addEventListener('mousemove', palette_mousemove, false);
    ctxPalette.canvas.addEventListener('mouseup', palette_mouseup, false);
    ctxPalette.canvas.addEventListener('touchstart', palette_touchstart, false);
    ctxPalette.canvas.addEventListener('touchmove', palette_touchmove, false);
    ctxPalette.canvas.addEventListener('touchend', palette_touchend, false);
    ctxPalette.canvas.addEventListener('touchcancel', palette_touchend, false);
}
// posicionar los botones
function initPaletteButtons() {

    palette_btn_pos[0] = [600,PALETTE_BTN_Y];
    palette_btn_pos[1] = [720,PALETTE_BTN_Y];
    //dibujar los botones
    drawButton(ctxPalette, palette_btn_pos[PALETTE_BTN_OK][0], palette_btn_pos[PALETTE_BTN_
    OK][1],palette_btn_label[PALETTE_BTN_OK], false);
    drawButton(ctxPalette, palette_btn_pos[PALETTE_BTN_CANCEL][0], palette_btn_pos[PAL-
    ETTE_BTN_OK][1],palette_btn_label[PALETTE_BTN_CANCEL], false);
}
//dibuja un cuadrado del color principal
function drawColor(index) {
    ctxPalette.fillStyle = 'rgb('+paleta[index]+')';
    ctxPalette.fillRect(320,32,128,128);
}
//dibuja un cuadrado del color secundario o fondo
function drawBgColor(index) {
    ctxPalette.fillStyle = 'rgb('+paleta[index]+')';
    ctxPalette.fillRect(368,80,128,128);
}
//dibuja la paleta de colores
function drawPalette() {
    // ctxPalette.lineWidth = 1;
    // ctxPalette.strokeStyle = '#FFFFFF';
    // ctxPalette.strokeRect(32, 32, SLIDER_WDT, SLIDER_HGT);
```

```
// ctxPalette.strokeRect(32, 80, SLIDER_WDT, SLIDER_HGT);
// ctxPalette.strokeRect(32, 128, SLIDER_WDT, SLIDER_HGT);
// ctxPalette.strokeRect(32, 176, SLIDER_WDT, SLIDER_HGT);
pal = 0;
y = PALETTE_Y;
for( j=0; j<8; j++ ) {
    x = PALETTE_X;
    for( i=0; i<32; i++ ) {
        ctxPalette.fillStyle = 'rgb('+paleta[pal]+')';
        ctxPalette.fillRect(x, y, COLOR_WDT, COLOR_HGT);
        x += COLOR_WDT;
        pal++;
    }
    y += COLOR_HGT;
}
drawBgColor(bgcolorTmp);
drawColor(colorTmp);
}

function palette_mousedown(event) {
  palette_pressed(event.pageX, event.pageY);
}

function palette_touchstart(event) {
    event.preventDefault();
    var touch = event.changedTouches[0];
    palette_pressed(touch.pageX, touch.pageY);
}
// Evento de manejo de botones
function palette_pressed(x,y) {
    if(x || y) {
        mousedown = true;
        buttonIdx = -1;
        if( inside(x, y, PALETTE_X, PALETTE_Y, PALETTE_WDT, PALETTE_HGT) ) {
            selectColor(x,y);
        } else if( inside(x, y, 368, 160, 128, 48) || inside(x, y, 448, 80, 48, 80) ) {
            swapColors();
        } else {
            for(i=0; i < PALETTE_BTN_MAX; i++) {
                if( inside(x, y, palette_btn_pos[i][0], palette_btn_pos[i][1], BUTTON_WDT,
BUTTON_HGT) ) {
                    buttonIdx = i;
                    break;
                }
            }
            if( buttonIdx > -1 ) {
                drawButton(ctxPalette, palette_btn_pos[buttonIdx][0], palette_btn_pos[but-
tonIdx][1], palette_btn_label[buttonIdx], true);
            }
        }
    }
}

function palette_mousemove(event) {
  palette_dragged(event.pageX, event.pageY);
}

function palette_touchmove(event) {
    event.preventDefault();
    var touch = event.changedTouches[0];
```

```
        palette_dragged(touch.pageX, touch.pageY);
    }

    function palette_dragged(x,y) {
        if(x || y) {
            if( mousedown && inside(x, y, PALETTE_X, PALETTE_Y, PALETTE_WDT, PALETTE_HGT) ) {
                selectColor(x,y);
            }
        }
    }
    //cambia el color del cuadrado principal al presionar la paleta
    function selectColor(x,y) {
        x = Math.floor((x - PALETTE_X) / COLOR_WDT);
        y = Math.floor((y - PALETTE_Y) / COLOR_HGT);
        colorTmp = (y << 5) | x;
        drawColor(colorTmp);
    }
    //intercambio del color principal y secundario
    function swapColors() {
        temp = colorTmp;
        colorTmp = bgcolorTmp;
        bgcolorTmp = temp;
        drawBgColor(bgcolorTmp);
        drawColor(colorTmp);
    }

    function palette_mouseup(event) {
        palette_released(0,0);
    }

    function palette_touchend(event) {
        palette_released(0,0);
    }
    // Evento de inactividad
    function palette_released(x,y) {
        mousedown = false;
        if( buttonIdx > -1 ) {
            drawButton(ctxPalette, palette_btn_pos[buttonIdx][0], palette_btn_pos[buttonIdx]
    [1], palette_btn_label[buttonIdx], false);
            palette_action(buttonIdx);
        }
    }
    //acciones de los botones
    function palette_action(idx) {
        switch(idx) {
            case PALETTE_BTN_OK:
                colorIdx = colorTmp;
                bgcolorIdx = bgcolorTmp;
                ctxSetColor(ctxBrush);
                document.getElementById('palette').style.display = 'none';
            break;
            case PALETTE_BTN_CANCEL:
                drawBgColor(bgcolorIdx);
                drawColor(colorIdx);
                document.getElementById('palette').style.display = 'none';
            break;
        }
    }
```

TOOLBAR.JS

```
/*** TOOLBAR ***/
//la barra superior de botones

TOOLBAR_BTN_Y = 4;
TOOLBAR_BTN_MAX = 7;
TOOLBAR_BTN_SPACING = 0;
TOOLBAR_BTN_NEW = 0;
TOOLBAR_BTN_LOAD = 1;
TOOLBAR_BTN_SAVE = 2;
TOOLBAR_BTN_SHARE = 3
TOOLBAR_BTN_BRUSHES = 4;
TOOLBAR_BTN_COLORS = 5;
TOOLBAR_BTN_HELP = 6;

var toolbar_btn_posx = new Array(TOOLBAR_BTN_MAX);
var toolbar_btn_label = ['Nuevo','Abrir','Guardar','Redes','Paleta','Colores','Ayuda'];

function initToolbar() {
    temp=document.getElementById('toolbar');
    ctxToolbar=temp.getContext('2d');
    ctxToolbar.canvas.width = WIN_WDT;
    ctxToolbar.canvas.height = TOOLBAR_HGT;
    ctxToolbar.canvas.style.background = '#404040';
    ctxToolbar.font = 'bold 16px Myriad Pro';
    ctxToolbar.textBaseline = 'top';
    ctxToolbar.fillStyle = '#000000';
    ctxToolbar.fillRect(0, TOOLBAR_HGT-1, WIN_WDT, TOOLBAR_HGT-1);
    initToolbarButtons();

    ctxToolbar.canvas.addEventListener('mousedown', toolbar_mousedown, false);
    ctxToolbar.canvas.addEventListener('mousemove', toolbar_mousemove, false);
    ctxToolbar.canvas.addEventListener('mouseup', toolbar_mouseup, false);
    ctxToolbar.canvas.addEventListener('touchstart', toolbar_touchstart, false);
    ctxToolbar.canvas.addEventListener("touchmove", toolbar_touchmove, false);
    ctxToolbar.canvas.addEventListener('touchend', toolbar_touchend, false);
}
//inicializa los botones de la barra principal
function initToolbarButtons() {
    free = WIN_WDT - (BUTTON_WDT * TOOLBAR_BTN_MAX);
    margin = (free % (TOOLBAR_BTN_MAX + 1)) >> 1;
    TOOLBAR_BTN_SPACING = Math.floor( free / (TOOLBAR_BTN_MAX + 1) );
    pos = margin + TOOLBAR_BTN_SPACING;
    for(i=0; i<TOOLBAR_BTN_MAX; i++) {
        toolbar_btn_posx[i] = pos;
        pos += (BUTTON_WDT + TOOLBAR_BTN_SPACING);
    }
    drawButton(ctxToolbar, toolbar_btn_posx[TOOLBAR_BTN_NEW], TOOLBAR_BTN_Y, toolbar_btn_
    label[TOOLBAR_BTN_NEW], false);
    drawButton(ctxToolbar, toolbar_btn_posx[TOOLBAR_BTN_LOAD], TOOLBAR_BTN_Y, toolbar_btn_
    label[TOOLBAR_BTN_LOAD], false);
    drawButton(ctxToolbar, toolbar_btn_posx[TOOLBAR_BTN_SAVE], TOOLBAR_BTN_Y, toolbar_btn_
    label[TOOLBAR_BTN_SAVE], false);
    drawButton(ctxToolbar, toolbar_btn_posx[TOOLBAR_BTN_SHARE], TOOLBAR_BTN_Y, toolbar_btn_
    label[TOOLBAR_BTN_SHARE], false);
    drawButton(ctxToolbar, toolbar_btn_posx[TOOLBAR_BTN_BRUSHES], TOOLBAR_BTN_Y, toolbar_
    btn_label[TOOLBAR_BTN_BRUSHES], false);
```

```
    drawButton(ctxToolbar, toolbar_btn_posx[TOOLBAR_BTN_COLORS], TOOLBAR_BTN_Y, toolbar_
    btn_label[TOOLBAR_BTN_COLORS], false);
    drawButton(ctxToolbar, toolbar_btn_posx[TOOLBAR_BTN_HELP], TOOLBAR_BTN_Y, toolbar_btn_
    label[TOOLBAR_BTN_HELP], false);
}

function toolbar_mousedown(event) {
    toolbar_pressed(event.pageX, event.pageY);
}

function toolbar_touchstart(event) {
    event.preventDefault();
    var touch = event.changedTouches[0];
    toolbar_pressed(touch.pageX, touch.pageY);
}

function toolbar_pressed(x,y) {
    mousedown = true;
    buttonIdx = -1;
    for(i=0; i < TOOLBAR_BTN_MAX; i++) {
        if( inside(x, y, toolbar_btn_posx[i], TOOLBAR_BTN_Y, BUTTON_WDT, BUTTON_HGT) ) {
            buttonIdx = i;
            break;
        }
    }
    if( buttonIdx > -1 ) {
        drawButton(ctxToolbar, toolbar_btn_posx[buttonIdx], TOOLBAR_BTN_Y, toolbar_btn_
        label[buttonIdx], true);
    }
}

function toolbar_mousemove(event) {
    toolbar_dragged(event.pageX, event.pageY);
}

function toolbar_touchmove(event) {
    if( mousedown ) {
        event.preventDefault();
        var touch = event.changedTouches[0];
        toolbar_dragged(touch.pageX, touch.pageY);
    }
}

function toolbar_dragged(x,y) {

}

function toolbar_mouseup(event) {
    toolbar_released(0,0);
}

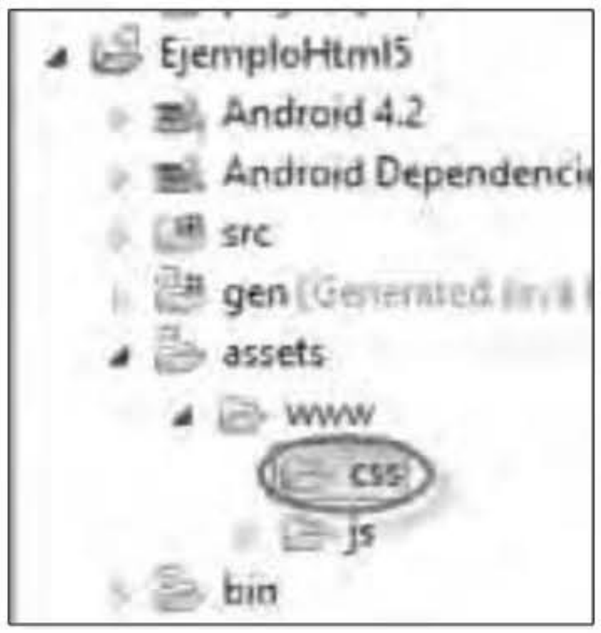
function toolbar_touchend(event) {
    toolbar_released(0,0);
}

function toolbar_released(x,y) {
    mousedown = false;
    if( buttonIdx > -1 ) {
        drawButton(ctxToolbar, toolbar_btn_posx[buttonIdx], TOOLBAR_BTN_Y, toolbar_btn_
        label[buttonIdx], false);
```

