



**UNIVERSIDAD AMERICANA DE ACAPULCO**  
"EXCELENCIA PARA EL DESARROLLO"

---

**FACULTAD DE INGENIERÍA EN COMPUTACIÓN**

INCORPORADA A LA UNIVERSIDAD NACIONAL

AUTÓNOMA DE MÉXICO

CLAVE DE INCORPORACIÓN 8852-16

**"IMPLEMENTACIÓN DE LOS DISPOSITIVOS  
PERIFÉRICOS EN EL SISTEMA OPERATIVO ANDROID"**

**T E S I S**

QUE PARA OBTENER EL TÍTULO DE

**INGENIERA EN COMPUTACIÓN**

PRESENTA

**IVONNE LÓPEZ MORALES**

DIRECTOR DE TESIS

**DR. RENÉ EDMUNDO CUEVAS VALENCIA**



**ACAPULCO, GUERRERO, MARZO 2017.**



Universidad Nacional  
Autónoma de México

Dirección General de Bibliotecas de la UNAM

**Biblioteca Central**



**UNAM – Dirección General de Bibliotecas**  
**Tesis Digitales**  
**Restricciones de uso**

**DERECHOS RESERVADOS ©**  
**PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

## **AGRADECIMIENTOS**

Agradezco a la Dra. Natividad Gutiérrez Chong, por darme la confianza y la oportunidad de participar en el proyecto de investigación SEP-CONACYT “Cultura política e Intelectuales Indígenas. Respuesta al resurgimiento étnico de América Latina” con clave 128183.

A los maestros que a lo largo de la carrera compartieron sus conocimientos y experiencias.

A la Ing. Eloísa Mercedes Vivas Villasana por el apoyo recibido durante el proceso de mi titulación.

A mi asesor Dr. René Edmundo Cuevas Valencia, por el asesoramiento en la revisión de la tesis.

Agradezco al † M.C. José Mario Martínez Castro, por los conocimientos otorgados en el transcurso del desarrollo de la presente tesis, por la disponibilidad que tenía para atenderme y por prepararme para la presentación de la misma.

Gracias por ser un excelente maestro, por su tiempo invertido en el salón de clases y por permitir aprender de él, sus enseñanzas, frases y consejos seguirán siempre presentes.

## **DEDICATORIA**

La conclusión de la presente tesis se debe al esfuerzo constante y dedicación que puse en ella. Este sueño por fin se ha realizado y no hubiese sido posible sin el apoyo de mi familia.

A mi madre, con su ejemplo he aprendido que con preparación y entrega se pueden alcanzar cosas extraordinarias.

A mi padre, por las historias y consejos recibidos.

A mi hermana, le agradezco por todo el cariño brindado y el ejemplo que he recibido de ella.

A mi sobrino por enseñarme muchas cosas a su corta edad, por hacerme reír a carcajadas y recordarme la pureza de los seres humanos.

A mis amigos gracias por sus palabras y consejos.

Agradezco a Dios y al universo por siempre cuidar de mí y guiarme por el camino correcto, por todas las alegrías que han llenado mi vida de bellos recuerdos.

Gracias por poner en mi camino a personas maravillosas y momentos perfectos.

# ÍNDICE

	Página
AGRADECIMIENTOS .....	0
DEDICATORIA.....	0
ÍNDICE .....	I
ÍNDICE DE FIGURAS.....	IV
ÍNDICE DE TABLAS.....	VI
INTRODUCCIÓN .....	1
PLANTEAMIENTO DEL PROBLEMA .....	3
JUSTIFICACIÓN .....	4
HIPÓTESIS .....	4
OBJETIVO GENERAL .....	4
OBJETIVO ESPECÍFICOS .....	5
<b>CAPÍTULO 1. MARCO TEÓRICO.....</b>	<b>7</b>
1.1 CARACTERÍSTICAS GENERALES DE LOS DISPOSITIVOS MÓVILES .....	8
1.1.1 <i>Movilidad</i> .....	9
1.1.2 <i>Tamaño reducido</i> .....	11
1.1.3 <i>Comunicación inalámbrica</i> .....	12
1.1.4 <i>Redes personales inalámbricas (WPAN)</i> .....	13
1.1.5 <i>Interacción: Interfaz Hombre-Máquina</i> .....	16
1.2 TIPOS DE DISPOSITIVOS MÓVILES .....	17
1.2.1 <i>Handheld PC</i> .....	18
1.2.2 <i>Consolas portátiles</i> .....	20
1.2.3 <i>Teléfono móvil</i> .....	21
1.2.4 <i>Laptop</i> .....	22
1.2.5 <i>Smartphones</i> .....	23
1.2.6 <i>Tabletas</i> .....	25
1.2.7 <i>Smartwatch</i> .....	26
1.2.8 <i>Google Glass</i> .....	27
1.3 SISTEMA OPERATIVO GNU/LINUX .....	29
1.3.1 <i>Características</i> .....	30
1.3.2 <i>Distribución Ubuntu</i> .....	32
1.3.3 <i>Ventajas</i> .....	33
1.4 HARDWARE: PERIFÉRICOS Y SENSORES .....	34
1.4.1 <i>Periféricos</i> .....	35
1.4.2 <i>Sensores</i> .....	37
1.5 WEBSERVICES.....	47
1.5.1 <i>Arquitecturas Webservices</i> .....	48
1.5.2 <i>Ventajas</i> .....	49
1.5.3 <i>Web Service RESTfull</i> .....	50
1.6 SERVIDOR WEB APACHE .....	51
1.7 METODOLOGÍA DE DESARROLLO.....	53
1.7.1 <i>Metodología RAD</i> .....	53
1.7.2 <i>Etapas de metodología RAD</i> .....	54

1.7.3	Ventajas.....	56
1.7.4	Desventajas.....	57
<b>CAPÍTULO 2. SISTEMA OPERATIVO ANDROID.....</b>		<b>59</b>
2.1	¿QUÉ ES ANDROID?.....	59
2.2	EL INICIO DE ANDROID.....	60
2.3	CARACTERÍSTICAS.....	63
2.4	ARQUITECTURA DEL SISTEMA OPERATIVO ANDROID.....	64
2.4.1	Núcleo de Linux.....	65
2.4.2	Android Runtime.....	65
2.4.3	Librerías Nativas.....	66
2.4.4	Entorno de Aplicación.....	67
2.4.5	Aplicaciones.....	69
2.5	VERSIONES DE ANDROID.....	70
2.5.1	Versión 1.0 Apple Pie.....	70
2.5.2	Versión 1.1 Banana Bread.....	71
2.5.3	Versión 1.5 Cupcake.....	71
2.5.4	Versión 1.6 Donut.....	72
2.5.5	Versión 2.0 / 2.1 Eclair.....	73
2.5.6	Versión 2.2 Froyo.....	74
2.5.7	Versión 2.3 Gingerbread.....	75
2.5.8	Versión 3.0 / 3.1 / 3.2 Honeycomb.....	76
2.5.9	Versión 4.0 Ice Cream Sandwich.....	77
2.5.10	Versión 4.1 Jelly Bean.....	78
2.5.11	Versión 4.4 Kit Kat.....	79
2.5.12	Versión 5.0 Lollipop.....	80
2.5.13	Versión 6.0 Marshmallow.....	82
2.5.14	Características y especificaciones actuales.....	83
2.6	ENTORNO DE PROGRAMACIÓN ANDROID STUDIO.....	84
2.6.1	Requisitos del sistema.....	86
2.6.2	Instalación de Android Studio en Linux.....	86
2.7	PRIMER PROYECTO HOLA MUNDO.....	96
2.8	ELEMENTOS DE UN PROYECTO ANDROID.....	100
2.8.1	Creación de un emulador (Android Virtual Device).....	104
2.9	BASES DE DATOS (ALMACENAMIENTO DE DATOS).....	109
2.9.1	SQLite.....	110
<b>CAPÍTULO 3. DESARROLLO DE APLICACIÓN: BIENES Y RAÍCES.....</b>		<b>114</b>
3.1	APLICACIÓN BIENES Y RAÍCES.....	114
3.2	PLANIFICACIÓN DE REQUISITOS.....	114
3.2.1	Modelado de Gestión.....	115
3.2.2	Diagrama de Casos de Uso.....	115
3.2.3	Diagrama de Actividades.....	116
3.2.4	Diagrama de Estados.....	118
3.2.5	Diagrama de Clase.....	119
3.2.6	Diagrama de Componentes.....	120
3.3	DISEÑO DE USUARIO.....	121
3.3.1	Interfaz de Usuario.....	121
3.3.2	Modelado de datos.....	125
3.3.3	Normalización.....	127
3.3.4	Modelo Entidad relación.....	130
3.4	DESARROLLO.....	132
3.4.1	Webservice.....	132

3.4.2	<i>Aplicación Android</i> .....	135
3.4.3	<i>Sincronización</i> .....	151
3.5	TRANSICIÓN .....	152
3.5.1	<i>Funcionamiento</i> .....	152
<b>CAPÍTULO 4.</b>	<b>RESULTADOS</b> .....	<b>157</b>
4.1	APLICACIÓN ANDROID BIENES Y RAÍCES .....	157
<b>CAPÍTULO 5.</b>	<b>CONCLUSIONES Y TRABAJO A FUTURO</b> .....	<b>164</b>
5.1	CONCLUSIONES .....	164
5.2	TRABAJO A FUTURO .....	165
<b>BIBLIOGRAFÍA</b>	.....	<b>166</b>
<b>ANEXOS</b>	.....	<b>170</b>

# ÍNDICE DE FIGURAS

Página

FIGURA 1.1 DISPOSITIVOS MÓVILES.....	9
FIGURA 1.2 TABLET ASUS.....	11
FIGURA 1.3 CLASIFICACIÓN DE LAS REDES INALÁMBRICAS.....	12
FIGURA 1.4 DECT OPERA EN LA BANDA DE FRECUENCIA DE 1.880 Y 1900 MHZ.....	14
FIGURA 1.5 IBM SIMON, PRIMER SMARTPHONE.....	17
FIGURA 1.6 PSION CON DISEÑO CLÁSICO CLAMSHELL.....	19
FIGURA 1.7 CONSOLAS PORTÁTILES.....	20
FIGURA 1.8 MOTHERBOARD ULTRABOOK ASUS ZENBOOK UX3 1E.....	22
FIGURA 1.9 TABLET SAMSUNG GALAXY TAB 3.....	25
FIGURA 1.10 RELOJ INTELIGENTE.....	27
FIGURA 1.11 GOOGLE GLASS.....	27
FIGURA 1.12 LOGOTIPO DEL SISTEMA OPERATIVO LINUX.....	29
FIGURA 1.13 LOGOTIPO DISTRIBUCIÓN UBUNTU.....	32
FIGURA 1.14 DASH DE UBUNTU.....	34
FIGURA 1.15 PERIFÉRICOS INTERNOS.....	35
FIGURA 1.16 PERIFÉRICOS.....	36
FIGURA 1.17 SATÉLITES GPS.....	43
FIGURA 1.18 PANTALLA URL API KEY.....	44
FIGURA 1.19 REGISTRO DE PROYECTO.....	45
FIGURA 1.20 OBTENCIÓN DE LA CLAVE.....	45
FIGURA 1.21 AÑADIR CLAVE.....	46
FIGURA 1.22 WEBSERVICE CON ARQUITECTURA REST.....	47
FIGURA 1.23 LOGOTIPO DEL SERVIDOR APACHE.....	52
FIGURA 1.24 FLUJO DE PROCESO DE RAD.....	54
FIGURA 1.25 ETAPAS DE MODELADO DE LA APLICACIÓN.....	55
FIGURA 2.1 ANDY EL ROBOT, LOGO DE LA COMPAÑÍA ANDROID INC.....	60
FIGURA 2.2 HTC DREAM, PRIMER SMARTPHONE CON SISTEMA OPERATIVO ANDROID.....	62
FIGURA 2.3 ARQUITECTURA POR CAPA DEL SISTEMA OPERATIVO ANDROID.....	64
FIGURA 2.4 FUNCIONAMIENTO DALVIK.....	66
FIGURA 2.5 INICIO O LAUNCHER EN ANDROID LOLLIPOP.....	69
FIGURA 2.6 LOGO VERSIÓN 1.0 APPLE PIE.....	70
FIGURA 2.7 LOGO VERSIÓN 1.1 BANANA BREAD.....	71
FIGURA 2.8 LOGO VERSIÓN 1.5 CUPCAKE.....	72
FIGURA 2.9 LOGO VERSIÓN 1.6 DONUT.....	73
FIGURA 2.10 LOGO VERSIÓN 2.0 ECLAIR.....	74
FIGURA 2.11 LOGO VERSIÓN 2.2 FROYO.....	75
FIGURA 2.12 LOGO VERSIÓN 2.3 GINGERBREAD.....	76
FIGURA 2.13 LOGO VERSIÓN 3.0 HONEYCOMB.....	77
FIGURA 2.14 LOGO VERSIÓN 4.0 ICE CREAM SANDWICH.....	78
FIGURA 2.15 LOGO VERSIÓN 4.1 JELLY BEAN.....	79
FIGURA 2.16 LOGO VERSIÓN 4.4 KIT KAT.....	80
FIGURA 2.17 LOGO VERSIÓN 5.0 LOLLIPOP.....	81
FIGURA 2.18 LOGO VERSIÓN 6.0 MARSHMALLOW.....	82
FIGURA 2.19 PORCENTAJE DE VERSIONES EN DISPOSITIVOS ANDROID.....	83
FIGURA 2.20 PÁGINA DE DESCARGA ANDROID SDK.....	87
FIGURA 2.21 SELECCIÓN DE SISTEMA OPERATIVO.....	87
FIGURA 2.22 COMPROBAR VERSIÓN DE JAVA INSTALADO.....	88
FIGURA 2.23 INSTALACIÓN DE JAVA.....	89
FIGURA 2.24 VENTANA AL EJECUTAR ARCHIVO ANDROID.SH.....	89



FIGURA 2.25 VENTANA INICIO DE INSTALACIÓN ANDROID STUDIO.....	90
FIGURA 2.26 INSTALACIÓN PERSONALIZADA.....	90
FIGURA 2.27 INSTALACIÓN DE COMPONENTES.....	91
FIGURA 2.28 COMPONENTES A INSTALAR.....	91
FIGURA 2.29 DESCARGA DE LOS COMPONENTES NECESARIOS.....	92
FIGURA 2.30 CONFIGURACIÓN ANDROID STUDIO.....	92
FIGURA 2.31 DESCARGA DE PAQUETES.....	93
FIGURA 2.32 VENTANA CREACIÓN DEL LANZADOR.....	94
FIGURA 2.33 UBICACIÓN ARCHIVO STUDIO.SH.....	95
FIGURA 2.34 CREAR LANZADOR EN ANDROID STUDIO.....	95
FIGURA 2.35 CREAR UN NUEVO POYECTO EN ANDROID STUDIO.....	96
FIGURA 2.36 NOMBRE DE LA APLICACIÓN.....	97
FIGURA 2.37 SELECCIÓN DE SDK DE LA APLICACIÓN.....	97
FIGURA 2.38 SELECCIÓN DE ACTIVIDAD.....	98
FIGURA 2.39 NOMBRAR ELEMENTOS DE LA ACTIVIDAD.....	99
FIGURA 2.40 HERRAMIENTAS EN ANDROID STUDIO.....	100
FIGURA 2.41 VENTANA DE SELECCIÓN DE DISPOSITIVOS.....	105
FIGURA 2.42 CONFIGURACIÓN PERFIL DEL HARDWARE.....	106
FIGURA 2.43 VENTANA IMAGEN DEL SISTEMA.....	107
FIGURA 2.44 VENTANA DE VERIFICACIÓN DE LA INFORMACIÓN.....	107
FIGURA 2.45 VENTANA SELECCIONAR EMULADOR.....	108
FIGURA 2.46 ARQUITECTURA SQLITE.....	112
FIGURA 3.1 DIAGRAMA DE CASOS DE USO.....	116
FIGURA 3.2 DIAGRAMA DE ACTIVIDADES.....	117
FIGURA 3.3 DIAGRAMA DE ESTADOS.....	118
FIGURA 3.4 DIAGRAMA DE CLASE.....	119
FIGURA 3.5 DIAGRAMA DE COMPONENTES.....	120
FIGURA 3.6 PANTALLA DE BIENVENIDA E INICIO DE SESIÓN.....	121
FIGURA 3.7 PROPIEDADES EN ALTA.....	122
FIGURA 3.8 ACTIVIDAD ALTA NUEVA PROPIEDAD.....	123
FIGURA 3.9 ACTIVIDAD ACTUALIZAR PROPIEDAD.....	124
FIGURA 3.10 MODELO ENTIDAD RELACIÓN.....	130
FIGURA 3.11 WEBSERVICE INICIO SESIÓN.....	153
FIGURA 3.12 WEBSERVICE PANEL DE USUARIO.....	154
FIGURA 3.13 DIÁLOGO CONFIRMACIÓN ELIMINAR PROPIEDAD.....	154
FIGURA 3.14 WEBSERVICE AGREGAR / MODIFICAR PROPIEDADES.....	155
FIGURA 3.15 PANTALLA DE INICIO Y BIENVENIDA.....	158
FIGURA 3.16 INICIO DE SESIÓN Y PANEL DE USUARIO.....	159
FIGURA 3.17 PANTALLA ALTA DE PROPIEDAD Y MAPA.....	160
FIGURA 3.18 PANTALLA ACTUALIZAR Y ELIMINAR PROPIEDAD.....	161
FIGURA 3.19 PANTALLA SINCRONIZAR BASE DE DATOS.....	162

# ÍNDICE DE TABLAS

	Página
TABLA 1.1 CARACTERÍSTICAS DE LOS DISPOSITIVOS MÓVILES.....	8
TABLA 1.2 PERIFÉRICOS.....	36
TABLA 1.3 TIPOS DE SENSORES SOPORTADOS EN ANDROID .....	39
TABLA 1.4 FRAMEWORK PARA SENSORES DE LA PLATAFORMA ANDROID .....	41
TABLA 2.1 REQUISITOS DEL SISTEMA (SDK, 2014) .....	86
TABLA 3.1 TABLA USUARIOS .....	126
TABLA 3.2 TABLAS PRIMERA FORMA NORMAL .....	128
TABLA 3.3 TABLAS CON SEGUNDA FORMA NORMAL .....	129
TABLA 3.4 DICCIONARIO DE DATOS .....	131

# INTRODUCCIÓN

En los últimos años se ha visto una creciente tendencia al uso de los dispositivos móviles, la facilidad que brinda para conectarnos a Internet, hacer video-llamadas, mandar y recibir mensajes e incluso hasta poder reservar un boleto de avión con un Smartphone, todo lo anterior se ha popularizado en poco tiempo.

Los nuevos teléfonos móviles ya poseen un sistema operativo el cual brinda la opción de poder instalar y remover aplicaciones dependiendo de los gustos y necesidades.

La presente, una investigación que tiene por objetivo desarrollar una aplicación que incorpore el uso de los sensores de localización, GPS, la cámara del celular, y las APIs de los mapas Google, que se utilizan en los nuevos Smartphones con Sistema Operativo Android.

Se expone en el caso de estudio, la creación de un catálogo de bienes y raíces virtual, dirigida a los agentes de ventas, el cual permite ingresar la información de las propiedades, por medio de mapas y fotos se conoce la ubicación y la construcción del inmueble así también los espacios en donde se encuentran, recámaras y baños.

Capítulo 1. En este apartado se narra la historia y sus antecedentes a las comunicaciones inalámbricas y móviles, cómo surge la necesidad de contar con dispositivos móviles, su evolución y las características que poseen.

Se explica y describe acerca de la API (Interfaz de Programación de Aplicaciones), los sensores de localización en Android y la cámara.

Capítulo 2. Se explica de manera detallada como se desarrolló el Sistema Operativo Android, como está integrado y el uso del entorno de desarrollo para trabajar sobre él.

Capítulo 3. En esta sección se presenta el caso de estudio de la aplicación bienes raíces, la interfaz de usuario que se desarrolló y los diagramas que dieron lugar a la metodología y desarrollo de la aplicación.

Capítulo 4. Por último, se tiene un reporte global de los logros a destacar y aportes que se proporcionan al presentar este trabajo de tesis en el apartado de las conclusiones, así como una visión de los trabajos pendientes para continuar a futuro.

## PLANTEAMIENTO DEL PROBLEMA

Debido a la demanda de Smartphones y la incursión de aplicaciones móviles es inevitable no voltear a ver el desarrollo de aplicaciones móviles en específico los manipulados por el sistema operativo Android.

Los Smartphones incluyen varias características que complementan la interacción que se tiene con el usuario, al incluir micrófonos, cámaras, acelerómetros, podómetros, indicadores de temperatura o detectores de luz, los Smartphones se han convertido en dispositivos extra sensoriales, que permiten aumentar la propia percepción.

Google ofrece herramientas y guías de desarrollo a los programadores, para complementar las aplicaciones con el uso de APIs, sensores y periféricos, éstas herramientas permiten aprovechar a gran escala el hardware y software del teléfono celular y construir aplicaciones más robustas que permiten la interacción entre usuario, dispositivo y el medio que lo rodea.

Actualmente es muy común que escuelas, gobiernos, empresas y hoteles empleen aplicaciones móviles para ofrecer una educación más interactiva, dar a conocer sus productos, reforzar los servicios que ofrecen a sus clientes o recomendar lugares a turistas, lo que permite que personas con un dispositivo móvil puedan interactuar o conocer lugares de su interés.

En el ámbito de compra y venta de bienes raíces existen empresas que no cuentan con una aplicación móvil la cual cubra sus necesidades principales y ayude a conectar compradores con vendedores de inmuebles.

## **JUSTIFICACIÓN**

Si se está interesado en introducirse al mundo del desarrollo de las aplicaciones móviles se encontrarán con muchas dudas acerca de ¿Cómo empezar?, buscando en Internet no se encontró suficiente información y la mayoría de los libros que abordan temas más laboriosos como: Obtener la información del GPS, el uso de periféricos con los que cuenta el celular y base de datos, no son suficientes y sus involucrados han publicado en inglés.

Es por eso que el tema de esta tesis nace por el interés de investigar, acerca de la programación en dispositivos móviles específicamente Android, a consecuencia de la falta de información que existe en la Universidad Americana de Acapulco, como un apoyo a las futuras generaciones para que sirva de referencia si se desea desarrollar una aplicación móvil robusta, que integre los periféricos de la cámara, el GPS y una base de datos local para sincronizar datos a un Webservice.

## **HIPÓTESIS**

Desarrollar una aplicación móvil, para empresas bienes y raíces, que permita agilizar, la obtención y manejo de los datos recabados por los agentes de bienes raíces, permitiendo la distribución de estos a un Webservice local, permitirá reducir tiempo, costos, por lo que mejorará la calidad de servicio y hacer más eficiente el sistema de venta y renta de inmuebles.

## **OBJETIVO GENERAL**

Desarrollar una aplicación para uso de agentes de ventas en bienes y raíces, en general que permita el manejo de los dispositivos periféricos aprovechando la filosofía de Software Libre.

## OBJETIVO ESPECÍFICOS

1. Investigar y analizar la evolución y características de dispositivos móviles, sistemas operativos móviles, sistemas de bienes y raíces, y la programación apoyada por el sistema operativo Android.
2. Instalar los programas para el desarrollo de la aplicación.
  - a) Android Studio
  - b) Xampp
3. Desarrollar una base de datos para soportar la aplicación.
4. Desarrollar un Webservice.
5. Desarrollar aplicación Android con base de datos local y que permita sincronización de datos con el Webservice.
6. Sincronizar los datos almacenados de manera local en la aplicación con el Webservice.
7. Analizar mediante un caso de estudio la aplicación de bienes y raíces.



# CAPÍTULO 1

## MARCO TEÓRICO





## CAPÍTULO 1. MARCO TEÓRICO

Actualmente, se es testigo de una de las más grandes revoluciones y no precisamente con armas sino tecnológica de comunicaciones inalámbricas, una revolución similar a la que protagonizaron en su momento la electricidad, la televisión, la computadora o las mismas comunicaciones alámbricas, que supusieron nuevos paradigmas. El no tener que estar conectado a cientos de cables lo hace todo más práctico, fácil y rápido.

La movilidad implica una ventaja sumamente alta, hoy en día los teléfonos móviles tienen acceso a Internet, sistema de mensajería y llamadas en cualquier lugar mientras los usuarios se encuentran en movimiento, sin tener que estar en una ubicación fija.

El hecho de que el punto de entrada en la red de comunicaciones no esté ligado a una ubicación fija y que el medio de transmisión ya esté preparado favorece su expansión, que puede ser más rápida que la de cualquier otro tipo de tecnología. Existe un ejemplo que lo corrobora: En solo cinco años de existencia, la telefonía móvil ya tuvo más usuarios que la telefonía fija. (Prieto Blázquez, Ramírez Vique, Morillo Pozo, & Domingo Prieto, 2011)

De acuerdo con el informe de movilidad de Ericsson anunciado en noviembre del 2014, se prevé que para el año del 2020 el 90% de la población mundial tendrá un Smartphone. (Ericsson, 2014).

Es sabido que la tecnología ha aumentado considerablemente, y esto ha permitido alcanzar logros que dentro de esta profesión ayuda a tener conocimientos amplios e innovadores, que permiten colaborar y aportar al crecimiento tecnológico en México.

Así pues el presente documento está enfocado a desarrollar la investigación de lo que en ella trata: Aplicaciones Móviles.

## 1.1 Características generales de los dispositivos móviles

Como se menciona anteriormente, la telefonía móvil ha ido aumentando aceleradamente, sobre todo por la llegada de los dispositivos móviles inteligentes “Smartphones”. (TICbeat, 2014)

Cuando se habla de dispositivos móviles solo se piensa en teléfonos móviles y Smartphones, sin embargo un dispositivo móvil puede ser cualquier dispositivo que cuente con las características de la Tabla 1.1:

Tabla 1.1 Características de los dispositivos móviles

Fuente: Reedición propia

	Son de tamaño pequeño
	Capacidades especiales de procesamiento
	Conexión inalámbrica
	Memoria Interna y externa
	Interacción con pantalla o teclado
	Es empleado para una función principal con tando con otras funciones secundarias
	Uso individual

Si se analizan las características antes mencionadas con los dispositivos de la actualidad, se puede decir que, son dispositivos móviles:

Laptop, kindle, ipod, smartwatch, Nintendo 3Ds, teléfono inalámbrico, celular, tablet, cámaras fotográficas, entre otros (Figura 1.1)



**Figura 1.1 Dispositivos móviles**

Fuente: Extraída de <https://arielrodrigoreyes.wordpress.com/que-es-un-dispositivo-movil/>

### **1.1.1 Movilidad**

Es imposible imaginar la vida cotidiana sin movilidad, los seres humanos están tan acostumbrados a la movilidad que no podrían vivir sin ella. Pero, ¿Qué es la movilidad?

Si se remonta al año 1981 en el mes de Junio, en donde se dio a conocer la primera computadora portátil, la OSBORNE 1 con un peso de 10 kilos, puede percatarse que eran los inicios de las computadoras portátiles, un nuevo paradigma, una nueva manera de usar una computadora, dando la posibilidad de trabajar no necesariamente en casa u oficina.

La movilidad no solamente empezó a surgir con las computadoras, sino también con otros aparatos electrónicos, como son los teléfonos, consolas de videojuegos e incluso impresoras. El ser humano ha ido transformando sus necesidades, con el pasar de los

años, la manera de pensar y forma de comunicarse ya no es la misma. Con los nuevos dispositivos portátiles le permiten estar en comunicación constante.

La tecnología avanza constantemente y en el presente hay dispositivos tan ligeros como la laptop Lenovo LaVie Z HZ550 que tiene un peso de 781grs, que permite su fácil transportación. La movilidad es entonces eso, la propiedad de transportar un dispositivo con facilidad, por ende tienen que ser pequeños, prácticos y livianos.

La movilidad en redes inalámbricas, es aquella que permite seguir “conectados” mientras las personas se desplazan de un lugar a otro. (ComputerWorld, 2014).

Aquellos que cuentan con un Smartphone con plan de renta, ni siquiera se percatan que al ir caminando este va cambiando la conexión de antenas móviles, dependiendo de la cercanía que se tenga con ellas, el cambio es tan rápido que ni siquiera se nota ni se pierde la Red. Eso es movilidad.

Cualquier dispositivo móvil debería funcionar y poder ser usado de forma fiable mientras se está en movimiento, independientemente de la proximidad de una fuente de energía (enchufe) o de una conexión física a Internet. Para facilitar la portabilidad, los dispositivos móviles suelen contener baterías recargables, que permiten varias horas de operación y funcionalidad sin necesidad de acceso a un cargador o a una fuente de energía externa. Aunque son móviles, estos dispositivos pueden sincronizarse con un sistema de escritorio para actualizar aplicaciones y datos. (Pozo, 2011)

### 1.1.2 Tamaño reducido

*Una de las cualidades de un dispositivo móvil, es la comodidad al ser transportado e igualmente para ser empleados, sin necesidad de utilizar algún soporte. (Pozo, 2011)*

Se denomina a algunos dispositivos handhelds o palmtops debido a que, en mayor o menor medida, sus dimensiones son parecidas a las de una mano. Por el contrario, otros dispositivos móviles ligeramente más grandes, como podría ser el caso de las tabletas, no suelen denominarse así. Un dispositivo móvil típico lo podemos llevar con una mano y cabe en un bolsillo o en un pequeño bolso. Algunos dispositivos móviles se pueden desplegar o desdoblar desde un modo portable o compacto a un tamaño ligeramente superior y descubrir teclados o pantallas más grandes. (Figura 1.2)



**Figura 1.2 Tablet ASUS**

**Fuente:** Extraída de [https://www.asus.com/latin/Tablets/Eee\\_Pad\\_Transformer\\_TF101](https://www.asus.com/latin/Tablets/Eee_Pad_Transformer_TF101)

Los dispositivos móviles pueden tener pantallas táctiles o pequeños teclados numéricos para recibir datos de entrada y mantener, al mismo tiempo, su tamaño pequeño e independencia de dispositivos de interfaz externos. (Soriano, 2014).

### 1.1.3 Comunicación inalámbrica

El inicio de la comunicación telefónica comenzó el 16 de Agosto de 1876 cuando el científico e inventor Graham Bell hizo la primera llamada telefónica entre 2 poblados situados a 10 km de distancia. Posteriormente en 1973 el estadounidense Martin Cooper realizó la primera llamada telefónica desde un móvil haciendo realidad esta nueva tecnología que nos permitiría comunicarnos en cualquier parte del mundo.

La definición de comunicaciones inalámbricas engloba desde una comunicación Bluetooth entre un teléfono móvil y una computadora portátil hasta una comunicación de dos terminales de telefonía móvil GSM. (Prieto Blázquez, Ramírez Vique, Morillo Pozo, & Domingo Prieto, 2011) Incluso la comunicación verbal entre dos personas sería una comunicación inalámbrica: Utilizan el aire como un canal para el intercambio de información.

#### Clasificación

De acuerdo al alcance, se establecen en la Figura 1.3, tres grandes grupos:

- Redes de área personal inalámbrica (WPAN Wireless Personal Area Networks).
- Redes de área local inalámbrica (WLAN Wireless Local Area Networks).
- Redes de área extendida inalámbrica (WWAN Wireless Wide Area Networks).

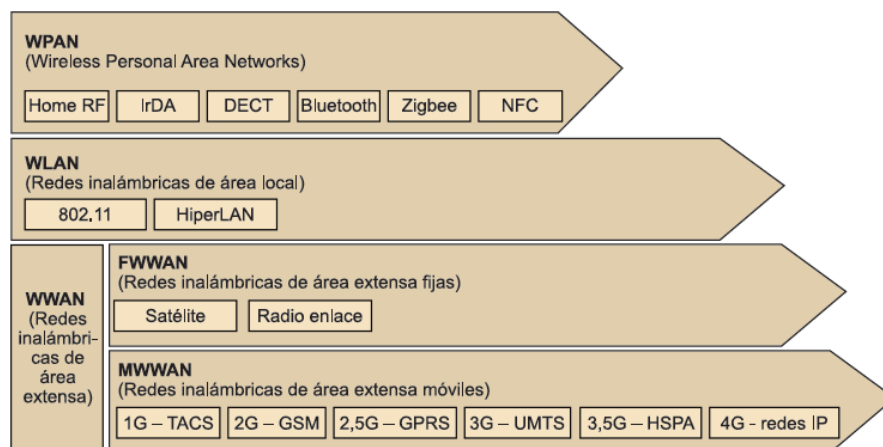


Figura 1.3 Clasificación de las Redes Inalámbricas

Fuente: Extraída de Introducción a los sistemas inalámbricos, 2011.

Se diferencian dos tipos de WWAN, según quién controle su acceso:

- Comunicación fija (FWWAN: Fixed Wireless Wide Area Networks).
- Comunicación móvil (MWWAN: Mobile Wireless Wide Area Networks).

#### 1.1.4 Redes personales inalámbricas (WPAN)

Las WPAN presentan una importante limitación de alcance: Los dispositivos que pretenden comunicarse han de estar poco separados. Generalmente, se acepta como límite el espacio de una habitación o un despacho.

##### Bluetooth

Es una especificación regulada por el grupo de trabajo IEEE 802.15.1, que permite la transmisión de voz y datos entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda ISM de 2,4 GHz.(Figura 1.4)



Figura 1.4 Logotipo de Tecnología Bluetooth

Fuente: Extraída de <https://www.bluetooth.com/>

Permite conectar inalámbricamente, a un alcance aproximadamente de 10 metros, diferentes dispositivos electrónicos, como asistentes digitales personales (PDA), teléfonos móviles, ordenadores portátiles, etcétera., lo que facilita y garantiza la interoperabilidad entre dispositivos de diferentes fabricantes. (Bluetooth, 2015)

## DECT

La tecnología Digital Enhanced Cordless Telecommunications (DECT) aparece oficialmente en el año de 1988, como una necesidad de la telefonía, para que las comunicaciones analógicas evolucionaran hacia comunicaciones digitales.

Esta tecnología se utiliza principalmente para sistemas telefónicos en el hogar o empresa (Figura 1.5), donde se aprovecha la posibilidad de ofrecer sistemas expandibles, más movilidad, permite múltiples terminales inalámbricos en una misma base, otorgando comunicaciones privadas y seguras. (DECT, 2014)



Figura 1.5 DECT Opera en la banda de frecuencia de 1.880 y 1900 MHz

Fuente: Extraída de [http://www.philips.com.hk/c-p/D1301B\\_90/-](http://www.philips.com.hk/c-p/D1301B_90/)

## IrDa

Es una asociación que integra más de ciento sesenta compañías. El estándar IrDA utiliza el espectro de frecuencia de infrarrojo para transmitir información.

El uso de la tecnología IrDA se ha extendido mucho, sobre todo en los años noventa y a principios de siglo, a causa de su bajo costo de implementación y su bajo consumo de batería. Además, es muy flexible y capaz de adaptarse fácilmente a un gran número de aplicaciones y dispositivos, como PDA, teléfonos, impresoras u ordenadores portátiles.



Los dispositivos que utilizan la IrDA se comunican mediante el uso del diodo LED (Light Emitting Diode). Es necesario que estos dispositivos estén alineados los unos con los otros. La desviación máxima permitida es de 30°. (Huertas, 2015)

## NFC

Permite la transmisión de datos de una manera simple entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda ISM de 13,56 MHz. Dado que la conexión se produce cuando dos dispositivos NFC están muy próximos entre sí, a menos de 20 centímetros, la comunicación es inherentemente segura. Como se muestra en la Figura 1.6.

Fue aprobado por el estándar ISO 18092 en el 2003. Philips, Sony y Nokia formaron el NFC Foro para avanzar en el desarrollo de las especificaciones NFC y velar por su interoperabilidad.

La tecnología NFC es una extensión del estándar ISO/IEC-14443 para tarjetas de proximidad sin contacto que combina la interfaz de una tarjeta inteligente y un lector en un único dispositivo, lo que la hace compatible con toda la infraestructura de pago sin contacto que existe actualmente. (NFC, 2014)



**Figura 1.6** Funcionamiento NFC en sistema de pago

Fuente: Extraída de <https://www.elandroidelibre.com/wp-content/uploads/2016/05/ing-direct-pagos-desde-movil-nfc.jpg>

## **Zigbee**

Es un estándar de comunicaciones inalámbricas, regulado por el grupo de trabajo IEEE 802.15.4 en el 2004, que permite habilitar redes inalámbricas con capacidades de control, y monitorizar que sean seguras, de bajo consumo energético y de bajo coste de procesador, de manera bidireccional.

Es promovida por la Zigbee Alliance, una comunidad internacional de más de cien compañías, como Motorola, Mitsubishi, Philips, Samsung, Honeywell y Siemens, entre otras. De hecho, Zigbee no es una tecnología, sino un conjunto estandarizado de soluciones que pueden ser implementadas por cualquier fabricante. (Prieto Blázquez, Ramírez Vique, Morillo Pozo, & Domingo Prieto, 2011)

### **1.1.5 Interacción: Interfaz Hombre-Máquina**

*Se entiende por interacción el proceso de uso que establece un usuario con un dispositivo. Entre otros factores, en el diseño de la interacción intervienen disciplinas como la usabilidad y la ergonomía. (Pozo, 2011)*

Un usuario realiza la interacción con un dispositivo móvil mediante la interfaz de usuario. Existen grandes diferencias en cuanto a la interacción que hay entre un PC y los dispositivos móviles (e incluso, entre distintos dispositivos móviles). Por ejemplo, la forma de utilizar los enlaces, es muy distinta entre un usuario que maneja un puntero de PDA y un usuario que avanza con teclado de un teléfono móvil saltando entre enlaces.