```
            toolbar_action(buttonIdx);
        }
    }

    function toolbar_action(idx) {
        switch(idx) {
            case TOOLBAR_BTN_NEW:
                ctxCanvas.fillStyle = 'rgb('+paleta[bgcolorIdx]+')';
                ctxCanvas.fillRect( 0, 0, ctxCanvas.canvas.width, ctxCanvas.canvas.height );
                break;
            case TOOLBAR_BTN_SHARE:
                var strDataURI = ctxCanvas.canvas.toDataURL('image/jpeg');
                console.log(strDataURI);
                break;
            case TOOLBAR_BTN_BRUSHES:
                lineTmp = lineWidth;
                document.getElementById('brushes').style.display = 'block';
                break;
            case TOOLBAR_BTN_COLORS:
                colorTmp = colorIdx;
                bgcolorTmp = bgcolorIdx;
                document.getElementById('palette').style.display = 'block';
                break;
        }
    }
```

Paso 5: Seleccione la hoja de estilo a utilizar. Para ello, cree la siguiente carpeta:



- **STYLES.CSS**

```
body {
    width: 100%;
    height: 100%;
    margin: 0px;
    background-color: #000000;
}

/*#toolbar {
    background-color: #404040;
}

#canvas {
    background-color: #FFFFFF;
}*/

#palette, #brushes {
    background-color: #404040;
    position: absolute;
    left: 0px;
    top: 0px;
    display: none;
}

table, tr, td {
    margin: 0px;
    padding: 0px;
    border: 0px;
    border-spacing: 0px;
}

canvas {
    margin: 0px;
    padding: 0px;
    border: 0px;
    border-spacing: 0px;
    display: block;
    /*opacity: 0.5;
    /*-moz-opacity: 0.5;
    filter:alpha(opacity=5);*/
}
```

La pagina principal index.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <title>Pintura</title>
        <link href="css/styles.css" rel="stylesheet" type="text/css" />
        <script type="application/javascript" src="js/main.js"></script>
        <script type="application/javascript" src="js/toolbar.js"></script>
        <script type="application/javascript" src="js/canvas.js"></script>
        <script type="application/javascript" src="js/palette.js"></script>
        <script type="application/javascript" src="js/brushes.js"></script>
    </head>
```

```
        <body>
            <div>
                <canvas id="toolbar" />
            </div>
            <div>
                <canvas id="canvas" />
            </div>
            <div>
                <canvas id="palette" />
            </div>
            <div>
                <canvas id="brushes" />
            </div>
        </body>
    </html>
```

EJECUTANDO LA APLICACIÓN

Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha.

- *Pantalla principal*

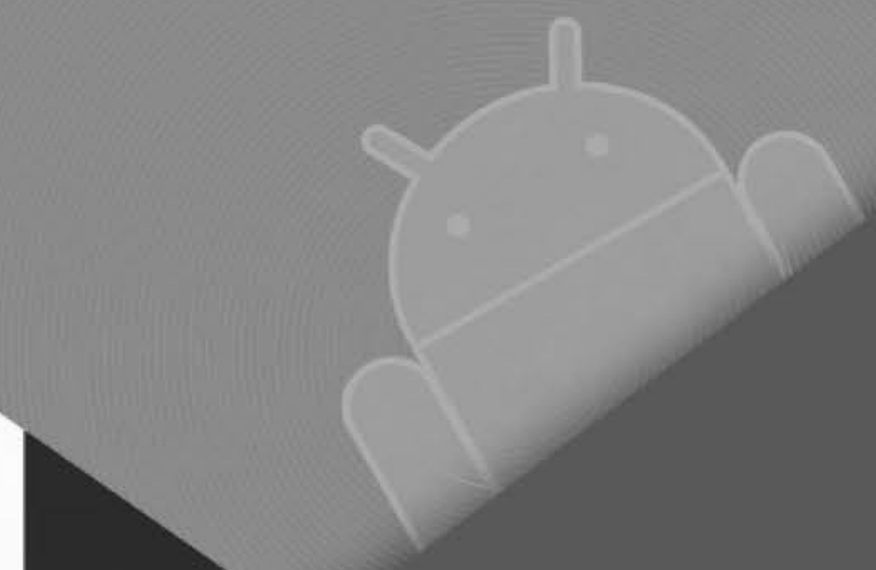


- *Manejo de la paleta*



- *Manejo de colores*





CAPÍTULO

8

Google Maps Android API V3

8.1 Introducción

Google Maps fue desarrollado originalmente por los hermanos daneses, Lars y Jens Rasmussen, cofundadores de Where 2 Technologies, una empresa dedicada a la creación de soluciones de mapeo. La empresa fue adquirida por Google en octubre de 2004 y los dos hermanos posteriormente crearon Google Maps (también son los que están detrás de Google Wave).

Antes de que hubiera una API pública, algunos desarrolladores descubrieron la manera de hackear Google Maps para incorporar los mapas en sus propios sitios web. Esto llevó a Google a la conclusión de que había una necesidad de una API pública y en junio de 2005 fue lanzado públicamente Google Maps.

El mashup por primera vez en Internet es, a menudo, considerado que lo ejerció Housingmaps.com, una combinación de Google Maps con los listados de bienes raíces de Craiglist.org. Fue creado de hecho antes de que la API pública fuera puesta en libertad y fue hackeado por el desarrollador Paul Rademacher. En mayo de 2010, se anunció la versión 3 del API. Ahora es la opción recomendada para las nuevas aplicaciones de Google Maps y el siguiente paso en la historia de Google Maps.

Versión 3 del API de JavaScript de Google Maps

Cómo declarar tu aplicación como HTML5

Te recomendamos que declares un DOCTYPE verdadero en tu aplicación web. En los ejemplos que te mostramos aquí, hemos declarado nuestras aplicaciones como HTML5 mediante el sencillo DOCTYPE HTML5, tal y como se muestra a continuación:

```
<!DOCTYPE html>
```

Los navegadores más habituales mostrarán contenido declarado con este DOCTYPE en "modo estándar", lo que significa que tu aplicación deberá ser más compatible con los navegadores. El DOCTYPE también está diseñado para ajustarse de la mejor manera; así, los navegadores que no lo entiendan lo ignorarán y utilizarán el "modo chapucero" para mostrar el contenido.

Ten en cuenta que algunas de las CSS que funcionan en el modo chapucero no son válidas en el modo estándar. En concreto, todos los tamaños basados en porcentajes deben heredarse de los elementos de bloque principales, y si cualquiera de estos antecesores no puede especificar un tamaño, se supondrá un tamaño de 0 x 0 píxeles. Por esta razón, incluimos la siguiente declaración <style>:

```
<style type="text/css">
    html { height: 100% }
    body { height: 100%; margin: 0; padding: 0 }
    #map_canvas { height: 100% }
</style>
```

Esta declaración de CSS indica que el contenedor del mapa (denominado map_canvas) debe ocupar el 100% de la altura del cuerpo de HTML. Ten en cuenta que debemos declarar de forma específica estos porcentajes tanto para <body> como para <html>.

Cómo cargar el API de Google Maps

```
<html>
    <head>
        <script type="text/javascript"
            src="http://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&sensor=SET_
            TO_TRUE_OR_FALSE">
  </script>
```

La URL incluida en la etiqueta script indica la ubicación de un archivo JavaScript que carga todos los símbolos y las definiciones que necesitas para utilizar el API de Google Maps. La etiqueta script es obligatoria.

El parámetro key incluye la clave de API de tu aplicación. Para obtener más información, consulta la sección Cómo obtener una clave de API. Ten en cuenta que esta clave no es la misma clave que se utiliza en la versión 2 del API y que se debe generar a partir de la consola de las API.

El parámetro sensor de la URL es obligatorio e indica si esta aplicación utiliza un sensor (por ejemplo, un localizador GPS) para determinar la ubicación del usuario. En este ejemplo hemos dejado el parámetro como la variable set_to_true_or_false para hacer hincapié en que se debe definir este valor en true o false de forma explícita.

Los usuarios del API de Google Maps for Business deben consultar la sección Cómo cargar el API de JavaScript de Google Maps de la documentación para empresas para obtener información importante sobre cómo cargar el API de Google Maps en sus aplicaciones.

HTTPS

Si tu aplicación es una aplicación HTTPS, es posible que quieras cargar el API de JavaScript de Google Maps a través de HTTPS:

```
<script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&sensor=SET_TO_
TRUE_OR_FALSE"
 type="text/javascript"></script>
```

El uso del protocolo HTTPS para cargar el API de Google Maps es necesario en las aplicaciones SSL para evitar que aparezcan advertencias de seguridad en la mayoría de los navegadores, y se recomienda para las aplicaciones que incluyen datos importantes de los usuarios en las solicitudes (por ejemplo, su ubicación).

Bibliotecas

Al cargar el API de JavaScript de Google Maps mediante la URL http://maps.googleapis.com/maps/api/js, tienes la opción de cargar bibliotecas adicionales utilizando el parámetro libraries. Las bibliotecas son módulos de código que proporcionan funciones adicionales al API de JavaScript, pero que no se cargan a menos que las solicites de forma específica. Para obtener más información, consulta la sección Bibliotecas en la versión 3 del API de Google Maps.

Cómo cargar el API de forma asíncrona

Te recomendamos que cargues el código JavaScript del API de Google Maps después de que la página haya terminado de cargarse o en el momento. Para ello, puedes insertar tu propia etiqueta <script> en respuesta a un evento window.onload o a una llamada de función. No obstante, también debes indicar a la secuencia de comandos de inicialización del API de JavaScript de Google Maps que retrase la ejecución del código de tu aplicación hasta que el código del API de JavaScript de Google Maps se haya cargado por completo. Para ello, puedes utilizar el parámetro callback, que utiliza como argumento la función que se debe ejecutar cuando finaliza la carga del API.

A continuación se muestra un fragmento de código que permite que la aplicación cargue el API de Google Maps una vez que se haya cargado por completo la página (mediante window.onload) y que se incluya el API de JavaScript de Google Maps en una etiqueta <script> dentro de la página. También se puede hacer que el API no ejecute la función initialize() hasta que se haya cargado por completo incluyendo callback=initialize en la secuencia de comandos de inicialización del API de Google Maps:

```
function initialize() {
 var mapOptions = {
  zoom: 8,
  center: new google.maps.LatLng(-34.397, 150.644),
  mapTypeId: google.maps.MapTypeId.ROADMAP
 }
 var map = new google.maps.Map(document.getElementById("map_canvas"), mapOptions);
}

function loadScript() {
 var script = document.createElement("script");
 script.type = "text/javascript";
 script.src = "http://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&sensor=TRUE_OR_
FALSE&callback=initialize";
 document.body.appendChild(script);
}

window.onload = loadScript;
```

Elementos DOM de mapas

```
<div id="map_canvas" style="width: 100%; height: 100%"></div>
```

Para que el mapa aparezca en una página web, se debe reservar un lugar para él. Normalmente, lo hacemos mediante la creación de un elemento div con nombre y la obtención de una referencia a este elemento en el modelo de objetos de documento (DOM) del navegador.

En el ejemplo anterior, definimos un objeto denominado "map_canvas" y establecemos su tamaño mediante atributos de estilo. Ten en cuenta que este tamaño se establece en "100%", lo que amplía el mapa hasta que ocupa toda la pantalla del dispositivo móvil. Es posible que sea necesario ajustar estos valores según el desplazamiento y el tamaño de la pantalla del navegador. Ten en cuenta que el mapa siempre adoptará el tamaño del elemento en el que esté contenido, por lo que siempre debes establecer un tamaño para el elemento de forma explícita.

Opciones de mapas

```
var mapOptions = {
 center: new google.maps.LatLng(-34.397, 150.644),
 zoom: 8,
 mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

Antes de inicializar un mapa, debemos crear un objeto Map options que contenga las variables de inicialización correspondientes. Este objeto no se construye, sino que se crea como un objeto literal.

```
var mapOptions = {};
```

Latitudes y longitudes

Como queremos center el mapa en un punto específico, creamos un objeto LatLng para mantener esta ubicación especificando las coordenadas de ubicación en el orden {latitud, longitud}:

```
center = new google.maps.LatLng(-34.397, 150.644)
```

El proceso de convertir una dirección en un punto geográfico se conoce como codificación geográfica. Esta versión del API de Google Maps incluye funciones de codificación geográfica. Para obtener más información, consulta la sección Codificación geográfica del capítulo Servicios de esta guía.

Niveles de zoom

La propiedad zoom especifica la resolución inicial con la que se mostrará un mapa, donde un zoom de 0 se corresponde con un mapa de la Tierra totalmente alejado y los niveles de zoom acercan el mapa con una resolución más elevada.

```
zoom: 8
```

Para ofrecer un mapa de todo el planeta como una única imagen, es necesario un mapa muy grande o un mapa pequeño con una resolución muy baja. Por consiguiente, las imágenes de mapa en Google Maps y el API de Google Maps se dividen en "mosaicos" de mapas y "niveles de zoom". A niveles bajos de zoom, un conjunto pequeño de mosaicos de mapas cubre una superficie amplia; a niveles de zoom más elevados, los mosaicos tienen una resolución mayor y cubren una superficie más pequeña [...].

Tipos de mapas

También debes establecer expresamente un tipo de mapa inicial en este momento.

```
mapTypeId: google.maps.MapTypeId.ROADMAP
```

Se admiten los siguientes tipos de mapas:

- *ROADMAP, que muestra los mosaicos normales en 2D predeterminados de Google Maps.*
- *SATELLITE muestra imágenes de satélite.*
- *HYBRID muestra una mezcla de mosaicos fotográficos y una capa de mosaicos para los elementos del mapa más destacados (carreteras, nombres de ciudades, etc.).*
- *TERRAIN muestra mosaicos de relieve físico para indicar las elevaciones del terreno y las fuentes de agua (montañas, ríos, etc.).*

Para obtener más información sobre los tipos de mapas, consulta la sección Tipos de mapas. Sin embargo, para la mayoría de los casos lo único que necesitas saber es cómo utilizar los tipos básicos descritos anteriormente.

El objeto "Map"

```
var map = new google.maps.Map(document.getElementById("map_canvas"),
  mapOptions);
```

La clase de JavaScript que representa a los mapas es Map. Cada objeto de esta clase define un único mapa en una página. (Puedes crear más de una instancia de esta clase; cada objeto definirá un mapa independiente en la página). Creamos una nueva instancia de esta clase mediante el operador new de JavaScript.

Al crear una nueva instancia de mapa, se especifica un elemento HTML en la página como contenedor para el mapa. Los nodos HTML son elementos secundarios del objeto document de JavaScript. Se obtiene una referencia a este elemento mediante el método document. getElementById().

Este código permite definir una variable (denominada map) y asignar dicha variable a un nuevo objeto Map, además de transmitir opciones definidas en el objeto mapOptions literal. Estas opciones se utilizarán para inicializar las propiedades del mapa. A continuación se muestra la definición de la función Map(), conocida como constructor:

Constructor	Descripción
Map(mapDiv:Node, opts?:MapOptions)	Crea un mapa nuevo dentro del contenedor HTML en cuestión, que suele ser un elemento DIV, mediante los parámetros (opcional) que se especifiquen.

Cómo cargar el mapa

```
<body onload="initialize()">
```

Mientras se procesa una página HTML, se crea el modelo de objetos de documentos (DOM) y las imágenes y secuencias de comandos externas se reciben y se incorporan al objeto document. Para garantizar que nuestro mapa se añada a la página cuando se cargue por completo, solo ejecutamos la función que crea el objeto Map cuando el elemento <body> de la página HTML ha recibido un evento onload. De este modo, evitamos un comportamiento impredecible y obtenemos más control acerca del modo y del momento en que se dibuja el mapa.

El atributo onload de la etiqueta body es un ejemplo de un controlador de eventos. El API de JavaScript de Google Maps también proporciona varios eventos que se pueden controlar para determinar cambios de estado. Para obtener más información, consulta la sección Eventos de mapa».

Fuente: developers.google.com

¿Cómo funciona Google Maps?

Es solo (sic) HTML, CSS y JavaScript trabajando juntos. Los mapas son solo imágenes que se cargan en el fondo a través de peticiones ejecutadas por la tecnología de AJAX, y se insertan en un en la página HTML. Mientras navegas en el mapa, el API envía información acerca de las nuevas coordenadas y los niveles de "zoom" de el mapa a través de AJAX y esto retorna las imágenes.

El API consiste de archivos JavaScript que contienen las clases, métodos y propiedades que se usan para el comportamiento de los mapas. ¿Cómo se usan?, de eso se trata esta guía. Esta guía tratará acerca de la última versión creada al momento, la versión 3.

Fuente: www.maestrosdelweb.com



Nota.

La versión 3 del API de JavaScript de Google Maps es ahora la versión oficial del API de JavaScript. La versión 2 de API se ha descartado oficialmente, de acuerdo con nuestra política de funciones obsoletas.

8.2 COORDENADAS

Las coordenadas se utilizan para expresar ubicaciones en el mundo. Hay varios sistemas de coordenadas diferentes. Google Maps utiliza el sistema geodésico 84 (WGS 84), que es el mismo sistema de posicionamiento global (GPS) que utiliza. Las coordenadas se expresan mediante la latitud y la longitud. Está relacionado con la idea que se tiene acerca de las ejes Y y los valores de X en una cuadrícula.

Las coordenadas están expresadas usando números decimales separados por coma. La latitud siempre precede la longitud. La latitud es positiva si va después del punto mostrado en el mapa y negativo si va antes. La longitud es positiva si va arriba del punto y negativa si va debajo.
En los mapas físicos, las coordenadas están expresadas en grados, así que la posición de Puerto Rico sería:

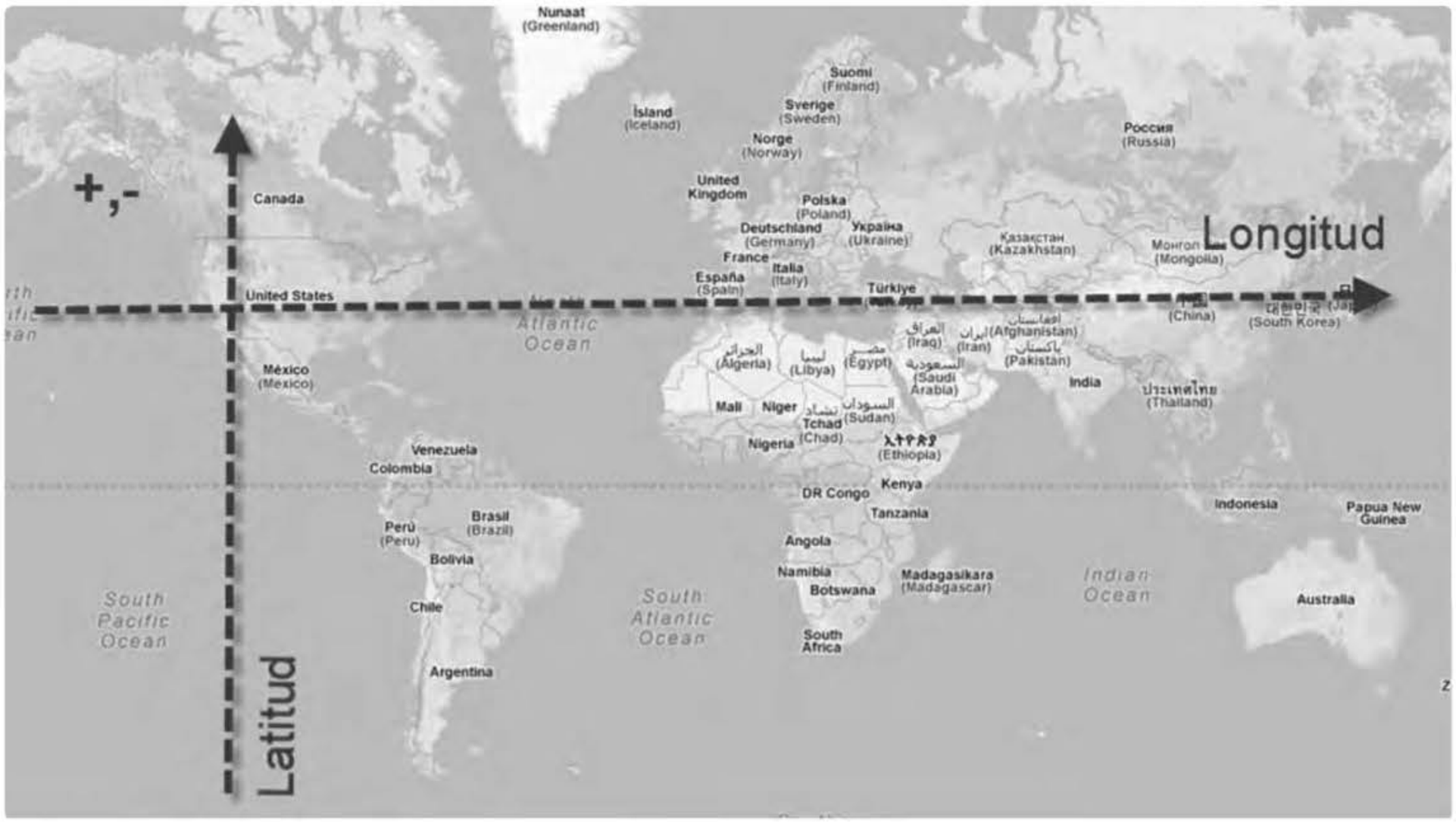
18°14'70" N 66°29'68" W

La forma de convertir estos datos a decimales sería:

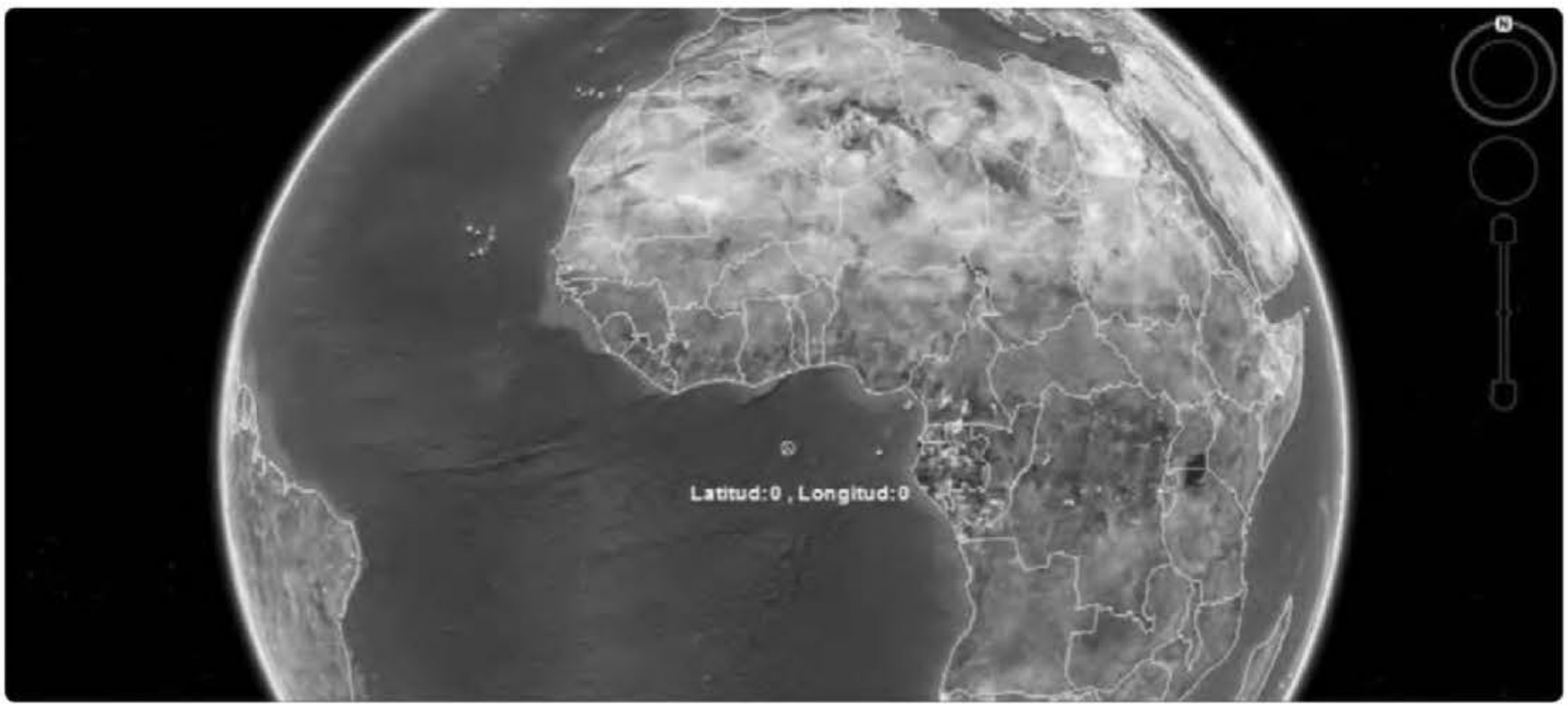
(18°14'70" N) = (18 + (14 / 60) + (70 / 3600)) = 18.252
(66°29'68" W) = -(66 + (29 / 60) + (68 / 3600)) = -66.8627

La longitud se multiplica por negativo, porque está a la izquierda (oeste) del punto 0,0. Para esta guía solo vamos a estar trabajando con decimales.

Fuente: www.maestrosdelweb.com



Por ejemplo, el centro del mundo, en la latitud 0 y longitud 0 (centro de la tierra).



¿Cuánto es el máximo de decimales?

Google maps no se limita a cierta cantidad de decimales. Sin embargo, según unas pruebas hechas, se notó que números mayores a 6 decimales es una perdida de tiempo. Así´también google estableció en varios métodos que la mayor cantidad a trabajar es 6 decimales, como por ejemplo el método toUrlValue(). Es decir, cuando vayamos a establecer los decimales se puede hacer así:

- *5 a 6 decimales: es el máximo que debemos usar para ser específicos*
- *4 decimales: para algún detalle en el mapa*
- *3 decimales: es bueno para centrar ciudades*
- *2 decimales: es apropiado para centrar paises o estados, tal vez 3 por monaco*

Fuente: www.maestrosdelweb.com

8.3 Ejemplo Mapa1

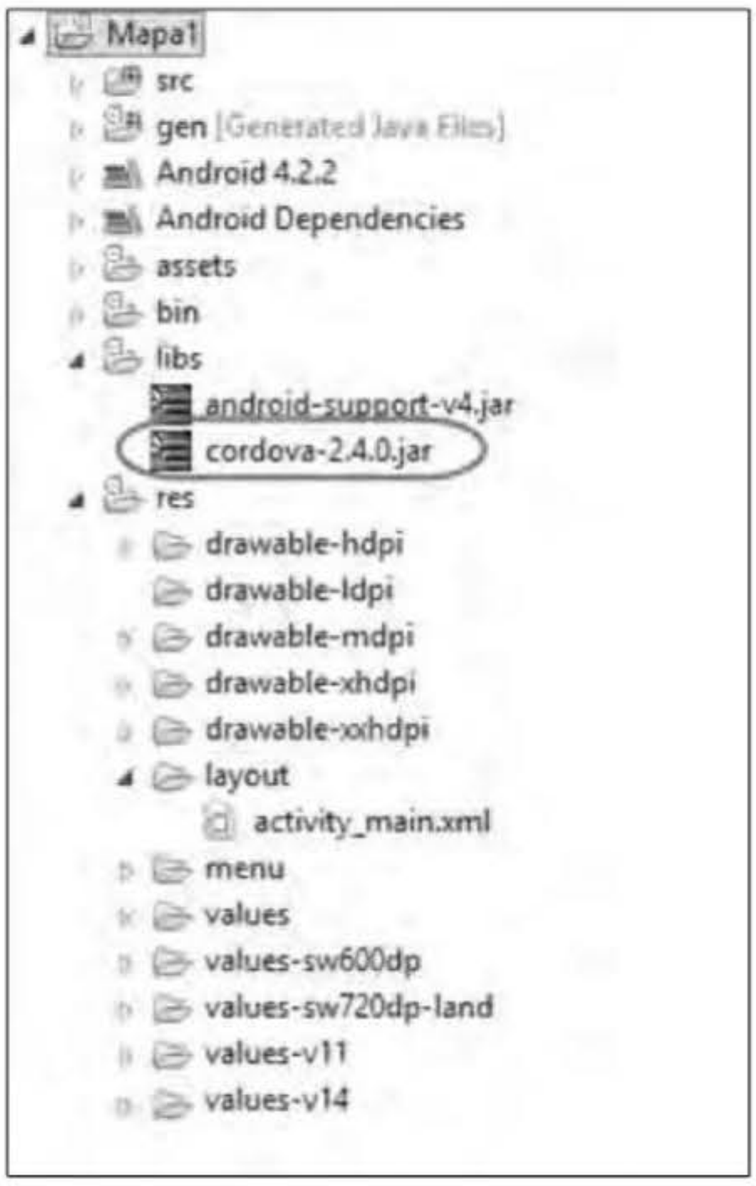


Ahora diseñaremos una aplicación la cual permitira el uso básico de los mapas:

Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android proyecto.*

Paso 2: Ingrese el nombre del proyecto: Mapa1 (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

Paso 3: Almacene el fichero .jar en una carpeta y añádala a su proyecto como una librería externa (cordova-2.4.0.jar).



Paso 4: Cree dentro de la carpeta assets, la carpeta www.

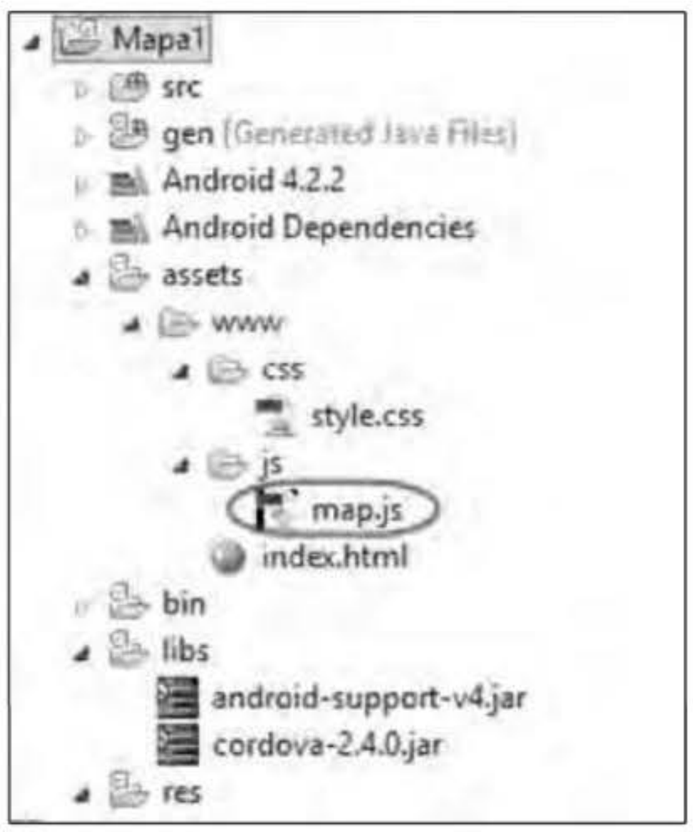


- Hoja de estilo styles.css



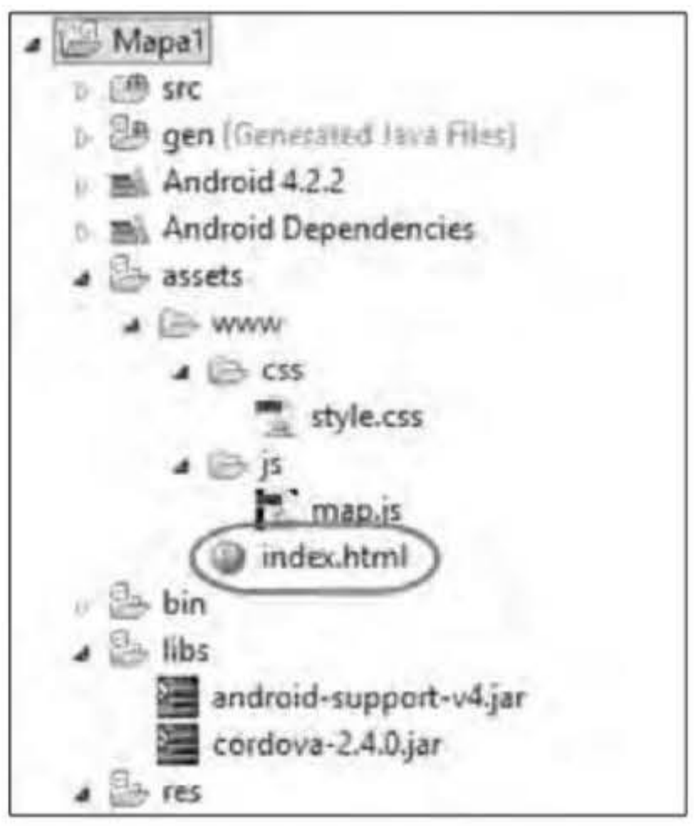