Eso sin contar con los distintos tipos de teclado móviles, por no mencionar que, en ambos casos, desaparece el evento mouseover y, por tanto, los tooltips y otras pistas para deducir el destino de un enlace que normalmente utilizamos cuando accedemos a Internet desde un ordenador.

Otros conceptos importantes, como la distancia en pantalla, el orden de los elementos, el tamaño de los textos y las imágenes, la forma de escanear visualmente las páginas, etc., cambian de unos dispositivos a otros.

Teniendo en cuenta las grandes diferencias físicas (pantalla, teclados, punteros, etc.), que acarrearán formas diferentes de interactuar, y el número, cada vez mayor, de capacidades de los dispositivos móviles, podemos dudar si será posible, e incluso acertado, alcanzar la famosa Web única. (Pozo, 2011)

## 1.2 Tipos de dispositivos móviles

20 años después de la primera llamada realizada con un móvil, en 1993 la empresa IBM lanzó al mercado el primer Smartphone oficialmente reconocido, bajo el nombre de "Simon" (Figura 1.7) este modelo de teléfono se adelantaba a su tiempo incorporando una pantalla táctil, agenda electrónica, calendario, bloc de notas, calculadora, reloj, juegos, correo electrónico, incluso permitía la recepción y envío de FAX. (Quees, 2014)



Figura 1.7 IBM Simon, primer smartphone

Fuente: Extraída de <http://www.neoteo.com/ibm-simon-el-primero-smartphone-de-la-historia>

Durante los años siguientes de la década de los 90 y principios del 2000 los fabricantes de móviles lanzaban al mercado diversos modelos de teléfonos inteligentes sin conseguir el éxito y la penetración de mercado deseado resultando un dispositivo caro y con escaso valor para los consumidores, pero en el año 2007 la empresa Apple

presenta ante todo el mundo su famoso iPhone revolucionando el concepto de teléfono inteligente al añadir una pantalla táctil a color, una excelente conexión a Internet así como unas aplicaciones realmente útiles para los consumidores, todo ello acompañado de una experiencia de uso jamás adquirida por ningún teléfono, su éxito batió todos los records de venta pronosticados convirtiéndose en la empresa líder del mercado de los Smartphone.

Durante los sucesivos años empresas como HTC, Samsung y LG, diseñaron y fabricaron sus Smartphone bajo el sistema operativo de código abierto Android compitiendo directamente con el iPhone e inundando el mercado de una amplia gama de teléfonos inteligentes alcanzables por los distintos perfiles de consumidores.

El futuro de los Smartphone pasa por converger y unificar la computadora de escritorio, las tabletas, teléfonos, relojes y lentes inteligentes en un único gadget capaz de desarrollar cualquier tipo de tarea y trabajar con cualquier tipo de software, basado en un sistema operativo con inteligencia artificial que nos permita interactuar con él para facilitarnos nuestro día a día. (Neoteo, 2014)

### **1.2.1 Handheld PC**

El concepto de *Handheld* PC es muy antiguo. El diseño puede ser similar al de un portátil, en el que la pantalla se dobla sobre el teclado y crea una carcasa compacta alrededor del dispositivo. Por esta razón, los *Handheld* PC fueron comúnmente conocidos como dispositivos *clamshell*<sup>1</sup>.

Los dispositivos clamshell, aparecieron mucho antes de que estuvieran disponibles los primeros PDA. (Pozo, 2011).

---

<sup>1</sup> Dispositivo que tiene dos o más secciones que se pliegan por una visagra.

A mediados de los años ochenta, Psion presentó un organizador-agenda que permitía a los usuarios ejecutar aplicaciones financieras, científicas y de datos de forma local en el dispositivo. (Figura 1.8)



**Figura 1.8 PSION con diseño clásico clamshell**

**Fuente:** Extraída de <http://www.topdesignmag.com/the-most-memorable-gadgets-from-the-90s/>

A principio de los años noventa, Psion lanzó un ordenador clamshell más funcional, el cual tenía un teclado y una pantalla con una interfaz gráfica de usuario (GUI) para ejecutar aplicaciones más sofisticadas.

Otras compañías, como Casio, lanzaron ofertas similares, a menudo demasiado basadas en sistemas operativos propietarios específicos del dispositivo. Estos sistemas operativos no soportaban aplicaciones de terceras partes, lo que era una limitación muy importante.

Una de las funciones más importantes de los organizadores digitales es la sincronización con las computadoras personales. Esto permite la actualización del directorio, haciendo que la información del computador y de la agenda digital sea la misma. Los handheld PC no se usan para sustituir a los portátiles, sino para complementarlos.

El uso común de un handheld PC no es el de ordenador general, sino el de un dispositivo para guardar información. (Prieto Blázquez, Ramírez Vique, Morillo Pozo, & Domingo Prieto, 2011)

## 1.2.2 Consolas portátiles

Son dispositivos electrónicos muy ligeros que permiten jugar con videojuegos. A diferencia de las videoconsolas de sobremesa, en las portátiles: Los controles, la pantalla, los altavoces y la alimentación (baterías), están todos integrados en la misma unidad. Son de pequeño tamaño, para poder llevarlas fácilmente a cualquier lugar. (Videoconsolas, 2014) (Figura 1.9)

Al igual que pasó con las videoconsolas de sobremesa, las primeras portátiles fueron diseñadas, exclusivamente para jugar solamente videojuegos. Pero a partir de la sexta generación, han sido convertidas en pequeños centros multimedia.



Figura 1.9 Consolas portátiles

Fuente: Extraída de <http://videoconsolas.jimdo.com/videoconsolas-port%C3%A1tiles/>

Por mencionar algunas características de las consolas portátiles de la actualidad:

- Cámara frontal y trasera.
- Pantalla táctil.
- Altavoces y micrófonos integrados.
- Comunicaciones inalámbricas: Bluetooth, WiFi y 3G.
- Reproducción de Música, videos y fotos.
- Almacenamiento MicroSD.

- Cartuchos de videojuegos pequeños.
- Interacción usuarios en línea.
- Instalar aplicaciones.
- Gráficos excelentes.

### 1.2.3 Teléfono móvil

Es un equipo inalámbrico electrónico que accede a una red telefónica gracias a ondas electromagnéticas.

En 1973 la FCC, estaba a punto de otorgarle a AT&T el monopolio del sistema celular. Motorola ve que ella puede entrar en este mercado, ya que conocía el negocio de las comunicaciones personales y convoca a sus trabajadores a un concurso para que en seis semanas presenten un prototipo de teléfono portátil. El 3 de abril de 1973, Marty Cooper vicepresidente de Investigación y Desarrollo de Motorola en ese entonces, hace una llamada telefónica a Joel Engel, ingeniero jefe de AT&T y propulsor de la telefonía celular, utilizando el prototipo desarrollado por su compañía, certificándose de esta manera el nacimiento del teléfono celular.

El primer prototipo de teléfono móvil comercializado a gran escala fue el diseñado por el ingeniero Rudy Krolopp, de la compañía Motorola, en 1983. Este teléfono pesaba 0,74 kg, y tenía un valor de 4000 dólares. (Ecured, 2015)

A continuación algunos de los componentes con los que cuentan los teléfonos móviles.

- Un micrófono.
- Un altavoz.
- Una pantalla de cristal o táctil.
- Un teclado.
- Una antena.
- Una batería.
- Una placa de circuitos.

El móvil posee un microprocesador que realiza cálculos a gran velocidad. El microprocesador trata todas las tareas del teclado y de la pantalla, gestiona los comandos y controla las señales de la estación de base, además de coordinar las demás funciones. (Pozo, 2011)

## 1.2.4 Laptop

Se le denomina a una computadora compacta y capaz de ser transportable debido al peso liviano que posee. (DefinicionMx, 2014). Una laptop tiene las mismas capacidades que puede observarse en una computadora de escritorio, con el aditivo de poder tener una autonomía razonable como consecuencia del uso de una batería.

Fueron en un pasado un producto de gran costo, quedando relegadas por las computadoras de escritorio en el trabajo diario. No obstante, con el paso del tiempo esta circunstancia fue cambiando debido al hecho de que su precio fue siendo cada vez menor. Los elementos que la componen son los mismos que pueden encontrarse en una computadora de escritorio. Un procesador, memorias, pantalla, disco duro, placa madre, etc. Todos estos elementos, están ordenados de modo tal que ocupen una porción mínima de espacio. Como se muestra en la Figura 1.10.



Figura 1.10 Motherboard Ultrabook ASUS ZENBOOK UX3 1E

Fuente: Extraída de [http://www.legitreviews.com/asus-zenbook-ux31e-ultrabook-review\\_1903/3](http://www.legitreviews.com/asus-zenbook-ux31e-ultrabook-review_1903/3)



Dada la posibilidad de ir reduciendo continuamente el tamaño de los componentes, hoy también existe otra categoría de ordenador portátil denominado ultrabook, más pequeña y con menos peso.

Las computadoras portátiles o laptops han significado todo un avance en lo que respecta al aumento de la productividad, en la medida en que ofrecieron las mismas prestaciones de performance que tiene una computadora de escritorio con el aditivo de posibilitar que el trabajo en una computadora pueda desplazarse de un lado a otro.

### **1.2.5 Smartphones**

Se llama así a los teléfonos móviles que disponen de un hardware y un sistema operativo propio capaz de realizar diferentes tareas y funciones similares a las realizadas por las computadoras de escritorio o portátiles, añadiéndole al teléfono funcionalidades extras a la realización y recepción de llamadas y mensajes telefónicos.

Los primeros Smartphones se diferenciaron de los móviles, añadiendo aplicaciones como un bloc de notas, calculadora, un calendario donde anotar citas, reuniones y alarmas, un gestor para la recepción y envío de correos electrónicos, un teclado QWERTY que facilitaba la escritura en el teléfono, etc. Estos móviles dieron un paso tecnológico con el objetivo de asemejarse a ciertas funcionalidades que solo las computadoras de escritorio y portátiles de aquella época podían ejecutar, pero con la ventaja de tenerlo en un pequeño dispositivo fácilmente transportable. (Quees, 2014)

Con el tiempo y el desarrollo tecnológico de los últimos años, los Smartphones que se encuentran hoy en día poseen una serie de características y funcionalidades extras que les diferencian claramente de los móviles convencionales, funcionalidades como:

- Disponen de una aplicación para el envío y recepción de emails así como la gestión de varias cuentas de correo electrónico.
- Disponen de una suite de aplicaciones focalizadas a realizar funciones de organizador personal como calendarios, recordatorios y alertas, bloc de notas;

los cuales pueden comunicarse y sincronizarse con desktops, tablets u otros móviles.

- Disponen de una conexión a Internet, gracias a la red 3G y 4G, que permite navegar por la red al igual que si se accediese desde una computadora.
- Se puede leer, editar y visualizar una amplia familia de archivos como: PDF, hojas de cálculo, editores de textos, archivos multimedia de video y música.
- Permiten la descarga y la ejecución de aplicaciones desarrollados por terceros los cuales amplían nuevas funcionalidades.
- Disponen de un teclado QWERTY físico o táctil el cual permite y facilita la escritura de datos en el teléfono.
- Cuentan con un sistema operativo capaz de desarrollar todas las funcionalidades descritas anteriormente. iOS de la empresa Apple, Android de la empresa Google o Windows Phone por parte de la empresa Microsoft son ejemplos de sistemas operativos diseñados y programados para realizar tareas que generalmente se hacían en una computadora. (AreaTec, 2014)

A parte de las funcionalidades descritas anteriormente, hoy en día los fabricantes de Smartphone incluyen en sus modelos cámaras de alta resolución que permiten realizar fotografías, grabar vídeos en alta definición y realizar videoconferencias, receptores GPS para conocer con exactitud la localización y trayecto de cualquier punto del mundo, giroscopios, acelerómetros, sensores de luminosidad y proximidad así como sensores para la identificación de huellas dactilares.

Con la filosofía descrita anteriormente los Smartphones parten del principio por el cual el teléfono no sirve solo para llamar y escribir mensajes, ahora en el teléfono se puede consultar páginas de Internet, leer y editar un documento de texto, descargar y reproducir una película o un álbum de música, los taxistas pueden utilizar el GPS como un navegador mientras conduce, con la tecnología SCREEN mirroring controlar la televisión a distancia. Con todas estas tareas que puede realizar un Smartphone, se puede concluir que es una pequeña computadora con una gran potencia de cálculo en un espacio reducido, fácilmente transportable y que sigue avanzando día con día.

## 1.2.6 Tabletas

Son dispositivos portátiles de diferentes tamaños (generalmente entre 7' y 10'), superior a un Smartphone.

Integran procesadores que consumen menos energía algunos modelos disponibles incluyen ranura para memoria micro SD, incrementando así las posibilidades de almacenamiento. No obstante, estos dispositivos de formato panorámico destacan por su ligereza, versatilidad y reducidas dimensiones lo que facilita enormemente su portabilidad. (Alsitecno, 2014)

Se puede decir que se hallan a medio camino entre un teléfono inteligente y un portátil. Las tablets están más enfocados al acceso de aplicaciones que a la creación de contenidos. Otra característica destacable de estos dispositivos es su naturaleza táctil lo que permite prescindir de teclado físico o ratón. (Figura 1.11).



**Figura 1.11 Tablet Samsung Galaxy tab 3**

**Fuente:** Extraída de <http://www.xataka.com/tablets/samsung-galaxy-tab-3-1>

Esto los convierte en herramientas intuitivas, rápidas y que no precisan de aprendizaje instrumental por parte del usuario.

Algunas actividades que se pueden realizar con una tableta son:

- Reproducir música, tomar fotos y videos.
- Sincronización en línea de contenidos multimedia.
- Proporcionar opciones de acceso a Internet a través de WI-FI o a redes 3G:
- Navegación Web.
- GPS.
- Envío y recepción de correos electrónicos.
- Llamadas por Internet sin costo.
- Sincronización de cuentas tipo Google.
- Seguimiento de redes sociales. (Outlectech, 2014)

### **1.2.7 Smartwatch**

Los primeros indicios de lo que podrían ser los Smartwatch se vieron en la década de los 70, cuando Seiko, una famosa compañía japonesa, empezó a desarrollar un prototipo de reloj muy moderno.

Un Smartwatch es un dispositivo que se traduce de forma habitual como “reloj inteligente” dicho dispositivo cuenta con un sistema operativo. Permite realizar tareas propias de Smartphones.

Es por ello que, por lo general, estos relojes permiten realizar funciones como:

- Controlar velocidades.
- Conocer el ritmo cardiaco.
- Funcionar como GPS
- Reproducir música.

Su funcionamiento suele basarse en la vinculación con un Smartphone, también funcionan como centro donde recibir las notificaciones.

Funcionan mediante tecnologías inalámbricas, como Wi-Fi, Bluetooth o NFC, para sincronizarse con el teléfono móvil o cualquier otro dispositivo del que se nutra. (AndroidExperto, 2014). En Figura 1.12 se pueden apreciar Smartwatches Sony.



**Figura 1.12** Reloj inteligente

**Fuente:** Extraída de <http://adicionz.blogspot.mx/2012/04/sony-smartwatch-la-competencia-del.html>

## 1.2.8 Google Glass

Es una pantalla con forma de lentes (Figura 1.13) que muestra la información en la esquina superior derecha del lente, sin necesidad de mirar la pantalla del teléfono celular. Además, permiten vincularse con un teléfono inteligente de una manera más eficiente: Usando solo comandos de voz.



**Figura 1.13** Google Glass

**Fuente:** Extraída de <http://www.thetechauthor.com/google-glass-the-room-for-improvement>

Estos lentes altamente tecnológicos usan una pantalla pequeña con un marco especial, que les da el aspecto de gafas o lentes de leer. Esta pantalla está conectada a una cámara, un micrófono y un altavoz. Mediante una conexión Wi-Fi y Bluetooth, el dispositivo puede comunicarse con otros dispositivos, como un teléfono celular inteligente o aplicaciones de Internet.

Para empezar a usarlos se debe decir el comando “¡OK Glass!” enseguida de las instrucciones y comandos que se deseen.

Google Glass tiene muchos usos, pero su función principal es comunicar al usuario con su Smartphone por medio de comandos de voz y mostrar los resultados en los lentes.

Algunas acciones que se pueden realizar con las Google Glass, son:

- Tomar fotos: Capturar fotografías usando las gafas.
- Grabar videos: Las gafas de Google graban video en alta definición (720p HD).
- Aplicaciones Google Glass: Se pueden utilizar aplicaciones como: Google Maps, Google Now, Google +, Gmail y otros servicios Google.
- Comandos de voz: Comunicación con el Smartphone mediante comandos de voz y ver los resultados en Google glass. Se pueden ver los círculos<sup>2</sup> usando Google+ Hangouts, navegar en Internet, traducir a otros idiomas, buscar imágenes, ver direcciones a cierto destino, ver el pronóstico del tiempo, enviar un mensaje de texto, ver el estado de un vuelo y mucho más.
- Google Glass opera bajo el sistema operativo Android, cuenta con 16 GB de espacio de almacenamiento. (Bernardo, 2013)

---

<sup>2</sup> Conjunto de usuarios agrupados en G+.

### 1.3 Sistema Operativo GNU/Linux

En 1983 Richard Stallman inició el Proyecto GNU, con el propósito de crear un sistema operativo similar y compatible con UNIX y los estándares POSIX. Dos años más tarde, en 1985, creó la Fundación del Software Libre (FSF) y desarrolló la Licencia pública general de GNU (GNU GPL), para tener un marco legal que permitiera difundir libremente el software. De este modo el software de GNU fue desarrollado muy rápidamente, y por varias personas. A corto plazo, se desarrolló una multiplicidad de programas, de modo que a principios de los años 1990 había un sinnúmero de software disponible como para crear un sistema operativo completo. Sin embargo, todavía le faltaba un núcleo. (GNU, 2015)

Linux es el kernel<sup>3</sup> del sistema operativo GNU/Linux, que nace del interés de su creador Linus Torvalds al crear un nuevo kernel de Unix, basado en el kernel de Minix<sup>4</sup>, creado por Andrew S. Tanenbaum, pero con mejoras significantes, y con la posibilidad de poder modificarlo periódicamente de manera que fuera capaz de ejecutar aplicaciones GNU. Es así que el 5 de octubre del año de 1991 Linus Torvalds, decide compartir el código fuente de su proyecto en un foro de Minix e invitó a usuarios del foro a participar en la modificación del código y aportar nuevas ideas. En la Figura 1.14 se muestra el logo del sistema operativo Linux.



**Figura 1.14** Logotipo del Sistema Operativo Linux.

Fuente: Extraída de <http://tecnologyc.com/software/linux/>

---

<sup>3</sup> El núcleo es la parte más importante de un sistema operativo, parte encargada de acceder a los distintos dispositivos de los que una computadora dispone.

<sup>4</sup> Es una copia del Sistema Operativo Unix creado para enseñar el diseño de sistemas operativos en la Universidad Vrije de Amsterdam.

Un gran número de programadores comenzaron a trabajar en el proyecto y después de varias revisiones la versión 1.0 salió el 14 de marzo de 1994. (Puttonen, 2001)

De este modo, el núcleo creado por Linus Torvalds, llenó el "espacio" final que quedaba en el sistema operativo de GNU.

### 1.3.1 Características

Gnu/Linux ofrece muchas características la cual lo hacen un sistema operativo bastante eficiente, y eso se ve reflejado en la actualidad, al ser un sistema estable es el más usado en servidores. A continuación se presentan algunas de las características de este sistema:

- **Compatibilidad con UNIX.** Al apegarse a las normas POSIX mantiene un máximo de compatibilidad con otras variantes de los sistemas operativos UNIX.
- **Multi-usuario.** Permite a más de un usuario utilizar los recursos del sistema.
- **Multi-tarea.** La característica de multi-tarea le permite a un usuario realizar varias tareas al mismo tiempo.
- **Portabilidad.** En la actualidad es usado en las plataformas Intel x86, PowerPC, Macintosh, Amiga, Atari, DEC Alpha, Sun Sparc, ARM y otras más.
- **Poderosas herramientas de desarrollo.** Linux es una plataforma ideal para el desarrollo de aplicaciones y la experimentación de nuevos lenguajes. Como parte de la instalación se encuentran diversos compiladores incluidos.
- **Estabilidad.** Linux se ha distinguido por su estabilidad de operación, se han conocido y comentado muchos casos de equipo trabajando por más de un año sin tener que apagar o reiniciarlo.



- Velocidad. Los equipos Linux también se han distinguido por su extraordinaria velocidad. El sistema operativo administra eficientemente los recursos como memoria, poder de CPU y espacio en disco.
- Librerías dinámicas compartidas. Linux usa extensivamente las librerías dinámicas compartidas. Estas librerías utilizan una sección común para todos los ejecutables, reduciendo el tamaño de la aplicación.
- Ejecutables compartidos. Si más de una copia de una aplicación o programa es cargado a memoria para ejecución, todas las tareas pueden compartir la misma área de memoria. Este proceso de ejecutables compartidos (shared executables), hace un uso eficiente de memoria RAM.
- Demanda de paginación. El kernel de linux soporta demanda de paginación, para los programas muy extensos significa que sólo una sección del programa es cargado a memoria.
- Para optimizar aún más la memoria Linux, usa un sólo espacio “pool” de memoria, y esto permite el uso de la memoria restante para poner secciones del disco permitiendo un acceso rápido a programas de uso común (disk cache).
- Espacio de Swap. Para soportar de manera simulada grandes cantidades de memoria para los diversos requerimientos.
- Diferentes sistemas de archivos. Linux soporta varios formatos de sistemas de archivos para compartir información. El propio sistema de archivos ext3, está diseñado para hacer un uso óptimo del espacio en el disco duro. Y ext4, mejora la eficiencia en velocidad e integridad de la información.
- Construcción Modularizada o Monolítica. El sistema operativo Linux puede ser compilado con los drivers en forma de módulos los cuales al ser necesarios

pueden ser incluidos, y se acoplan en el corazón del sistema operativo, reconociendo el hardware para el cual fueron diseñados.

- Para dar velocidad al sistema operativo se puede compilar el kernel con todos los drivers necesarios, para todo el hardware del equipo en el cual se cargará y construirá un sistema operativo que no necesite incluir módulos. (EILinux, 2015)

### 1.3.2 Distribución Ubuntu

Sistema operativo basado en GNU/Linux, y que se distribuye como software libre, el cual incluye su propio entorno de escritorio denominado Unity. Su nombre proviene de la ética homónima, en la que habla de la existencia de uno mismo como cooperación de los demás "Yo soy por que nosotros somos". El logo de la distribución muestra a tres personas tomadas de las manos, formando un círculo (Figura 1.15)



Figura 1.15 Logotipo Distribución Ubuntu

Fuente: Extraída de <http://design.ubuntu.com/brand/ubuntu-logo>

Está orientado al usuario novato y promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia de usuario. Está compuesto de múltiple software normalmente distribuido bajo una licencia libre o de código abierto.

Estadísticas Web sugieren que la cuota de mercado de Ubuntu dentro de las distribuciones Linux es, aproximadamente, del 49%, y con una tendencia a aumentar como servidor Web. (Y un importante incremento activo de 20 millones de usuarios para fines del 2011). (Reyes, 2015).

Su patrocinador, Canonical, es una compañía británica propiedad del empresario sudafricano Mark Shuttleworth. Ofrece el sistema de manera gratuita y se financia por medio de servicios vinculados al sistema operativo y vendiendo soporte técnico.

Además, al mantenerlo libre y gratuito, la empresa es capaz de aprovechar a los desarrolladores de la comunidad, para mejorar los componentes de su sistema operativo.

Extraoficialmente, la comunidad de desarrolladores proporciona soporte para otras derivaciones de Ubuntu, con otros entornos gráficos, como Kubuntu, Xubuntu, Ubuntu MATE, Edubuntu, Ubuntu Studio, Mythbuntu, Ubuntu GNOME y Lubuntu. (LinuxZone, 2015)

### **1.3.3 Ventajas**

Ubuntu se ha convertido en una de las distribuciones Linux más usada a nivel mundial, sus principales ventajas son:

- La interfaz de usuario con la que cuenta es muy fácil e intuitiva e incluso ha servido de modelo a otros sistemas operativos. Cuenta con un dash que permite acceder a las aplicaciones rápidamente. (Figura 1.16)
- Permite instalar y desinstalar aplicaciones desde Ubuntu software center.
- Es muy rápido al abrir aplicaciones y al momento de programar con Android Studio carga y realiza las tareas de forma eficiente.

- Con una configuración adecuada se adapta a cualquier tipo de computadora Laptop, notebook o desktop. (VIX, 2015)



Figura 1.16 Dash de Ubuntu

Fuente: Extraída de <http://elblogdeliher.com/que-es-el-dash-o-tablero-de-ubuntu/>

## 1.4 Hardware: Periféricos y Sensores

Las diferentes características de hardware, en los dispositivos Android se han convertido en un punto clave para los consumidores a la hora de adquirir un dispositivo móvil, los Smartphones están compuestos por periféricos y sensores, los cuales son utilizados día a día sin percatarse de su presencia pero que cumplen una función importante en el teléfono. Para desarrollar una aplicación móvil se deben tener en cuenta los periféricos y sensores que existen, esto permitirá una mayor interacción entre el usuario, el dispositivo y el medio que lo rodea.

El kit de desarrollo de software de Android (SDK), proporciona una variedad de interfaces de programación de aplicaciones (API), que permiten acceder a las características del hardware en el teléfono de una forma segura y robusta. Además de la cámara, los dispositivos Android pueden tener un número de sensores y hardware. No todos los Smartphones Android cuentan con el mismo número de sensores o periféricos.

## 1.4.1 Periféricos

Como se ha mencionado, los Smartphones están compuestos por periféricos que complementan y mejoran el funcionamiento de los equipos. Ver Figura 1.17



Figura 1.17 Periféricos internos

Fuente: Extraída de <http://www.think-dash.com/2011/12/samsung-galaxy-nexus-teardown-reveals.html>

Los periféricos son aquellos dispositivos que se conectan o vinculan a un dispositivo, para complementar funciones y obtener características extras. Permitiendo al usuario mejores interacciones entre el medio exterior y virtual. Los Smartphones modernos cuentan con periféricos internos como la cámara, el micrófono, la tarjeta microSD y bocinas.

Con la ayuda de los periféricos (Figura 1.18) los smartphones se pueden transformar en herramientas que se pueden utilizar en el trabajo, permitiendo expandir características con las que el teléfono ya cuenta, agregar un teclado físico inalámbrico, mejorar la cámara interna del dispositivo o agregar funcionalidades completamente nuevas.



**Figura 1.18 Periféricos**

**Fuente: Edición propia**

A continuación se presenta la Tabla 1.2 que muestra los diferentes periféricos y accesorios que se pueden utilizar en un Smartphone.

**Tabla 1.2 Periféricos**

**Fuente: Edición propia**

Nombre	Funcionalidad
Cargador tradicional	Permite el ingreso de la energía a la batería del equipo.
Cargadores para automóvil	Permite el ingreso de la energía a la batería del equipo con un adaptador especial para conectar en el automóvil.
Cable de datos USB	Permite la conexión a un puerto USB de otro equipo
Cable de alta definición	Permite conectar y exportar video de Alta definición.
Cable de audio	Permite conectar y exportar sonido a otros dispositivos.
Lápiz óptico	Permite trabajar con los teclados táctiles de los dispositivo
Bocinas	

	Dispositivos para reproducir y amplificar el sonido del dispositivo móvil.
Antenas	Permite la aplicación de la señal y permite recibir señales de radio.
Manos Libres	Dispositivo en forma de auricular para colocarse en el oído.
Doble SIM	Permite utilizar dos SIM en el mismo dispositivo móvil.
Gafas	Permite visualizar las imágenes de la pantalla del Smartphone.

## Cámara

La popularidad de las cámaras digitales ha provocado que sus precios sean cada vez más bajos y el tamaño de estas sea más reducido. En la actualidad es difícil encontrar un Smartphone sin una cámara y los dispositivos Android no son la excepción.

El framework de Android incluye soporte para varias cámaras y sus características que están disponibles en los dispositivos, permitiéndole a las aplicaciones tomar fotos y videos.

### 1.4.2 Sensores

Los teléfonos móviles de hoy en día son algo más que simples dispositivos de comunicación, que tienen acceso a Internet. Al incluir micrófonos, cámaras, acelerómetros, podómetros, indicadores de temperatura, detectores de luz, los Smartphones se han convertido en dispositivos extra sensoriales, permitiendo aumentar la propia percepción. (Meier, 2010).

Los sensores que detectan propiedades físicas y ambientales, brindan una emocionante innovación para mejorar la experiencia del usuario al momento de usar distintas aplicaciones móviles. La incorporación de compases electrónicos, sensores

gravitacionales, detectores de luz y sensores de proximidad en los dispositivos modernos proporciona un conjunto de nuevas posibilidades para interactuar con ellos.

## **Sensores disponibles en Android**

El framework de sensores de Android permite acceder a muchos tipos de sensores. Algunos de estos sensores están basados en hardware y algunos son basados en software. Los sensores basados en hardware son componentes físicos integrados en un dispositivo de teléfono o Tablet (Figura 1.19). Derivan sus datos midiendo directamente las propiedades ambientales específicas, tales como aceleración, la fuerza del campo geomagnético, o el cambio angular.



**Figura 1.19** Sensor de proximidad y de luz

Fuente: <https://goo.gl/rnreTj>

Los sensores basados en software no son dispositivos físicos, a pesar de que imitan a los sensores basados en hardware. Derivan sus datos de uno o más de los sensores de hardware y algunas veces son llamados sensores virtuales o sensores sintéticos. El sensor de aceleración lineal y el sensor de la gravedad son ejemplos de sensores basados en software. La Tabla 1.3, resume los sensores que son compatibles con la plataforma Android.



Tabla 1.3 Tipos de sensores soportados en Android

Fuente: Edición propia

Sensor	Tipo	Usos
TYPE_ACCELEROMETER	Hardware	Detección de movimiento
TYPE_AMBIENT_TEMPERATURE	Hardware	Monitoreo de la temperatura
TYPE_GRAVITY	Software o Hardware	Detección de movimiento
TYPE_GYROSCOPE	Hardware	Detección de rotación
TYPE_LIGHT	Hardware	Control de brillo de la pantalla.
TYPE_LINEAR_ACCELERATION	Software o Hardware	Monitoreo de la aceleración en un solo eje.
TYPE_MAGNETIC_FIELD	Hardware	Crear una brújula
TYPE_ORIENTATION	Software	Determinar la posición del dispositivo.
TYPE_PRESSURE	Hardware	Monitoreo de los cambios de presión en el aire.
TYPE_PROXIMITY	Hardware	Posición del teléfono durante una llamada.
TYPE_RELATIVE_HUMIDITY	Hardware	Monitoreo de la humedad.
TYPE_ROTATION_VECTOR	Software o Hardware	Detección de movimiento y de rotación.
TYPE_TEMPERATURE	Hardware	Monitoreo de temperaturas.

- **Near Field Communication.** NFC es una forma de comunicación sin contacto entre dispositivos como Smartphones o tabletas. Permite a un usuario enviar la información (con teléfonos compatibles), sin tener que colocar juntos los dispositivos o pasar por pasos múltiples que establecen una conexión. (NFC, 2014)

- **Bluetooth.** Envía información dentro de su red de área personal a distancias de hasta 100 metros (328 pies) -dependiendo de la implementación del dispositivo. (Bluetooth, 2015).
- **Localización GPS.** Es un sistema que sirve para determinar la posición con coordenadas de Latitud, Longitud y Altura. Se basa en una constelación de 21 satélites que orbitan a la tierra a una altura de 20,200 Km, necesitando 11h 58m para describir una órbita completa.
- **Giroscopio.** Permite al Smartphone medir y mantener la orientación. Los sensores giroscópicos pueden monitorear y controlar posiciones del dispositivo como la orientación, la dirección, el movimiento angular y la rotación. Cuando se aplica a un teléfono inteligente, un sensor giroscópico comúnmente lleva a cabo funciones de reconocimiento de gestos.  
Además, los giroscopios en los teléfonos inteligentes ayudan a determinar la posición y orientación del teléfono.
- **Sensor de luz.** Este sensor mide el brillo de la luz ambiental. Así, el software del teléfono utiliza estos datos para ajustar el brillo de la pantalla automáticamente, haciéndola más clara o más oscura.
- **Sensor de proximidad.** Este sensor, puede medir la distancia que existe entre el Smartphone, y algún otro objeto. Es por eso que cuando se encuentra haciendo una llamada y se coloca el Smartphone en la oreja, el dispositivo apaga la luz de la pantalla.
- **Podómetro.** Es capaz de medir con precisión el número de pasos que se han realizado.
- **Magnetómetro:** Detecta campos magnéticos. Como es el caso de las fundas imantadas o el google Cardboard.  
El magnetómetro, es uno de los sensores que utilizan las aplicaciones de la brújula la cual permite al Smartphone detectar en qué dirección está orientado respecto a los puntos cardinales.
- **Sensor de ritmo cardiaco.** Se encarga de recopilar las pulsaciones por minuto que tiene el usuario con tan sólo colocar el dedo sobre él. (Meier, 2010)

## Framework de Android para sensores

Proporciona mecanismos para acceder y obtener datos de los sensores, con la excepción del GPS, al cual se accede mediante los servicios de ubicación para Android. Este framework es parte del paquete de hardware. La (Tabla 1.4) incluye las clases e interfaces principales del framework para sensores. (Milette & Stroud, 2012).

Tabla 1.4 Framework para sensores de la plataforma Android

Fuente: Edición propia.

Nombre	Tipo	Descripción
<b>SensorManager</b>	Clase	Se usa para crear una instancia del servicio de sensores. Proporciona varios métodos para el acceso a sensores, el registro y la eliminación de registros de las escuchas de eventos de sensores, etc.
<b>Sensor</b>	Clase	Se usa para crear la instancia de un sensor específico.
<b>SensorEvent</b>	Clase	El sistema lo usa para publicar datos del sensor. Incluye los valores de datos de sensores sin procesar, el tipo de sensor, la precisión de los datos y una marca de hora.
<b>SensorEventListener</b>	Interfaz	Proporciona métodos de llamada de regreso para recibir avisos del SensorManager cuando los datos o la precisión del sensor han cambiado.

Tiene varios métodos, que hacen que sea fácil determinar en tiempo de ejecución, los sensores que se encuentran disponibles en el dispositivo. La API, también proporciona métodos que permiten determinar las capacidades de cada sensor, tales como su alcance máximo, su resolución, y sus requisitos de energía.

## GPS

Con aplicaciones que hacen uso de la localización para proporcionar una experiencia de usuario más rica y con un nivel de funcionalidad, que antes no era posible, la localización es cada vez más importante en el desarrollo de aplicaciones móviles.

La capacidad para obtener datos y ubicación fácilmente, para después ser utilizados por diversas aplicaciones, se está convirtiendo en una característica importante de las plataformas móviles, de hoy en día. Android proporciona esta funcionalidad con su servicio de localización.

El servicio de localización de Android proporciona acceso a servicios que pueden ser utilizados para determinar la ubicación actual de un dispositivo.

Android, hace uso de diferentes métodos para proporcionar la información de ubicación en una aplicación. En Android, se conocen como proveedores de localización, y cada uno tiene su propio conjunto único de fortalezas y debilidades.

Además, debido a que cada uno de los proveedores de localización tienen características únicas, se prestan a ser utilizados de manera diferente en diferentes situaciones.

Las siguientes secciones dan algunas explicaciones de alto nivel en cuanto a cómo funcionan los diferentes métodos de adquisición de ubicación. Aunque una aplicación tiene poco control sobre el funcionamiento de los proveedores, puede decidir qué proveedor de ubicación de su uso. La comprensión de cómo funciona cada proveedor va un largo camino en la comprensión de sus limitaciones y características. (Meier, 2010)

## GPS Provider

El Sistema de Posicionamiento Global (GPS) utiliza un sistema de satélites que orbitan el planeta para ayudar a un receptor (un teléfono Android en este caso) a determinar su ubicación actual. El término GPS se refiere a todo el sistema GPS, que consiste en los satélites, receptores y los puestos de control que supervisa.

El receptor que se encuentra en el teléfono es inútil sin el resto del sistema.

## Funcionamiento

En general, un receptor GPS utiliza información de los satélites GPS que orbitan alrededor de la tierra, para calcular su ubicación actual. El sistema GPS contiene 27 satélites que orbitan la tierra continuamente.