```
body {
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size: small;
    background: #fff;
}
#mapa_peru {
    width: 90%;
    height: 500px;
    border: 3px solid #000;
}
```

- archivo javascript : map.js



```
(function() {
    window.onload = function() {
        alert('Ubicacion del Perú....!');
        // refrencia al div que mostrará el mapa
        var mapDiv = document.getElementById('mapa_peru');
        // referencia de longitud y latitud
        var latlng = new google.maps.LatLng(-12.05,-77.05);
        // propiedades del mapa
        var options = {
        center: latlng,
        zoom: 5,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    // creación del mapa y lo inserta en la capa
    var map = new google.maps.Map(mapDiv, options);
 }
})();
```

- Archivo html : index.html



- Index.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>Mi Primer Ejemplo</title>
        <link type="text/css" href="css/style.css" rel="stylesheet" media="all" />
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sen-
        sor=false"></script>
        <script type="text/javascript" src="js/map.js"></script>
    </head>
    <body>
        <h1>Ubicacion del Perú</h1>
        <div id="mapa_peru"></div>
    </body>
</html>
```

Paso 5: Codifique lo siguiente en la actividad MainActivity.java:

```
import android.os.Bundle;
import org.apache.cordova.*;

public class MainActivity extends DroidGap
{
    public void onCreate(Bundle savedInstanceState)
    {
    super.onCreate(savedInstanceState);
    super.loadUrl("file:///android_asset/www/index.html");
```

Paso 6: Antes de ejecutar la aplicación no se olvide de los permisos necesarios, en el archivo de manifiesto:

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

EJECUTANDO LA APLICACIÓN

Paso 1: *Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:*



Nota:

Tipos de mapas que se puede utilizar en Google Maps son los siguientes:

- *MapTypeId.ROADMAP muestra la vista de mapa de carretera predeterminada.*
- *MapTypeId.SATELLITE muestra imágenes de satélite de Google Earth.*
- *MapTypeId.HYBRID muestra una mezcla de vistas normales y de satélite.*
- *MapTypeId.TERRAIN muestra un mapa físico basado en información del relieve.*

Paso 2: *Para tener un visualización del mapa tipo satelital, modifique el archivo map.js:*

```
(function() {
  window.onload = function() {
    alert('Ubicacion del Peru....!');
    // referencia al div que muestra el mapa
    var mapDiv = document.getElementById('mapa_peru');

    // referencia de longitud y latitud
    var latlng = new google.maps.LatLng(-12.05,-77.05);
    // propiedades del mapa
    var options = {
      center: latlng,
      zoom: 5,
      mapTypeId:google.maps.MapTypeId.SATELLITE
    };
    // creacion del mapa
    var map = new google.maps.Map(mapDiv, options);
  }
})();
```

EJECUTANDO LA APLICACIÓN

Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:



8.4 Ejemplo Mapa2



Ahora diseñe una aplicación que permita mostrar la ubicación de Perú y permita alternar un mapa de la ciudad de Machu Picchu y la ubicación de Perú; además que muestre los valores siguientes:

Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android proyecto.*

Paso 2: Ingrese el nombre del proyecto: Mapa2 (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

Paso 3: Almacene el fichero .jar en una carpeta y añádala a su proyecto como una librería externa (cordova-2.4.0.jar).



Paso 4: Cree dentro de la carpeta assets, la carpeta www.