Cada satélite sigue un camino definido, asegurando que al menos cuatro satélites son "visibles" desde cualquier punto de la tierra en un momento dado. Es necesario tener una línea de visión por lo menos de 4 satélites, para determinar la ubicación a través del GPS. La Figura 1.20 muestra una representación de la constelación de satélites GPS. (Milette & Stroud, 2012)

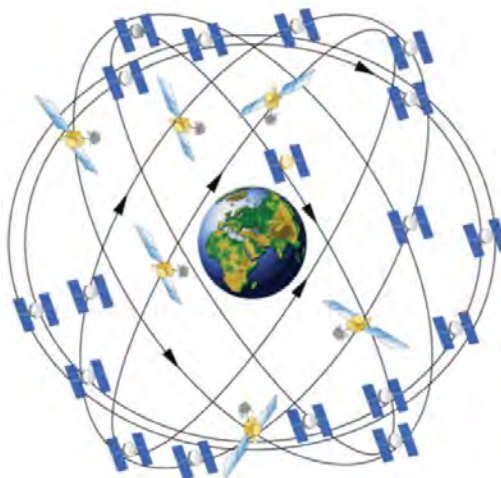


Figura 1.20 Satélites GPS

Fuente: Extraída de Professional Android sensor programming

## Obtener Google Maps API key

Para utilizar la versión 2 del API de Google Maps, se debe registrar el proyecto de la aplicación en la consola de desarrolladores de Google y obtener una clave API de Google, la cual después se agrega a la aplicación.

Todas las aplicaciones Android están firmadas con un certificado digital para el que se mantiene la clave privada.

Las claves API de Android están vinculadas entre certificados y paquetes. No importa cuántos usuarios tenga la aplicación. Sólo se necesita una clave para cada certificado. Para obtener una clave para la aplicación se requiere de varios pasos. Estos pasos se describen a continuación:

1. Obtener información sobre el certificado de la aplicación.

En el documento `google_maps_api.xml`, copiar en el navegador la url que genera, direcciona a la consola de desarrolladores. Ver Figura 1.21.



```
<resources>
  <!--
  TODO: Before you run your application, you need a Google Maps API key.

  To get one, follow this link, follow the directions and press "Create" at the end:
  https://console.developers.google.com/flows/enableapi?apiId=maps_android_backend&keyType=CLIENT_SIDE_MANUAL

  You can also add your credentials to an existing key, using this line:
  0A:40:13:EA:DC:D9:B3:19:55:D8:B1:A2:5F:C1:AB:87:6B:F9:A3:9A;com.example.ivonne.gps

  Once you have your key (it starts with "AIza"), replace the "google_maps_key"
  string in this file.
  -->
  <string name="google_maps_key" translatable="false" templateMergeStrategy="preserve">
    YOUR_KEY_HERE
  </string>
</resources>
```

Figura 1.21 Pantalla URL API key

Fuente: Edición propia

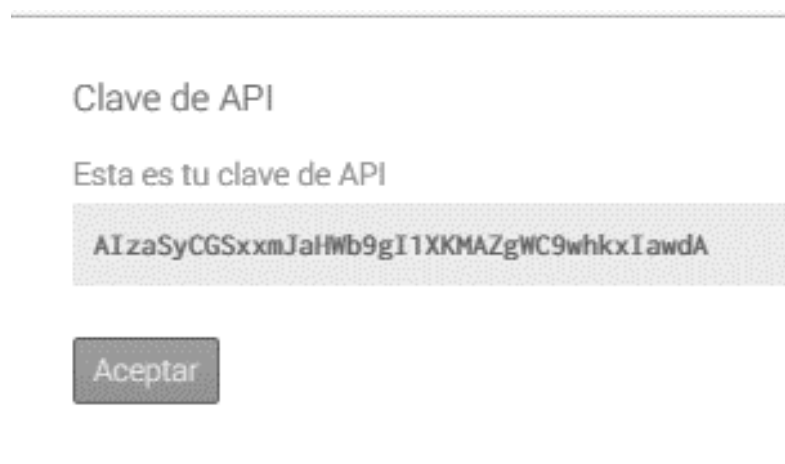
2. Registrar un proyecto en la consola de desarrolladores de Google y añadir la v2 API de Google Maps para Android como un servicio para el proyecto. Dar clic en continuar. Ver Figura 1.22



**Figura 1.22 Registro de proyecto**

**Fuente: Edición propia**

3. Crear una clave. En la siguiente ventana dar clic en "Crear" y después aparece la clave que se ha generado la cual se añadirá al proyecto existente. Ver Figura 1.23



**Figura 1.23 Obtención de la clave**

**Fuente: Edición propia**

4. Añadir la clave en la aplicación. Reemplazar el texto "YOUR\_KEY\_HERE" por la clave de la aplicación. Ver Figura 1.24

```
<string name="google_maps_key" translatable="false" templateMergeStrategy="preserve">  
YOUR_KEY_HERE  
</string>
```

Figura 1.24 Añadir clave

Fuente: Edición propia

Una vez agregada la clave ya se tiene permiso para utilizar los mapas en la aplicación Android



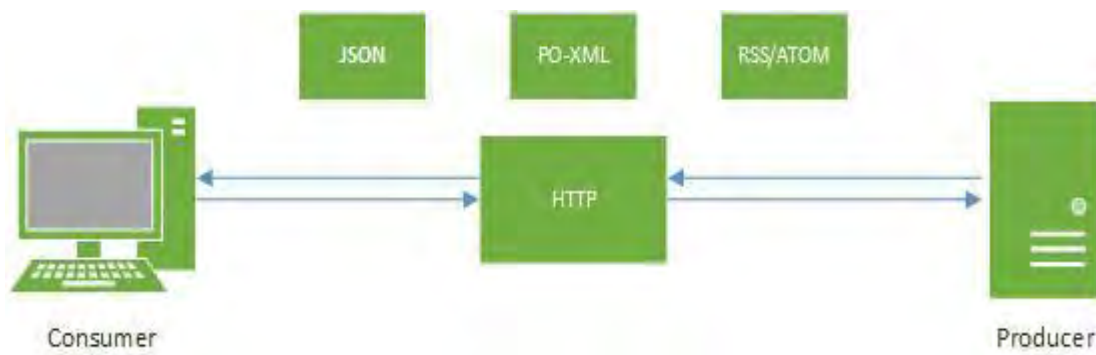
## 1.5 Webservices

Es un sistema software diseñado para soportar una interacción interoperable máquina a máquina sobre una red. Los Servicios Web suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja. (Navarro Marset, 2015). La idea de los servicios Web es eliminar la necesidad de crear nuevos protocolos, proporcionando un mecanismo estandarizado para las llamadas a procedimientos remotos, basado en XML y HTTP. (Tatroe, MacIntyre, & Lerdof, 2013)

Un conjunto de protocolos y estándares que sirven para intercambiar datos entre diferentes aplicaciones de software han sido desarrolladas en distintos lenguajes de programación, incluso ejecutadas sobre cualquier plataforma.

Generalmente, la interacción se basa en el envío de solicitudes y respuestas entre un cliente y un servidor, que incluyen datos.

El cliente solicita información, enviando datos al servidor para que pueda procesar su solicitud. El servidor genera una respuesta que envía de vuelta al cliente, adjuntando otra serie de datos que forman parte de esa respuesta. (Figura 1.25)



**Figura 1.25 WebService con arquitectura REST**

**Fuente:** Extraída de <http://crunchify.com/how-to-create-restful-java-client-with-jersey-client-example>

En los últimos años se ha popularizado un estilo de arquitectura Software conocido como REST (Representational State Transfer). Este nuevo estilo ha supuesto una

nueva opción de uso de los Servicios Web. A continuación se listan los tres estilos de usos más comunes:

- Remote Procedure Calls (RPC, Llamadas a Procedimientos Remotos): Los Servicios Web basados en RPC presentan una interfaz de llamada a procedimientos y funciones distribuidas. Típicamente, la unidad básica de este tipo de servicios es la operación WSDL (WSDL es un descriptor del Servicio Web, es decir, el homólogo del IDL para COM). Las primeras herramientas para Servicios Web estaban centradas en esta visión.
- Service-Oriented Architecture (SOA, Arquitectura Orientada a Servicios): En los servicios Web basados en ésta arquitectura la unidad básica de comunicación es el mensaje más que la operación, son servicios orientados a mensajes.
- Representation State Transfer (REST, Transferencia de Estado Representacional): Es un estilo híbrido derivado de varios estilos de arquitectura de red y combinado con restricciones adicionales que definen una interfaz de conector uniforme. (Fielding, 2016). Este servicio intenta emular al protocolo HTTP o protocolos similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar.<sup>5</sup> (Fielding, 2016)

### 1.5.1 Arquitecturas Webservices

Estos están contruidos con varias tecnologías que trabajan conjuntamente con los estándares que están emergiendo para la seguridad y operatividad de los mismos. A continuación se describen brevemente los estándares que están ocupando los Web Services.

**XML:** Abreviación de Extensible Markup Language. El XML es una especificación desarrollada por W3C. Permite a los desarrolladores crear sus propios tags, que les

---

<sup>5</sup> Métodos HTTP, GET, PUT, POST y DELETE utilizados para el manejo de datos.

permiten habilitar definiciones, transmisiones, validaciones, e interpretación de los datos entre aplicaciones y entre organizaciones.

**SOAP:** Abreviación de Simple Object Access Protocol, es un protocolo de mensajería construido en XML que se usa para codificar información de los requerimientos de los Web Services y para responder los mensajes antes de enviarlos por la red. Los mensajes SOAP son independientes de los sistemas operativos y pueden ser transportados por los protocolos que funcionan en la Internet, como ser: SMTP, MIME y HTTP.

**WSDL:** Abreviación de Web Services Description Language, es un lenguaje especificado en XML que se ocupa para definir los Web Service, como colecciones de punto de comunicación capaces de intercambiar mensajes. El WSDL es parte integral de UDDI y parte del registro global de XML, en otras palabras es un estándar de uso público (no se requiere pagar licencias ni royalties para usarlo).

**UDDI:** Abreviación de Universal Description, Discovery and Integration. Es un directorio distribuido que opera en la Web y que permite a las empresas publicar sus Web Services, para que otras empresas conozcan y utilicen los Web Services que publican. Opera de manera análoga a las páginas amarillas. (Navarro Marset, 2015)

## 1.5.2 Ventajas

Usar un Web Service brinda muchos beneficios a empresas y programadores, los principales se mencionan a continuación:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder al contenido y entender su funcionamiento.

- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Proporcionan la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta, la W3C<sup>6</sup>, por tanto no hay secretismos por intereses particulares de fabricantes concretos, y se garantiza la plena interoperabilidad entre aplicaciones. (Navarro Maset, 2015)

### 1.5.3 Web Service RESTfull

REST es un estilo de arquitectura de software, para sistemas hipermedias distribuidos tales como la Web. El término fue introducido en la tesis doctoral de Roy Fielding en 2000, quien es uno de los principales autores de la especificación de HTTP. En realidad, REST se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen como los recursos son definidos y diseccionados. El término frecuentemente es utilizado en el sentido de describir a cualquier interfaz, que transmite datos específicos de un dominio sobre HTTP sin una capa adicional, como hace SOAP. Estos dos significados pueden chocar o incluso solaparse. Es posible diseñar un sistema software de gran tamaño de acuerdo con la arquitectura propuesta por Fielding sin utilizar HTTP o sin interactuar con la Web. Así como también es posible diseñar una simple interfaz XML+HTTP que no sigue los principios REST, y en cambio seguir un modelo RPC.

REST no es un estándar, ya que es tan solo un estilo de arquitectura. Aunque REST no es un estándar, está basado en estándares:

- HTTP
- URL

---

<sup>6</sup> World Wide Web Consortium, es una comunidad internacional que desarrolla estándares que aseguran el crecimiento de la Web a largo plazo. Fundada en 1994 por Tim Berners-Lee

- Representación de los recursos: XML/HTML/GIF/JPEG/...
- Tipos MIME: Text/xml, text/html.

Permite diferentes formatos de datos, dando por ejemplo al desarrollador la posibilidad de utilizar JSON que normalmente es más rápido y tiene mejor escalabilidad y rendimiento. Las lecturas del REST se pueden guardar temporalmente en la caché<sup>7</sup>, las lecturas basadas en SOAP no se pueden.

La arquitectura a utilizar para realizar el Web Service en la presente tesis será RESTfull puesto que al utilizar HTTP, el manejo de datos, el Desarrollo de APIs y la creación de clientes es más estable y practico, también es importante mencionar que a diferencia de SOAP permite el envío y recepción de datos en formato JSON lo que permite un mejor soporte a los clientes del explorador. (Vaqqas, 2015)

## 1.6 Servidor Web Apache

Un servidor Web es un programa especialmente diseñado para transferir datos de hipertexto, utilizan el protocolo http. Los servidores Web están alojados en una computadora que cuente con acceso a Internet. El servidor Web, se encuentra a la espera de una petición por parte del navegador, por ejemplo, acceder a una página Web y responder a la petición, enviando código HTML mediante una transferencia de datos en red.

Apache es un servidor Web desarrollado por la Apache Software Foundation, utilizado para dar servicio a páginas Web estáticas o dinámicas, es de código abierto y puede ser instalado en diferentes sistemas operativos, debido a que es un servidor estable, seguro y sólido, se mantiene como uno de los servidores Web más usados en el mundo. (Culturación, 2015)

---

<sup>7</sup> Es la memoria de acceso rápido de una computadora, que guarda temporalmente los datos recientemente procesados.

La historia de Apache se remonta a febrero de 1995, donde empieza el proyecto del grupo Apache, el cual está basado en el servidor Apache http de la aplicación original de NCSA<sup>8</sup>. El desarrollo de esta aplicación original se estancó por algún tiempo tras la marcha de Rob McCool, por lo que varios Webmaster<sup>9</sup> siguieron creando sus parches para sus servidores Web, hasta que se contactaron vía email para seguir en conjunto el mantenimiento del servidor Web, fue ahí cuando formaron el grupo Apache. (Figura 1.26)



Figura 1.26 Logotipo del servidor Apache

Fuente: Extraída de [https://commons.wikimedia.org/wiki/File:Apache\\_HTTP\\_server\\_logo\\_\(2016\).svg](https://commons.wikimedia.org/wiki/File:Apache_HTTP_server_logo_(2016).svg)

Fueron Brian Behlendorf y Cliff Skolnick quienes a través de una lista de correo coordinaron el trabajo y lograron establecer un espacio compartido de libre acceso para los desarrolladores. En 1999, se formó la Fundación de Software Apache (Apache Software Foundation) para obtener apoyo financiero, organizativo y legal para el servidor. (Apache, 2015).

Al igual que en el inicio del Sistema Operativo Linux, Apache nace por el interés de un grupo de programadores que por medio de una conexión a Internet, compartían los parches que realizaban al sistema, llegando a crear un sistema estable que cuenta con una gran documentación y comunidad de desarrolladores.

Entre las principales características de Apache, se encuentran las siguientes:

- Soporte de seguridad SSL y TLS.
- Puede realizar autenticación de datos utilizando SGDB.
- Puede dar soporte a diferentes lenguajes, como Perl, PHP, Python y tcl.

---

<sup>8</sup> National Center for Supercomputing Applications, es un organismo de Estados Unidos relacionado con la investigación en el campo de la Informática y las Telecomunicaciones.

<sup>9</sup> El Webmaster es el dueño del sitio Web o el encargado de mantener el sitio Web habilitado.

## 1.7 Metodología de desarrollo

Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información.

Una gran variedad de estos marcos de trabajo han evolucionado durante años, cada uno con sus propias fortalezas y debilidades. Una metodología de desarrollo de sistemas no tiene que ser necesariamente adecuada para usarla en todos los proyectos. Cada una de las metodologías disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo.

Una metodología:

- Optimiza el proceso y el producto software. - Métodos que guían en la planificación y en el desarrollo del software.
- Define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto. (Herrera, 2014)

### 1.7.1 Metodología RAD

Es una metodología que permite a las organizaciones, desarrollar sistemas estratégicamente importantes, de manera más rápida reduciendo a la vez los costos de desarrollo manteniendo la calidad.

El método RAD tiene una lista de tareas y una estructura de desglose de trabajo diseñada para la rapidez. El método comprende el desarrollo iterativo, la construcción de prototipos y el uso de utilidades CASE, (Computer Aided Software Engineering). Tradicionalmente, el desarrollo rápido de aplicaciones tiende a englobar también la usabilidad, utilidad y rapidez de ejecución. (Curiosimos, 2015)

La gran mayoría de las herramientas gráficas orientadas a objetos, tienen interiorizadas el concepto general de RAD. Además, con la creación bien planificada, la programación de nuevos módulos se vuelve cada vez más simplificada, reutilizando los objetos creados anteriormente.

## 1.7.2 Etapas de metodología RAD

La metodología RAD tiene 4 etapas que permiten un mejor análisis y comprensión del proyecto a desarrollar, como se muestra en la (Figura 1.27)



Figura 1.27 Flujo de proceso de RAD

Fuente: Reedición propia

1. En la etapa planificación de requisitos se definen las funciones del negocio y las áreas sujeto de datos que el sistema apoyará y determina el alcance del sistema.
2. La etapa de Diseño usa los talleres para modelar los datos y los procesos del sistema y para construir un prototipo de trabajo de los componentes críticos del sistema.
3. La etapa de Desarrollo completa la construcción física de la base de datos y del sistema de aplicación, construye el sistema de conversión y elabora ayudas de usuarios y planes de trabajo a desarrollar o de despliegue.



4. La etapa de Transición o Implementación incluye la puesta a prueba y la capacitación del usuario final, la conversión de datos y la implementación del sistema de aplicación.

A continuación se muestra en la Figura 1.28 las etapas que se utilizan en el proceso de modelado de la aplicación.



**Figura 1.28 Etapas de modelado de la aplicación**

**Fuente: Reedición propia**

**Modelado de gestión:** El flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la procesó?

**Modelado de datos:** El flujo de información definido como parte de la fase de modelado de gestión, se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características (llamadas atributos) de cada uno de los objetos y las relaciones entre estos objetos.

**Modelado de proceso:** Los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información, necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos. Es la comunicación entre los objetos.

**Generación de aplicaciones:** El RAD asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación de tercera generación, el proceso RAD trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible), o a crear componentes reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.

**Pruebas de entrega:** Como el proceso RAD enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfaces a fondo. (RAD, 2015)

### 1.7.3 Ventajas

Las principales ventajas que puede aportar este tipo de desarrollo son las siguientes:

- Velocidad de desarrollo.
- Visibilidad temprana debido al uso de técnicas de prototipado.
- Mayor flexibilidad que otros modelos.
- Ciclos de desarrollo más cortos.
- Calidad.

Las ventajas que puede añadir sobre el seguimiento de un método en cascada, es que en este método hay un largo periodo de tiempo hasta que el cliente puede ver cualquier resultado. El desarrollo puede llevar tanto tiempo que el negocio del cliente haya cambiado sustancialmente en el momento en el que el software está listo para usar. Con este tipo de métodos no hay visibilidad del producto hasta que el proceso no está finalizado al 100%, que es cuando se entrega el software.

## 1.7.4 Desventajas

Entre los principales inconvenientes que se pueden encontrar en el uso del desarrollo rápido de aplicaciones se encuentra:

- Características reducidas.
- Escalabilidad reducida.
- Más difícil de evaluar el progreso porque no hay hitos clásicos.

Una de las críticas principales que suele generar este tipo de desarrollo es que, el desarrollo rápido de aplicaciones es un proceso iterativo e incremental, puede conducir a una sucesión de prototipos, que nunca culmine en una aplicación de producción satisfactoria. Tales fallos pueden ser evitados si las herramientas de desarrollo de la aplicación son robustas, flexibles y colocadas para el uso correcto. (Mena, 2015).



# CAPÍTULO 2

## SISTEMA OPERATIVO ANDROID



## CAPÍTULO 2. SISTEMA OPERATIVO ANDROID

Hace 10 años los teléfonos celulares eran muy diferente a lo que son ahora y simplemente eran utilizados para hacer y recibir llamadas, poco a poco se fueron agregando más funciones como envío de mensajes, correo electrónico, juegos con gráficos más elaborados, pero aún carecían de un sistema operativo propio. Nokia fue una de las primeras compañías que decidieron introducir el sistema operativo Symbian a sus celulares, causó revuelo debido a la disponibilidad y en cierto modo facilidad de instalar aplicaciones que se podían utilizar en la vida diaria. A pesar que tuvo mucho éxito, Nokia no supo innovar y Symbian se convirtió en un sistema operativo pesado y no muy eficaz. Después de Symbian llegaron otros sistemas operativos pero no tan robustos.

Android fue creado en el año 2000 por Andy Rubin como un sistema operativo para teléfonos móviles, fue en el año 2005 que Google decide comprar Android Inc. En el año 2007 hubo un suceso de gran importancia: El lanzamiento del primer Iphone de Apple el cual revolucionó totalmente la forma que un celular debía de ser. Un año después fue presentado el primer celular HTC Dream que incorporaba el sistema operativo Android, a partir de esa fecha año tras año ambos sistemas operativos han tenido actualizaciones importantes y hoy en día Android cuenta con un sistema muy estable, seguro, funcional, innovador y sobre todo con una gran comunidad de desarrolladores alrededor del mundo. (López M. , 2014)

### 2.1 ¿Qué es Android?

Es una plataforma libre para teléfonos móviles basada en GNU/Linux con licencia GPL. Está basado en una versión modificada del Kernel de Linux. Google y otros miembros de la Open Handset Alliance colaboran en el desarrollo y lanzamiento de Android. El AOSP<sup>10</sup> está encargado del mantenimiento y desarrollo de Android.

---

<sup>10</sup> Android Open Source Project.

En el cuarto trimestre del 2010, el sistema operativo Android fue la plataforma de Smartphones más vendida del mundo, destronando a Symbian de la primera posición por primera vez en 10 años. El logotipo de Android es el robot llamado Andy (Figura 2.1).



**Figura 2.1 Andy el robot, logo de la compañía Android Inc.**

**Fuente:** Extraída de <https://www.android.com/>

## **2.2 El inicio de Android**

En octubre del 2003, Andy Rubin, Rich Miner, Nick Sears y Chris White fundaron Android Inc. en Palo Alto, California (EE. UU).

Entre otros empleados iniciales importantes se incluyen Andy McFadden, que trabajó en WebTV y MOXI, y Chris White, que trabajó en el diseño y la interfaz de WebTV antes de ayudar a fundar Android. (Karch, 2010)

Rubin, cofundador de Danger Inc.<sup>11</sup> (Danger es conocida por crear los teléfonos de Sidekick T-Mobile)<sup>12</sup>, Miner, cofundador de Wildfire Communications Inc. y vicepresidente de tecnología e innovación en Orange (compañía de telefonía en Reino Unido), y los otros empleados iniciales llevaron una considerable experiencia en la industria inalámbrica a la compañía. A pesar de los logros obvios del pasado de los

---

<sup>11</sup> Compañía que se especializó en el diseño de hardware, software y servicios para dispositivos móviles.

<sup>12</sup> El Sidekick incluía varias características incluyendo una cámara digital, mensajes multimedia y un reproductor multimedia. Las opiniones en CNET elogiaron el Sidekick por sus capacidades de mensajería, pero fueron críticas por la falta de conectividad 3G o Wi-Fi.

fundadores y de los primeros empleados, Android Inc. funcionó de forma reservada y simplemente admitió que estaba trabajando en software para teléfonos móviles.

En agosto del 2005, Google compró en ese entonces la pequeña empresa Android Inc. Los empleados principales de Android Inc., entre los que se encontraban Andy Rubin, Rich Miner y Chris White, permanecieron en la compañía después de la adquisición. En el momento de la adquisición, debido al poco conocimiento que se tenía sobre el trabajo de Android Inc., se conjeturó que Google estaba planeando entrar en el mercado de los teléfonos móviles. (Gargenta, 2011)

En Google, el equipo liderado por Rubin desarrolló una plataforma para dispositivos móviles basada en el kernel de Linux. Google puso en el mercado la plataforma para los fabricantes de dispositivos móviles y los operadores con la premisa de proporcionar un sistema flexible y actualizable. Google alineó a una serie de socios dedicados a componentes hardware y software e indicó a los operadores que estaba abierto a varios grados de cooperación por su parte. (Basterra, 2013)

La especulación sobre que el Sistema Operativo Android de Google entraría en el mercado de la telefonía móvil se incrementó en diciembre de 2006. Informes de la BBC<sup>32</sup> y The Wall Street Journal indicaron que Google quería su sistema de búsqueda y sus aplicaciones en teléfonos móviles y que estaba trabajando duro para conseguirlo. Algunos medios de comunicación escritos y en Internet publicaron rápidamente rumores de que Google estaba desarrollando un dispositivo con la marca de fábrica Google.

En el 2007 la Open Handset Alliance<sup>13</sup>, es anunciada, con el fin de desarrollar estándares abiertos para dispositivos móviles. Google forma parte de la Open Handset Alliance por lo que Android es una plataforma de Código abierto.

---

<sup>13</sup> Es un consorcio es una alianza comercial de 84 compañías que se dedica a desarrollar estándares abiertos para dispositivos móviles. Algunos de sus miembros son Google, Dell, Intel y Motorola.

En el 2008, es liberada la versión 1.0 del Android SDK. En octubre del mismo año, se empieza a comercializar lo que es el primer Smartphone con sistema operativo Android, G1 o HTC Dream, manufacturizado por HTC (Figura 2.2) y desarrollado por Google y la Open Handset Alliance. Contaba con una pantalla táctil capacitiva LCD de 3.2 pulgadas y una resolución de 320x480, un teclado QWERTY físico, cámara de 3.15 Megapíxeles, con un procesador Qualcomm MSM7201A a 528 MHZ, 192 MB de RAM y 256 MB de almacenamiento interno que se podía expandir a 16GB.



**Figura 2.2 HTC Dream, primer Smartphone con Sistema Operativo Android**

**Fuente:** Extraída de <http://phonesdata.com/es/smartphones/htc/dream-397/>

En 2009 ya había más de 20 dispositivos que funcionaban con Android como su Sistema Operativo. Se lanzaron las versiones: 1.5 Cupcake, 1.6 Donut y 2.0 Eclair. En el 2010 más de 60 dispositivos contaban con Android y se convierte en la segunda plataforma móvil más vendida. Se lanza la versión 2.2 Froyo. (Gargenta, 2011)



## 2.3 Características

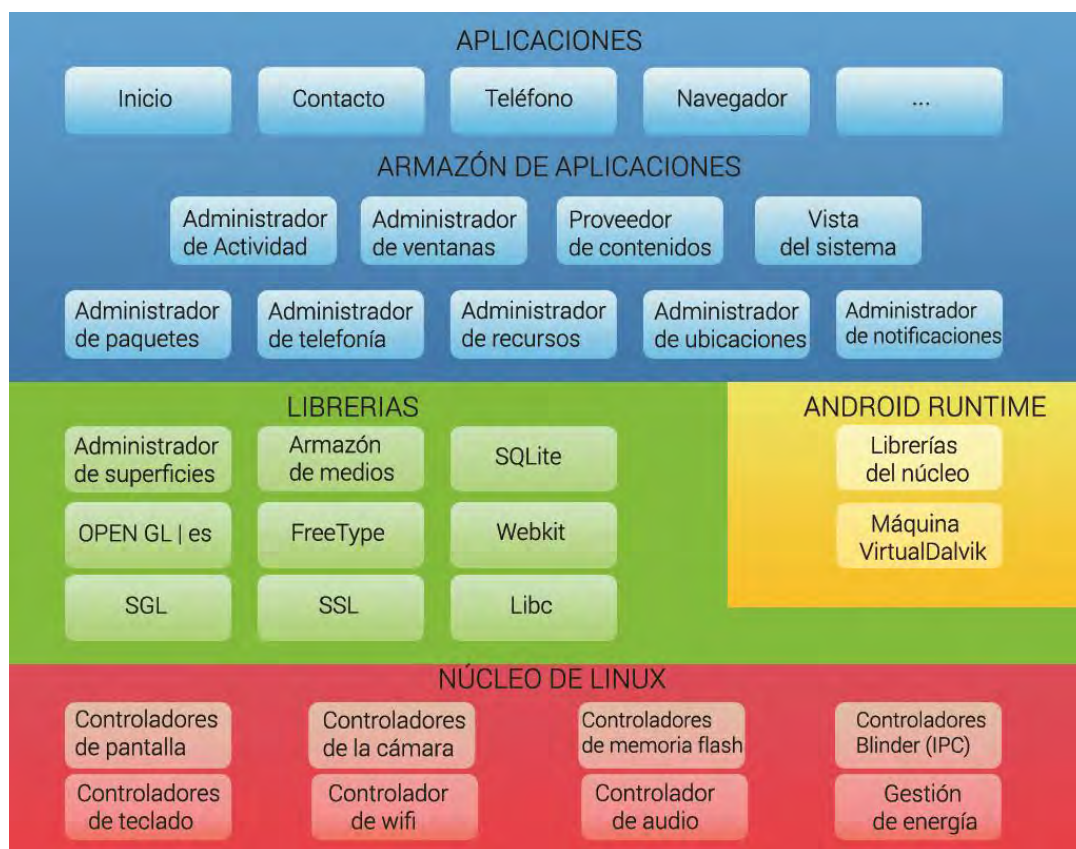
Con la alta demanda que tienen los Smartphones, hoy en día existen una gran variedad de sistemas operativos móviles (IOS, Windows Phone, Firefox OS, Blackberry, Ubuntu, etc.) diferentes sistemas operativos que cuentan con características propias que hacen que se distingan de los demás. A continuación se mencionan las características que hacen que Android sea el sistema operativo más usado en todo el mundo:

- **Plataforma abierta.** Android incluye el Android Open Source Project el cual es de código abierto, lo que permite usar y modificar el sistema. Plataforma de código abierto y desarrollo libre basado en Linux
- **El diseño de la interfaz de usuario (UI).** Al incorporar XML para el diseño de la interfaz de usuario, permite la ejecución de las aplicaciones en dispositivos con diferentes resoluciones y código mejor estructurado.
- **Soporte de Java y muchos formatos multimedia.**
- **Gran cantidad de servicios, sensores y periféricos.** Permite desarrollar aplicaciones en las que se puede interactuar con el medio que nos rodea.
- **Nivel de seguridad.** Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja que hereda de Linux. Además cada aplicación dispone de una serie de permisos que limitan su rango de actuación. (Gironés, 2012).
- **Multitarea real de aplicaciones.**
- **Optimizado para baja potencia y poca memoria.** A partir de la versión 4.4 Kitkat Google ha implementado nuevas medidas para disminuir el excesivo gasto de batería en los Smartphones, incorporando la nueva máquina virtual ART.
- **Utiliza SQLite para el almacenamiento de datos.**
- **Alta calidad de gráficos y sonidos.** Gráficos vectoriales suavizados, gráficos en 3 dimensiones basados en OpenGL. Incorpora los codecs más comunes en audio y video.

- **Variedad de aplicaciones en Play Store.**
- **Soporte de HTML, HTML5, Adobe Flash Player.** (Androidos, 2013)

## 2.4 Arquitectura del Sistema Operativo Android

La estructura (Figura 2.3) se compone de aplicaciones que se ejecutan en un Framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en Lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una API gráfica OpenGL ES 2.0 3D, un motor de renderizado Blink<sup>14</sup>, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic.



**Figura 2.3** Arquitectura por capa del sistema operativo Android

**Fuente:** Reedición propia.

<sup>14</sup> Software que toma contenido marcado (HTML, XML) e información de formateo (CSS, XSL) y luego muestra el contenido con el formato correspondiente en la pantalla de aplicaciones.

El Sistema Operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de XML, 2,8 millones de líneas de lenguaje C, 2.1 millones de líneas de Java. (Gargenta, 2011)

### **2.4.1 Núcleo de Linux**

Está formado por el sistema operativo Linux versión 2.6.

Es el encargado de que el software y el hardware del Smartphone trabajen en conjunto. Las funciones más importantes del mismo, aunque no las únicas, son:

- Administración de la memoria para todos los programas y procesos en ejecución.
- Multiproceso.
- Seguridad.
- Soporte de drivers para dispositivos.
- Administración del tiempo de procesador que los programas y procesos en ejecución utilizan.
- Es el encargado poder acceder a los periféricos del Smartphone.
- Actúa como capa de abstracción entre el hardware y el resto de la pila. (Gironés, 2012)

### **2.4.2 Android Runtime**

Está basado en el concepto de máquina virtual utilizado en Java. Dado las limitaciones de los dispositivos donde ha de correr Android (poca memoria y procesador limitado) no fue posible utilizar una máquina virtual Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones.

Algunas características de la máquina virtual Dalvik que facilitan esta optimización de recursos son: Que ejecuta ficheros Dalvik ejecutables (.dex) - formato optimizado para ahorrar memoria. Además, está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Delega al kernel de Linux algunas funciones como threading y el manejo de la memoria a bajo nivel. También se incluye en el Runtime de Android el core libraries con la mayoría de las librerías disponibles en el lenguaje Java. (AndroidCurso, 2014).

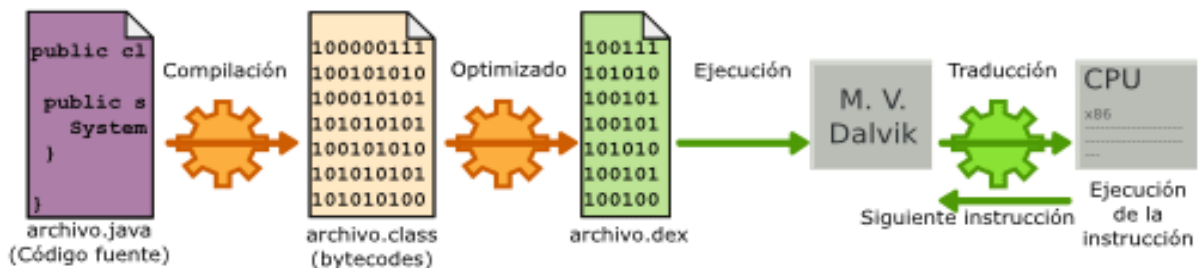


Figura 2.4 Funcionamiento Dalvik

Fuente: Extraído de <http://padinsolutions.blogspot.mx/>

### 2.4.3 Librerías Nativas

La siguiente capa que se sitúa justo sobre el kernel la componen las bibliotecas nativas de Android, también llamadas librerías. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma “más eficiente”.

Las librerías son las siguientes:

- **Librería C del sistema / System C library.** Una derivación de la librería BSD de C estándar (libc), adaptada para dispositivos embebidos basados en Linux.
- **Estructura de los medios / Media Framework.** Librería basada en Packet Video's OpenCORE; soporta códecs de reproducción y grabación de multitud

de formatos de audio vídeo e imágenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.

- **Administrador de superficie / Surface Manager.** Maneja el acceso al subsistema de representación gráfica en 2D y 3D.
- **WebKit.** Soporta un moderno navegador Web utilizado en el navegador Android y en la vista Webview. Se trata de la misma librería que utiliza Google Chrome y Safari de Apple.
- **SGL.** Motor de gráficos 2D.
- **FreeType.** Fuentes en bitmap y renderizado vectorial.
- **SQLite.** Potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.
- **Capa de conexión segura / SSL (Secure Socket Layer).** Proporciona servicios de encriptación. (Gironés, 2012)

#### 2.4.4 Entorno de Aplicación

No se considera una capa en sí mismo, dado que también está formado por librerías. Aquí se encuentran las librerías con las funcionalidades habituales de Java así como otras específicas de Android.

El componente principal del entorno de ejecución de Android es la máquina virtual Dalvik. Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación. (Androidos, 2013)

Cabe aclarar que Dalvik es una variación de la máquina virtual de Java, por lo que no es compatible con el bytecode<sup>15</sup> Java. Java se usa únicamente como lenguaje de programación, y los ejecutables que se generan con el SDK de Android tienen la extensión .dex que es específico para Dalvik, y por ello no podemos correr aplicaciones Java en Android ni viceversa.

Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones, etc.). También conocido como Java SDK.

La arquitectura ha sido diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas (sujetas a las restricciones de seguridad). Este mismo mecanismo permite a los usuarios reemplazar componentes.

Una de las mayores fortalezas del entorno de aplicación de Android es que se aprovecha el lenguaje de programación Java. El SDK de Android no acaba de ofrecer todo lo disponible para su estándar del entorno de ejecución Java (JRE), pero es compatible con una fracción muy significativa de la misma.

Los servicios más importantes que incluye son:

- **Vistas / Views.** Conjunto de vistas, parte visual de los componentes.
- **Administrador de Recursos / Resource Manager.** Proporciona acceso a recursos que no son en código.
- **Administrador de Actividades / Activity Manager.** Maneja el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.
- **Administrador de Notificaciones / Notification Manager.** Permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.
- **Proveedor de contenidos / Content Providers.** Mecanismo sencillo para acceder a datos de otras aplicaciones (como los contactos). (Gargenta, 2011)

---

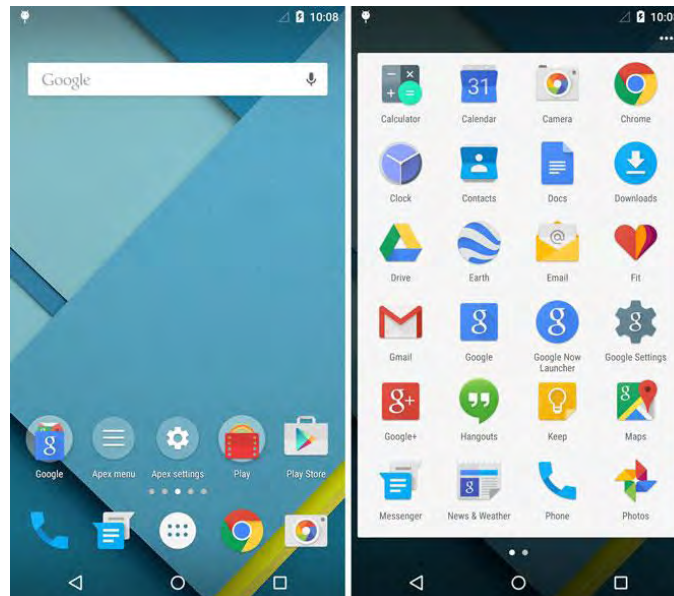
<sup>15</sup> El bytecode un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un archivo binario que contiene un programa ejecutable similar a un módulo objeto, que es un archivo binario producido por el compilador cuyo contenido es el código objeto o código máquina .

## 2.4.5 Aplicaciones

En la última capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++) y las administradas (programadas en Java), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado.

En esta capa se encuentra la aplicación principal del sistema: Inicio (Figura 2.5), es la encargada de ejecutar otras aplicaciones mediante una lista y muestra diferentes escritorios donde se pueden colocar accesos directos a aplicaciones o incluso widgets, que son también aplicaciones de esta capa.

Android proporciona un entorno poderoso para poder programar aplicaciones que cumplan con los requerimientos necesarios. Nada dentro de Android es inaccesible y se puede jugar siempre con las aplicaciones del Smartphone para optimizar cualquier tarea. El potencial de Android se sitúa en el control total que brinda al usuario para que haga de su teléfono un dispositivo a su medida. (AndroidCurso, 2014)



**Figura 2.5 Inicio o Launcher en Android Lollipop**

**Fuente: Edición propia.**

## 2.5 Versiones de Android

Desde el 23 de septiembre de 2008 cuando salió la primera versión de Android, los desarrolladores han mantenido actualizaciones constantes en donde han arreglado bugs y añadido nuevas características en versiones posteriores, esto ha influido a que el sistema no quede estancado como sucedía con plataformas pasadas, que al no tener actualizaciones constantes, el sistema se volvía lento y obsoleto.

A continuación se mencionan las versiones existentes que han sido liberadas.

### 2.5.1 Versión 1.0 Apple Pie

La primera versión de Android es lanzada el 23 de septiembre de 2008, el primer celular en contar con esta versión fue el HTC Dream, el logo es un pay de manzana (Figura 2.6), carecía de muchas funciones con las que ahora cuenta un Smartphone, no se comercializó, entre sus principales características se encuentran:

- Incorporación de una tienda de aplicaciones Android Market.
- Un navegador Web capaz de abrir páginas Web en HTML y XHTML.
- Soporte básico para cámara de fotos. Creación de carpetas.
- Acceder a servidores de correo electrónico por Web soportando los protocolos POP3, IMAP4 y SMTP. Sincronización con los productos de Google: Gmail, Google Calendar y Google Contacts. Mensajería instantánea, SMS y MMS.
- Soporte para fondos de pantalla y Widgets. (Espeso, 2014)



Figura 2.6 Logo versión 1.0 Apple Pie

Fuente: Extraída de <https://www.android.com/>



## 2.5.2 Version 1.1 Banana Bread

Una pequeña actualización fue presentada el 9 de Febrero de 2009 para el HTC Dream, su logo es un pan de plátano (Figura 2.7), en esta versión se resolvían los errores que se encontraban en la versión anterior y se agregaron nuevas características:

- Se añadieron detalles y reseñas sobre lugares y negocios en Google Maps.
- Cambio en la pantalla en llamada así como nueva pantalla en caso de uso de manos libres y posibilidad de mostrar-ocultar el teclado numérico.
- Posibilidad de guardar archivos adjuntos en los mensajes.



Figura 2.7 Logo Versión 1.1 Banana Bread

Fuente: Extraída de <https://www.android.com/>

## 2.5.3 Versión 1.5 Cupcake

Basado en el kernel de Linux 2.6.27 y liberada el 30 de abril de 2009, su logo es un pastelito (Figura 2.8). Hubo varias características y actualizaciones en la interfaz de usuario, mejorando la usabilidad con las siguientes características:

- Camcorder para la grabación y reproducción de videos.
- Capacidad de subir videos a YouTube e imágenes a Picasa directamente desde el teléfono.

- Soporte para Bluetooth A2DP y AVRCP
- Capacidad de conexión automática para conectar a auricular Bluetooth a cierta distancia.
- Transiciones de pantalla animadas
- Teclado táctil QWERTY en pantalla con predicción de texto.
- Widgets de escritorio.
- SDK para el desarrollo de widgets de escritorio por parte de terceros.
- Funciones del portapapeles ampliadas.



Figura 2.8 Logo versión 1.5 Cupcake

Fuente: Extraída de <https://www.android.com/>

### 2.5.4 Versión 1.6 Donut.

Presentada el 15 de septiembre de 2009. Basado en el kernel de Linux 2.6, su logo es una dona glaseada (Figura 2.9).

En esta actualización se incluyó:

- Mejora en el Android Market
- Una interfaz integrada de cámara, video y galería
- La galería ahora permite a los usuarios seleccionar varias fotos para eliminarlas

- Búsqueda por voz actualizada, con respuesta más rápida y mayor integración con aplicaciones nativas, incluyendo la posibilidad de marcar a contactos
- Experiencia de búsqueda mejorada que permite buscar marcadores, historiales, contactos y páginas Web desde la pantalla de inicio.
- Actualización de soporte para CDMA/EVDO, 802.1x, VPN y text-to-speech
- Soporte para resoluciones de pantalla WVGA
- Mejoras de velocidad en las aplicaciones de búsqueda y cámara
- Framework de gestos y herramienta de desarrollo GestureBuilder
- Navegación gratuita turn-by-turn de Google. (Android, 2014)



**Figura 2.9 Logo versión 1.6 Donut**

Fuente: Extraída de <https://www.android.com/>

## 2.5.5 Versión 2.0 / 2.1 Eclair

Basado en el kernel de Linux 2.6.29 liberado el 26 de octubre de 2009. Su logo es un pan alargado cubierto de chocolate (Figura 2.10).

Los cambios incluyeron:

- Velocidad de hardware optimizada
- Soporte para más tamaños de pantalla y resoluciones
- Interfaz de usuario renovada
- Nuevo interfaz de usuario en el navegador y soporte para HTML5
- Una mejor relación de contraste para los fondos
- Mejoras en Google Maps 3.1.2

- Soporte para Microsoft Exchange
- Soporte integrado de flash para la cámara
- Zoom digital
- MotionEvent mejorado para captura de eventos multi-touch
- Teclado virtual mejorado
- Bluetooth 2.1
- Fondos de pantalla animados

El SDK 2.0.1 fue liberado el 3 de diciembre de 2009 y el SDK 2.1 fue liberado el 12 de enero de 2010.



Figura 2.10 Logo versión 2.0 Eclair

Fuente: Extraída de <https://www.android.com/>

## 2.5.6 Versión 2.2 Froyo.

Liberado el 20 de mayo del 2010, basado en el kernel de Linux 2.6.32. Su logo es un vaso con helado de yogurt (Figura 2.11). El nexus one fue el primer smartphone en contar con esta actualización, los cambios incluyeron:

- Optimización general del sistema Android, la memoria y el rendimiento.
- Nueva galería de imágenes.
- Mejoras en la velocidad de las aplicaciones, gracias a la implementación de JIT53.
- Integración del motor JavaScript V8 del Google Chrome en la aplicación Browser.

- Lanzador de aplicaciones mejorado con accesos directos a las aplicaciones de teléfono y Browser.
- Funcionalidad de Wi-Fi hotspot y tethering por USB.
- Permite desactivar el tráfico de datos a través de la red del operador.
- Marcación por voz y compartir contactos por Bluetooth.
- Pantalla de desbloqueo con código PIN
- Soporte para la instalación de aplicación en la memoria expandible.
- Soporte para pantallas de alto número de Puntos por pulgada, tales como 720p.



Figura 2.11 Logo versión 2.2 Froyo

Fuente: Extraída de <https://www.android.com/>

### 2.5.7 Versión 2.3 Gingerbread

El 6 de diciembre de 2010, fue liberado. Basado en el kernel de Linux 2.6.35.7. Su logo es una galleta de jengibre con la forma de Andy el robot (Figura 2.12) Los cambios incluyeron:

- Actualización del diseño de la interfaz de usuario
- Soporte para pantallas extra grandes y resoluciones WXGA y mayores
- Soporte nativo para telefonía VoIP SIP
- Soporte para reproducción de videos WebM/VP8 y decodificación de audio AAC
- Nuevos efectos de audio como reverberación, ecualización, virtualización de los auriculares y refuerzo de graves.
- Soporte para Near Field Communication (NFC).

- Funcionalidades de cortar, copiar y pegar disponibles a lo largo del sistema.
- Teclado multi-táctil rediseñado.
- Mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos.
- Recolección de elementos concurrentes para un mayor rendimiento
- Soporte nativo para más sensores (como giroscopios y barómetros)
- Un administrador de descargas para descargar archivos grandes



Figura 2.12 Logo versión 2.3 Gingerbread

Fuente: Extraída de <https://www.android.com/>

### 2.5.8 Versión 3.0 / 3.1 / 3.2 Honeycomb

Lanzada en febrero del 2011. Actualización únicamente para tablets, la Tablet Xoom de Motorola fue la primera en contar con esta actualización. Su logo es una abeja con características de Andy el robot (Figura 2.13), llamada panal de miel.

Sus características son:

- Mejor soporte para tablets
- Escritorio 3D con widgets rediseñados
- Sistema multitarea mejorado
- Mejoras en el navegador Web predeterminado.
- Soporte para videochat mediante Google Talk
- Mejor soporte para redes Wi-Fi
- Añade soporte para una gran variedad de periféricos y accesorios con conexión USB: Teclados, ratones, hubs, dispositivos de juego y cámaras digitales.

- Se añade soporte opcional para redimensionar correctamente las aplicaciones inicialmente creadas para móvil para que se vean bien en Tablets.



Figura 2.13 Logo versión 3.0 Honeycomb

Fuente: Extraída de <https://www.android.com/>

### 2.5.9 Versión 4.0 Ice Cream Sandwich.

Interfaz estilo Honeycomb, en cualquier dispositivo, se busca la homogeneidad entre teléfonos, televisiones, tablets y netbooks. Su logo es un sándwich de helado con la forma de Andy el robot (Figura 2.14).

- Nueva fuente tipográfica Roboto
- Interfaz Holo
- Sistema de gestión de notificaciones mejorado
- Multitarea mejorada
- Sugerencias y diccionarios para el teclado virtual
- Nuevo diseño y funcionalidades para la pantalla “Home”
- Android Beam, funcionalidad para transferir datos entre dos dispositivos vía NFC
- Función de desbloqueo mediante el rostro
- Nuevas funciones para la visualización y gestión del consumo de datos
- Nuevas aplicaciones de correo y calendario
- Herramienta integrada de captura de (botones de volumen y encendido simultáneamente)

- Soporte MKV
- Soporte Stylus (lápiz táctil).



Figura 2.14 Logo versión 4.0 Ice Cream Sandwich

Fuente: Extraída de <https://www.android.com/>

## 2.5.10 Versión 4.1 Jelly Bean

Lanzada en julio del 2012, en esta versión se destaca la desaparición del soporte para Flash Player. Su logo es un recipiente con la forma de Andy el robot, el cual en su interior tiene grageas dulces

- Rendimiento del sistema y gráfico mejorado gracias a Project Butter
- Estreno de Google Now, el servicio-asistente de voz inteligente de Google
- Navegador Google Chrome
- Búsqueda mediante voz mejorada
- Rediseño de la tipografía Roboto
- Mejoras en la corrección ortográfica y la predicción del teclado
- Dictado de voz offline

En noviembre, llegó Android 4.2 una actualización que mantenía el mismo nombre de la versión (Jelly Bean).

- Nuevo panel de control
- Acceso a widgets y cámara fotográfica desde la pantalla de bloqueo



- Soporte para Miracast (función de streaming de vídeo y audio desde el terminal)
- Soporte para varios perfiles de usuario
- Photosphere, captura de fotografías panorámicas de 360°
- Gestual Mode para personas invidentes

El 24 de julio del 2013 Google anuncia Android 4.3

- Soporte multiusuario y de perfiles mejorado
- Soporte OpenGL ES 3.0
- Compatible con TRIM
- Servicios de localización Wi-Fi mejorados (Figura 2.15).



Figura 2.15 Logo versión 4.1 Jelly Bean

Fuente: Extraída de <https://www.android.com/>

### 2.5.11 Versión 4.4 Kit Kat

Lanzado el 31 de octubre del 2013 KitKat ofrece nuevas posibilidades al usuario al tiempo que corrige uno de sus principales defectos: La fragmentación de versiones. Google llegan a un acuerdo con Nestle para utilizar el nombre de uno de sus productos en la versión del SO. Su logo es un KitKat con la forma de Andy el robot.

- Rebaja de requisitos hardware para corregir la fragmentación de versiones
- Compatible con terminales con 512 MB de memoria RAM
- Reducción del consumo de batería mediante la optimización de los sensores

- Incluye la suite ofimática QuickOffice
- Servicios de almacenamiento online integrados: Google Drive, Box...
- Soporte para infrarrojos. Usa el móvil como mando de TV.
- Soporte Bluetooth HID a través de GATT y Bluetooth Message Access Profile
- Captura de pantalla en vídeo.

Google presentó KitKat 4.4 al mismo tiempo que el nuevo Nexus 5, otro smartphone de Google firmado junto a LG y basado en el modelo de gama alta G2.

Android amplía sus horizontes en 2014 y se instala en todo tipo de dispositivos: Android TV, smarwatches, Android Auto y otros dispositivos para el entretenimiento como Nexus Player. (Figura 2.16)



**Figura 2.16 Logo versión 4.4 Kit Kat**

Fuente: Extraída de <https://www.android.com/>

## 2.5.12 Versión 5.0 Lollipop

El 12 de noviembre del 2014 fue lanzada la versión 5.0 Lollipop y uno de sus rasgos más llamativos ha sido la inclusión Material Design<sup>16</sup>, un nuevo lenguaje de diseño que unificará la usabilidad en cualquier tipo de dispositivo. El logo es Andy sosteniendo una paleta de dulce.

---

<sup>16</sup> Lenguaje de diseño desarrollado por google el cual busca un diseño más limpio, en el que predominan animaciones y transiciones de respuesta,

- Nuevo diseño basado en Material Design que logra un flujo de trabajo más fluido
- Interfaz que se adapta a cualquier tamaño de dispositivo
- Renovado sistema de notificaciones inteligente
- Aumento del rendimiento energético.
- Función Android Smart Lock.
- Modo "Invitado".
- El entorno de ejecución de aplicaciones ART (Android RunTime) pasa a ser el predeterminado.
- Soporte para sistemas de 64 bits

El 9 de marzo del 2015 es lanzada la actualización 5.1 para Lollipop. Entre las novedades se encuentra una gestión mejorada de la batería para mayor autonomía, y mejor rendimiento del procesador.

Otra de las grandes novedades de esta actualización es la llegada del soporte nativo para múltiples tarjetas SIM, además de un nuevo sistema de protección en caso de robo (Gizmodo, 2014).

- Modo silencioso añadido tras su breve desaparición en la versión 5.0.
- Mejoras en la gestión de la memoria RAM.
- Problema de aplicaciones que cierran inesperadamente arreglado.
- Mejora de gestión de la batería.
- Problema de excesivo consumo de red en WiFi, arreglado.



**Figura 2.17** Logo versión 5.0 Lollipop

Fuente: Extraída de <https://www.android.com/>

### 2.5.13 Versión 6.0 Marshmallow

El 5 de octubre del 2015 fue lanzada para Nexus 5, Nexus 6, Nexus 7 y Nexus 9. Con Marshmallow por fin hubo una mejora muy significativa respecto al consumo de batería cuando el teléfono no está en uso. El logo de esta versión es Andy sosteniendo un malvavisco (Figura 2.18). Entre las características principales se encuentran las siguientes.

- Mejoras en la batería.
- Menú para la gestión de la memoria RAM.
- Permite controlar el volumen de todo, multimedia y alarma.
- Soporte para identificación por huella dactilar.
- Android pay basado en NFC y Host Card Emulation.
- Respaldo en la nube con Google Drive.
- Incluye un menú de impresión.
- Cambio en el menú de ajustes.
- Mejoras para el ahorro de batería.
- Permite acceder a Google Now desde la pantalla de bloqueo.
- Función "Now On Tap".
- Eliminar aplicaciones desde el escritorio.
- El Administrador de archivos permite buscar, copiar, compartir, filtrar y borrar.



Figura 2.18 Logo versión 6.0 Marshmallow

Fuente: Extraída de <https://www.android.com/>



Cuenta con almacenamiento de datos SQLite, tecnologías de conectividad 3G, 4G, Mensajería SMS y MMS son formas de mensajería, incluyendo mensajería de texto y ahora Firebase Cloud Messaging<sup>17</sup> es parte del servicio de Push Messaging de Android.

El navegador Web, incluido en Android está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript V8 de Google Chrome. El navegador obtiene una puntuación de 93/100 en el test Acid3.

Soporte de Java, aunque las aplicaciones son escritas en Java, no hay una Máquina Virtual de Java en la plataforma. El código no es ejecutado. Este se compila en el ejecutable Dalvik y corre en la Máquina Virtual Dalvik. Es una máquina virtual especializada diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados. El soporte para J2ME, puede ser agregado mediante aplicaciones de terceros como el J2ME MIDP Runner.

Soporte para streaming Streaming RTP/RTSP (3GPP PSS, ISMA), descarga progresiva de HTML. (Scoello, 2014).

## **2.6 Entorno de programación Android Studio**

Para el desarrollo de aplicaciones móviles es esencial contar con un entorno de programación completo con las herramientas necesarias, diseñado para manipular elementos básicos y no tan básicos, como botones, etiquetas, layouts o controles de zoom, aunado a esto también permite depurar y correr aplicaciones con diferentes dispositivos físicos y virtuales.

---

<sup>17</sup> Es un servicio de Google cuya función popular es el envío de push notifications desde una aplicación servidor hacia una aplicación cliente. Esto es posible gracias a la intervención del servicio de mensajería del servidor de Firebase.

Actualmente existen diversos y excelentes IDEs, para desarrollar en Android: Netbeans, Motodev, Eclipse y Android Studio, por citar algunos.

Eclipse fue uno de los entornos de programación más utilizados para programar aplicaciones en Android, debido a la facilidad con la que se instalaban las extensiones y librerías, hacía que se contara con un entorno óptimo.

La presente tesis se enfocará en Android Studio, por ser este mismo, el entorno desarrollado por Google, resulta ser el más recomendable.

El 16 de mayo en la edición 2013 del Google /IO, Google anunció un nuevo IDE: Android Studio, en su versión 0.1 en etapa de prueba, después de varias mejoras que se le hicieron, el 8 de diciembre del 2014 Google lanzan la versión estable 1.0 con esto se anunciaba lo que sería el entorno de programación oficial de desarrollo para Android, incorporando nuevas características como:

- Soporte para programar aplicaciones para Android Wear.
- Herramientas Lint, que detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador.
- Utiliza ProGuard, optimiza y reduce el código del proyecto al exportar a APK, útil para dispositivos de gama baja con limitaciones de memoria interna.
- Gradle herramienta encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de testing, compilación o empaquetado.
- Nueva interfaz específica para el desarrollo en Android.
- Permite la importación de proyectos realizados en el entorno Eclipse, que a diferencia de Android Studio (Gradle) utiliza ANT.
- Posibilita el control de versiones accediendo a un repositorio como Mercurial, Git, Github o Subversion.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- Vista previa en diferentes dispositivos y resoluciones.
- Integración con Google Cloud Platform, para el acceso a los diferentes servicios que proporciona Google en la nube.

- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo xml. (López E. , 2014)

## 2.6.1 Requisitos del sistema

Android Studio está disponible para Linux, Mac Os y Windows, antes de instalarlo en la computadora es necesario tener en cuenta los siguientes requisitos (véase Tabla 2.1) para un mejor funcionamiento de este.

Tabla 2.1 Requisitos del sistema (SDK, 2014)

Fuente: Edición propia

Linux	Mac Os	Windows
Escritorio GNOME o KDE, la librería GNU C glibc en su versión 2.11 o superior.	Mac OS X 10.8.5 o superior	Microsoft Windows 8/7/Vista/2003 (32 o 64 bit)
Mínimo 2GB de RAM como mínimo		
400 MB de espacio libre en el disco duro		
1GB extra para Android SDK, Imágenes del sistema para emulador, etc.).		
Resolución de pantalla de 1280×800 o superior.		
Oracle Java Development Kit (JDK) 7 o superior.		

## 2.6.2 Instalación de Android Studio en Linux

En la página oficial para desarrolladores Android se encuentra la descarga del SDK de Android: <http://developer.android.com/sdk/index.html>.

Este kit de desarrollo proporciona las librerías APIs y aquellas herramientas para desarrollar y compilar la aplicación. A continuación se mencionan los pasos a seguir para la instalación de Android Studio en Ubuntu.



## Paso 1. Descarga del SDK

1. Ir a la dirección de la página oficial de Desarrolladores Android <http://developer.android.com/sdk/index.html> para descargar Android Studio (Figura 2.20), elegir sistema operativo y la versión de 64 o 32 bits correspondiente (Figura 2.21).

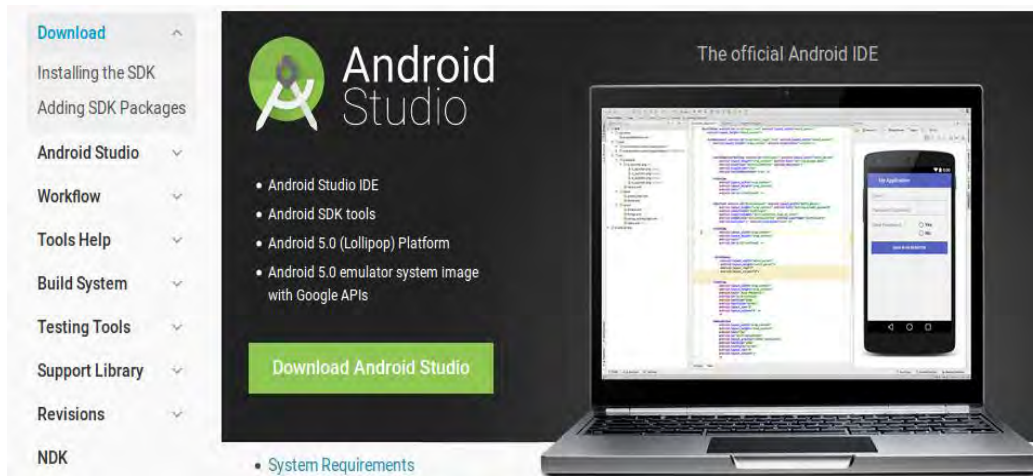


Figura 2.20 Página de descarga Android SDK

Fuente: Edición propia

Linux	<a href="#">android-sdk_r24.1.2-linux.tgz</a>	168121693 bytes	68980e4a26cca0182abb1032abffbb72a1240c51
-------	---	-----------------	--

All Android Studio Packages

Select a specific Android Studio package for your platform. Also see the [Android Studio release notes](#).

Platform	Package	Size	SHA-1 Checksum
Windows	<a href="#">android-studio-bundle-135.1740770-windows.exe</a> (Recommended)	856233768 bytes	7484b9989d2914e1de30995fbaa97a271a514b3f
	<a href="#">android-studio-ide-135.1740770-windows.exe</a> (No SDK tools included)	242135128 bytes	5ea77661cd2300cea09e8e34f4a2219a0813911f
	<a href="#">android-studio-ide-135.1740770-windows.zip</a>	261667054 bytes	e903f17cc6a57c7e3d460c4555386e3e65c6b4eb
Mac OS X	<a href="#">android-studio-ide-135.1740770-mac.dmg</a>	261303345 bytes	f9745d0fec1eefd498f6160a2d6a1b5247d4cda3
Linux	<a href="#">android-studio-ide-135.1740770-linux.zip</a>	259336386 bytes	e8d166559c50a484f83ebfec6731cc0e3f259208

Figura 2.21 Selección de sistema operativo

Fuente: Edición propia

Una vez que se ha descargado completamente el SDK, generar una carpeta nueva con nombre “Programas” en la ubicación /home.

Descomprimir el archivo y moverlo en la carpeta que se ha creado.

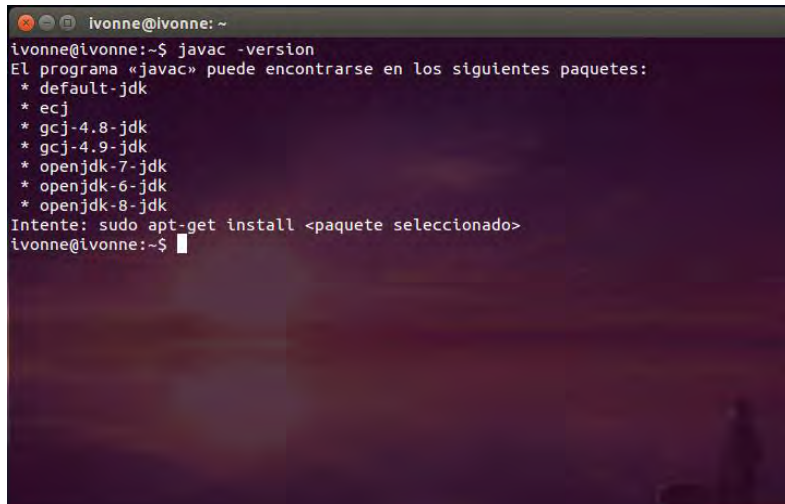
Si se cuenta con un sistema de 64bits se deben instalar las librerías de 32bits con el siguiente comando en la terminal

```
$ sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1
```

## Paso 2. Instalación de Java

Antes de iniciar la instalación es necesario comprobar la versión de Java con la que cuenta el sistema (Figura 2.22), abrir una terminal y ejecutar el comando:

```
javac -versión
```



```
ivonne@ivonne:~$ javac -version
El programa «javac» puede encontrarse en los siguientes paquetes:
* default-jdk
* ecj
* gcj-4.8-jdk
* gcj-4.9-jdk
* openjdk-7-jdk
* openjdk-6-jdk
* openjdk-8-jdk
Intente: sudo apt-get install <paquete seleccionado>
ivonne@ivonne:~$
```

Figura 2.22 Comprobar versión de Java instalado

Fuente: Edición propia

Si en la terminal aparece el mensaje: “El programa java puede encontrarse en los siguientes paquetes”, Java no está instalado. En ese caso se agregará el repositorio PPA Webupd8team Java, en el sistema para instalar Oracle java 8 usando los siguientes comandos:

```
sudo add-apt-repository ppa:Webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

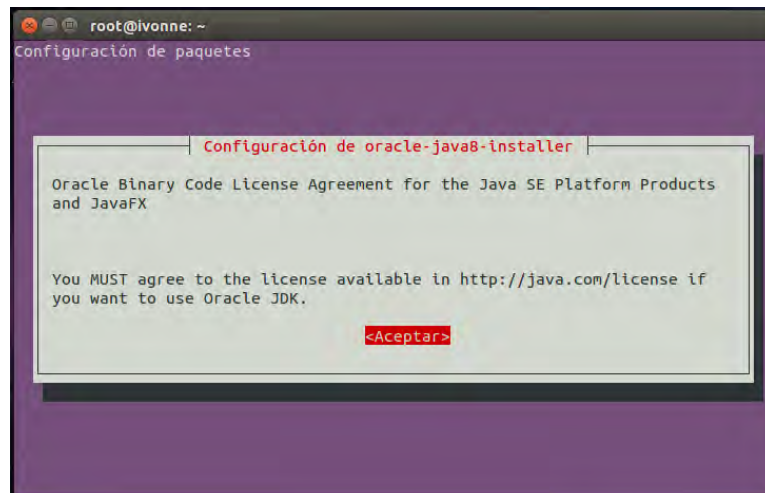


Figura 2.23 Instalación de Java

Fuente: Edición propia

Con esto el instalador de java se ha ejecutado y aparecerá en pantalla (Figura 2.23), seguir las instrucciones del instalador y esperar a que se instale.

### Paso 3. Instalación Android Studio

En la ubicación Programas/android-studio/bin se encuentra el archivo Shell “studio.sh” el cuál se ejecutará desde la terminal.

```
cd Programas/android-studio/bin
./studio.sh
```

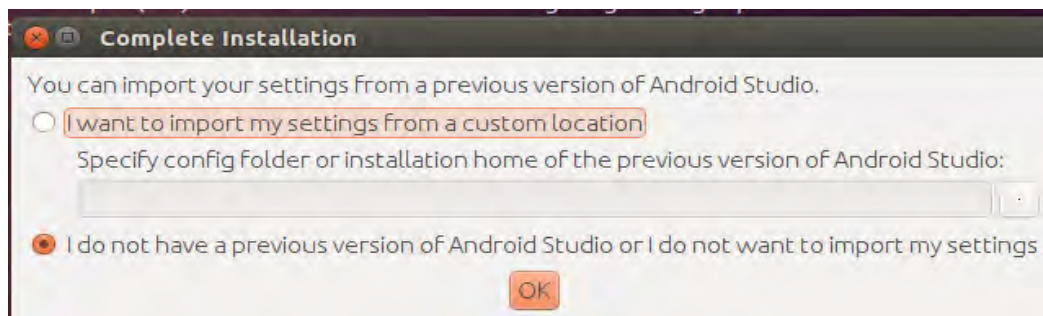
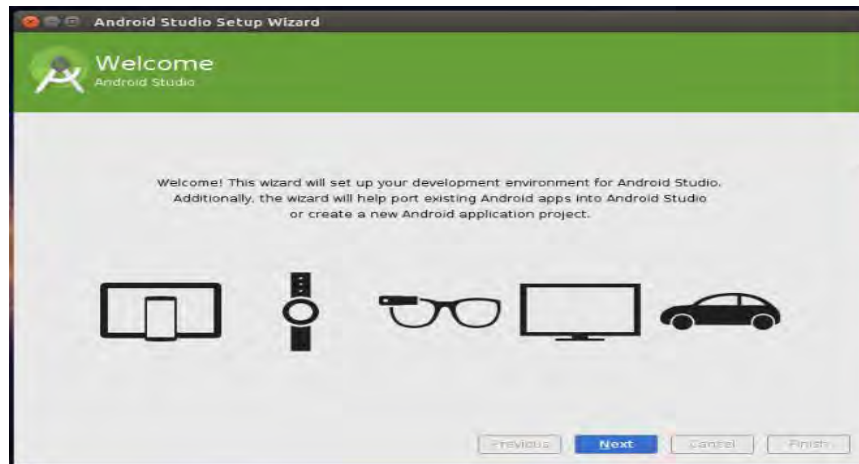


Figura 2.24 Ventana al ejecutar archivo android.sh

Fuente: Edición propia

Seleccionar la opción, “I do not have a previous version of Android Studio or I do not want to import my settings” (No tengo una version anterior de Android Studio o no quiero importar mis configuraciones) (Figura 2.24).

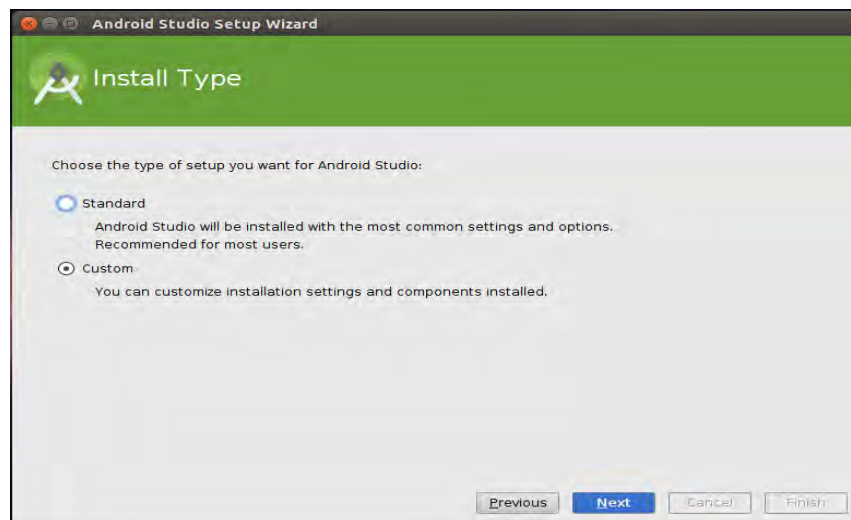
Presionar “ok”, una vez hecho esto aparecerá la pantalla de bienvenida de Android Studio (Figura 2.25). Seguir el tutorial de Android Studio.



**Figura 2.25** Ventana inicio de instalación Android Studio

**Fuente:** Edición propia

En la siguiente pantalla se elegirá el tipo de instalación, estándar o personalizada (custom). Seleccionar Personalizada y clic en el botón Next (Figura 2.26).



**Figura 2.26** Instalación personalizada

**Fuente:** Edición propia

Seleccionar los componentes que se quieren instalar y/o actualizar. (Figura 2.27). Clic en el botón Next (siguiente).

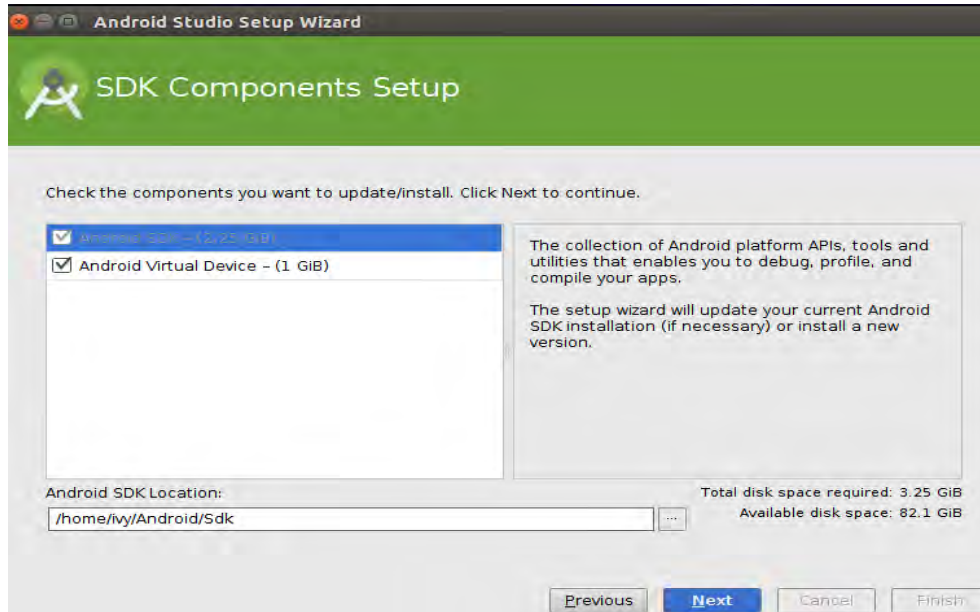


Figura 2.27 Instalación de componentes

Fuente: Edición propia

En la siguiente ventana (Figura 2.28) se muestran los componentes que se desean instalar, marcar Android Virtual Device (Dispositivo Virtual Android) y dar clic en Next (Siguiente).

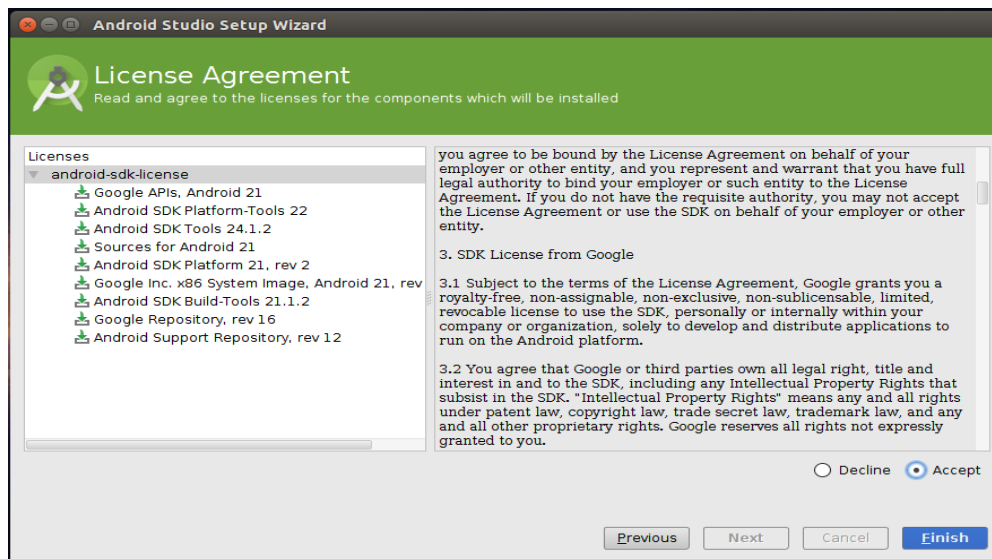


Figura 2.28 Componentes a instalar

Fuente: Edición propia

Marcar la opción Accept (Aceptar) para aprobar los acuerdos de licencia. Después dar clic en Finish (Finalizar) (Figura 2.28). Cuando los componentes terminen de descargarse dar clic en el botón Finish (Finalizar). (Figura 2.29).

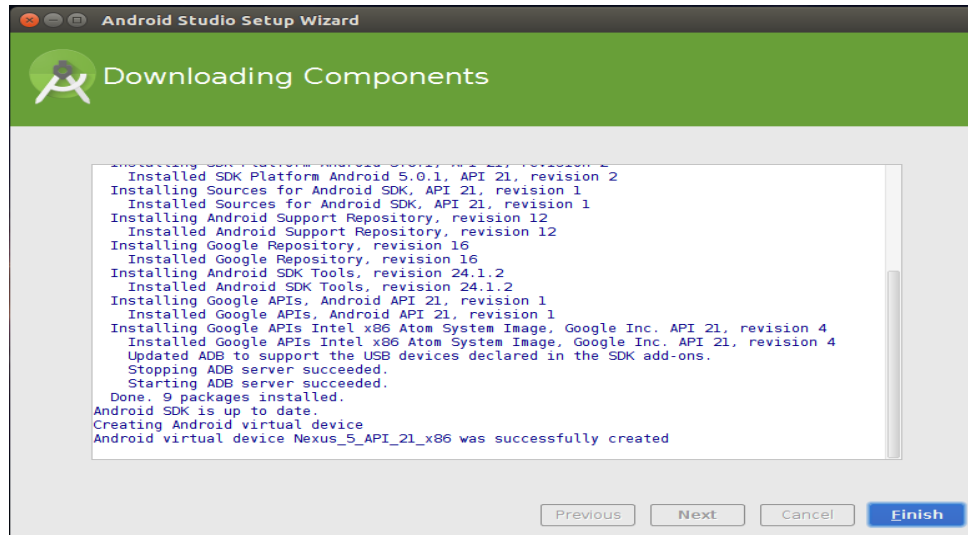


Figura 2.29 Descarga de los componentes necesarios

Fuente: Edición propia

Si después de esto aparece una ventana como la de la (Figura 2.30), Android Studio se ha instalado, ahora es necesario configurar y descargar las herramientas y APIs que se utilizarán para el desarrollo de aplicaciones.



Figura 2.30 Configuración Android Studio

Fuente: Edición propia

En el menú que se muestra dar clic en Configure (Configurar) y seleccionar Android SDK Manager (Administrador del SDK de Android). En el SDK Manager (Figura 2.31), seleccionar la carpeta tools (herramientas), seleccionar también la API 20 Android 4.4.

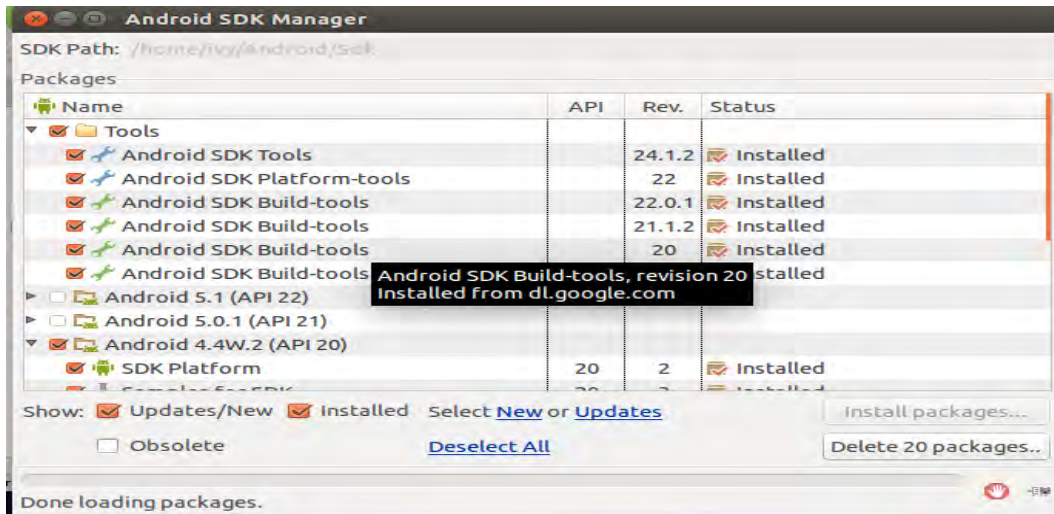


Figura 2.31 Descarga de paquetes

Fuente: Edición propia

Para desarrollar con las API's de Google, es indispensable instalar el paquete de los servicios de Google Play, Abrir el directorio "Extras" y agregar a la instalación:

- Android Support Repository.
- Android Support Library.
- Google Repository.
- Google Play Services.

#### Paso 4. Crear lanzador

Para finalizar la instalación, es esencial la creación de un lanzador en el escritorio para acceder fácilmente a la aplicación en este caso Android Studio.

En la terminal escribir lo siguiente para instalar gnome-panel en caso de que no esté instalado en el equipo.