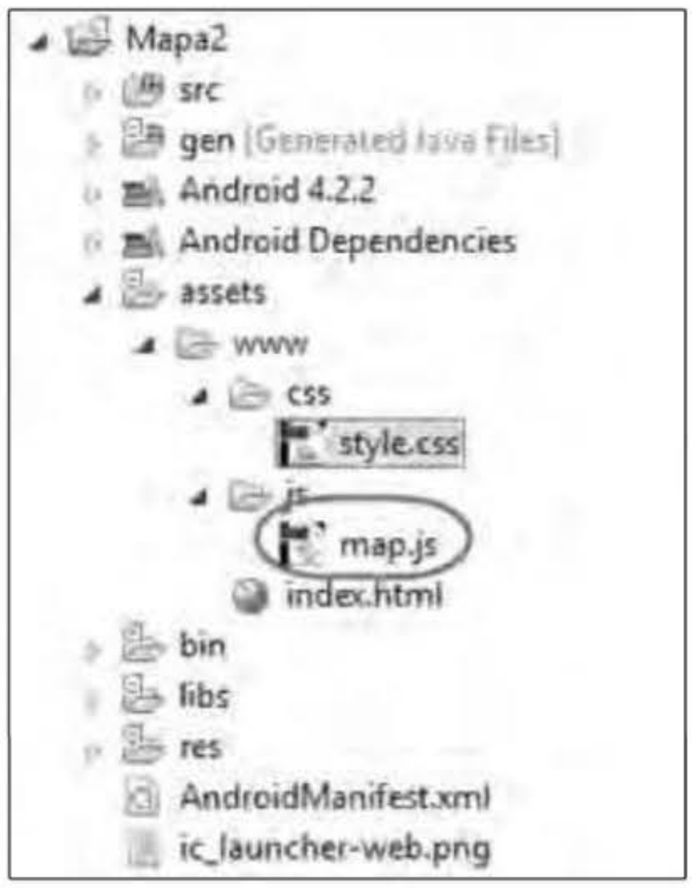


- La hoja de estilo styles.css




```
body {
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size: small;
    background: #fff;
}
#mapa_peru {
    width: 90%;
    height: 500px;
    border: 3px solid #000;
}
```

- El archivo javascript : map.js



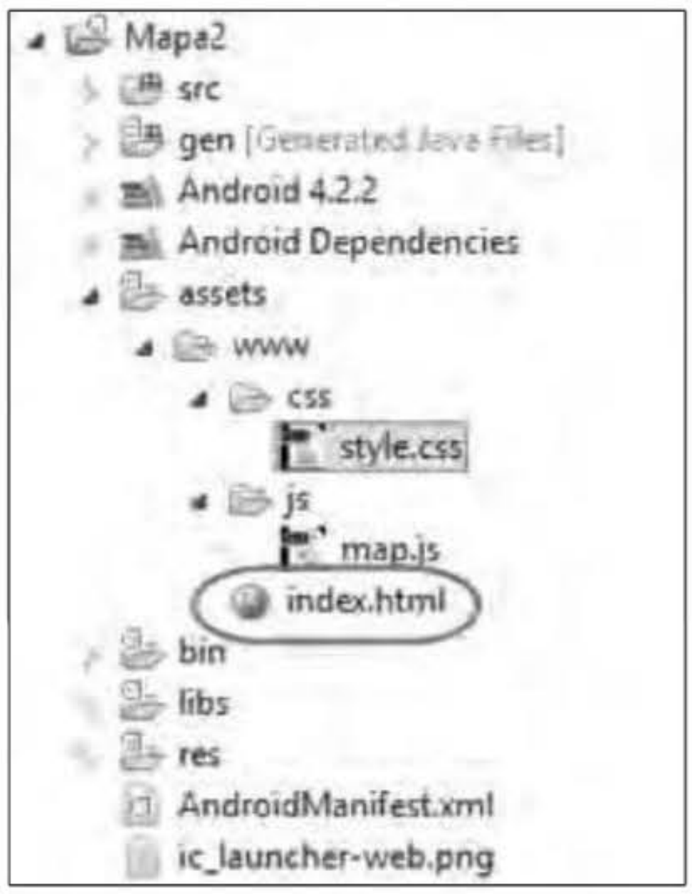
```
(function() {
    window.onload = function() {

        var options = {
            zoom: 7,
            center: new google.maps.LatLng(-12.05,-77.051),
            mapTypeId: google.maps.MapTypeId.ROADMAP,
            mapTypeControl: true,
            mapTypeControlOptions: {
                mapTypeIds: [
                    google.maps.MapTypeId.ROADMAP,
                    google.maps.MapTypeId.SATELLITE
                ]
            }
        };

        var map = new google.maps.Map(document.getElementById('mapa_peru'), options);

        document.getElementById('getValues').onclick = function() {
            alert('Nivel ' + map.getZoom());
            alert('Centrado ' + map.getCenter());
            alert('Tipo' + map.getMapTypeId());
        }
        document.getElementById('changeValues2').onclick = function() {
            map.setOptions({
                center: new google.maps.LatLng(-12.05,-77.051),
                zoom: 7,
                mapTypeId: google.maps.MapTypeId.SATELLITE
            });
        }
            document.getElementById('changeValues').onclick = function() {
                map.setOptions({
                    center: new google.maps.LatLng(-13.164317,-72.545372),
                    zoom: 17,
                    mapTypeId: google.maps.MapTypeId.SATELLITE
                });
            }
        };
})();
```

- archivo html : index.html



- Index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Segundo Ejemplo</title>
        <link rel="stylesheet" href="css/style.css" type="text/css" media="all" />
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
        <script type="text/javascript" src="js/map.js"></script>
    </head>

    <body>
        <input type="button" value="Valores" id="getValues" />
        <input type="button" value="Machu Picchu" id="changeValues" />
        <input type="button" value="Perú" id="changeValues2" />
        <div id="mapa_peru"></div>
    </body>
</html>
```

Paso 5: Codifique lo siguiente en la actividad MainActivity.java.

```
import android.os.Bundle;
import org.apache.cordova.*;

public class MainActivity extends DroidGap
{
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        super.loadUrl("file:///android_asset/www/index.html");
```

Paso 6: Antes de ejecutar la aplicación, no se olvide de los permisos necesarios, en el archivo de manifiesto:

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

EJECUTANDO LA APLICACIÓN

Paso 1: *Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:*



- *Vista satélite*



- *Machu Picchu*



8.5 EJEMPLO MAPA3



Ahora diseñe una aplicación que permita el uso básico de los mapas y rotación.

Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android proyecto*.

Paso 2: Ingrese el nombre del proyecto: Mapa3 (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

Paso 3: Almacene el fichero .jar en una carpeta y añádala a su proyecto como una librería externa (cordova-2.4.0.jar).





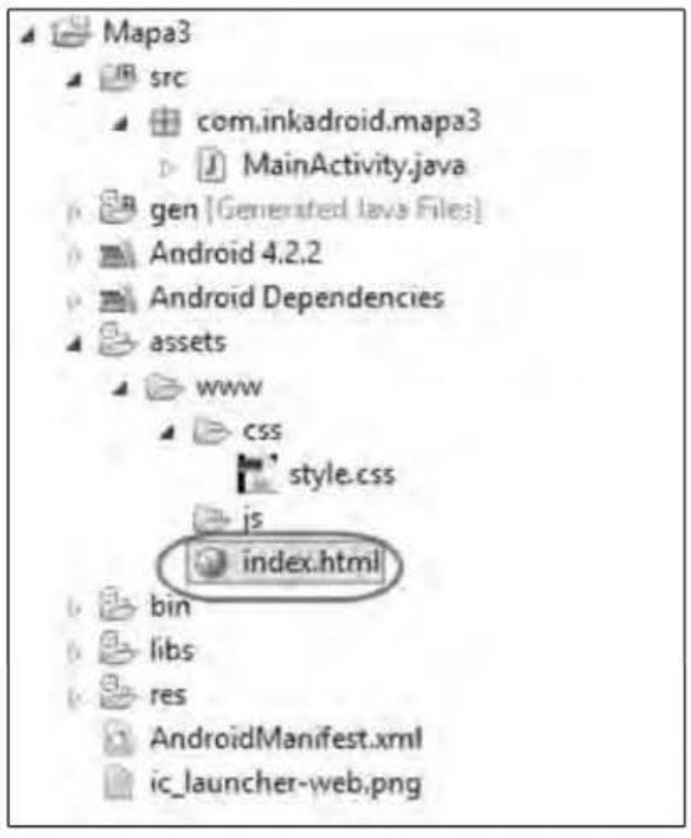
Paso 4: Cree dentro de la carpeta assets, la carpeta www.

- Hoja de estilo styles.css




```
html, body {
    height: 100%;
    margin: 0;
    padding: 0;
}
#mapa{
    height: 100%;
}
@media print {
    html, body {
     height: auto;
    }
    #mapa {
     height: 650px;
    }
}
```

- Archivo html : index.html



- Index.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
        <meta charset="utf-8">
        <title>Google Maps JavaScript API v3 Example: Simple Aerials</title>

        <link type="text/css" href="css/style.css" rel="stylesheet" media="all" />
        <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>
        <script>
            var map;
            function initialize() {
                var mapOptions = {
                      center: new google.maps.LatLng(45.518970, -122.672899),
                      zoom: 18,
                      mapTypeId: google.maps.MapTypeId.SATELLITE,
                      heading: 90,
                      tilt: 45
                };
                map = new google.maps.Map(document.getElementById('mapa'), mapOptions);
            }

            function rotate90() {
                var heading = map.getHeading() || 0;
                map.setHeading(heading + 90);
            }

                function autoRotate() {
                    // Determine if we're showing aerial imagery
                    if (map.getTilt() != 0) {
                          window.setInterval(rotate90, 3000);
                    }
                }

                google.maps.event.addDomListener(window, 'load', initialize);
            </script>
        </head>
        <body>
            <input type="button" value="Rotar" onclick="autoRotate();">
            <div id="mapa"></div>
        </body>
    </html>
```

Paso 5: Codifique lo siguiente en la actividad MainActivity.java.

```
import android.os.Bundle;
import org.apache.cordova.*;

public class MainActivity extends DroidGap
{
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        super.loadUrl("file:///android_asset/www/index.html");
```

Paso 6: Antes de ejecutar la aplicación, no se olvide de los permisos necesarios en el archivo de manifiesto:

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

EJECUTANDO LA APLICACIÓN

Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:



8.6 MARCADORES

Los marcadores identifican ubicaciones en el mapa. De manera predeterminada, utilizan un ícono estándar, aunque puede establecer un ícono personalizado dentro del constructor del marcador o mediante la ejecución de setIcon() en el marcador. El constructor google.maps.Marker toma un único objeto literal Marker options que especifica las propiedades iniciales del marcador. A continuación, se indican algunos campos especialmente importantes que se suelen definir al crear un marcador.

- *position (obligatorio):* Especifica un valor de LatLng que identifica la ubicación inicial del marcador.
- *map (opcional):* Especifica el objeto Map en el que se sitúa el marcador.

Considere que debe especificar el mapa en el que se va a añadir el marcador dentro del constructor Marker. Si no se especifica este argumento, el marcador se creará pero no se añadirá al mapa (o no se mostrará). Puede añadir el marcador más tarde, mediante la ejecución del método setMap() del marcador. Para eliminar un marcador, ejecute el método setMap() y transmita null como el argumento. Los marcadores están diseñados para ser interactivos. De forma predeterminada, reciben eventos clic, por ejemplo, y se suelen utilizar dentro de detectores de eventos para abrir ventanas de información. Puede establecer la propiedad draggable de un marcador en true para que los usuarios puedan editar el marcador en el mapa.

8.7 Ejemplo Mapa4

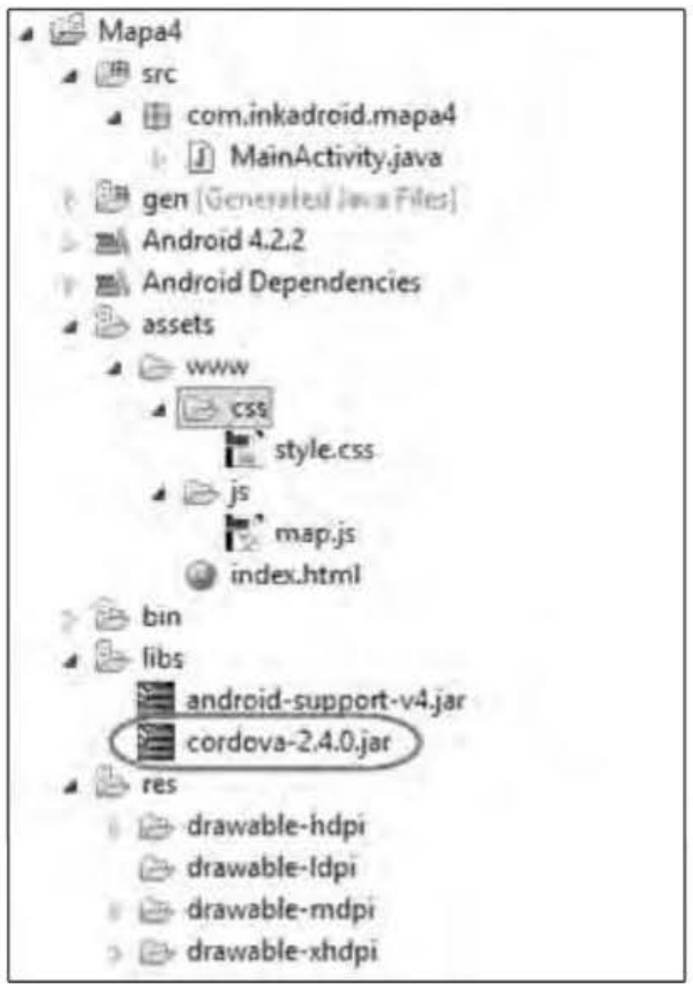


Ahora diseñe una aplicación que permita el uso básico de marcadores.