```
sudo apt-get update
```

```
sudo apt-get install gnome-panel
```

Una vez instalado se procederá a crear el lanzador con el siguiente comando.

```
gnome-desktop-item-edit ~/Escritorio --create-new
```

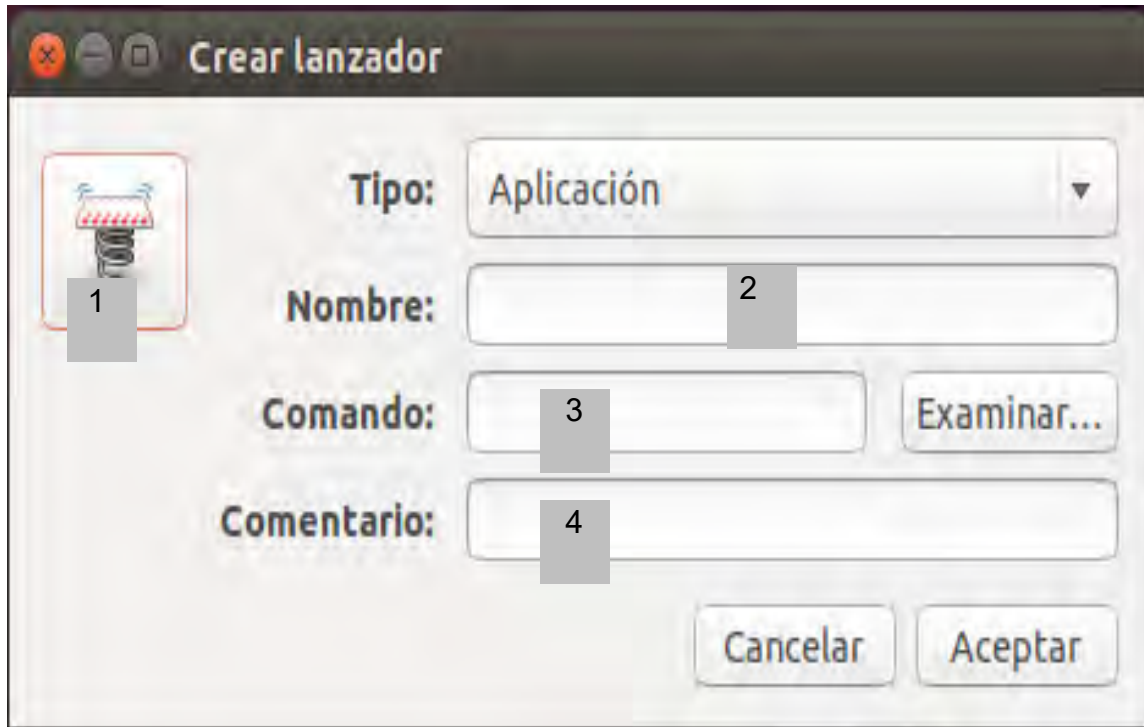


Figura 2.32 Ventana creación del lanzador

Fuente: Edición propia

En la ventana “Crear lanzador”:

- 1) Escoger un icono para el lanzador.
- 2) Seleccionar tipo Aplicación
- 3) Nombrar el lanzador



- 4) Clic en el botón Examinar para buscar el archivo ejecutable “studio.sh (Programas/android-studio/bin) (Figura 2.33)

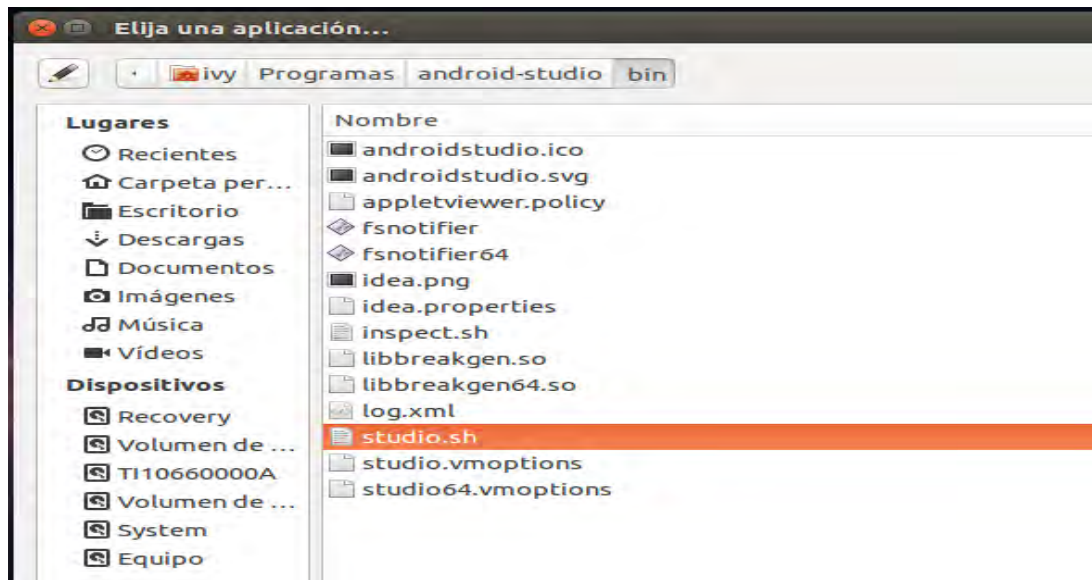


Figura 2.33 Ubicación archivo studio.sh

Fuente: Edición propia

También se puede crear un lanzador desde Android Studio, ir a la pestaña Tools (Herramientas) / Create desktop entry (Crear entrada en el escritorio) y dar clic en ok y después reiniciar Android Studio. (Figura 2.34).

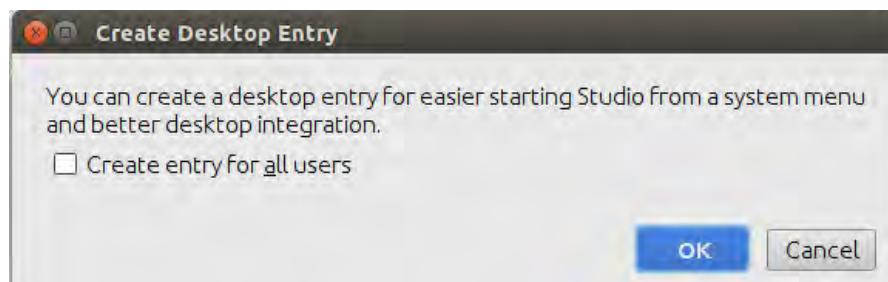


Figura 2.34 Crear lanzador en Android Studio

Fuente: Edición propia

En el dash de Ubuntu buscar Android Studio y arrastrarlo a la barra del launcher para tener un lanzador en el escritorio.

## 2.7 Primer proyecto Hola Mundo

El entorno de programación Android Studio ya ha sido instalado sin embargo aún es necesario configurar las herramientas de trabajo y conocer lo esencial acerca del entorno de programación Android Studio. En este subtema se conocerán los elementos de un proyecto Android y los componentes de una aplicación, todo esto con el fin de comprender mejor el IDE Android Studio y poder realizar la primera aplicación “Hola Mundo”.

Abrir Android Studio, en el menú de inicio de seleccionar la opción Start a new Android Studio Project (Crear un nuevo proyecto de Android) (Figura 2.35)

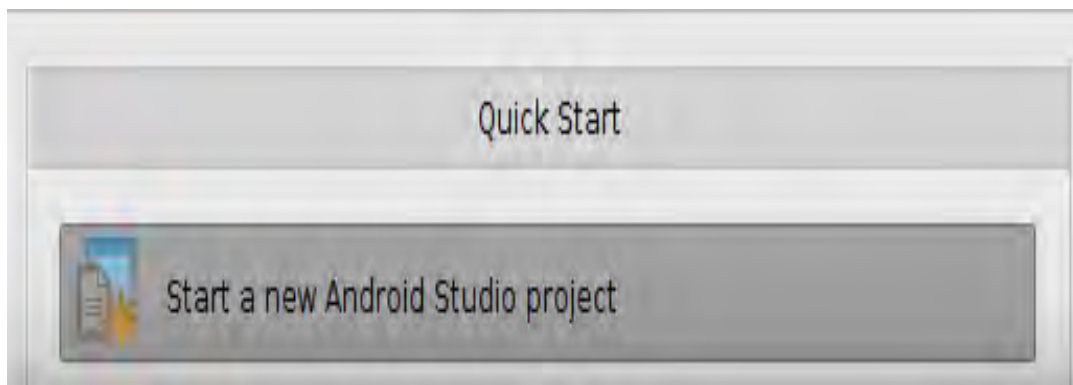
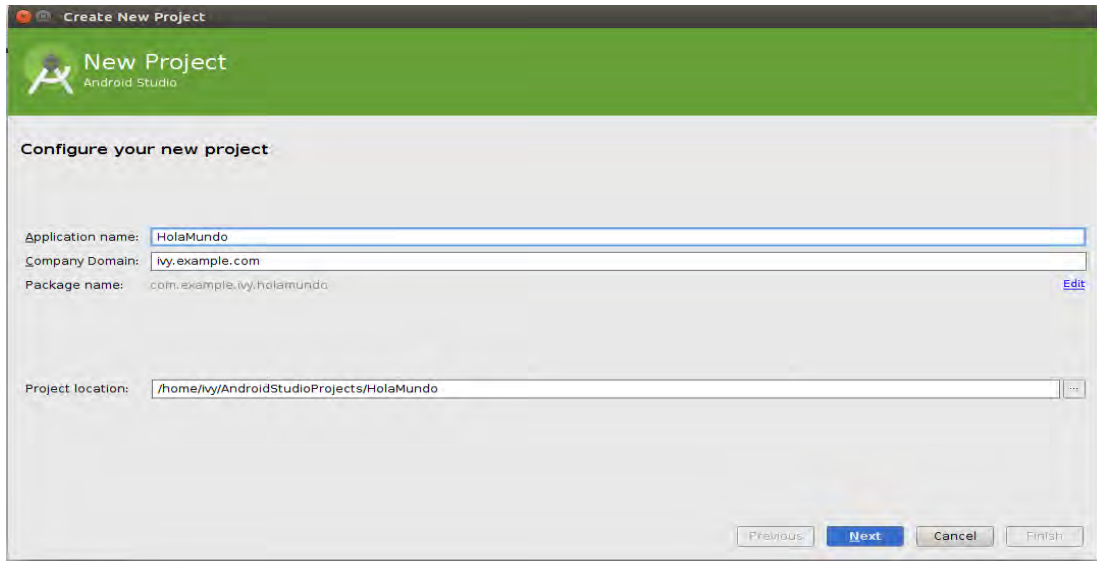


Figura 2.35 Crear un nuevo proyecto en Android Studio

Fuente: Edición propia

Mostrará una ventana en donde se llenarán los campos con la siguiente información.

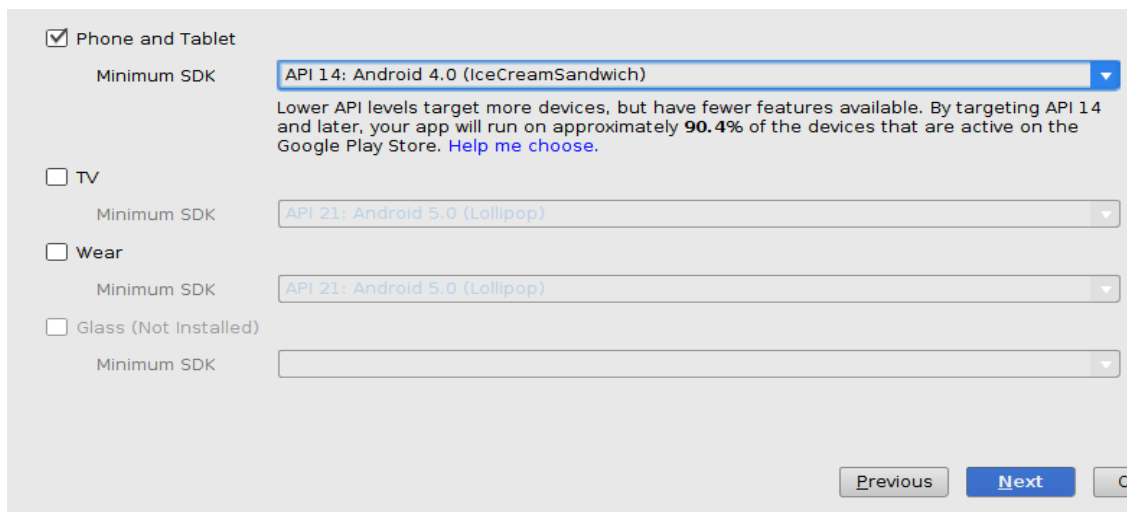
- Application name (Nombre de la aplicación), nombrar la aplicación como HolaMundo.
- Company Domain (Dominio de la compañía) en caso de contar con dominio ponerlo sino escribir my.example.com, como se muestra en la (Figura 2.36) dar clic en botón Next (Siguiete).



**Figura 2.36 Nombre de la aplicación**

**Fuente: Edición propia**

En la siguiente ventana (Figura 2.37) se elegirá el SDK mínimo que se requiere para la aplicación a realizar, también muestra el porcentaje aproximado de los dispositivos que cuentan con ese nivel de API activos en el Play Store, también se debe tomar como referencias las características que se han incluido en cada versión, esto con el fin de trabajar de manera congruente entre el porcentaje de dispositivos y características incluidas en cada versión, como se ha mencionado en el subtema 2.5.



**Figura 2.37 Selección de SDK de la aplicación**

Fuente: Edición propia

Para el ejemplo que se va a desarrollar seleccionar API 14 Android 4.0 IceCreamSandwich. Android Studio también permite programar en dispositivos de tv, auto, wear y Glass, una vez que se ha seleccionado el nivel de API, seleccionar el dispositivo en el cual se desea trabajar: Phone and Tablet (Telefono y Tablet ). Dar clic en el botón Next (Continuar).

En la siguiente ventana se muestran las actividades en las cuales se va a desarrollar la aplicación (Figura 2.38). Antes de empezar un proyecto es esencial saber con qué tipos de actividades se va a trabajar y cuál va a ser su finalidad. Seleccionar Blank Activity (Actividad en blanco) y clic en el botón Next (Siguiete) para continuar con el asistente.

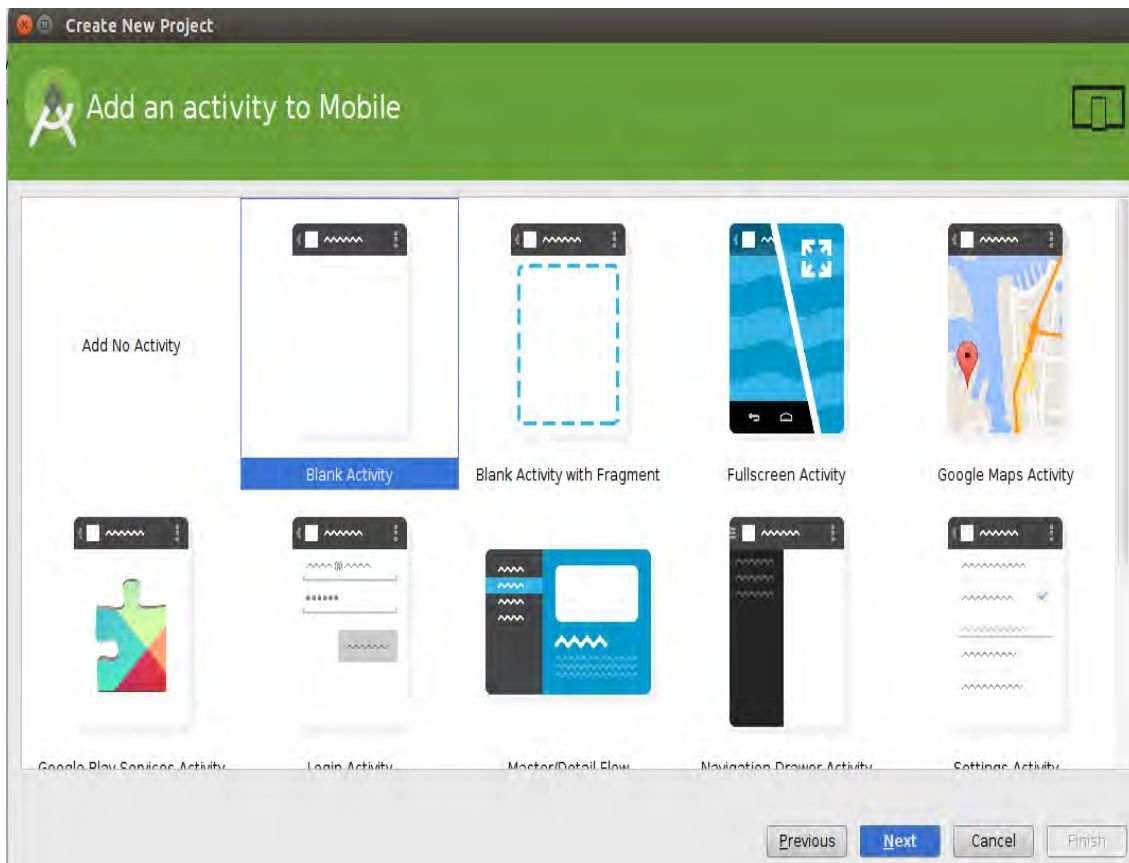


Figura 2.38 Selección de Actividad

Fuente: Edición propia

Por último nombrar la actividad principal y dejar los valores por defecto.

- Activity Name (nombre de la actividad): MainActivity
- Layout Name (nombre del layout): Activity\_main
- Title (Titulo): MainActivity
- Menu Resource Name (nombre del recurso del menu): Menu\_main.

Dar clic en el botón Finish (Finalizar). (Figura 2.39).

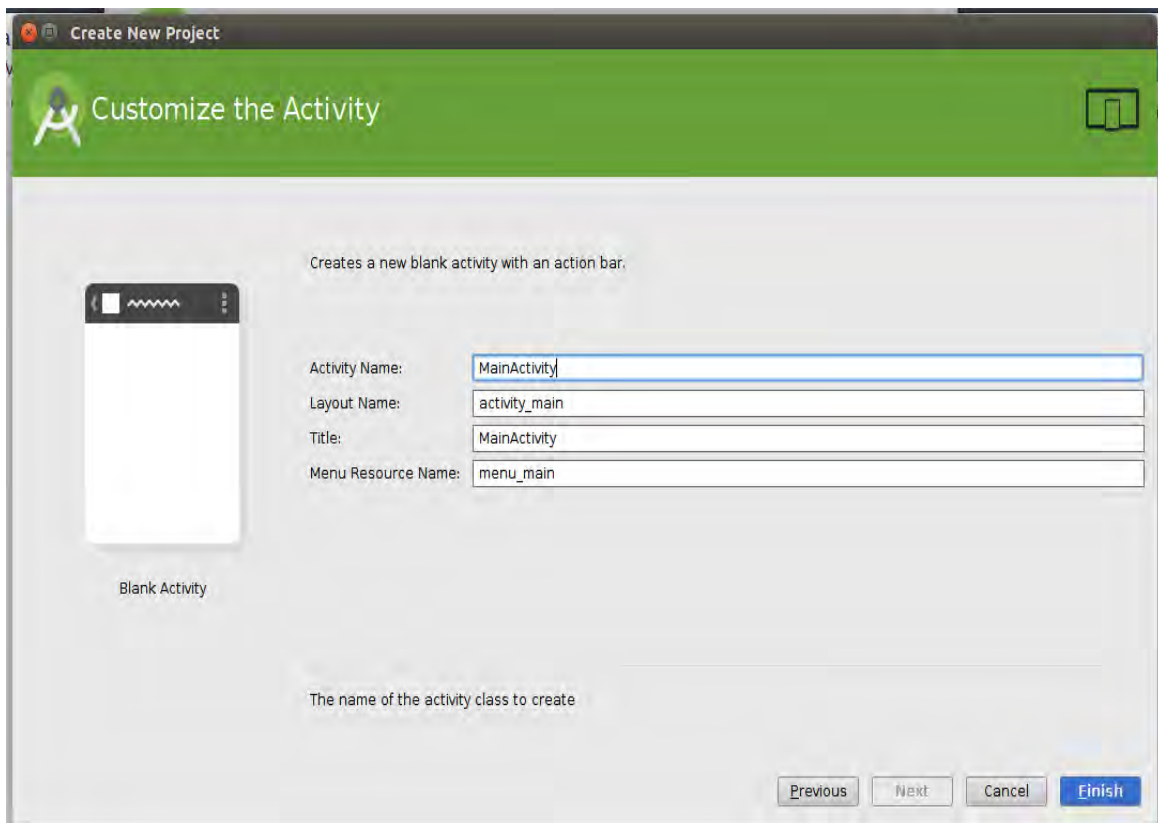


Figura 2.39 Nombrar elementos de la actividad

Fuente: Edición propia

## 2.8 Elementos de un proyecto Android

El proyecto ha sido creado, Android Studio muestra el `activity_main.xml` que es la vista en la que se observan las herramientas que proporciona el IDE de desarrollo. En la (Figura 2.40) se muestran la interfaz de Android Studio, a continuación una explicación de las ventanas y herramientas

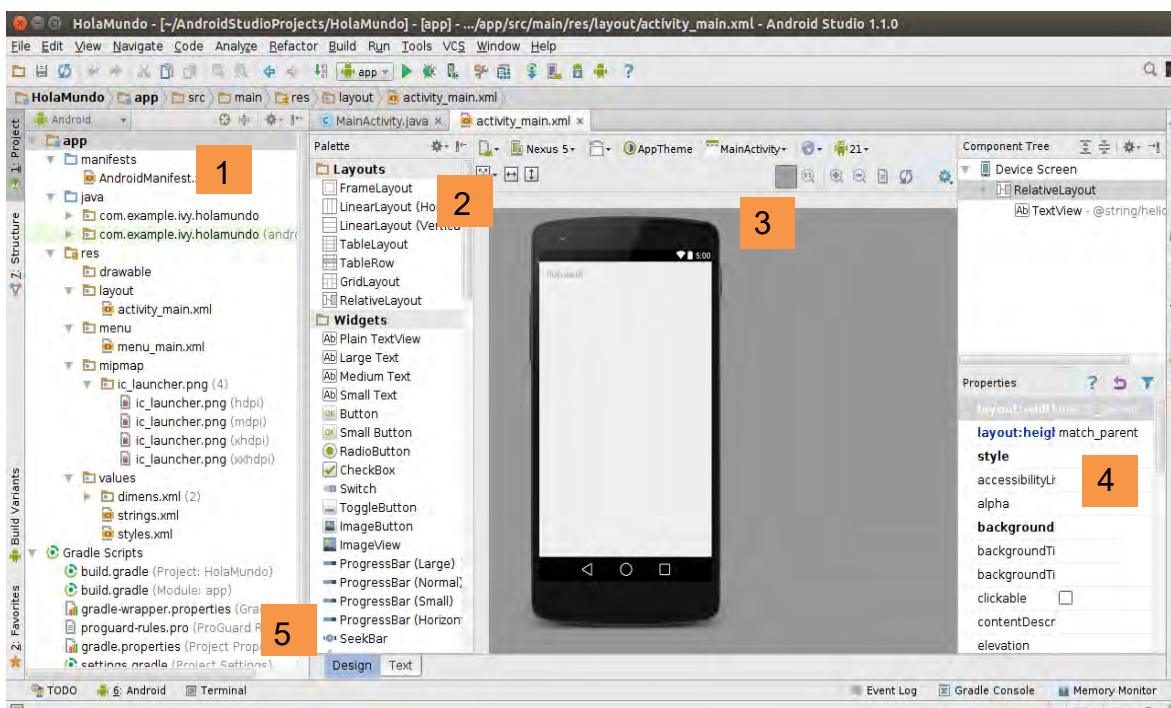


Figura 2.40 Herramientas en Android Studio

Fuente: Edición propia

- 1) Ventana de herramientas proyecto. Es el directorio del proyecto muestra los archivos y carpetas que se encuentran en la aplicación a través de una vista de árbol o jerarquía. Este directorio está formado básicamente por un descriptor de la aplicación (`AndroidManifest.xml`), el código fuente y una serie de ficheros con recursos. Cada elemento se almacena en una carpeta específica.

- Java. Carpeta que contiene el código fuente de la aplicación.
- Res. Carpeta que contiene los recursos usados por la aplicación.
- Drawable. En esta carpeta se almacenan los archivos de imágenes y descriptores de imágenes.
- Layout. Contiene archivos XML con vistas de la aplicación. Las vistas permiten configurar las diferentes pantallas que compondrán la interfaz del usuario de la aplicación.
- Menu. Archivos XML con los menús de la aplicación.
- Values. Archivos XML para indicar valores del tipo string, color o estilo.
- Xml. Otros archivos XML requeridos por la aplicación.
- Raw. Archivos adicionales que no se encuentran en formato XML.
- Gradle. Se encuentra la información del proyecto como numero de versión mínima de Api que ejecutará la aplicación.

- 2) Paleta de componentes. Se encuentran las vistas/views, los elementos que componen la interfaz del usuario de una aplicación. (Button, ImageView, CheckBox, TextView, etc.). Todas las vistas van a ser objetos descendientes de la clase View, y por tanto, se pueden definir utilizando código Java. Sin embargo, lo habitual va a definir las vistas utilizando un fichero XML y dejar que el sistema cree los objetos a partir de este fichero. Esta forma de trabajar es muy parecida a la estructura de una página Web utilizando código HTML.
- 3) Editor de diseño/ editor de código. Dependiendo del archivo que se haya seleccionado esta vista cambia, si se selecciona un layout la vista es de diseño, muestra una vista previa de la aplicación en donde se pueden agregar los elementos al dispositivo que aparece de ejemplo. Si se abre un archivo java el editor de código mostrará el código que hay en este.
- 4) Ventana de propiedades. Muestra las opciones o propiedades del elemento añadido a la vista que ha sido seleccionado.

5) Ventana de herramientas Android Monitor. Android Studio, proporciona monitores de rendimiento para realizar de manera más sencilla un seguimiento del uso de CPU y memoria de la app, busca objetos sin asignar, localizar pérdidas de memoria, optimizar el rendimiento de los gráficos y analizar solicitudes de la red. Con la App ejecutándose en un dispositivo o emulador, abrir la ventana de herramientas Android Monitor, y hacer clic en la pestaña Monitors. Cuando se compila y ejecuta la app con Android Studio, se pueden ver mensajes adb de salida y mensajes de registro del dispositivo (logcat) haciendo clic en Android Monitor en la parte inferior de la ventana. (López E. , 2014).

Existen una serie de componentes Android que se utilizan para desarrollar cualquier aplicación, estos componentes son los siguientes:

### **Android Manifest**

Todo proyecto Android tiene un archivo Android Manifest, el cual es un archivo XML que contiene información acerca de las características de la aplicación que se está desarrollando en Android. Describe los componentes: Activities, Services, Broadcast Receivers y Content Providers con los que está compuesta. También se declaran los permisos que requerirá la aplicación.

Estas declaraciones permiten al sistema de Android conocer que componentes son y bajo qué condiciones pueden ser utilizadas.

### **Actividad (Activity).**

Una aplicación en Android, va a estar formada por un conjunto de elementos básicos de visualización, coloquialmente conocidos como pantallas de la aplicación. En Android cada uno de estos elementos, o pantallas, se conoce como actividad. Su



función principal es la creación de la interfaz de usuario. Toda actividad ha de pertenecer a una clase descendiente de Activity.

### **Servicio (Service).**

Un servicio es un proceso que se ejecuta “detrás”, sin la necesidad de una interacción con el usuario. Es algo parecido a un demonio en Unix o a un servicio en Windows. Android dispone de dos tipos de servicios: Servicios locales, que son ejecutados en el mismo proceso y servicios remotos, que son ejecutados en procesos separados.

### **Receptor de anuncios (Broadcast Receiver).**

Recibe y reacciona ante determinados tipos de anuncios, pueden ser originados por el sistema o aplicaciones. Por ejemplo un broadcast puede ser generado por llamada entrante o conexión a un WiFi.

### **Intención (Intents).**

Una intención representa la voluntad de realizar alguna acción. Se utiliza cada vez que se desea: Lanzar una actividad, lanzar un servicio, enviar un anuncio de tipo broadcast o comunicar con un servicio. Los componentes lanzados pueden ser internos o externos a la aplicación. También se utilizan las intenciones para el intercambio de información entre estos componentes.

### **Fragmentos (Fragment).**

La llegada de las tabletas trajo el problema de que las aplicaciones de Android ahora deben soportar pantallas más grandes. Para ayudar al diseñador a resolver este problema, en la versión 3.0 de Android aparecen los fragments. Un fragment está

formado por la unión de varias vistas para crear un bloque funcional de la interfaz de usuario. Una vez creados los fragments, se pueden combinar uno o varios fragments dentro de una actividad, según el tamaño de pantalla disponible.

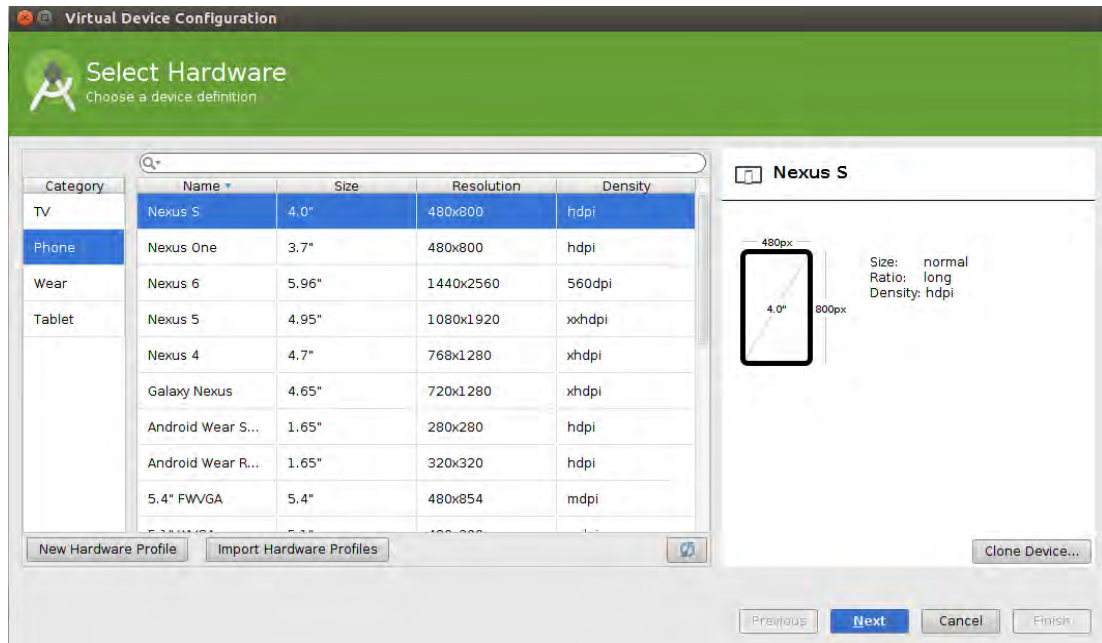
### **Proveedores de contenido (Content Provider).**

En muchas ocasiones las aplicaciones instaladas en un terminal Android necesitan compartir información. Android define un mecanismo estándar para que las aplicaciones puedan compartir datos sin necesidad de comprometer la seguridad del sistema de ficheros. Con este mecanismo se puede acceder a datos de otras aplicaciones. (Gómez, 2014)

### **2.8.1 Creación de un emulador (Android Virtual Device)**

Una herramienta muy útil es Android Virtual Device Manager (Administrador de dispositivos virtuales), la cual permite crear emuladores para probar las aplicaciones en diferentes dispositivos virtuales en caso de no contar con un Smartphone físico. Para crear un emulador dar clic en el icono AVD Manager y seleccionar el botón Create Virtual Device (Crear dispositivo virtual).

En la siguiente pantalla (Figura 2.41), se muestra una lista de dispositivos con sus características (Tamaño, Resolución y Densidad).

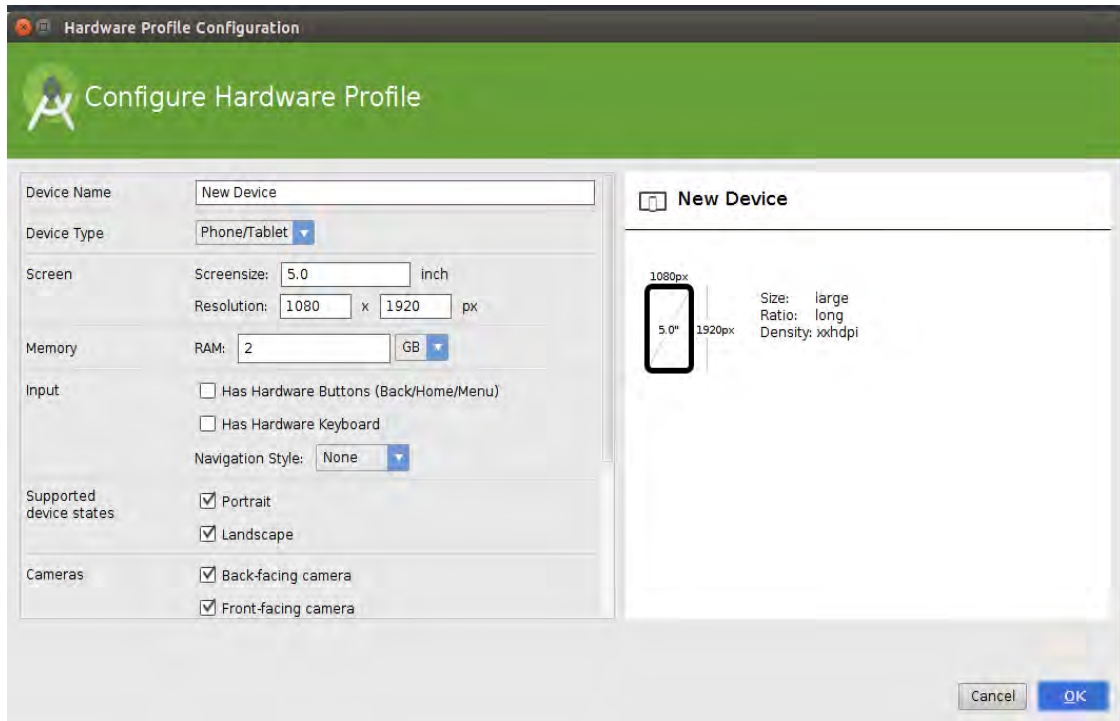


**Figura 2.41** Ventana de selección de dispositivos.

**Fuente:** Edición propia

Seleccionar uno de ellos para este ejemplo Nexus 4 y en New Hardware Profile (Nuevo perfil de hardware), modificar las características dependiendo de qué tipo de dispositivos se desea emular.

En device name (nombre del dispositivo). Nexus 4. Device type (tipo de dispositivo). Phone/Tablet. (Figura 2.42)



**Figura 2.42 Configuración perfil del hardware**

**Fuente: Edición propia**

En la siguiente ventana (Figura 2.43), elegir la API con la que se desea trabajar para este ejemplo se seleccionará la api 4.03 Ice Cream Sandwich y dependiendo del procesador de la computadora en que se está trabajando el proyecto, elegir la arquitectura a utilizar, x86 para Intel y armeabi-v7a para AMD.

Dar clic en siguiente.

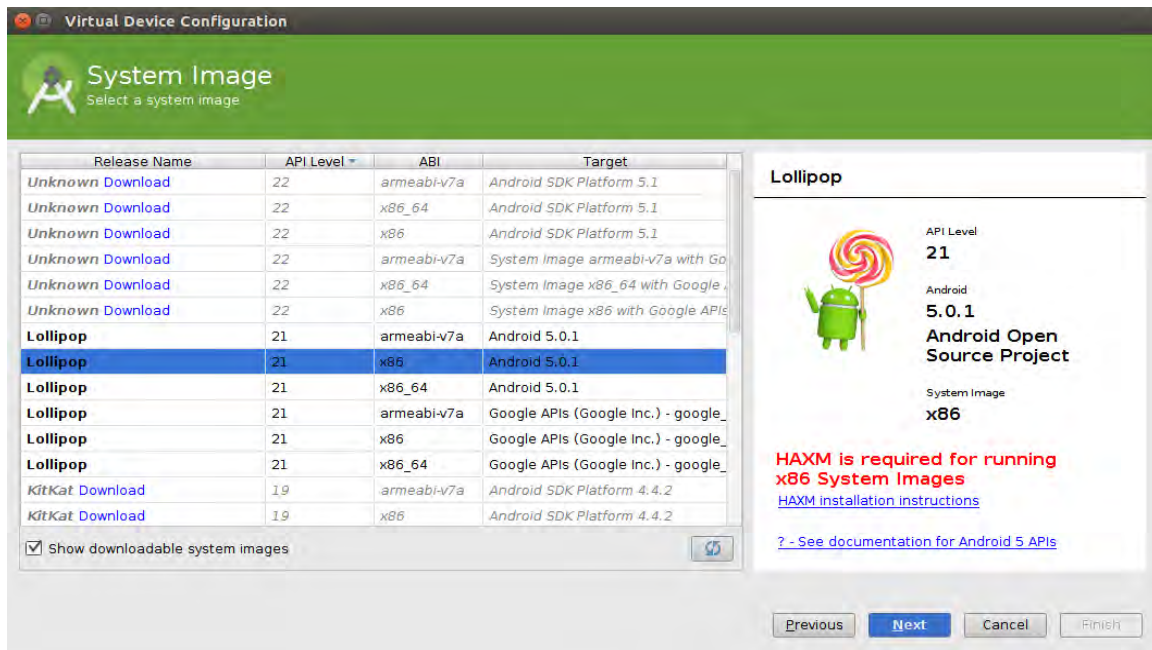


Figura 2.43 Ventana imagen del sistema.

Fuente: Edición propia

En la siguiente ventana verificar si todos los datos están correctos y de ser así, dar clic en el botón Finish (Finalizar) para terminar con la creación del emulador. (Figura 2.44)

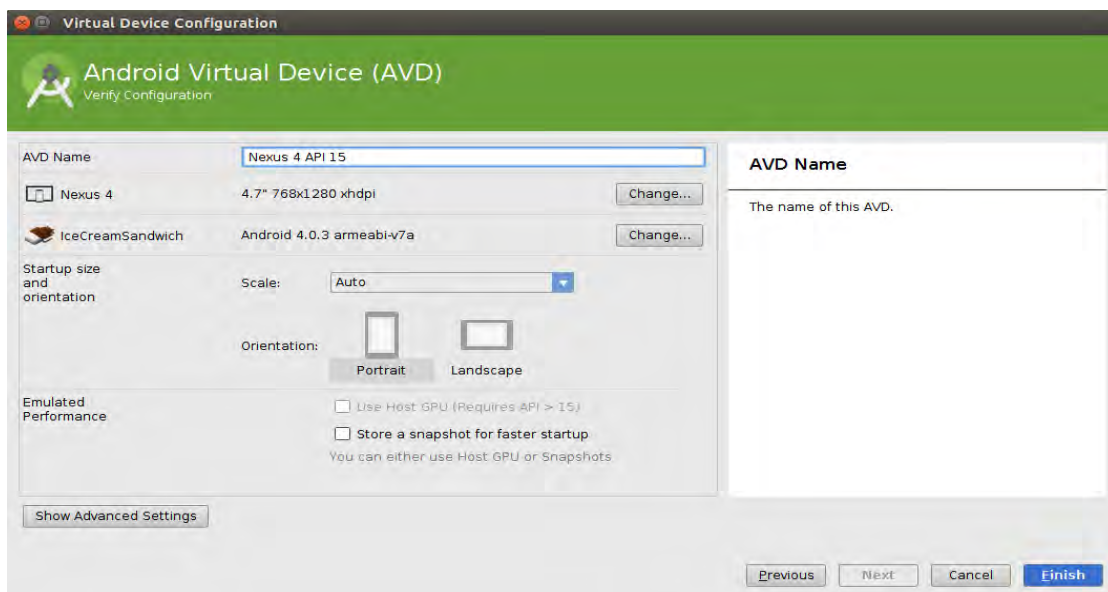
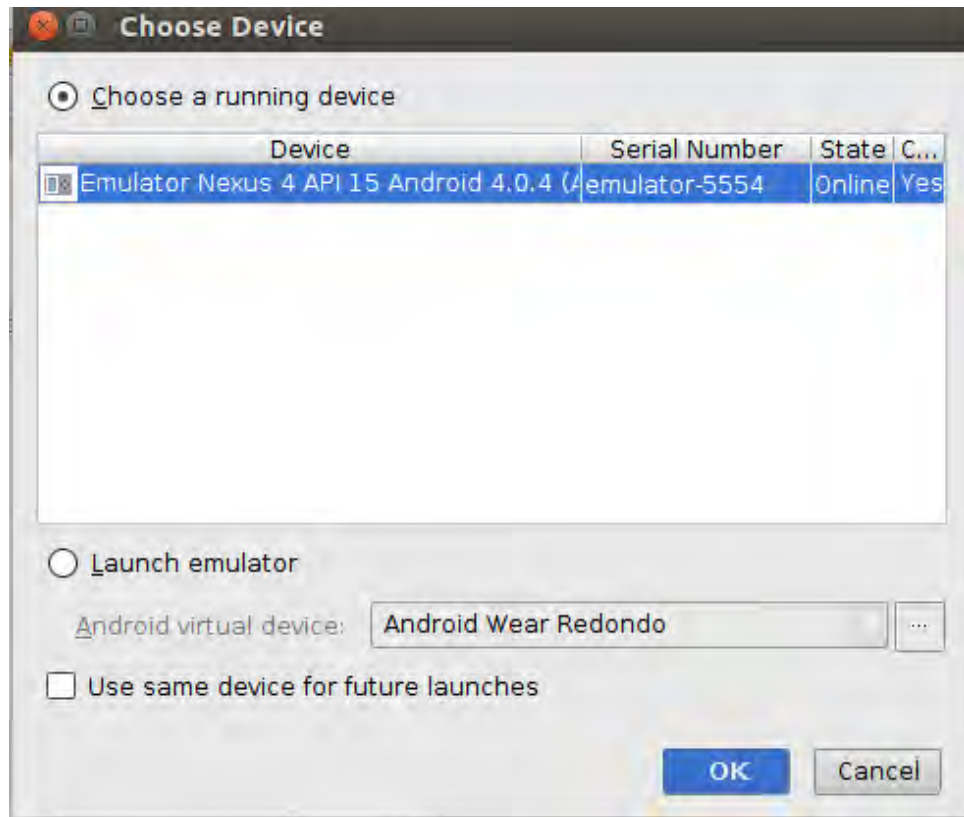


Figura 2.44 Ventana de verificación de la información.

Fuente: Edición propia

Una vez creado el emulador, se mostrará en la lista el dispositivo que se ha creado, seleccionar el icono start (iniciar) ► para iniciar el emulador.

En la pantalla donde se encuentra la aplicación, en la barra de herramientas dar clic en start ► para lanzar la aplicación. Seleccionar el dispositivo que se está ejecutando. (Figura 2.45).



**Figura 2.45** Ventana seleccionar emulador

**Fuente:** Edición propia

Los requisitos para instalar Android Studio pide contar con al menos 1 GB de memoria RAM, pero es cierto que en algunas computadoras con muy poca memoria RAM el emulador tardará demasiado tiempo en cargar, es por eso que también existen emuladores alternativos que se pueden encontrar en la Web como Genymotion.

Asimismo es altamente recomendable contar con uno o varios dispositivos físicos, para probar las aplicaciones que se van a desarrollar.

## 2.9 Bases de Datos (Almacenamiento de Datos)

En la actualidad el uso de datos es una parte esencial del día a día, grandes empresas como Google, Amazon y Facebook, por mencionar algunas, almacenan una gran cantidad de información referente a las búsquedas que ha hecho el usuario o sus gustos por medio de likes en publicaciones de Facebook (Wei, 2012), esto con el fin de mostrar un entorno más personalizado a cada usuario. En México existen cadenas de supermercados, cines y heladerías que en un principio manejaban sus datos por medio de una tarjeta, pero hoy en día aunado a eso tienen una aplicación móvil en donde los datos de esa tarjeta se ven reflejados en tiempo real en el Smartphone, entonces, con todo esto se puede decir que, se vive en un mundo cada vez más centrado en los datos, por lo que es de vital importancia desarrollar aplicaciones con una perspectiva centrada en ellos.

Almacenar y recuperar datos es esencial para la mayoría de las aplicaciones, en Android, existen cuatro maneras de almacenamiento de datos, dependiendo del contexto que se desee llevar a cabo y características del proyecto, dependerá usar una u otra función. (Báez, y otros, 2012).

Los datos de cada aplicación son privados a dicha aplicación, pero se pueden compartir si se desea. A continuación se mencionan las cuatro maneras de almacenamiento:

- Preferencias: Mecanismo liviano que permite almacenar y recuperar datos primitivos en la forma de pares clave/valor.
- Archivos locales: Se puede almacenar los archivos en la memoria interna y externo como una tarjeta SD. También se pueden utilizar ficheros añadidos a la aplicación como recursos.
- Base de datos: Las APIs de Android proveen soporte para SQLite. Las aplicaciones pueden crear y usar bases de datos SQLite de forma muy sencilla y con toda la seguridad que brinda el lenguaje SQL.

- Proveedores de contenidos: Es un mecanismo para compartir datos entre aplicaciones, no se trata con ellos directamente. Los proveedores de contenidos implementan una sintaxis estándar para solicitar datos (URIs) y un mecanismo de acceso para devolver los datos, de manera muy parecida a SQL. (Gargenta, 2011).

## 2.9.1 SQLite

Apareció en mayo del año 2000 de la mano de su creador D. Richard Hip, quién ha liberado las diferentes versiones de SQLite en base a la licencia GPL por lo que su código es de dominio público y puede ser modificado por cualquier persona. Gracias a esto, SQLite ha sido mejorada a lo largo de 7 años y también ha sido migrada a diversas plataformas. (Rómmel, 2007).



Figura 2.46 Logo de SQLite

Fuente: <https://goo.gl/Dqfnt3>

SQLite (Figura 2.46) es una herramienta de software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser un teléfono celular. Implementa el estándar SQL92 y también agrega extensiones que facilitan su uso en cualquier ambiente de desarrollo.



Permite que soporte desde las consultas más básicas hasta las más complejas del lenguaje SQL, se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, existe compatibilidad al 100% entre las diversas plataformas disponibles, haciendo la portabilidad entre dispositivos y plataformas transparente. (Owens, 2014)

## **Arquitectura**

El Motor SQLite tiene una arquitectura modular para la administración de bases de datos relacionales. A continuación se muestra la arquitectura interna de SQLite.

La arquitectura interna del motor SQLite (Figura 2.47), consiste de ocho módulos separados (Interface, Tokenizer, Parser, Code Generator, Virtual Machine, B-Tree, Pager y OS Interface).

Los módulos se encuentran agrupados en tres subsistemas mayores (Compiler, Core y Backend) que dividen el procesamiento de una consulta en tareas discretas que trabajan como una línea de ensamble, donde la parte superior compila la consulta, la parte intermedia la ejecuta y la parte más baja es la que se encarga del almacenamiento y la interacción con el Sistema Operativo.

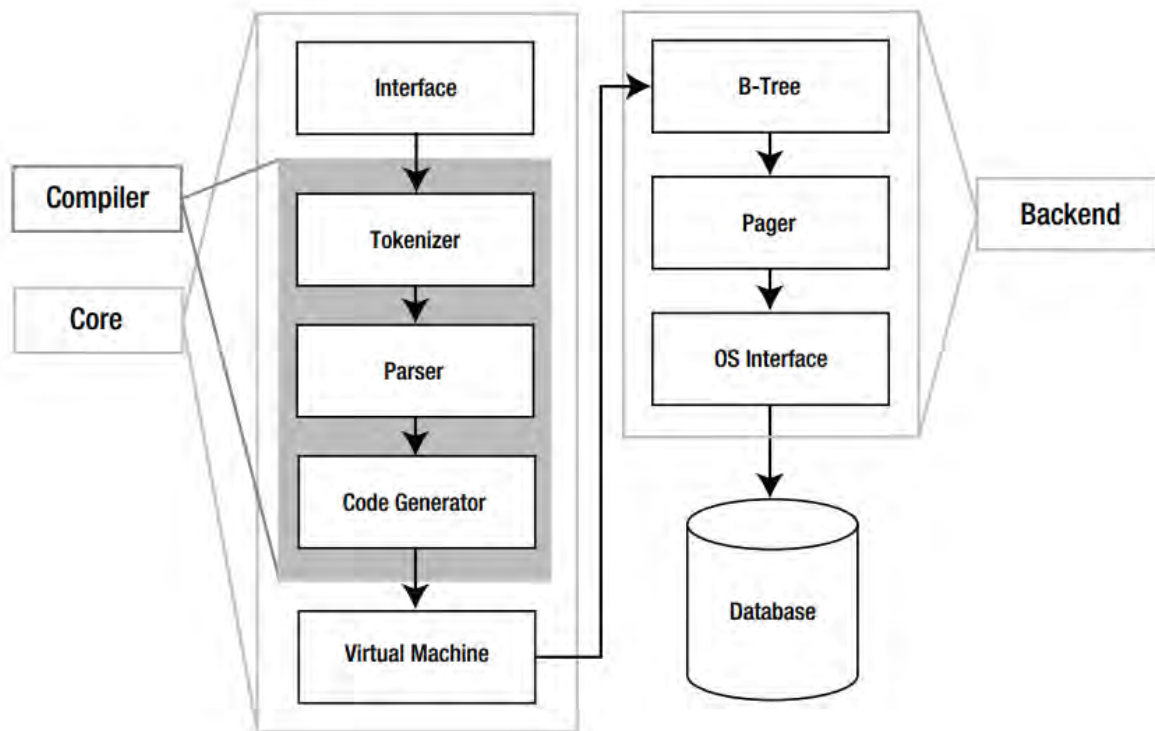


Figura 2.47 Arquitectura SQLite

Fuente: Extraída de The definitive guide to SQLite

## Características

A continuación se mencionan las características principales de SQLite:

- La base de datos completa se encuentra en un solo archivo.
  - Puede funcionar enteramente en memoria, lo que la hace muy rápida.
  - Tiene un footprint menor a 230KB.
  - Es totalmente auto contenida (sin dependencias externas).
  - Cuenta con librerías de acceso para muchos lenguajes de programación.
  - Soporta texto en formato UTF-8 y UTF-16, así como datos numéricos de 64 bits.
  - Soporta funciones SQL definidas por el usuario (UDF).
  - El código fuente es de dominio público y se encuentra muy bien documentado.
- (Rómmel, 2007)



# CAPÍTULO 3

DESARROLLO DE  
APLICACIÓN:  
BIENES Y RAÍCES



## **CAPÍTULO 3. DESARROLLO DE APLICACIÓN: BIENES Y RAÍCES**

Con las herramientas antes mencionadas se pueden desarrollar sistemas completos y eficientes que minimicen el tiempo en empresas del tipo bienes raíces.

### **3.1 Aplicación Bienes y Raíces**

La aplicación se conforma con una base de datos en MySQL, una aplicación móvil con una base de datos local SQLite y un Webservice, que permite la interacción entre ambas bases de datos.

### **3.2 Planificación de requisitos**

La función principal de la aplicación bienes raíces, es permitirle al agente de ventas recolectar una serie de datos que conforman las características de las propiedades que están en venta o renta.

Utiliza un sistema de logueo el cual permite a cada agente de bienes raíces contar con una sesión propia, ver un listado de las propiedades que ha agregado y tener un control sobre ellas tales como crear, consultar, actualizar y eliminar, agregando a los formularios una interfaz fácil e intuitiva que le permite realizar dichas tareas muy fácilmente, además se complementa con el uso de mapas, localización (GPS) y cámara.

Todos los datos son guardados automáticamente en la base de datos local de la aplicación, esto le permite trabajar sin conexión a Internet y una vez que ya cuente con una conexión a una red WiFi, se pueden sincronizar con la base de datos MySQL que se encuentra en el Webservice, permitiendo así contar con dicha información en tiempo real, una vez que el agente suba los datos al Webservice.

El Webservice, se ha desarrollado con las siguientes herramientas:

Con un servidor Apache, y con los lenguajes de programación, PHP, JavaScript, PDO, para la conexión segura de la base de datos en MySQL, CSS3 para la parte visual. En el Webservice se crearon los mismos métodos que en la base de datos local, Crear, modificar, eliminar, visualizar y sincronizar propiedades.

### **3.2.1 Modelado de Gestión**

En el desarrollo del proyecto se cuenta con información de las propiedades en venta y renta (título, descripción, dirección, colonia, coordenadas, foto, etc.) que ha sido generada por el agente de ventas de bienes raíces, una vez agregada una nueva propiedad, la información es almacenada en la base de datos local de la aplicación Android; con una conexión WiFi la información almacenada en la base de datos local, es enviada a la base de datos MySQL del servidor Apache por medio de los métodos del Webservice.

### **3.2.2 Diagrama de Casos de Uso**

El diagrama de casos de uso representa la forma en como el Agente de ventas, interactúa con la aplicación en desarrollo.

En el diagrama de casos de uso (Figura 3.1), el agente inicia sesión con su usuario y contraseña, si estos son correctos la aplicación le permite visualizar las propiedades que han sido dadas de alta. Además de realizar las siguientes operaciones que son Agregar, modificar, eliminar y sincronizar las propiedades.

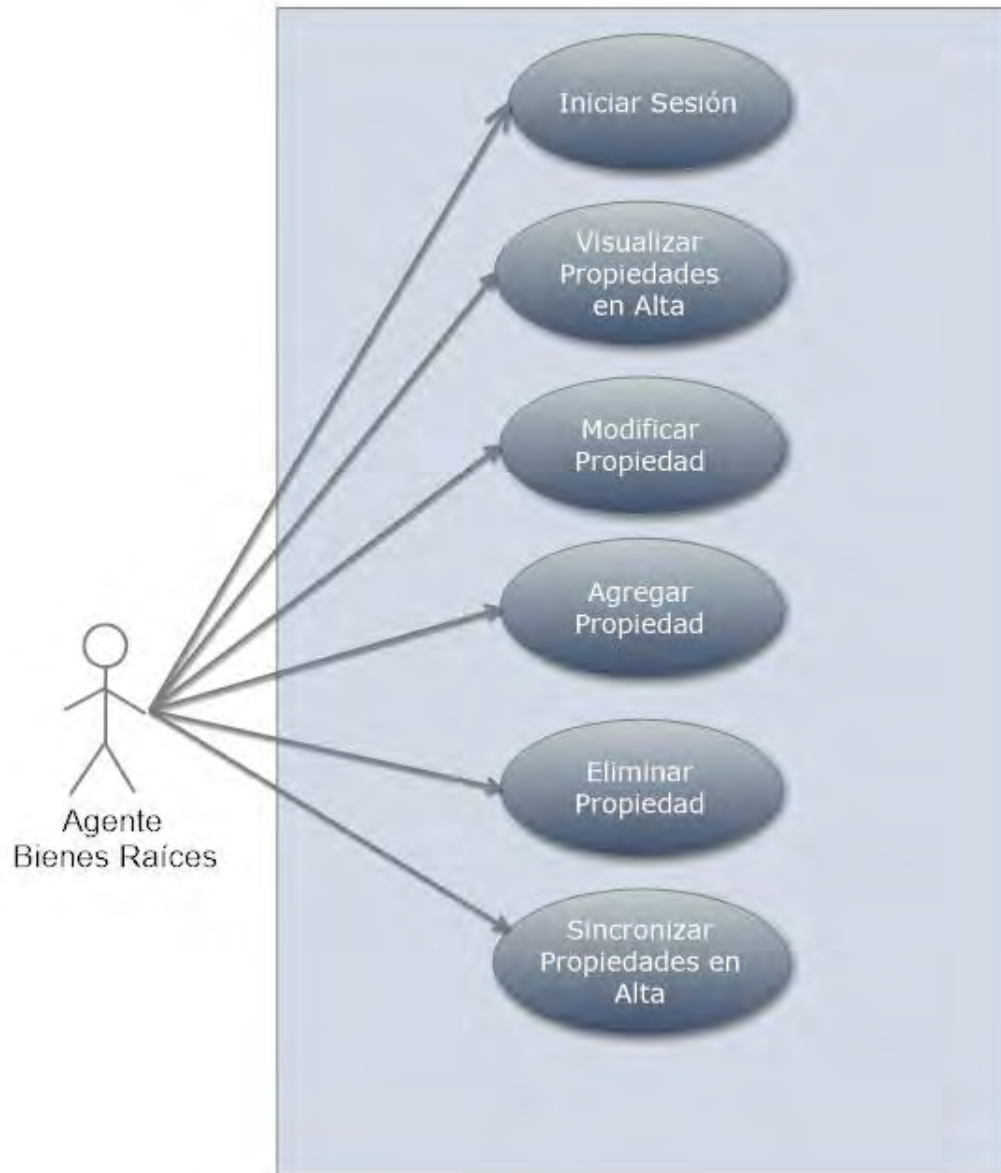


Figura 3.1 Diagrama de Casos de Uso

Fuente: Edición propia

### 3.2.3 Diagrama de Actividades

Un diagrama de Actividad demuestra la serie de actividades que deben ser realizadas en un caso de uso, así como las distintas rutas que pueden irse

desencadenando en el caso de uso. Teniendo como base la descripción y el diagrama de casos de uso, el diagrama de actividades se representa de la siguiente manera en la (Figura 3.2)

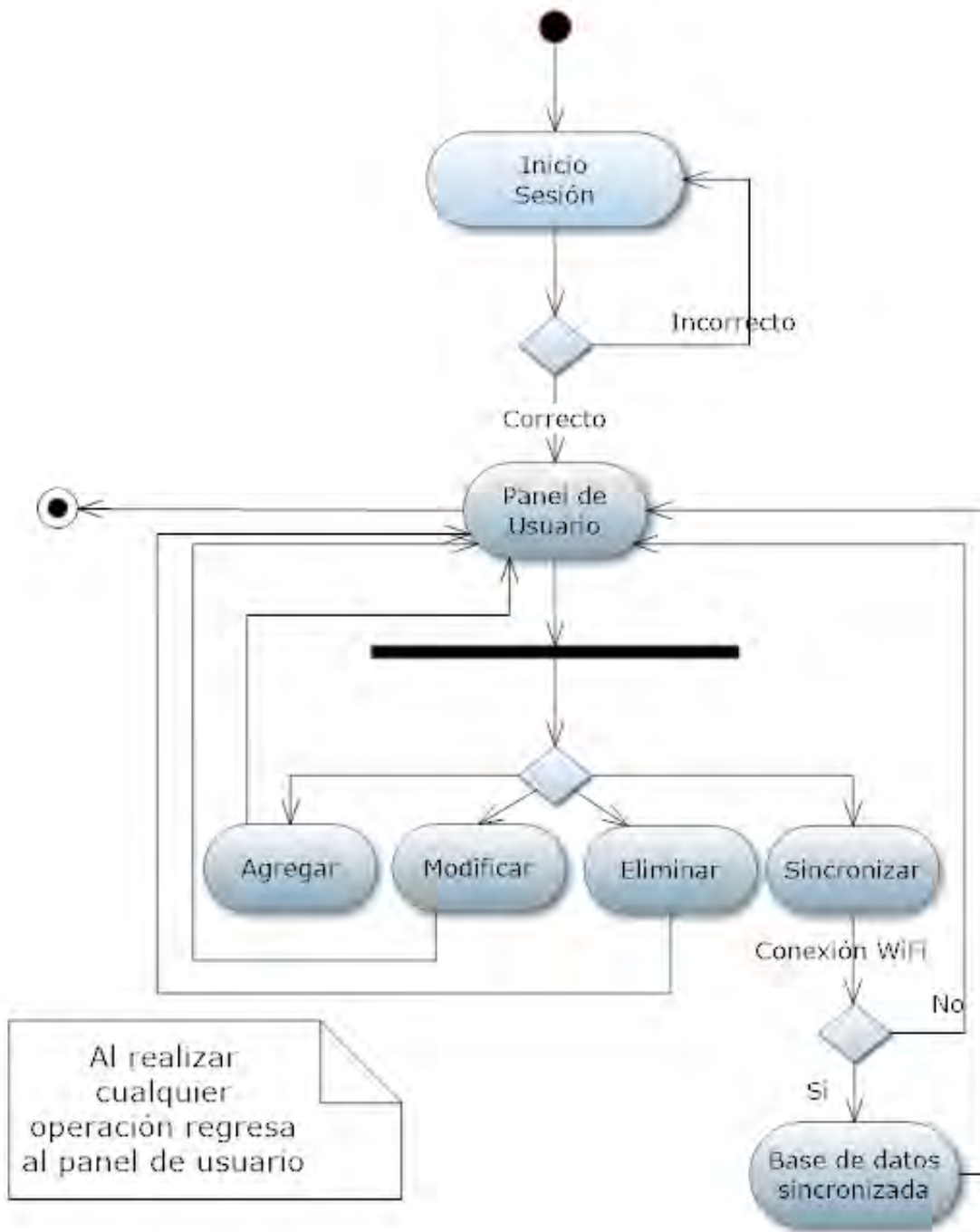


Figura 3.2 Diagrama de Actividades

Fuente: Edición propia

### 3.2.4 Diagrama de Estados

El diagrama de estados es utilizado para identificar cada una de las rutas o caminos que puede tomar un flujo de información luego de ejecutarse cada proceso (Figura 3.3)

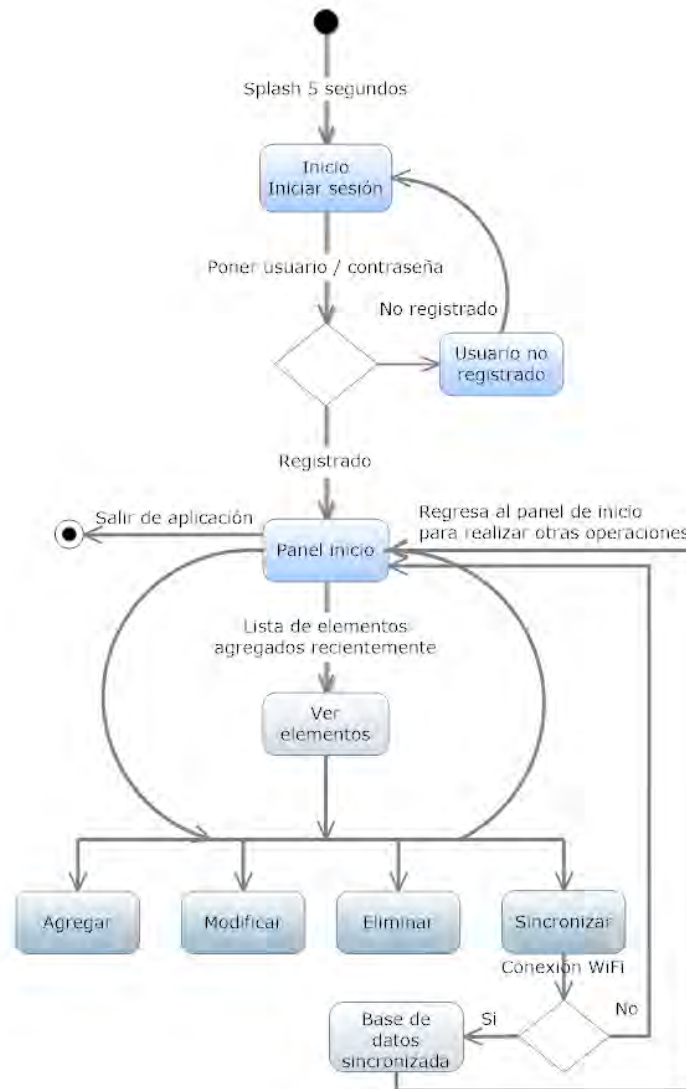


Figura 3.3 Diagrama de Estados

Fuente: Edición propia

Al abrir la aplicación muestra un splash (pantalla de bienvenida), que dura 5 segundos, si el usuario no puede iniciar sesión sigue estando en la pantalla de inicio de sesión. Una vez que ya ha accedido accede al panel de usuario en donde puede realizar las



operaciones ya mencionadas. Siempre que se termina un proceso la aplicación vuelve al panel de usuario para seguir realizando operaciones o salir de la aplicación.

### 3.2.5 Diagrama de Clase

Un diagrama de clases es una representación gráfica que sirve para representar la estructura de un sistema que será implementado, utilizando un lenguaje orientado a objetos. La idea de estos diagramas es representar las clases que tendrá el sistema así como su contenido y sus relaciones con otras.

En el diagrama de clase (Figura 3.4) se muestran las dos clases que se manejarán en el sistema: La clase usuario y la clase propiedad, la relación es de 1 a varios ya que un usuario puede tener más de una propiedad agregada.

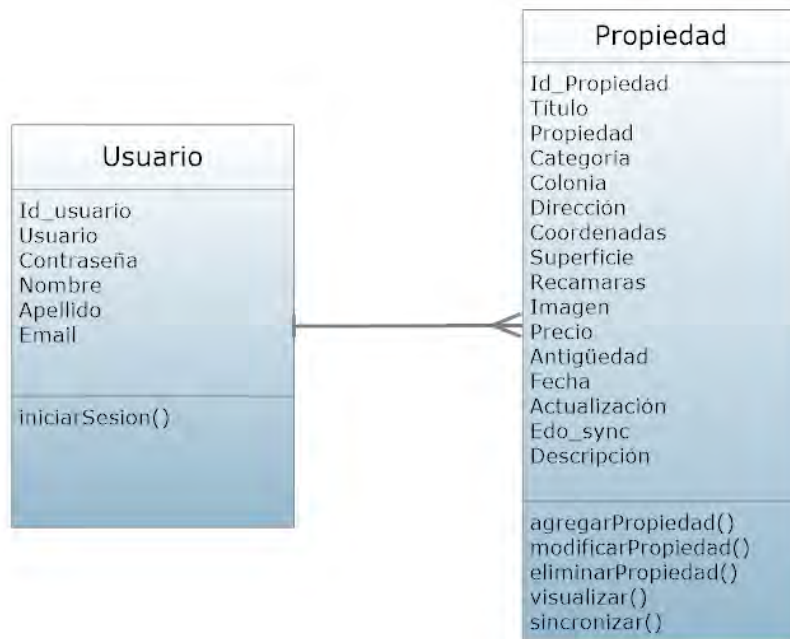


Figura 3.4 Diagrama de Clase

Fuente: Edición propia

En el diagrama, también se pueden ver los atributos y los métodos que realiza cada clase.

### 3.2.6 Diagrama de Componentes

Los Diagramas de Componentes ilustran las piezas del software, librerías, controladores embebidos que conformarán un sistema. Tiene un nivel más alto de abstracción que un diagrama de clase.

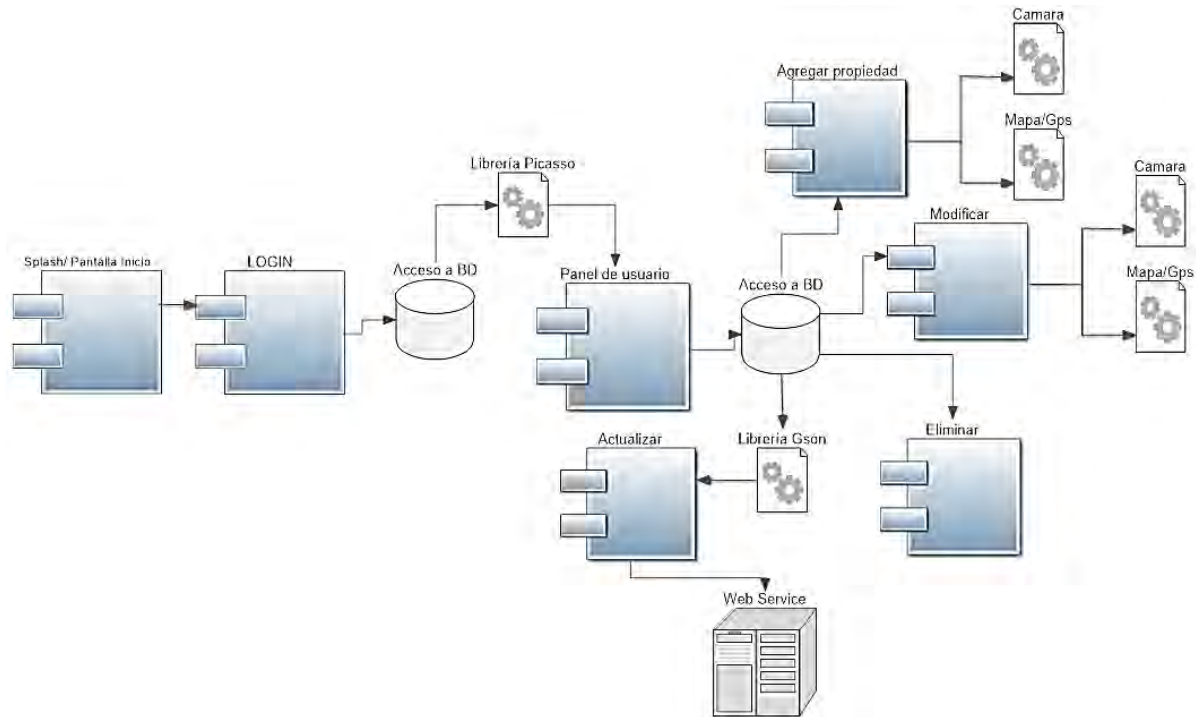


Figura 3.5 Diagrama de Componentes

Fuente: Edición propia

En este diagrama (Figura 3.5) se muestran los componentes que se usan en la aplicación. Se utiliza la API de los mapas y la cámara, también se accede a la base de datos para obtener los datos de las operaciones que se realizan, se hace uso de librerías como Picasso, para la presentación de imágenes, y la librería de Google GSON, para obtener los datos de la base de datos, convertirlo en cadena JSON y así enviar la información al Webservice.

### 3.3 Diseño de Usuario

Para la aplicación de bienes raíces se decidió contar con un diseño limpio, sencillo e intuitivo.

#### 3.3.1 Interfaz de Usuario

La interfaz de usuario es el medio visual por el cual, el usuario puede interactuar con el dispositivo móvil, para compartir información, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

Al iniciar la aplicación aparece una pantalla de bienvenida la cual muestra el logo de la aplicación y 5 segundos después muestra la actividad de inicio de sesión, como se muestra en la Figura 3.6.



Figura 3.6 Pantalla de Bienvenida e Inicio de Sesión

Fuente: Edición propia

En ésta pantalla sólo se muestran elementos necesarios 2 EditText para escribir el usuario y la contraseña y un botón “Iniciar Sesión”. Después de que el usuario ha iniciado sesión correctamente con su usuario y contraseña, la aplicación muestra la siguiente actividad: Panel de usuario, en donde muestra las propiedades que ya se han dado de alta en la aplicación, permitiendo eliminar propiedades, al mantener seleccionada dicha propiedad, aparece un mensaje para confirmar si se desea eliminar la propiedad (Figura 3.7), en la barra de menú de esta misma actividad se puede visualizar un botón de sincronización, al ser presionado y si se cuenta con conexión a Internet WiFi, las propiedades que no se han actualizado se agregarán a la base de datos del Webservice.

Para modificar una propiedad se presiona la propiedad que se desea modificar y después aparece una pantalla con los datos ya agregados, permitiendo modificar la propiedad.



**Figura 3.7** Propiedades en alta

**Fuente:** Edición propia

En la parte inferior derecha se observa un botón flotante el cual permite agregar una nueva propiedad. La actividad “Alta propiedad” despliega un formulario para dar de alta los datos, imagen y coordenadas, de una nueva propiedad. Para obtener las coordenadas que se muestran en el mapa se mueve el marcador de posición para establecer la ubicación deseada y se da clic en el botón “obtener coordenadas”. (Figura 3.8Figura 3.7).

Al realizar este procedimiento la actividad del mapa se cerrará y mostrará las coordenadas obtenidas en el EditText de la actividad “Alta propiedad”. Una vez capturados los datos se agregan al dar clic en el botón que aparece en la esquina inferior derecha.



**Figura 3.8 Actividad Alta nueva propiedad**

**Fuente: Edición propia**

De igual forma, para modificar una propiedad que ya se ha dado de alta, mostrará un formulario con los datos que ya se encuentran en la base de datos y un botón para completar la acción como se muestra en la Figura 3.9.

Los iconos que se utilizan en los botones y en la barra de menú son los iconos de Material Design que se encuentran en la página de Material Design Library<sup>18</sup>, iconos que simbolizan acciones conocidas, como enviar, editar o sincronizar.

Una de las razones por la que Google desarrolló estas normas de diseño es para unificar la interfaz en las aplicaciones ya sean móviles o Web.



**Figura 3.9 Actividad Actualizar Propiedad**

**Fuente: Edición propia**

---

<sup>18</sup> Página Web de Material Design Library <https://material.io/icons/>

### 3.3.2 Modelado de datos

Un modelo de datos es un lenguaje orientado a hablar de una Base de Datos. Típicamente un modelo de datos permite describir:

- Las estructuras de datos de la base: El tipo de los datos que hay en la base y la forma en que se relacionan.
- Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar la realidad deseada.
- Operaciones de manipulación de los datos: Típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base.
- Otro enfoque es pensar que un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí.

A continuación se presenta el proceso que se llevó a cabo para el modelado de datos del proyecto de bienes raíces, en la Tabla 3.1

**Tabla 3.1 Tabla usuarios**  
Fuente: Edición propia

No.	Nombre	Apellido	Usuario	Contraseña	Email	Título	Colonia	Coordenadas	Imagen	Tipo	Precio
1	Ana	Blancas	anab	764858	ab@... .com	Casa en venta	Progreso	-19.565, 16988	IMG_78 5.jpg	Venta	1,500,000
1	Ana	Blancas	anab	764858	ab@... .com	Casa en renta	La mira	-19.415, 17,697	MG_78 6.jpg	Renta	2,500
1	Ana	Blancas	anab	764858	ab@... .com	Depto. Renta	Alfareros	-19.259, 17,680	MG_78 7.jpg	Venta	700,000
1	Ana	Blancas	anab	764858	ab@... .com	Depto. venta	Anáhuac	-19.355, 17,690	MG_78 8.jpg	Renta	1,500
2	José	Díaz	joser	12365	joser @....c om	Casa en venta	Azteca	-19.665, 17,986	MG_78 9.jpg	Venta	500,000
2	José	Díaz	joser	12365	joser @....c	Depto. renta	Aztlán	-19.598, 17,987	MG_79 0.jpg	Renta	4,000
3	Daniel	Flores	danif	75159	danif @....c om	Casa en renta	Morelos	-19.125, 17,753	MG_79 1.jpg	Renta	5,000



### 3.3.3 Normalización

En la Tabla 3.1 se observa la tabla de usuarios sin normalizar, en un inicio, en ésta tabla se incluían los datos de los usuarios y las características de las propiedades, esto generaba muchos problemas a la hora de almacenar datos, porque se incluía almacenamiento innecesario, al agregar los mismos campos en repetidas ocasiones.

En la tabla usuario se muestran las propiedades que han sido agregadas por 3 agentes de venta diferentes, el usuario Ana blancas llena en 4 ocasiones los campos nombre, apellido, usuario, contraseña e Email, y de igual manera los otros usuarios.

#### La primera forma normal

Satisface las siguientes cinco condiciones:

1. No hay orden de arriba a abajo en las filas.
2. No hay orden de izquierda a derecha en las columnas.
3. No hay filas duplicadas.
4. Cada intersección de fila-y-columna contiene exactamente un valor del dominio aplicable (y nada más).
5. Todas las columnas son regulares, es decir, las filas no tienen componentes como IDs de fila, IDs de objeto, o timestamps ocultos.

Aplicando las condiciones anteriores, la tabla usuarios se divide en tabla usuario y tabla propiedades. En la tabla usuarios contiene los datos del agente de ventas. En la tabla propiedades se encuentra las características de la propiedad (Tabla 3.2)

Tabla 3.2 Tablas Primera forma normal

Fuente: Edición propia

NO.
TITULO
TIPO
CATEGORIA
COLONIA
DIRECCION
COORDENADAS
SUPERFICIE
RECAMARAS
IMAGEN
PRECIO
ANTIGUEDAD
FECHA
EDO_SYNC
DESCRIPCION

NO.
NOMBRE
APELLIDO
USUARIO
CONTRASEÑA
EMAIL

## Segunda forma normal

La Segunda Forma Normal debe cumplir satisfactoriamente ciertos requisitos aparte de los ya establecidos en la Primera Forma Normal. Una tabla en 1FN está en 2FN si y solo si todos sus atributos no-principales se encuentran en dependencia funcional de una parte llamada clave primaria (Atributo o conjunto de atributos que permite identificar a una entidad de las otras).

En este paso se deben verificar las dependencias de las distintas tablas, donde no existan dependencias parciales, es decir cada atributo debe tener una dependencia completa a la clave primaria.

Una relación se encuentra en segunda forma normal, cuando cumple con las reglas de la primera forma normal y todos sus atributos que no son claves primarias presenten dependencia funcional completa con el atributo escogido como la clave primaria.

Se ha identificado que la tabla propiedades depende de la tabla usuarios, puesto que un agente de ventas va a ser el encargado de crear las propiedades. La llave primaria será el Id del usuario y se agrega una llave secundaria prop\_id en la tabla propiedades. Tabla 3.3

**Tabla 3.3 Tablas con segunda forma normal**

**Fuente: Edición propia**

ID USUARIO	PK
NOMBRE	
APELLIDO	
EMAIL	
USUARIO	
CONTRASEÑA	

<b>_ID</b>	
TITULO	
TIPO	
CATEGORIA	
COLONIA	
DIRECCION	
COORDENADAS	
SUPERFICIE	
RECAMARAS	
IMAGEN	
PRECIO	
ANTIGUEDAD	
FECHA	
ACTUALIZACION	
EDO_SYNC	
DESCRIPCION	
PROP_ID	FK

Las tablas ya se encuentran normalizadas, eliminando duplicado de datos y ambigüedades que puedan contener.

### 3.3.4 Modelo Entidad relación

En la

Figura 3.10 se muestra el modelo entidad relación de las dos tablas de la base de datos.

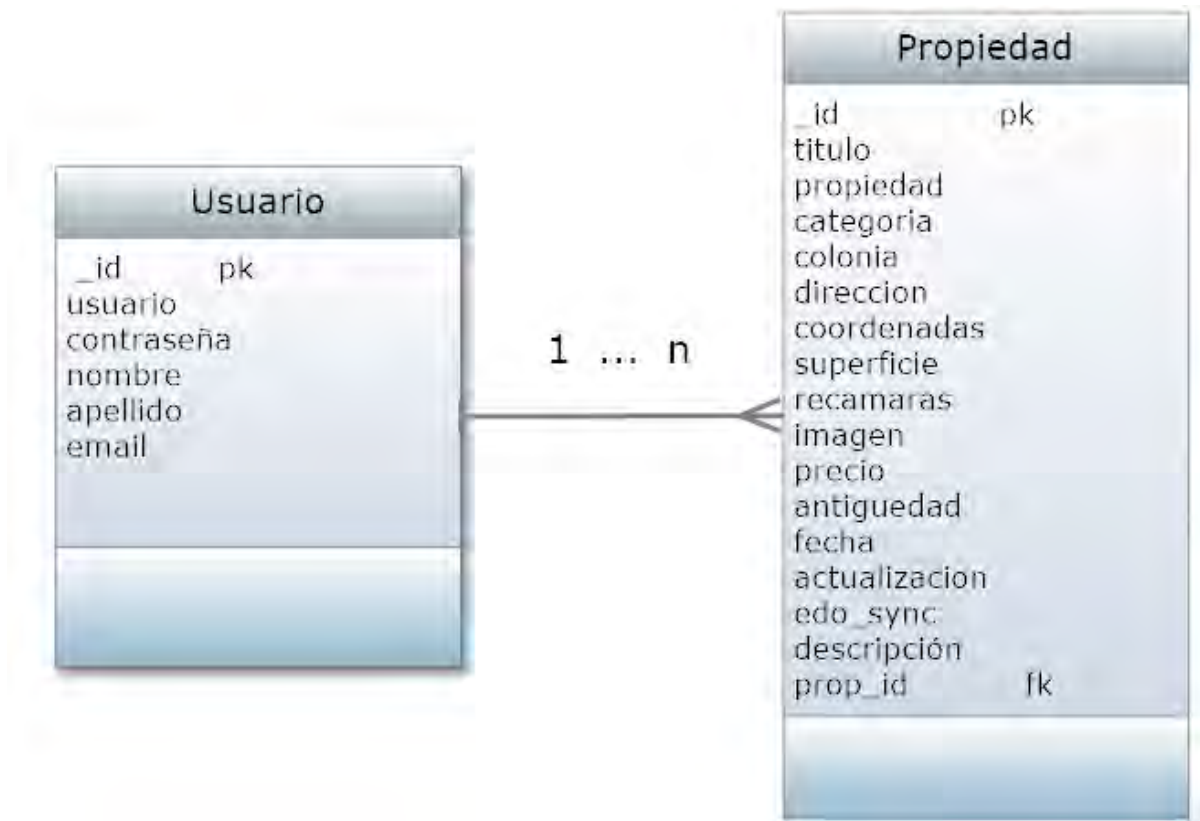


Figura 3.10 Modelo entidad relación

Fuente: Edición propia

En la Tabla 3.4 se muestra el diccionario de datos de la aplicación aquí se encuentran todos los datos que se utilizan en la base de datos, sus características y descripción, esto con el fin de comprender mejor el funcionamiento del proyecto.

**Tabla 3.4 Diccionario de datos**

Fuente: Edición propia

Nombre	Campo	Tipo	Tamaño	Descripción
<b>Usuarios</b>				
<b>ID usuario</b>	_id	Auto numérico	5	Identificador del agente
<b>Nombre</b>	nombre	Varchar	10	Almacena el nombre del agente
<b>Apellido</b>	apellido	Varchar	10	Almacena el apellido del agente
<b>Email</b>	email	Varchar	20	Almacena el email del agente
<b>Usuario</b>	usuario	Varchar	10	Almacena el usuario del agente
<b>Contraseña</b>	contraseña	Varchar	10	Almacena la contraseña del agente
<b>Propiedades</b>				
<b>ID propiedad</b>	_id	Auto numérico	5	Identificador de la propiedad
<b>Título</b>	titulo	Varchar	20	Título de la propiedad
<b>Tipo</b>	tipo	Varchar	12	Tipo de propiedad casa o departamento
<b>Categoría</b>	categoria	Varchar	5	Categoría de la propiedad venta o renta.
<b>Colonia</b>	colonia	Varchar	20	Colonia de la propiedad
<b>Dirección</b>	direccion	Varchar	25	Dirección de la propiedad
<b>Ubicación / coordenadas</b>	coordenadas	Varchar	50	Coordenadas de la propiedad (Latitud, Longitud)
<b>Superficie</b>	superficie	Varchar	10	Superficie de la propiedad.
<b>Recamaras</b>	recamaras	Numérico	4	Número de recámaras con las que cuenta la propiedad
<b>Imagen</b>	imagen	Varchar	50	Ruta de la imagen capturada con la cámara
<b>Precio</b>	precio	Varchar	10	Precio de la propiedad
<b>Antigüedad</b>	antigüedad	Varchar	3	Antigüedad de la propiedad

<b>Fecha de publicación</b>	fecha	Varchar	8	Fecha de alta de la propiedad
<b>Actualización</b>	actualizacion	Varchar	15	Fecha de la última actualización
<b>Estado de sincronización</b>	edo_sync	Varchar	2	Estado en que se encuentra la sincronización. (Si / No)
<b>Descripción</b>	descripcion	Varchar	20	Descripción de la propiedad
<b>ID usuario propiedad</b>	prop_id	Numérico	5	Id del usuario que agregó la propiedad

### 3.4 Desarrollo

Una vez que ya se han analizado los requisitos y el diseño de la aplicación se prosigue al desarrollo de ésta. Primero se creará el Webservice y los métodos a utilizar.

#### Instalación de apache

Descargar el paquete de apache Xaamp que es utilizado en distribuciones de Linux. Esto instalará Php, MySql y una serie de herramientas a utilizar.

#### 3.4.1 Webservice

Para desarrollar el esquema del Webservice se utiliza HTML, se creó una página de inicio de sesión, y una página para visualizar las propiedades del usuario ver Anexo A, una página para agregar propiedad y modificarlas.

Organización de carpetas

#### Webservice.php

En este documento se crea el servidor y se menciona la ubicación de este.