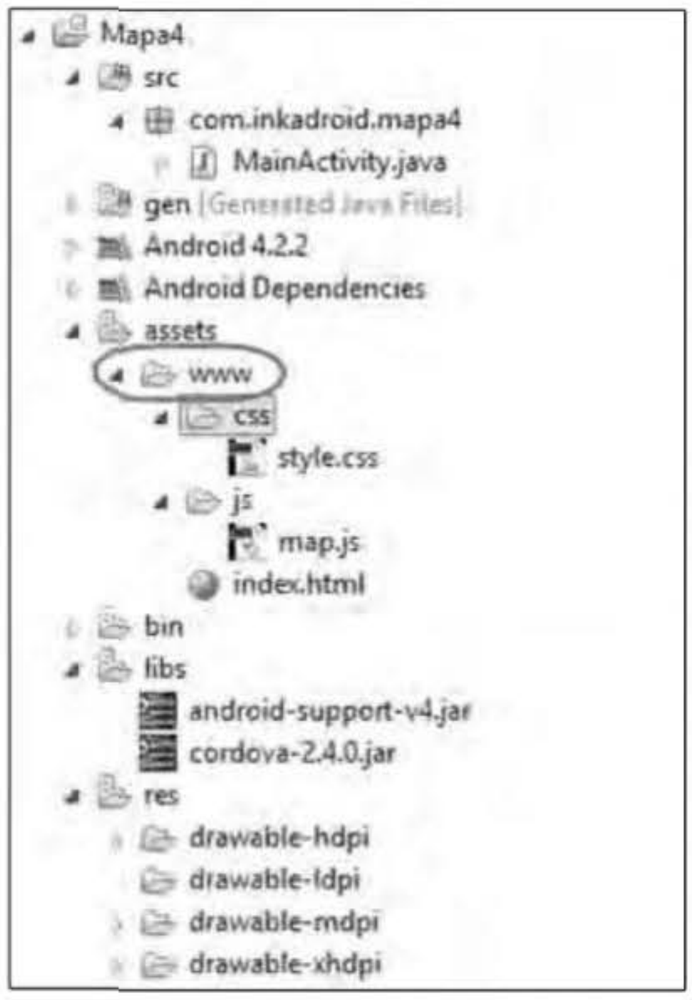
Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android proyecto.*

Paso 2: Ingrese el nombre del proyecto: Mapa4 (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

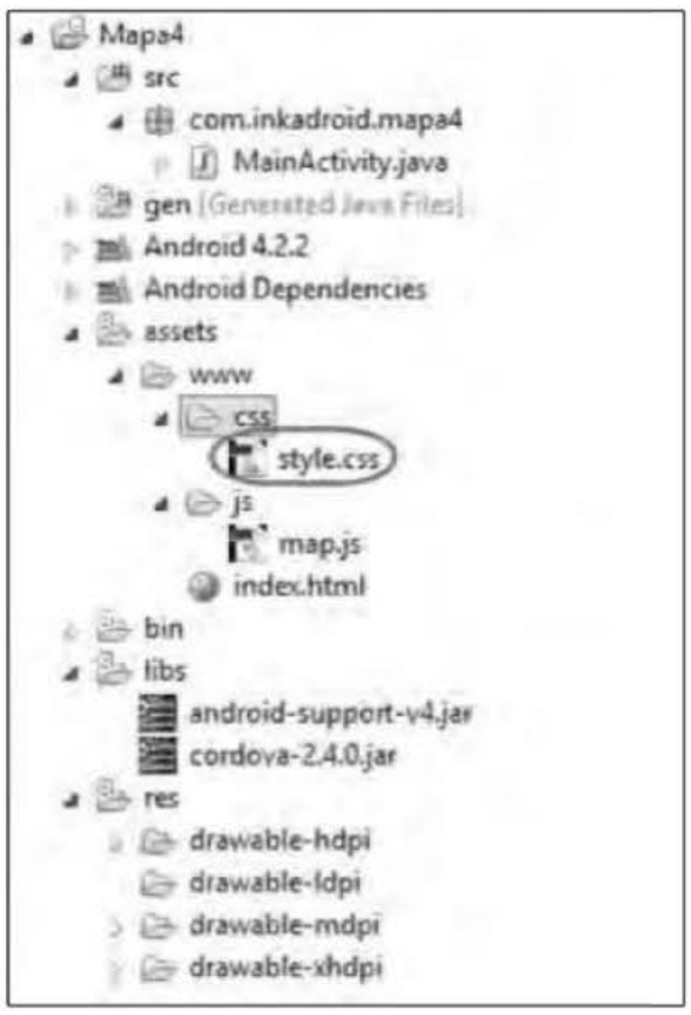
Paso 3: Almacene el fichero .jar en una carpeta y añádala a su proyecto como una librería externa (cordova-2.4.0.jar).



Paso 4: Cree dentro de la carpeta assets, la carpeta www.



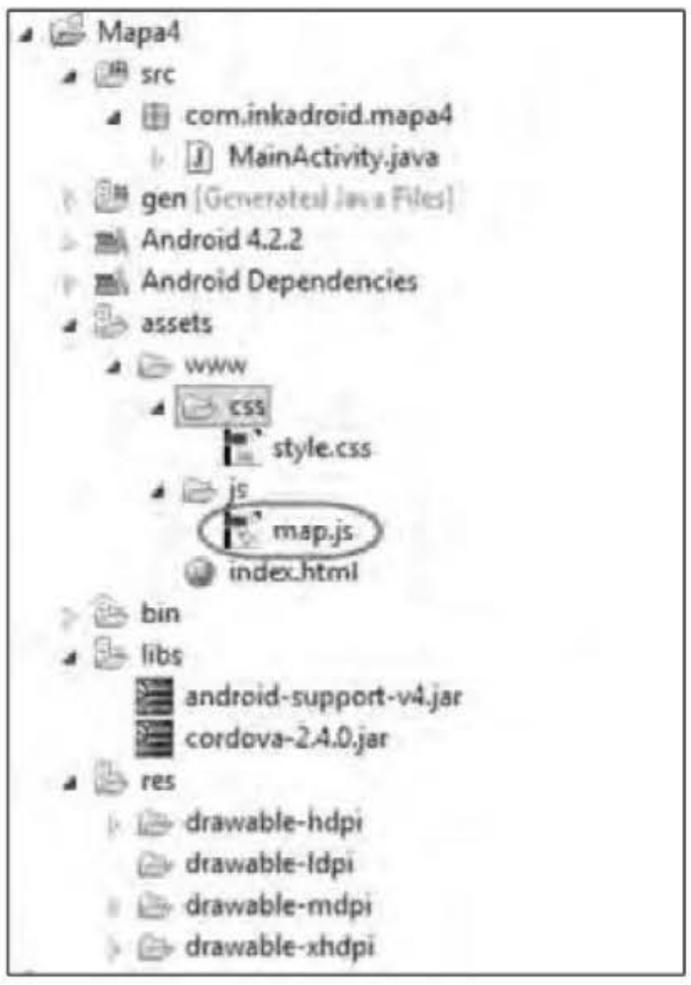
- Hoja de estilo styles.css



- Style.css

```
body {
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size: small;
    background: #fff;
}
#mapa_peru {
    width: 90%;
    height: 500px;
    border: 3px solid #000;
}
```

- Archivo html : map.js



- map.js

```
(function() {
    window.onload = function() {
        alert('Ubicacion del Perú....!');
        // refrencia al div que mostrará el mapa
        var mapDiv = document.getElementById('mapa_peru');

        // referencia de longitud y latitud
        var latlng = new google.maps.LatLng(-12.05,-77.05);
        // propiedades del mapa
        var options = {
            center: latlng,
            zoom: 4,
            mapTypeId:google.maps.MapTypeId.ROADMAP
        };
        // creación del mapa
        var map = new google.maps.Map(mapDiv, options);

        var marker = new google.maps.Marker({
        position: latlng,
        map: map,
            title: 'Hola Perú!'
         });
       }
})
();
```

- archivo html : index.html



- Index.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>My first map</title>
        <link type="text/css" href="css/style.css" rel="stylesheet" media="all" />
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=-
        false"></script>
        <script type="text/javascript" src="js/map.js"></script>
    </head>
    <body>
        <h1>Ubicacion del Perú</h1>
        <div id="mapa_peru"></div>
    </body>
</html>
```

Paso 5: Codifique los siguientes en la actividad MainActivity.java.

```
import android.os.Bundle;
import org.apache.cordova.*;