```
<?php
include('clases/service.class.php');
```

```

try
{
    $options = array('uri' => 'http://localhost/wsbienenes');
    $server = new SoapServer(NULL,$options);
    $server->setClass('ClaseCRUD');
    $server->handle();
}
catch(PDOException $e){
    echo "Error:". $e->getMessage();
}
?>

```

### **consta.php**

En este documento se muestran los valores necesarios para poder acceder a la base de datos, si se cuenta con un host comprado aquí se agrega la dirección y se configura con los datos de este.

```

<?php
const HOST      = "localhost";
const BD        = "BienesRaices";
const USER      = "ivonne";
const PASSWORD  = "123ivy*";
?>

```

### **Configuración**

Para la configuración de la base de datos se mandan a llamar los elementos ya instanciados en consta.php

### **Database.php**

```

<?php
include("consta.php");
try

```

```

{
    $DB_con = new PDO("MySQL:host=".HOST.";dbname=".BD,USER,PASSWORD);
    $DB_con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch(PDOException $e)
{
    echo "Error Conexion: ".$e->getMessage();
}
?>

```

## Constantes.php

Se muestra la ruta del Webservice y la carpeta que lo contiene.

```

<?php
    const WEBSERVICE= "http://localhost/wsbieness/dataws/Webservice.php";
    const URL_BASE = "http://localhost/wsbieness";
?>

```

## guardarPropiedad.php

Aquí se encuentra el método para guardar las propiedades que se han almacenado anteriormente en la base de datos local de Android. Se recibe la cadena JSON, se decodifica y se procede a insertar los campos utilizando la clase Insertar propiedad. Por último se obtiene una respuesta para saber si, los datos enviados se han insertado correctamente o hubo un error, el código se encierra en un try catch.

```

<?php
    include("constantes.php");
    $Propiedad = $_POST["PropiedadPost"];
    if (!get_magic_quotes_gpc()) {
        $Propiedad = stripslashes($Propiedad);
        $data = json_decode($Propiedad);
    }
}

```



```

$options = array('uri' => URL_BASE,'location' => WEBSERVICE);
$client = new SoapClient(NULL, $options);
    for($i=0; $i<count($data) ; $i++)
    {
        $encoding = json_encode($data[$i]);
        $response = $client->InsertarPropiedad($encoding);
        echo $response;
    }
?>

```

### ListarPropiedad.php

Llama a la clase propiedades la cual muestra obtiene las propiedades ya agregadas en la base de datos.

```

<?php
    try
    {
        include("constantes.php");
        $Propiedad = $_POST["IdPost"];
        $options = array('uri' => URL_BASE,'location' =>
WEBSERVICE);
        $client = new SoapClient(NULL, $options);
        $response = $client->Propiedades($Propiedad);
        echo $response;
    }catch(Exception $e){
        echo "Error:". $e->getMessage();
    }
?>

```

### 3.4.2 Aplicación Android

Una vez que ya se ha desarrollado el Webservice, se procede a crear el código de la aplicación Android. Se desarrolla una actividad la cual es la responsable de crear la base de datos. Ver Anexo B.

## Permisos Android Manifest

Para que la aplicación haga uso de la API de la cámara es necesario establecer los permisos pertinentes en el Android manifest.

Agregar los permisos en el android manifest:

```
<uses-permission android:name="android.permission.CAMERA" />
```

Agregar permisos para características de la cámara

```
<uses-feature android:name="android.hardware.camera"
    android:required="false" />
```

Cuando se agregan las características en el manifest, Google Play previene de instalar la aplicación en celulares que no cuenten con una cámara o que no soporte ciertas características que han sido especificadas.

Si la aplicación guardará imágenes o videos en el almacenamiento externo del dispositivo también se debe de especificar, con el siguiente código en el manifest.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

## AltaPropiedadActivity.java

Con los permisos agregados se procede a crear el código para dar de alta las propiedades, para guardar los datos que se han ingresado, primero se obtienen los valores de los datos del formulario para después insertarlos en la base de datos con el método `reg.put(adaptador.columna, nombre del campo.getText().toString());` Por último, establecer el resultado como OK, como se muestra en el siguiente código.

```

private void guardarPropiedad()
{
    String edo_sync = "No";
    String accesoID = getIntent().getStringExtra("accesoID");
    Integer.parseInt(accesoID.toString());
    ContentValues reg = new ContentValues();
    reg.put(PropiedadDbAdapter.C_COLUMNA_TITULO, titulo.getText().toString());
    setResult(RESULT_OK);
    finish();
}

```

En el método abrirMapa, se abre la actividad de maps activity, y una vez que se han obtenido las coordenadas, regresa el valor en un intent.

```

public void abrirMapa(View view) {
    Intent pickCoordIntent = new Intent(this, MapsActivity.class);
    startActivityForResult(pickCoordIntent,
        PICK_COORDINATES_ACTIVITY_REQUEST);
}

```

En el método abrirCamara() se crea un intent para abrir la cámara y regresar un valor, después se crea un archivo para guardar la imagen y el nombre de este. Con Environment.getExternalStorageState(); se obtiene el estado de almacenamiento para saber si la tarjeta SD está montada en el dispositivo. Se crea el directorio BienesRaices el cual existirá cuando la aplicación sea desinstalada.

```

public void abrirCamara(View view) {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri
        startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
    Toast toast = Toast.makeText(getApplicationContext(), "Se ha capturado la
        imagen", Toast.LENGTH_LONG);
}

private static Uri getOutputMediaFileUri(int type) {
    return Uri.fromFile(getOutputMediaFile(type));
}

```

```

    }
private static File getOutputMediaFile(int type){
    String estado = Environment.getExternalStorageState();
    Log.d("STORAGE", estado);

    File mediaStorageDir = new File (Environment.
getExternalStoragePublicDirectory (Environment.DIRECTORY_PICTURES),
"BienesRaices");

    if (! mediaStorageDir.exists()){
        if (! mediaStorageDir.mkdirs()){
            Log.d("CAMARA", "Error al crear el directorio");
            return null;
        }
    }
}
}

```

Para la creación del nombre de la imagen que se capturará, se obtiene la fecha, Año, mes, día, hora, minutos y segundos, y se guarda en la variable timeStamp para crear el nombre del archivo como se muestra en el siguiente código.

```

String timeStamp = new SimpleDateFormat("yyyyMMdd_HH:mm:ss").format(new
Date());
File mediaFile;
    if (type == MEDIA_TYPE_IMAGE){
        mediaFile = new File(mediaStorageDir.getPath() + File.separator +
"IMG_"+ timeStamp + ".jpg");
    } else {
        return null;
    }
return mediaFile;

```

## FormularioPropiedadesActivity.java

Se crean los métodos para obtener los datos de las propiedades y mostrarlos en un formulario.

```

public class FormularioPropiedadesActivity extends AppCompatActivity
implements Serializable {

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_formulario_propiedades);
    Intent intent = getIntent();
    Bundle extra = intent.getExtras();
    if (extra == null) return;
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    CollapsingToolbarLayout toolBarLayout = (CollapsingToolbarLayout)
    findViewById(R.id.toolbar_layout);
    toolBarLayout.setTitle(getTitle());
    dbAdapter = new PropiedadDbAdapter(this);
    dbAdapter.abrir();
    if (extra.containsKey(BienesRaDbAdapter.C_COLUMNA_ID))
    {
        id = extra.getLong(BienesRaDbAdapter.C_COLUMNA_ID);
        consultar(id);
    }
    establecerModo(extra.getInt(ListaUsuariosActivity.C_MODO));
    FloatingActionButton fab = (FloatingActionButton)
    findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            actualizarPropiedad();
        }
    });
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
}

```

En el método consultar, se muestran en el formulario los datos que corresponden al id de la propiedad, se hace una consulta en la base de datos y pone los valores en cada objeto correspondiente.

```

private void consultar(long id)
{

```

```

        cursor = dbAdapter.getRegistro(id);
        String compareValue = "Casa";

        ArrayAdapter<CharSequence> adapter =
        ArrayAdapter.createFromResource(this, R.array.array_propiedad,
        android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dro
        pdown_item);

        titulo.setText(cursor.getString(cursor.getColumnIndex((PropiedadDbA
        dapter.C_COLUMNNA_TITULO))));

        colonia.setText(cursor.getString(cursor.getColumnIndex(PropiedadDbA
        dapter.C_COLUMNNA_COLONIA)));
    }

```

Para actualizar una propiedad se obtienen los datos que están en los textview y se envían en un ContentValues. Con el método dbAdapter.update(reg);

```

    private void actualizarPropiedad() {
        String edo_sync = "No";
        ContentValues reg = new ContentValues();
        reg.put(PropiedadDbAdapter.C_COLUMNNA_ID, id);
        dbAdapter.update(reg);

        Toast.makeText(this, R.string.confirmacion_crear_propiedad,
        Toast.LENGTH_SHORT).show();

        setResult(RESULT_OK);
        finish();
    }
}

```

## InicioSesionActivity.java

Se crea los métodos para validar el usuario y contraseña. Aquí se abre la base de datos una vez que el usuario ha iniciado sesión, y la base de datos no ha sido creada es aquí en donde se crea.

```

public class InicioSesionActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {

```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_inicio_sesion);
usuario = (EditText) findViewById(R.id.etUsuario);
contrasena = (EditText) findViewById(R.id.etContrasena);
dbAdapter = new BienesRaDbAdapter(this);
dbAdapter.abrir();
BienesRaDbHeper dbHelper = new BienesRaDbHeper(getBaseContext());
SQLiteDatabase db = dbHelper.getWritableDatabase();
Toast.makeText(getBaseContext(), "Base de datos preparada",
Toast.LENGTH_LONG).show();
}

```

Para iniciar sesión se capturan los datos ingresados por el agente de ventas, en las variables, `getUsuario` y `getContrasena`, y se hace una consulta a la base de datos, obtiene la contraseña del usuario y después sigue el proceso de validación en donde compara la contraseña que escribió el usuario, con la contraseña que se obtuvo de la base de datos, de ser correctos abre la actividad de panel de usuario.

```

public void iniciarSesion(View v) {
    String getUsuario = usuario.getText().toString();
    String getContrasena = contrasena.getText().toString();
    String contraGuardada = dbAdapter.accesoUsuario(getUsuario);
    String accesoID = dbAdapter.obtenerIDusuario(getUsuario);

    if (getContrasena.equals(contraGuardada)) {
        Intent intent = new Intent(this, PanelActivity.class);
        intent.putExtra("accesoID", accesoID);
        startActivity(intent);

        Toast.makeText(getBaseContext(), "Contraseña correcta",
        Toast.LENGTH_LONG).show();

        Toast.makeText(getBaseContext(), "ID " + accesoID,
        Toast.LENGTH_SHORT).show();
    } else if (getContrasena.equals("")) {
        Toast.makeText(getBaseContext(), "Contraseña incorrecta" +
        getUsuario + getContrasena, Toast.LENGTH_LONG).show();
    }
}

```

```
}
```

## MapsActivity.java

Se crea una actividad del tipo mapas que se utiliza para obtener la ubicación actual del usuario y obtener las coordenadas.

```
public class MapsActivity extends FragmentActivity implements
    OnMapReadyCallback, GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener {

    protected Location mLastLocation;

    private CameraUpdate mCamera;

    public boolean isObtenerCoordenadas() {
        return obtenerCoordenadas;
    }

    public void setObtenerCoordenadas(boolean obtenerCoordenadas) {
        this.obtenerCoordenadas = obtenerCoordenadas;
    }
}
```

En el método `onConnected`, se obtienen la ubicación actual, latitud y longitud y hace que la cámara del mapa se mueva hacia esa dirección.

```
@Override
public void onConnected(Bundle bundle) {
    Toast.makeText(this, "onConnected", Toast.LENGTH_SHORT).show();
    mLastLocation = LocationServices.FusedLocationApi.getLastLocation(
        mGoogleApiClient);
    if (mLastLocation != null) {
        mLatitudeText = mLastLocation.getLatitude();
        mLongitudeText = mLastLocation.getLongitude();
        mCamera = CameraUpdateFactory.newLatLngZoom(new
            LatLng(mLatitudeText, mLongitudeText), 14);
        mMap.animateCamera(mCamera);
    }
}
```

Para obtener la ubicación actual del usuario se utiliza el método, `setUpMapIfNeeded`. Primero se configura el objeto Google Maps, se instancia el objeto `mMap` a partir del `MapFragment` "map".



Con la propiedad `setMyLocationEnable` activada (`true`), se obtiene la ubicación del usuario.

```
private void setUpMapIfNeeded() {
    if (mMap == null) {
        mMap = ((SupportMapFragment)
            getSupportFragmentManager().findFragmentById(R.id.map))
            .getMap();
        if (mMap != null) {
            mMap.setMyLocationEnabled(true);
            mMap.setOnMapClickListener(new
                GoogleMap.OnMapClickListener() {
                    @Override
                    public void onMapClick(LatLng point) {
                        Log.d("Map", "Map clicked");
                        drawMarker(point);
                    }
                });
            mMap.setOnInfoWindowClickListener(new
                GoogleMap.OnInfoWindowClickListener() {
                    @Override
                    public void onInfoWindowClick(Marker marker) {

                        marker.remove();
                    }
                });
        }
    }
}
```

En la aplicación se muestra un mensaje al dar clic en el marcador, en el evento `onInfoWindowClick`, muestra lo que ocurre al hacer clic sobre este, con `marker.remove()`; elimina el marcador. En el caso que el usuario se haya equivocado de ubicación.

En el siguiente método se obtienen las coordenadas y se envía el intent a la siguiente actividad.

```
public void obtenerCoordenadas (View v) {
```

```

        Intent pickCoordIntent = getIntent();
        pickCoordIntent.putExtra("latitude", markLatitude);
        pickCoordIntent.putExtra("longitude", markLongitude);
        this.setResult(RESULT_OK, pickCoordIntent);
        finish();
    }
}

```

Los marcadores cuentan con diferentes propiedades para ser personalizados, en el siguiente código se ha establecido un marcador color naranja, que tiene como título “Bienes Raices” y el texto “Toca aquí para quitar la marca”. Aquí se obtiene la posición del marcador (latitud y longitud).

```

private void drawMarker(LatLng point) {
    Marker markerOptions = mMap.addMarker(new MarkerOptions()
        .position(point)
        .snippet("Tocar aquí para quitar la marca")
        .title("Bienes Raices")
        .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_ORANGE)));
    if (markerOptions != null) {
        markLatitude = markerOptions.getPosition().latitude;
        markLongitude = markerOptions.getPosition().longitude;
        Toast.makeText(this, "Lat:" + markLatitude + "long" +
            markLongitude, Toast.LENGTH_SHORT).show();
    }
}
}

```

## PanelActivity.java

Se crea el panel principal en donde se realizaran los métodos crear, eliminar, actualizar, ver y sincronizar. Se crea un cursor para mostrar las propiedades que se encuentran en el RecyclerView ver Anexo C.

```

public class PanelActivity extends AppCompatActivity {

```

```

private List<PropiedadDbAdapter> mPropiedadDbAdapterList = new
ArrayList<>();

private SQLiteDatabase db;
ProgressDialog prgDialog;
private void consultar(final long id){
    cursor = dbAdapter.getIdProp(id);
    startManagingCursor(cursor);

    propiedadAdapter = new PropiedadCursorAdapter(context, cursor );
    RecyclerView.LayoutManager mLayoutManager = new
    LinearLayoutManager(getApplicationContext());
    listRecyclerView.setAdapter(propiedadAdapter);

    listRecyclerView.setOnItemClickListener(new
    RecyclerViewTouchListener(getApplicationContext(), listRecyclerView, new
    RecyclerViewTouchListener.ClickListener() ...

```

Aquí se muestran los eventos que se realizan al pulsar una propiedad por un largo o corto tiempo. En el método onClick se obtiene la posición del elemento, se guarda en la variable id que es de tipo long y la envía en el método visualizar. Lo mismo ocurre con el método onLongClick, al enviar el id en el método eliminarPropiedad.

```

@Override
public void onClick(View view, int position) {
    propiedadAdapter = new PropiedadCursorAdapter(context, cursor);
    long id = propiedadAdapter.getItemId(position);
    visualizar(id);
}

@Override
public boolean onLongClick(View view, int position) {
    long id = propiedadAdapter.getItemId(position);
    eliminarPropiedad(id);
    return true;
}
}));
}

```

Al eliminar una propiedad muestra un AlertDialog, con dos opciones, cancelar y aceptar. Al hacer clic en el botón aceptar, llama al método de la base de datos delete y le pasa el id de la propiedad que se presionó.

```
private void eliminarPropiedad(final long id) {
    AlertDialog.Builder alertaEliminar = new
    AlertDialog.Builder(context);
    alertaEliminar.setTitle(R.string.titulo_msj)
    .setMessage(R.string.msj_eliminar)
    .setPositiveButton(R.string.ok,
    new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dbAdapter.delete(id);
            Toast.makeText(getBaseContext(), "OK",
            Toast.LENGTH_LONG).show();
            setResult(RESULT_OK);
            cargarLista();
        }
    })
    .setNegativeButton(R.string.cancelar,
    new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(getBaseContext(), "CANCELAR",
            Toast.LENGTH_LONG).show();
        }
    });
    alertaEliminar.show();
}
```

En el método visualizar, se abre la actividad para modificar una propiedad, se obtiene el id de la propiedad que se seleccionó y se envía esa información para mostrar los campos del id de la propiedad

```
private void visualizar(long id)
```

```

    {
        Intent i = new Intent(PanelActivity.this,
FormularioPropiedadesActivity.class);
        i.putExtra(C_MODO, C_ACTUALIZAR);
        i.putExtra(PropiedadDbAdapter.C_COLUMNNA_ID, id);
        startActivityForResult(i, C_ACTUALIZAR);
    }

```

En el método cargar lista se obtiene el id del usuario y de ahí se llama al método consultar con el id que se ha recibido.

```

private void cargarLista(){
    String accesoID= getIntent().getStringExtra("accesoID");
    long id = Long.parseLong(accesoID);
    consultar(id);
}

```

## PropiedadCursorAdapter.java

Se crea un cursor adapter, se utiliza para mostrar la lista de propiedades agregadas.

Primero se crean las variables y se hace referencia a los objetos que se encuentran en las actividades.

```

public class PropiedadCursorAdapter extends
CursorRecyclerViewAdapter<PropiedadCursorAdapter.ViewHolder> {
    public PropiedadCursorAdapter(Context context, Cursor cursor) {
        super(context, cursor);
    }
    private PropiedadDbAdapter dbAdapter =null;
    public static class ViewHolder extends RecyclerView.ViewHolder {
        public ImageView ivCasa;
        public TextView tvTitulo, tvUbicacion;
        public ViewHolder(View view) {
            super(view);

```

```

        tvTitulo = (TextView) view.findViewById(R.id.tv_titulo);
        tvUbicacion = (TextView) view.findViewById(R.id.tv_ubicacion);
        ivCasa = (ImageView) view.findViewById(R.id.ivCasa);
    }
}

```

En el método onCreateViewHolder se manda a llamar la actividad, en este caso es lista\_propiedades.

```

@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View itemView = LayoutInflater.from(parent.getContext())
        .inflate(R.layout.layout_lista_propiedades, parent, false);
    ViewHolder vh = new ViewHolder(itemView);
    return vh;
}

```

En el método onBindViewHolder se llama el viewHolder antes creado, se obtienen los valores de la base de datos y se guardan en una variable.

```

@Override
public void onBindViewHolder(ViewHolder viewHolder, Cursor cursor) {
    int icol_ivCasa = cursor.getColumnIndex(C_COLUMNA_IMAGENES);
    int icol_tvTitulo =
        cursor.getColumnIndex(PropiedadDbAdapter.C_COLUMNA_TITULO);
    int icol_tvUbicacion =
        cursor.getColumnIndex(PropiedadDbAdapter.C_COLUMNA_DIRECCION);
    String stxtCasa = cursor.getString(icol_ivCasa);
    String stxtTitulo = cursor.getString(icol_tvTitulo);
    String stxUbicacion = cursor.getString(icol_tvUbicacion);
    Context context = viewHolder.ivCasa.getContext();
}

```

Después se pone el texto o imagen en el objeto de la lista. Aquí se hace uso de la librería de Picasso, puesto que permite facilitar la presentación de imágenes y comprimirlas, para un uso mínimo de datos.

```

try {
    Picasso.with(context).load(stxtCasa).placeholder(R.drawable.logo)
        .error(R.drawable.ic_autorenew).resize(100,100).centerCrop().into
        (viewHolder.ivCasa);

    viewHolder.tvUbicacion.setText(stxUbicacion);

    viewHolder.tvTitulo.setText(stxtTitulo);
} catch (Exception e){
    e.printStackTrace();
}

```

Es importante contar con la captura de errores, en caso, que el agente no haya tomado la foto, muestra una imagen con el logo de la aplicación, esto permite que la aplicación no se cierre y deje de funcionar inesperadamente.

```

Picasso.with(context).load(R.drawable.logo).placeholder(R.drawable.logo).err
or(R.drawable.ic_autorenew).resize(100,100).centerCrop().into(viewHolder.ivC
asa);

```

## SplashActivity.java

Se crea la actividad de bienvenida que se mostrará 5 segundos al iniciar la aplicación.

```

public class SplashActivity extends Activity {
private static final long SPLASH_SCREEN_DELAY = 3000;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    setContentView(R.layout.activity_splash);
    TimerTask task = new TimerTask() {
        @Override
        public void run() {
            Intent mainIntent = new Intent().setClass(
                SplashActivity.this, InicioSesionActivity.class);
            startActivity(mainIntent);
        }
    };
}
}

```

```
        finish();  
    }  
};  
Timer timer = new Timer();  
timer.schedule(task, SPLASH_SCREEN_DELAY); }}
```



### 3.4.3 Sincronización

En el método `onOptionsItemSelected` del menú de opciones, se agrega el icono de sincronización. Para sincronizar primero se obtiene el ID de usuario que ha iniciado sesión, después se comprueba que haya datos agregados en la base de datos local, de no ser así el proceso se cancela.