public class MainActivity extends DroidGap
{
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        super.loadUrl("file:///android_asset/www/index.html");
```

Paso 6: Antes de ejecutar la aplicación, no se olvide de los permisos necesarios en el archivo de manifiesto:

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

EJECUTANDO LA APLICACIÓN

Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:



8.8 EJEMPLO MAPA5

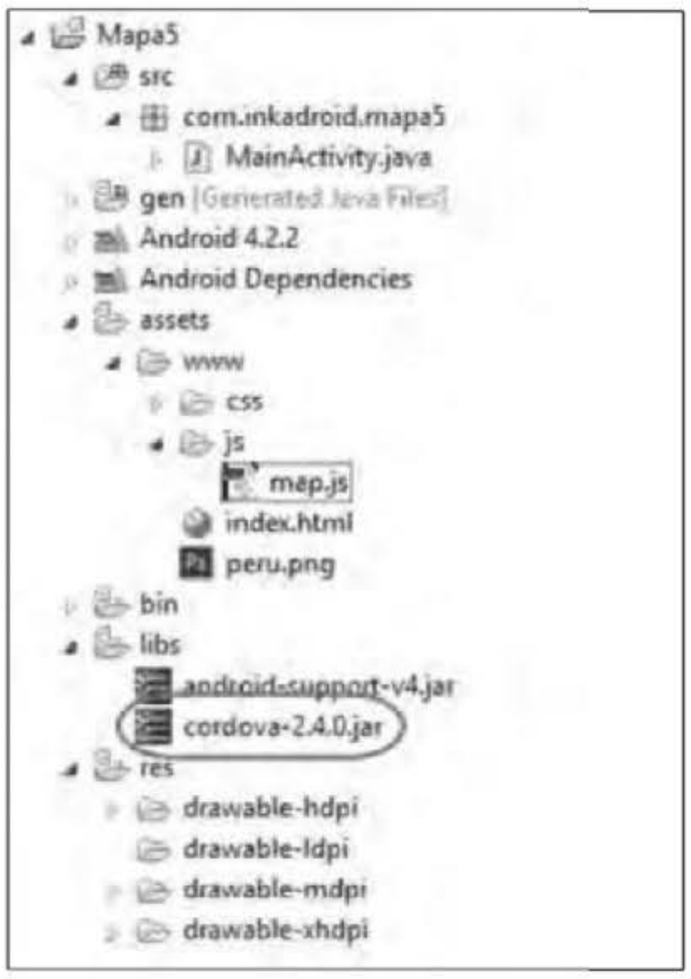


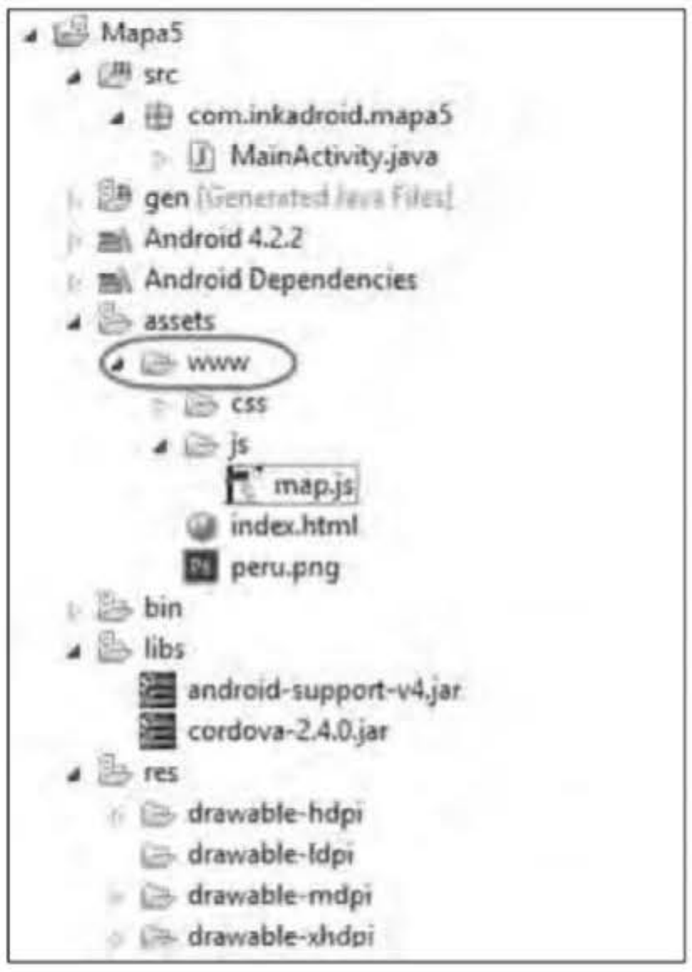
Ahora diseñe una aplicación que permita la personalizacion del marcador.

Paso 1: Después de haber creado la primera aplicación para Android, cree un nuevo proyecto seleccionando *Archivo > Nuevo > Android proyecto.*

Paso 2: Ingrese el nombre del proyecto: Mapa5 (los pasos de creación de un proyecto ya se detallaron en ejemplos anteriores).

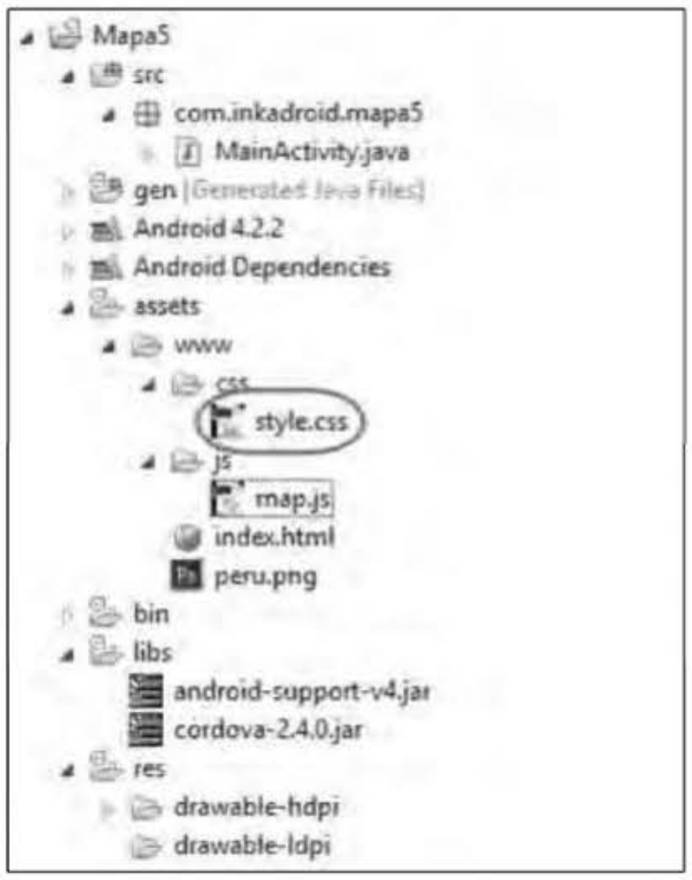
Paso 3: Almacene el fichero .jar en una carpeta y añádala a su proyecto como una librería externa (cordova-2.4.0.jar).





Paso 4: Cree dentro de la carpeta assets, la carpeta www.

- Hoja de estilo styles.css



- Style.css

```
body {
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size: small;
    background: #fff;
}
#mapa_peru {
    width: 90%;
    height: 500px;
    border: 3px solid #000;
}
```

- Archivo html: map.js



- map.js

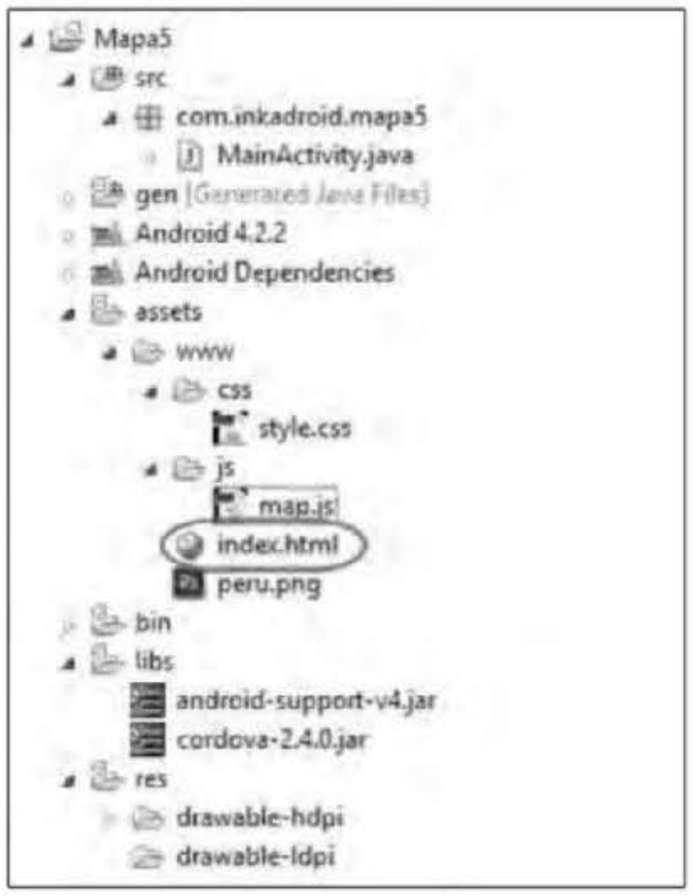
```
(function() {
    window.onload = function() {
        alert('Ubicación del Perú....!');
        // refrencia al div que mostrará el mapa
        var mapDiv = document.getElementById('mapa_peru');

        // referencia de longitud y latitud
        var latlng = new google.maps.LatLng(-12.05,-77.05);
        // propiedades del mapa
        var options = {
            center: latlng,
            zoom: 4,
            mapTypeId:google.maps.MapTypeId.ROADMAP
        };
        // creación del mapa
        var map = new google.maps.Map(mapDiv, options);

        var marker = new google.maps.Marker({
            position: latlng,
            map: map,
            title: 'Hola Perú!'
        });

    }
})
();
```

- *archivo html : index.html*



- Index.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>My first map</title>
        <link type="text/css" href="css/style.css" rel="stylesheet" media="all" />
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=-
        false"></script>
        <script type="text/javascript" src="js/map.js"></script>
    </head>
    <body>
        <h1>Ubicacion del Perú</h1>
        <div id="mapa_peru"></div>
    </body>
</html>
```

Paso 4: Codifique lo siguiente en la actividad MainActivity.java.

```
import android.os.Bundle;
import org.apache.cordova.*;

public class MainActivity extends DroidGap
{
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        super.loadUrl("file:///android_asset/www/index.html");
```

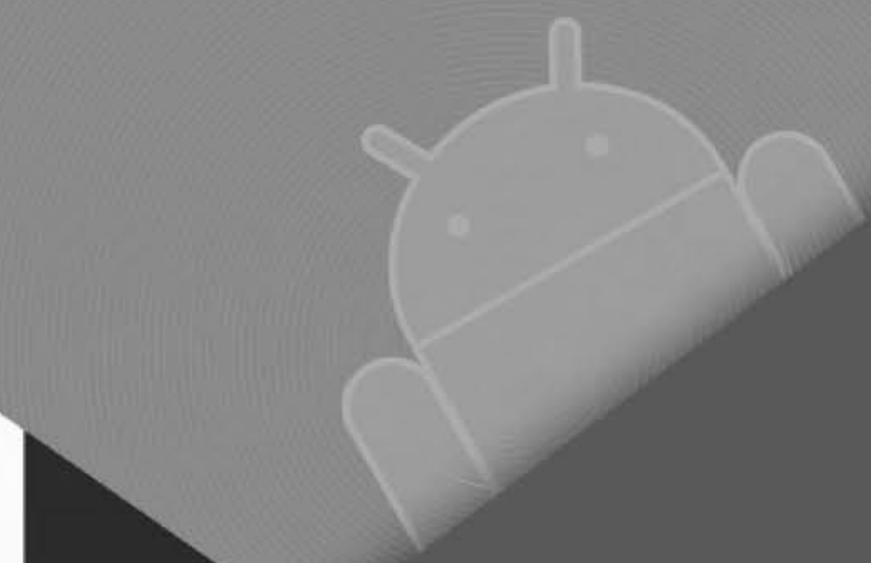
Paso 5: Antes de ejecutar la aplicación, no se olvide de los permisos necesarios, en el archivo de manifiesto:

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

EJECUTANDO LA APLICACIÓN

Seleccione el nombre del proyecto en Eclipse. Haga clic derecho, seleccione la opción Run As > Android Application y observe que el emulador se pondrá en marcha:





CAPÍTULO

9

Códigos ocultos en su dispositivo móvil

9.1 Código



Anteriormente, se ha revisado cómo obtener información útil del dispositivo Android, mediante MOBILedit! En este capítulo se abordará algunos códigos secretos con los que realizar determinadas funciones. En esta ocasión, los códigos sirven para reiniciar o formatear el móvil como venía de fábrica. Cabe advertir que el uso de estos códigos están bajo tu entera responsabilidad.

9.1.1 Lista de los diferentes códigos

```
*#*#8255#*#*
```

Este código sirve para lanzar el monitor de servicio de GTalk:

```
*#*#4636#*#*
```

La información que debe tener su batería es la siguiente:

- Información del teléfono
- Información de la batería
- Historial de la batería
- Estadísticas de uso

```
*#*#7780#*#*
```

Este código sirve para hacer un reset de fábrica, por lo que borrará lo siguiente:

- Ajustes de su cuenta Google en el teléfono.
- Ajustes del sistema y de las aplicaciones.
- Aplicaciones descargadas en la SD (descargadas, no instaladas).

Por otro lado, no borrará lo siguiente:

- El software y las aplicaciones que estén instaladas.
- Los archivos de la SD.

```
*2767*3855#
```

Cabe advertir que este código hará un reset de fábrica completo (wipe total). Borrará todos los archivos de la memoria interna y reinstalará el firmware.

```
*#*#34971539#*#*
```

Información de su cámara:

- Actualizar el firmware de la cámara (no seleccione esta opción).
- Actualizar el firmware de la cámara en la SD.
- Sobre la versión del firmware de la cámara.

```
*#*#7594#*#*
```

Con este código, se apagará directamente su teléfono, sin pasar por el menú:

```
*#*#273283*255*663282*#*#*
```

Este código abre una pantalla que le permite copiar sus archivos multimedia:

```
*#*#197328640#*#*
```

Sirve para acceder al modo Servicio, que sirve para realizar diferentes tests. Asimismo, permite utilizar los siguientes códigos:

```
*#*#232339#*#* O bien *#*#526#*#* OR *#*#528#*#* - WLAN test
*#*#232338#*#* - Enseña la dirección MAC actual
*#*#1472365#*#* - GPS test
*#*#1575#*#* - Otro GPS test
*#*#232331#*#* - Bluetooth test
*#*#232337#*# - Muestra la dirección del terminal Bluetooth
*#*#8255#*#* Abres el monitor de Gtalk.
```

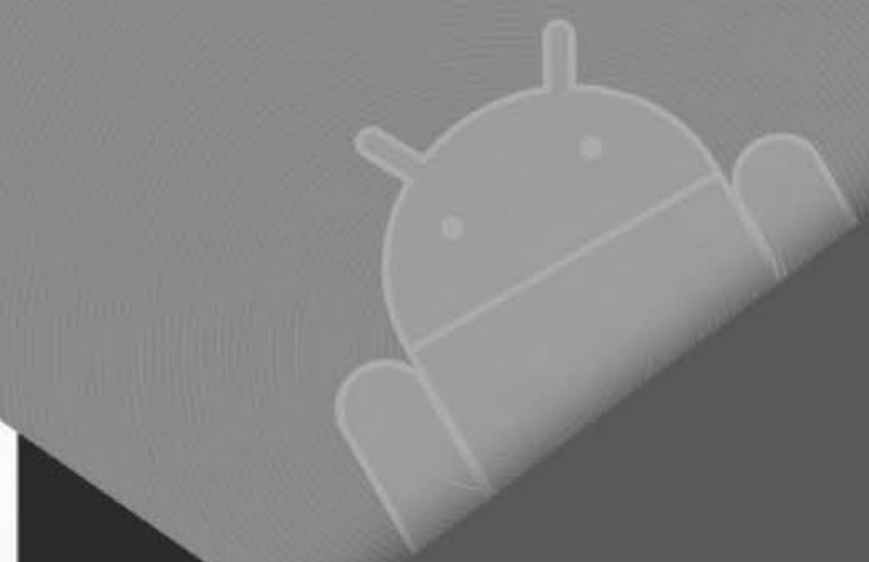
9.1.2 Códigos para obtener información del firmware

```
*#*#4986*2650468#*#* - PDA, Phone, H/W, RFCallDate
*#*#1234#*#* - PDA and Phone
*#*#1111#*#* - FTA SW Version

*#*#2222#*#* - FTA HW Version
*#*#44336#*#* - PDA, Phone, CSC, Build Time, Changelist number
```

9.1.3 Códigos para ejecutar diferentes tests de fábrica

```
*#*#0283#*#* - Packet Loopback
*#*#0*#*#* - LCD test
*#*#0673#*#* OR *#*#0289#*#* - Melody test
*#*#0842#*#* - Device test (Vibration test and BackLight test)
*#*#2663#*#* - Touch screen version
*#*#2664#*#* - Touch screen test
*#*#0588#*#* - Proximity sensor test
*#*#3264#*#* - RAM versión
```



CAPÍTULO

10

Las mejores aplicaciones para Android

10.1 WhatsApp

WhatsApp Messenger es un mensajero multiplataforma disponible en Android y en otros teléfonos inteligentes. La aplicación utiliza su conexión 3G o Wi-Fi (cuando está disponible) para recibir mensajes de amigos, colegas y familia. Olvídese de los SMS y envíe mensajes, fotos, notas de audio y videos sin costo alguno.



10.2 Facebook

Manténgase en permanente contacto con sus amigos de forma más rápida.



- Vea qué están haciendo sus amigos.
- Comparta actualizaciones de estado, fotos y videos.
- Reciba notificaciones cuando sus amigos comenten sus publicaciones o marquen que les gusta.
- Envíe SMS, chatee y mantenga conversaciones en grupo.
- Juegue y utilice sus aplicaciones favoritas.

Para usar Facebook, el usuario debe tener 13 años como mínimo. Las condiciones del servicio las puede encontrar en:

> http://m.facebook.com/terms.php

10.3 Spotbros

Esta aplicación permite enviar mensajes gratuitos a su grupo de contactos y comunicarse con otros usuarios a su alrededor. Asimismo, presenta las SBApps, aplicaciones que ofrecen servicios útiles y gratuitos a través de mensajería instantánea.



10.3.1 Características

- *Mensajería instantánea:* Le permite enviar y recibir mensajes gratuitos para estar siempre conectado con su grupo de contactos. Podrá chatear con amigos y crear grupos privados, sin límite de miembros, donde se podrá compartir todo tipo de archivos.
- *Seguridad*: Spotbros se preocupa por la privacidad de sus usuarios, por eso cifra sus conversaciones con AES256bits (el mismo sistema de cifrado que utiliza EE.UU. para sus documentos Top Secret) que luego borra de los servidores en un plazo máximo de 30 días.
- *Spots*: Son grupos de chat públicos geolocalizados que se interesan por un tema en común. Además, permite que el usuario pueda crear sus propios Spots.

- *Shout*: Esta aplicación resulta útil para ubicar lugares. Por ejemplo, si una turista busca un lugar para comer, recibirá mensajes de los restaurantes más cercanos a su ubicación actual.
- *SBoole*: Es una forma rápida y fácil de encontrar lo que le interesa al usuario, en cada momento. Encontrará Spots, SBApps y otros usuarios. Para ello, escriba una palabra o, si prefiere, busque Spots por ubicación o categoría.
- *SBapps:* Consiste en aplicaciones que ofrecen servicios vía mensajería instantánea. Existen SBapps sobre información de transporte público, resultados de fútbol, cambio de divisas, etc.
- *Newsfeed*: Permite compartir sus momentos y enterarse de las últimas noticias sobre sus amigos y contactos. Actualice su estado con fotos, mapas, videos y audios.

10.4 TWITTER

Presenta las siguientes características:



- *Esencial:* Un flujo organizado de tweets que ofrece el mejor contenido para el usuario.
- *Instantáneo:* Se encuentran todos los medios de comunicación, noticias, eventos e información que necesitas.
- *Personal:* Permite expresarse por medio de tweets y fotos.
- *Búsqueda en tiempo real:* Permite seguir a las personas que le interesan al usuario y conocer las tendencias que se encuentran en auge.

10.5 OTRAS APLICACIONES

Angry Birds para móviles y tablets



"[…] Angry Birds Android es uno de los juegos android más descargados y, sobre todo, más populares de estos tiempos. El objetivo del juego es derribar las casas de los cerditos, para esto deberás lanzar a las aves gruñonas con la ayuda de las divertidas Angry Birds. Así, acabarás con todos los cerditos de cada escenario para avanzar en cada uno de los niveles pero, cuidado, porque Angry Birds se irá poniendo cada vez más complicado […]".

Fuente: imovil.org

Camera ZOOM FX



- *Zoom óptico / digital (hasta 6 aumentos).*
- *Flash (si está disponible).*
- *Cámara frontal (si está disponible).*
- *Personalización de todos los botones físicos. Por ejemplo: Botones de volumen para controlar el zoom, trackball para tomar la fotografía.*
- *Previsualización en vivo de algunos efectos.*
- *Cámara silenciosa (si el dispositivo lo soporta).*
- *Modifica opciones del dispositivo: enfoque automático, balance de blancos, disparo nocturno…*
- *Envía y comparte tus tomas.*
- *Sube las imágenes con 1 click a Facebook, Twitter, Flickr, etc… ¡a la vez!*
- *Cuadrículas sobreimpresionadas personalizables.*

Fuente: play.google.com

Avast! Mobile Security



DESCRIPCIÓN

Una aplicación antivirus y de seguridad para Android con todas las características y con la mejor valoración (4.7 de 5 estrellas).

Localizador/rastreador de teléfonos y protección contra malware (con bloqueador de USSD).

Altamente recomendado por autoridades en Android: AV-Test, PCAdvisor, Techworld y, ahora, por 14 millones de usuarios en todo el mundo.

ANÁLISIS

- *AndroidAuthority: "El mejor, aún mejor... Nada se le compara".*
- *Android and Me: "La mejor solución antirrobo del mercado".*
- *AndroidPolice: "Básicamente, lo que usted desearía tener si su dispositivo se pierde o es robado"*
- *Droid-Life: "No se puede pedir mucho más que esto".*
- *PCAdvisor: "El primer puerto en el que recalar cuando busque un producto de seguridad".*

PRINCIPALES CARACTERÍSTICAS

- *Disponible en los siguientes idiomas: Catalán, checo, holandés, inglés, francés, alemán, húngaro, chino (simplificado), chino (tradicional), italiano, japonés, coreano, polaco, portugués, ruso, español.*
- *Network Meter: Controla los flujos de datos de entrada y salida.*
- *Widget: Le permite controlar la aplicación desde el escritorio de Android.*
- *Interfaz web: Proporciona control remoto de las opciones de anti-theft (bloqueo de teléfono, seguimiento mediante GPS, eliminación remota de datos, etc.).*
- *SiteCorrect: Corrige automáticamente las URLs mal tecleadas.*
- *Control de escudos: Opciones avanzadas para el análisis de aplicaciones, web y mensajes.*
- *Diseño UI: Excelente legibilidad en teléfonos y tablets Android.*
- *Antivirus: Analiza, bajo demanda, las aplicaciones instaladas, y el contenido de la tarjeta de memoria y cada nueva aplicación antes del primer uso.*
- *Asesor de privacidad: Analiza y muestra los derechos de acceso de las aplicaciones instaladas.*
- *Filtro SMS y de llamadas: Filtra los contactos que usted seleccione.*
- *Anti-Theft (componente oculto): Proporciona acceso remoto mediante SMS o Web (bloqueo, localización, sirena, borrado de datos, etc.).*
- *Gestor de aplicaciones: Realiza el listado de aplicaciones y su tamaño (MB), carga de CPU, memoria usada, etc.*
- *Escudo web: Analiza cada URL cargada y le avisa si está infectada con malware. Incluye bloqueo de USSD.*
- *Ahorro de batería: Funciona solo cuando está ejecutando alguna tarea.*
- *Cortafuegos (solo funciona en teléfonos rooteados): Sirve como filtro para el tráfico de red.*

Para más información, visite: http://www.avast.com/free-mobile-security.
Controle de forma remota su teléfono en: https://my.avast.com.

Fuente: play.google.com

Battery Widget Reborn



Battery Widget Reborn es una excelente aplicación para controlar el uso de la batería; cuenta con una interfaz estilo Holo muy cuidada y fácil de usar. Gracias a su widget y a su notificación configurable en la barra de estado podemos estar siempre informados sobre la batería restante.

Fuente: www.xatakandroid.com

Cerberus



Cerberus es un programa completo para la protección contra el robo de sus dispositivos. Es la mejor opción para que pueda recuperar sus dispositivos de Android extraviados, perdidos o robados. Esta es una versión de prueba gratuita durante una semana. Luego, usted puede comprar una licencia de por vida a un pequeño precio (2.99€) desde la aplicación: no hay cuotas mensuales o anuales, solo un pago por única vez. La licencia está asociada a su cuenta de Cerberus. Si tiene hasta 5 dispositivos, los puede utilizar con la misma cuenta.

Tiene tres maneras de proteger su dispositivo:

- *Via control remoto a través de la página web www.cerberusapp.com.*
- *Via control remoto a través de de mensajes de textos SMS.*
- *Con SIM Checker (para los dispositivos que tienen una tarjeta SIM) recibirá automáticamente alertas si alguien utiliza su teléfono con una tarjeta SIM no autorizada.*

El control remoto le permite realizar muchas operaciones en el dispositivo, como las siguientes:

- *Localizar y rastrear su dispositivo.*
- *Iniciar una fuerte alarma, incluso si el dispositivo está en modo silencioso.*

- *Borrar la memoria interna y de la tarjeta SD.*
- *Ocultar Cerberus del menu de las aplicaciónes.*
- *Bloquear el dispositivo con un código.*
- *Grabar audios desde el micrófono.*
- *Obtener una lista de las últimas llamadas enviadas y recibidas.*
- *Obtener información sobre la red y el operador a cual el dispositivo está conectado.*
- *¡Y mucho más!*

Cuando intenta localizar su dispositivo, Cerberus habilita automáticamente GPS si está apagado, (solo en Android <2.3.3) y lo puede proteger de la desinstalación no autorizada —más información en la aplicación.

La aplicación funciona incluso si el dispositivo no tiene una conexión a Internet, esto es gracias a la función de control remoto por medio de mensajes SMS. Asimismo, la función SIM Checker le permite conocer el nuevo número si se cambia la tarjeta SIM, para que pueda enviar textos a este número. Los textos recibidos y enviados para el control remoto de Cerberus no iniciarán ninguna notificación ni aparecerán en la aplicación de mensajes.

Cerberus no agota la batería porque no hay servicio en segundo plano ejecutándose: se activará solo si hay una solicitud por control remoto y se detendrá después [...].

Fuente: play.google.com

Kingosft Office

Kingsoft Office Reader for Android Free es un excelente visor de documentos de Microsoft Office para teléfonos moviles y tablets con sistema Android.

El programa se compone fundamentalmente de tres partes: el Word reader, el Excel reader y el PowerPoint reader. Gracias a cada uno de estos tres componentes podremos abrir en nuestro smartphone o tablet cualquier archivo creado con Office y que tenga diferentes extensiones como doc, .docx, .txt, .ppt, .xls y .xlsx.

Kingsoft Office Reader for Android Free es completamente funcional, permite la visualización de todos los elementos incrustados en los documentos, desde simples gráficos hasta las más sofisticadas imágenes. Además, nos permite tener la visión en pantalla completa y un potente gestor de archivos.

Fuente: es.kioskea.net

Referencias bibliográficas

Ableson, F.; Collins, C. y Sen, R. (2009). *Android: guía para desarrolladores*. Madrid: Editorial Anaya Multimedia.

Web:

<https://www.android.com/>
<https://developer.android.com/studio/intro/index.html?hl=es-419>

Impreso en los talleres gráficos de



EDITORIAL
MACRO®

Surquillo
Octubre 2016