Si hay datos agregados y se encuentran datos que no han sido sincronizados, con ayuda de una tarea asíncrona y con uso de la librería GSON se convierten, los datos agregados en la base de datos, en una cadena JSON, la guarda en la variable `PropiedadPost` y la envía con el método `put` al servidor (como es un servidor local se encuentra en la ruta: `http://192.168.0.20`) con la ruta del archivo `guardarPropiedad.php`.

```
public void sincronizar () {
    String accesoID= getIntent().getStringExtra("accesoID");
    long id = Long.parseLong(accesoID);
    AsyncHttpClient client = new AsyncHttpClient();
    RequestParams params = new RequestParams();
    ArrayList<HashMap<String, String>> arrayList =
    dbAdapter.getJson(id);
    if(arrayList.size() !=0) {
        Log.d("Panel Activity", "arrayList" );
        if(dbAdapter.contadorSync(id) != 0){
            prgDialog.show();
            Log.d("Panel Activity", dbAdapter.cadenaJSONsQLITE(id));
            params.put("PropiedadPost", dbAdapter.cadenaJSONsQLITE(id));

            client.post("http://192.168.0.20/wsbieness/guardarPropiedad.php
", params, new AsyncHttpResponseHandler()...
```

El servidor manda una respuesta, (`responseBody`), en la cual incluye el id de la propiedad y el estado de sincronización, si se agregaron los datos con éxito, sobrescribe el valor `edo_sync` por "Si".

```

@Override
public void onSuccess(int statusCode, Header[] headers, byte[]
responseBody) {
    prgDialog.hide();

    try {
        String str = new String(responseBody, "UTF-8");
        System.out.println("RESPONSE: "+str);
        JSONArray array= new JSONArray(str);
        for (int i=0; i<array.length(); i++){
            JSONObject object = (JSONObject)array.get(i);
            ContentValues reg = new ContentValues();
            reg.put(PropiedadDbAdapter.C_COLUMNNA_ID,
            object.get("IdRegistro").toString());
            reg.put(PropiedadDbAdapter.C_COLUMNNA_USR_PROP_ID,
            object.get("prop_id").toString());
            reg.put(PropiedadDbAdapter.C_COLUMNNA_EDO_SYNC,
            object.get("edo_sync").toString());
            dbAdapter.actualizarEdoSync(reg);
        }
    }
}
}

```

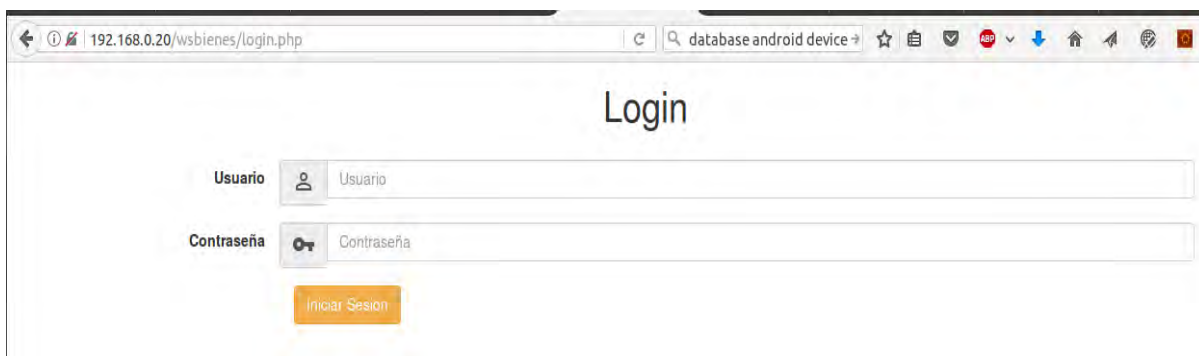
## 3.5 Transición

Con el proyecto ya terminado, se prosigue a poner en práctica el funcionamiento del mismo. Generalmente las empresas bienes raíces cuentan con un Webservice, de ser este el caso, se debe de poner su dirección IP, en el método sincronizar() de la aplicación Bienes Raíces.

### 3.5.1 Funcionamiento

El Webservice que se desarrolló, tiene las mismas herramientas que la aplicación móvil, cuenta con un sistema de acceso y un panel de usuario, en donde se realizan las operaciones CRUD, crear una nueva propiedad, listar las propiedades sincronizadas, modificar y eliminar.

El Webservice está de forma local con la dirección IP 192.168.0.20. La página que proporciona acceso al usuario es login.php, en ella se muestran dos campos, usuario y contraseña, que tienen que ser validados por el servidor. (Figura 3.11).



**Figura 3.11 Webservice inicio sesión**

**Fuente: Edición propia**

Una vez que el usuario ha iniciado sesión de manera satisfactoria, muestra un panel que despliega las propiedades que han sido sincronizadas.

Además de contar con dos botones del lado izquierdo para editar y eliminar la propiedad respectivamente.

Del lado inferior derecho, se encuentra un botón flotante, con el ícono de un lápiz, éste permite agregar una nueva propiedad. (Figura 3.12)

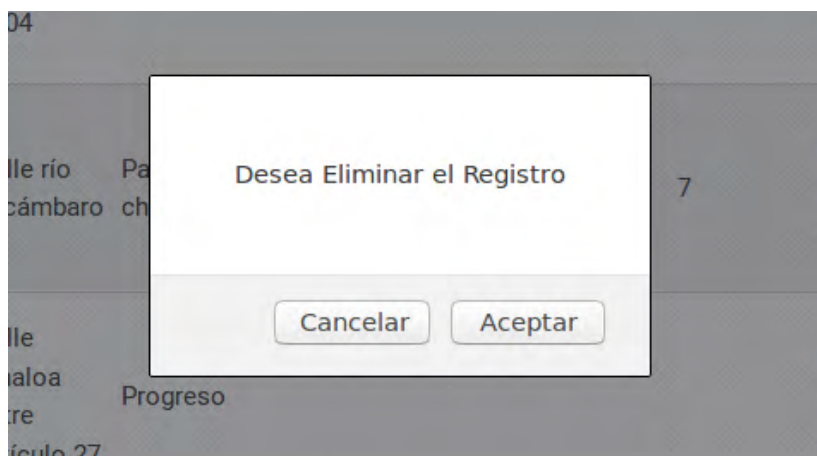
## Webservice Bienes Raices

Herramienta	Título	Propiedad	Categoría	Dirección	Colonia	Coordenadas	Superficie	Recamaras	Imagenes	Precio	Antigüedad	Fecha
<input type="button" value="EDITAR"/> <input type="button" value="ELIMINAR"/>	Casa de prueba	Casa	Venta	Calle sin calle	La mira	19.388135540379682,-	800m2	8	file:///storage/emulated/0/Pictures/BienesRaices/IMG_20161027_133235.jpg	2,000,000	5 años	0000-
<input type="button" value="EDITAR"/> <input type="button" value="ELIMINAR"/>	Casa en venta	Casa	Venta	Calle río zape	Paseos de churubusco	19.38477584287268,-9	100m2	8	file:///storage/emulated/0/Pictures/BienesRaices/IMG_20161025_190552.jpg	6800	10 años	0000-
<input type="button" value="EDITAR"/> <input type="button" value="ELIMINAR"/>	Casa en venta	Casa	Venta	Calle Tacámbaro	Paseos de Churubusco	19.385183197633104,-	50m2	5	file:///storage/emulated/0/Pictures/BienesRaices/IMG_20161025_190735.jpg	100000000	5 años	0000-
<input type="button" value="EDITAR"/> <input type="button" value="ELIMINAR"/>	Departamento en renta	Casa	Venta	Calle el progreso	Progreso	19.386786040166186,-	30m2	3	file:///storage/emulated/0/Pictures/BienesRaices/...	450000	11 años	0000-

**Figura 3.12 Webservice Panel de usuario**

**Fuente: Edición propia**

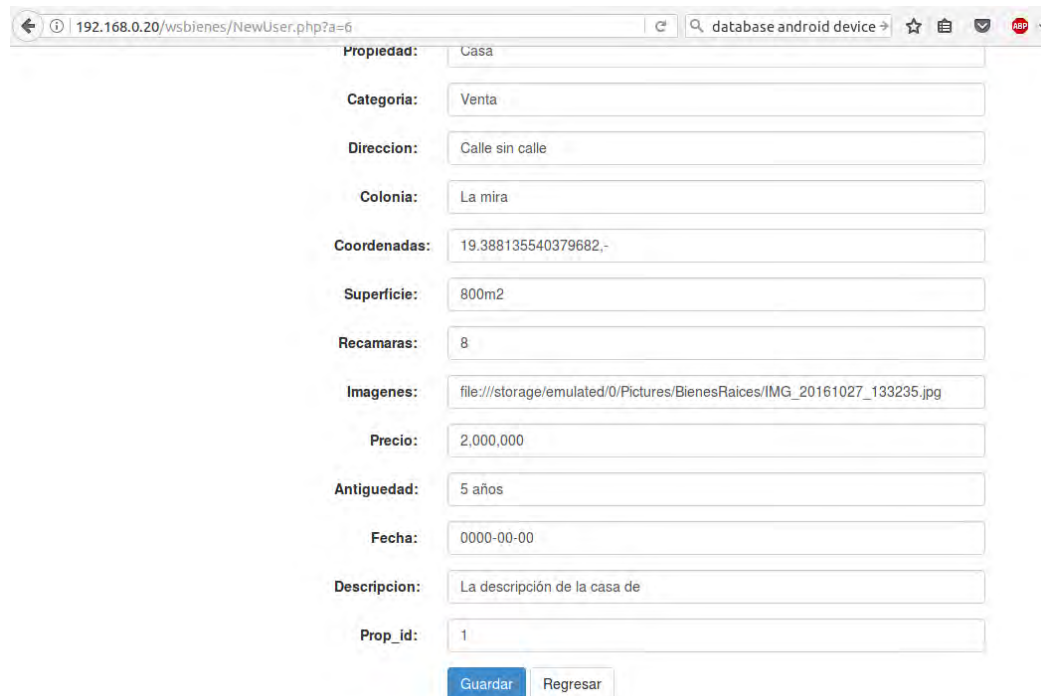
Al dar clic sobre el botón de eliminar, aparece un diálogo de confirmación para corroborar que desea eliminarla. (Figura 3.13)



**Figura 3.13 Diálogo confirmación eliminar propiedad**

**Fuente: Edición propia**

La página Web para agregar y modificar una propiedad (Figura 3.14), muestra los campos que se encuentran en la base de datos de MySQL y la base de datos local SQLite, de la aplicación en Android.



The screenshot shows a web browser window with the URL `192.168.0.20/wsbienes/NewUser.php?a=6`. The page contains a form with the following fields and values:

Propiedad:	Casa
Categoría:	Venta
Dirección:	Calle sin calle
Colonia:	La mira
Coordenadas:	19.388135540379682,-
Superficie:	800m2
Recamaras:	8
Imágenes:	file:///storage/emulated/0/Pictures/BienesRaices/IMG_20161027_133235.jpg
Precio:	2,000,000
Antigüedad:	5 años
Fecha:	0000-00-00
Descripción:	La descripción de la casa de
Prop_id:	1

At the bottom of the form are two buttons: "Guardar" (Save) and "Regresar" (Return).

**Figura 3.14 Webservice Agregar / Modificar propiedades**

**Fuente: Edición propia**

Se crearon todos los métodos para poder manipular los datos de una manera más sencilla.



# CAPÍTULO 4

## RESULTADOS



## **CAPÍTULO 4. Resultados**

A continuación se plantean los requerimientos y el procedimiento necesario para el funcionamiento de la aplicación Bienes y Raíces, por último se muestran los resultados obtenidos.

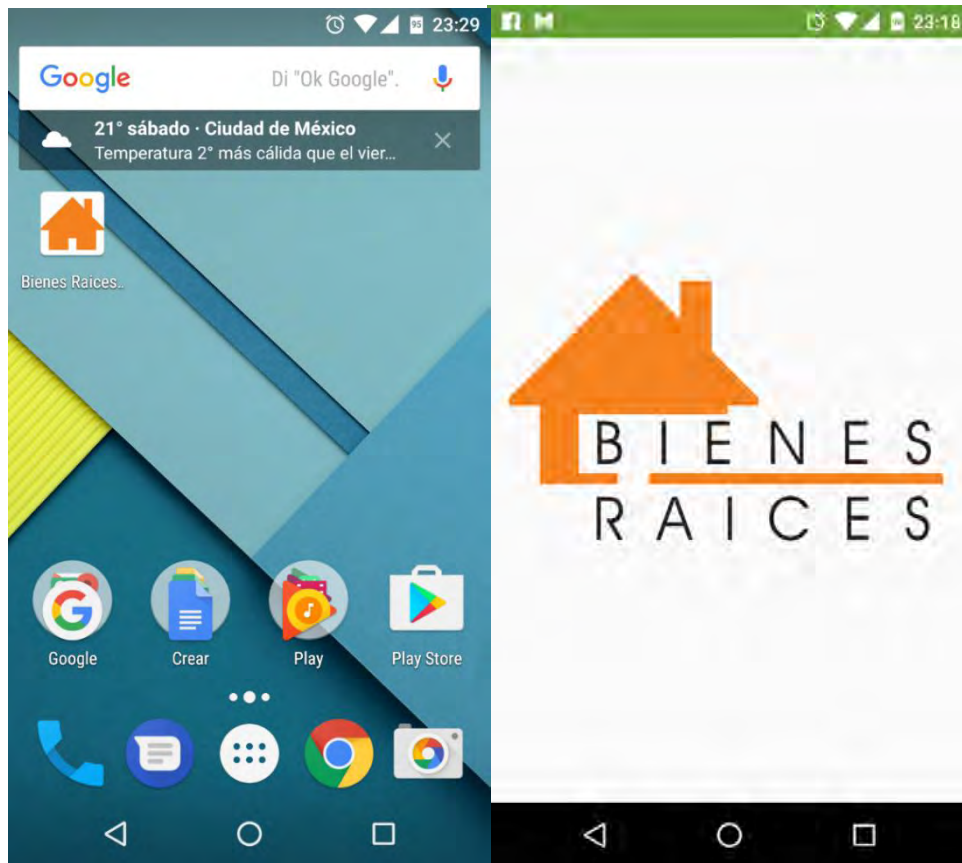
### **4.1 Aplicación Android Bienes y Raíces**

La aplicación de Bienes Raíces puede ser instalada en Smartphones que tengan instalada la versión 4.4 KitKat, la aplicación pedirá permiso para usar la cámara, ubicación del GPS y para almacenamiento interno.

Instalar la aplicación en el dispositivo y aceptar los permisos antes mencionados.

El icono de la aplicación es una casa naranja con un fondo blanco con el nombre Bienes Raíces app.

Iniciar la aplicación Bienes Raíces que se encuentra en el launcher, dando clic en el logo, aparecerá la pantalla de bienvenida por 5 segundos, donde se muestra el logo de la compañía, como se muestra en la Figura 4.1



**Figura 4.1** Pantalla de inicio y Bienvenida

**Fuente:** Edición propia

En la siguiente pantalla se muestran los campos que se tienen que llenar para poder iniciar sesión y así acceder al panel de usuario, esto permite que cada agente de bienes raíces cuente con su propia sesión.

Una vez que ha ingresado correctamente al sistema, se muestra el panel de usuario. En el panel de usuario, se encuentra lo esencial a la vista del agente de ventas, en el menú, un botón con el icono universal para sincronizar, en el centro se encuentra el listado de las propiedades agregadas, con la foto de la propiedad a la izquierda y una breve descripción a la derecha, la cual incluye el título y la dirección de la propiedad. Para agregar una nueva propiedad, se toca el botón anaranjado que se encuentra en la esquina inferior derecha. Figura 4.2.



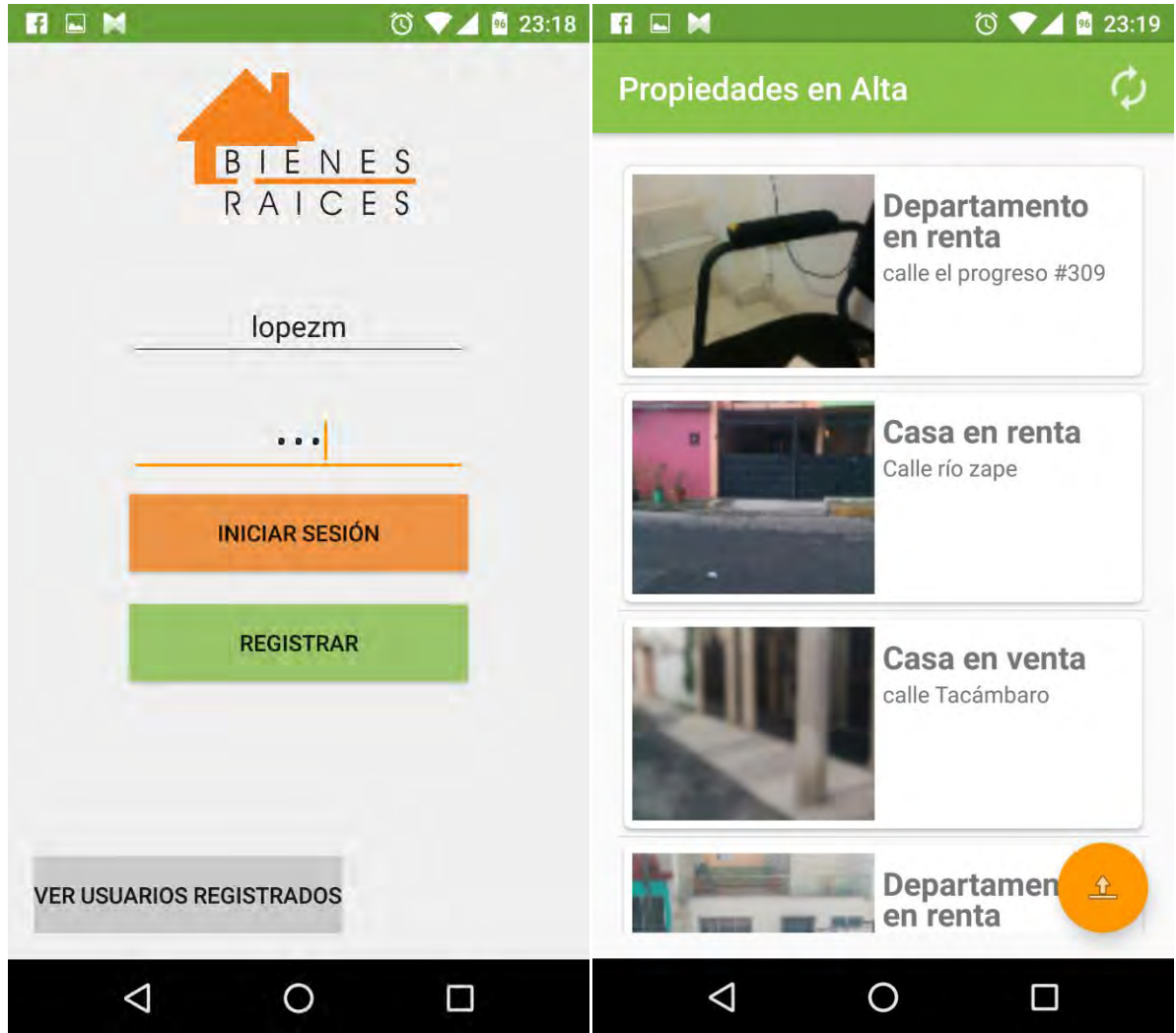
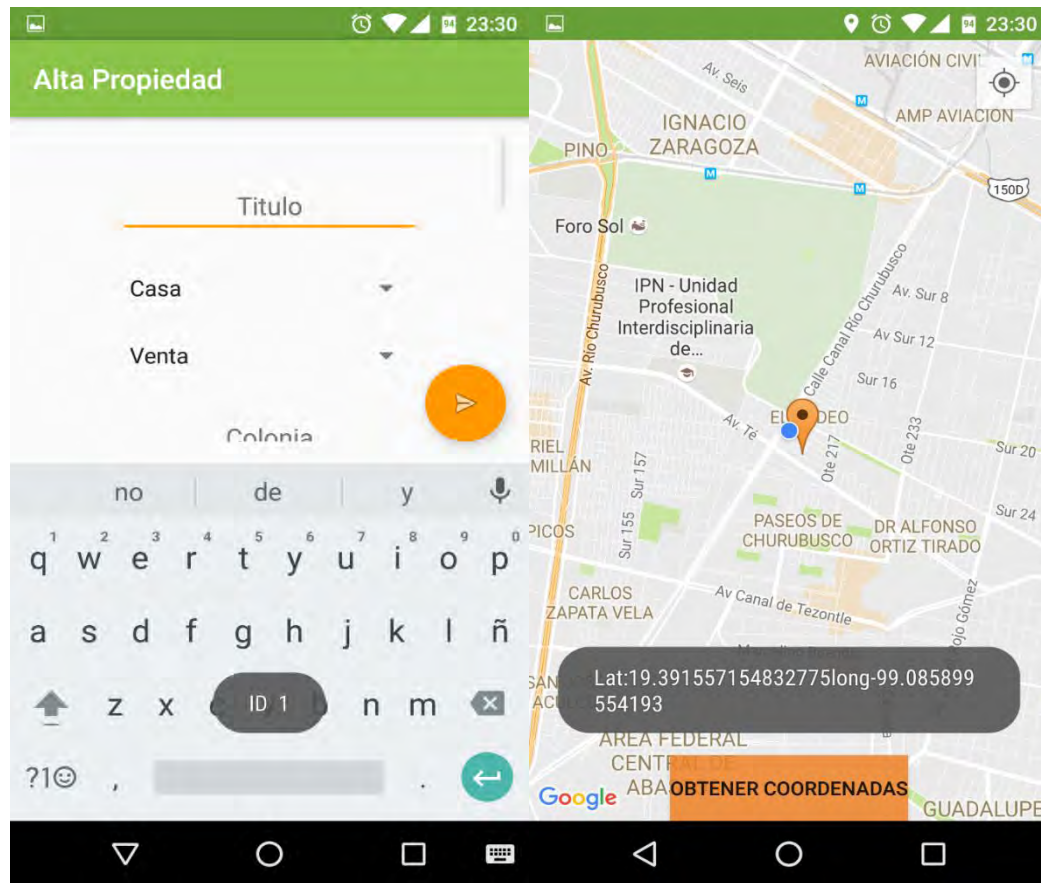


Figura 4.2 Inicio de sesión y panel de usuario

Fuente: Edición propia

La pantalla para dar de alta una propiedad, muestra un formulario, en forma de lista, para llenar los datos de las propiedades, contiene dos botones (Figura 4.3). El primero para la cámara, el cual manda a llamar a la actividad de la cámara, y una vez que se ha tomado la foto la muestra en el formulario. El segundo botón es para acceder a la actividad del mapa, automáticamente se muestra la ubicación actual del agente de ventas, con un punto azul, al igual que en otras aplicaciones de mapas, esto es muy útil porque permite una mejor ubicación de la zona.

Para agregar la coordenada, indicar la ubicación tocando el mapa en el lugar deseado y después dar clic en el botón “Obtener coordenadas”, si se desea cambiar la ubicación, seleccionar el marcador de color naranja para eliminar la marcación y volver a elegirla.

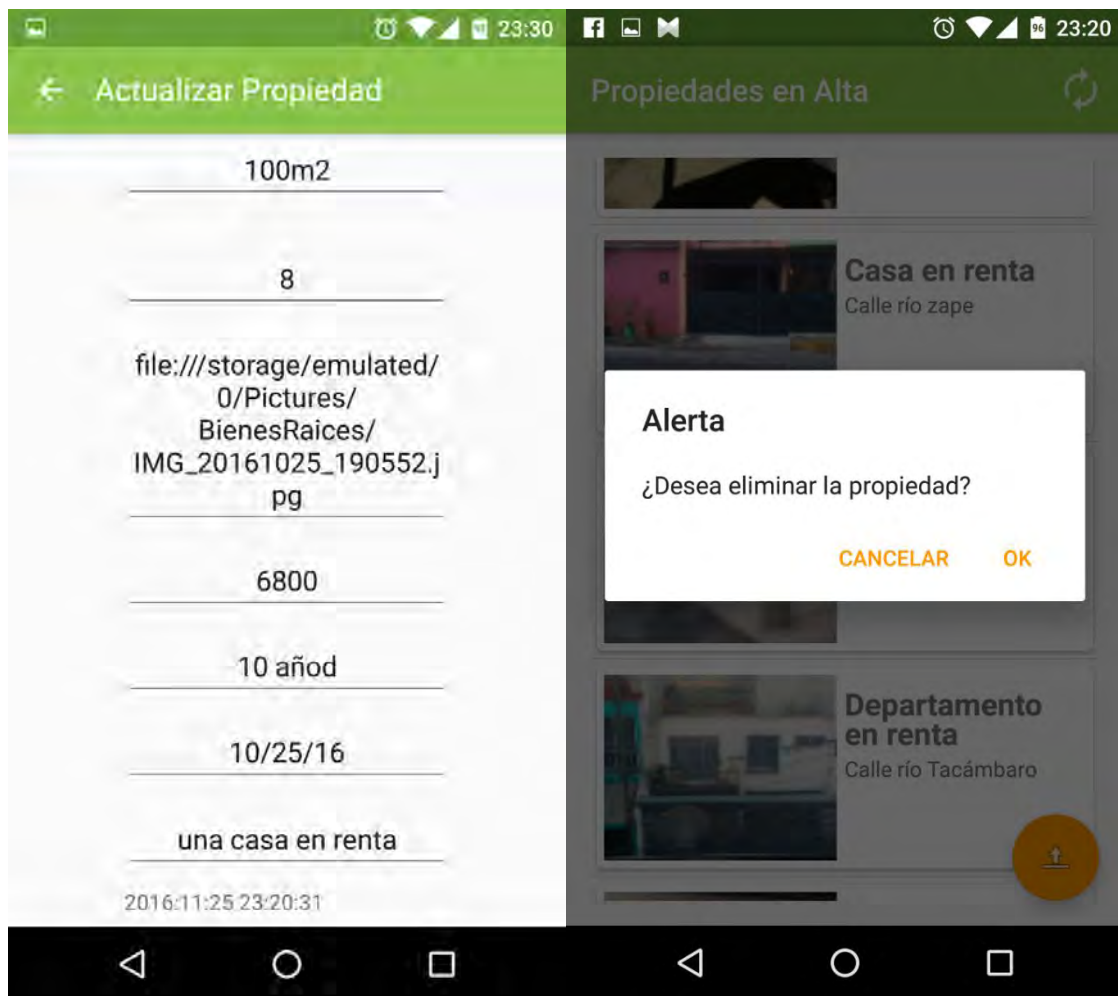


**Figura 4.3 Pantalla Alta de propiedad y Mapa**

**Fuente: Edición propia**

Al dar clic sobre el botón “Obtener coordenadas” se cerrará la actividad del mapa y regresará al formulario de alta de propiedad, colocando en el Textview, las coordenadas, antes seleccionadas.

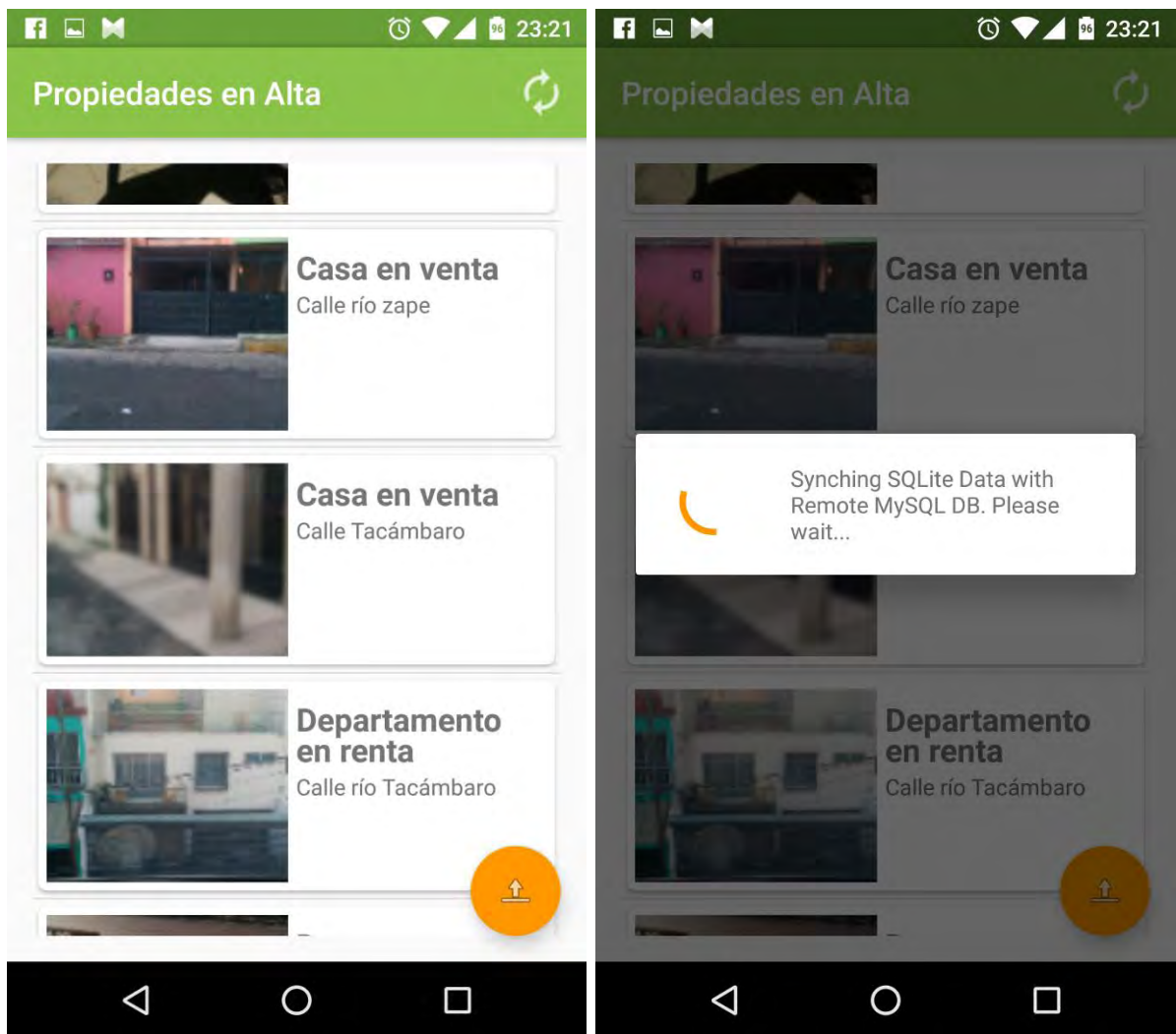
En el listado de las propiedades agregadas, al seleccionar una propiedad, se muestran los datos contenidos en ella para su modificación, como se muestra en la Figura 4.4, para eliminarla mantener presionada la propiedad, y después aparecerá un mensaje de confirmación.



**Figura 4.4 Pantalla Actualizar y Eliminar Propiedad**

**Fuente: Edición propia**

Una vez que se han agregado las propiedades se procede a sincronizarlas, tocar el botón de sincronización, que se encuentra en el menú de la aplicación y esperar a que se suban los datos al Webservice. (Figura 4.5)



**Figura 4.5** Pantalla sincronizar base de datos

Fuente: Edición propia

Si las propiedades se han subido al Webservice mostrará el mensaje de confirmación “Las propiedades han sido sincronizadas correctamente”, de no ser así aparecerá un mensaje de error, estos errores se dan generalmente con la comunicación del Webservice o por que no se cuenta con una conexión a Internet.



# CAPÍTULO 5

## CONCLUSIONES Y TRABAJO A FUTURO



## **CAPÍTULO 5. Conclusiones y Trabajo a futuro**

En la actualidad las empresas con visión, se dan la oportunidad de actualizarse, utilizando tecnologías novedosas. Para ahorrar tiempo y costos de producción, capacitan al personal para el uso de estas tecnologías y el uso de software que en ellas utilizan.

### **5.1 Conclusiones**

Con el desarrollo de esta aplicación, se comprueba que, con las nuevas tecnologías móviles, la implementación de periféricos, sensores y bases de datos, se pueden crear aplicaciones móviles robustas y de muy buena calidad, que permiten hacer más eficiente el trabajo a empresas que manejan y dependen de bases de datos.

Al realizar una aplicación interactiva, con inclusión de mapas, GPS, y de periféricos como la cámara del Smartphone, y con una interfaz limpia, el usuario puede completar sus tareas, que antes le tomaban mucho tiempo. Esto le da la capacidad, al agente de ventas, de enfocarse en otras actividades y obtener una mayor productividad.

Es así que, con el desarrollo del presente trabajo, los objetivos antes mencionados se cumplieron, y se corrobora que utilizando los sensores y dispositivos periféricos, las aplicaciones móviles brindan más interacción al usuario, con la obtención y distribución de los datos recabados para su almacenamiento en un Webservice, le permite agilizar su trabajo, reducir tiempo, costos y eficientar el sistema de bienes y raíces.

## 5.2 Trabajo a futuro

Para el desarrollo de la aplicación se decidió utilizar el sistema operativo Android, sin embargo con la metodología y el razonamiento que se ha desglosado en este trabajo, se puede realizar en otros sistemas operativos como iOS o Windows Phone.

Con el término del trabajo, se identificaron áreas en donde la aplicación puede mejorar, con el fin de ser más eficiente al usuario:

- El inicio de sesión, se puede optimizar, con un control de registro mediante el correo electrónico Gmail, y permitir mantener la sesión abierta al iniciar la aplicación.
- Agregar un sistema de notificaciones push para el envío de mensajes a los usuarios, que tengan la aplicación instalada.
- Permitir seleccionar de la galería de imágenes, la foto que se desea subir al servidor.
- Incorporación de los nuevos dispositivos móviles (Smartwatch) para recibir notificaciones y visualizar información esencial de las propiedades en alta.

Este trabajo solo se enfoca en el lado del agente de bienes raíces, pero con este mismo proyecto, se puede crear una aplicación, del lado del cliente, que solicita la compra o renta de algún inmueble, obteniendo la petición al Webservice, de las propiedades que han sido alojadas previamente.

## BIBLIOGRAFÍA

- Alsitecno. (16 de Marzo de 2014). *Tablet, definición y características*. Obtenido de Alsitecno: <http://alsitecno.com/2014/09/17/tablet-definicion-y-caracteristicas/>
- Android. (17 de Abril de 2014). *Donut*. Obtenido de Android: <https://www.android.com/history/#/donut>
- AndroidCurso. (13 de Abril de 2014). *Arquitectura Android*. Obtenido de Android Curso: <http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android>
- AndroidExperto. (17 de Marzo de 2014). *Que es un Smartwatch*. Obtenido de Android Experto: <http://www.androidexperto.com/dispositivos-con-android/que-es-sirve-smartwatch/>
- Androidos. (16 de Abril de 2013). *Características*. Obtenido de Androidos: <http://androidos.readthedocs.io/en/latest/data/caracteristicas/>
- Apache. (9 de Diciembre de 2015). *Http Server Project*. Obtenido de Apache: [https://httpd.apache.org/ABOUT\\_APACHE.html](https://httpd.apache.org/ABOUT_APACHE.html)
- AreaTec. (16 de Marzo de 2014). *Que es un smartphone*. Obtenido de Areatecnología: <http://www.areatecnologia.com/Que-es-un-smartphone.htm>
- Báez, M., Borrego, Á., Cordero, J., Cruz, L., González, M., Hernández, F., . . . Zapata, Á. (2012). *Introducción a Android*. Madrid: E.M.E.
- Basterra. (16 de Abril de 2013). *Historia*. Obtenido de Androidos: <http://androidos.readthedocs.io/en/latest/data/historia/>
- Bernardo, A. (18 de Diciembre de 2013). *Características Google Glass*. Obtenido de Hipertextual: <https://hipertextual.com/2013/04/caracteristicas-de-google-glass>
- Bluetooth. (19 de Enero de 2015). *What is bluetooth technology*. Obtenido de Bluetooth: <http://www.bluetooth.com/Pages/Fast-Facts.aspx>
- ComputerWorld. (2 de Marzo de 2014). *Las tendencias de 2014 en las telecomunicaciones*. Obtenido de Computerworld: <http://www.computerworld.es/tendencias/las-tendencias-de-2014-en-las-telecomunicaciones>
- Culturación. (21 de Diciembre de 2015). *Que es Apache*. Obtenido de Culturacion: <http://culturacion.com/que-es-apache/>
- Curiosimos. (28 de Diciembre de 2015). *Modelo de desarrollo rápido de aplicaciones*. Obtenido de Curiosisimos: <https://curiosisimos.files.wordpress.com/2009/12/modelo-de-desarrollo-rapido-de-aplicaciones.pdf>
- DECT. (22 de Noviembre de 2014). *Tecnología inalámbrica DECT*. Obtenido de Teléfonos inalámbricos: <http://telefonosinalambricos.info/tecnologia-inalambrica-dect/>
- DefinicionMx. (Marzo de 20 de 2014). *Definición de laptop*. Obtenido de Definicion: <http://definicion.mx/laptop/>
- Ecured. (5 de Octubre de 2015). *Telefono celular*. Obtenido de Ecured: [https://www.ecured.cu/Tel%C3%A9fono\\_celular](https://www.ecured.cu/Tel%C3%A9fono_celular)
- ElLinux. (20 de Diciembre de 2015). *Principales características de Linux*. Obtenido de El Linux: <http://www.elinux.com.mx/1-aprendiendo-linux/11-blog-de-historia/114-principales-caracteristicas-de-linux>

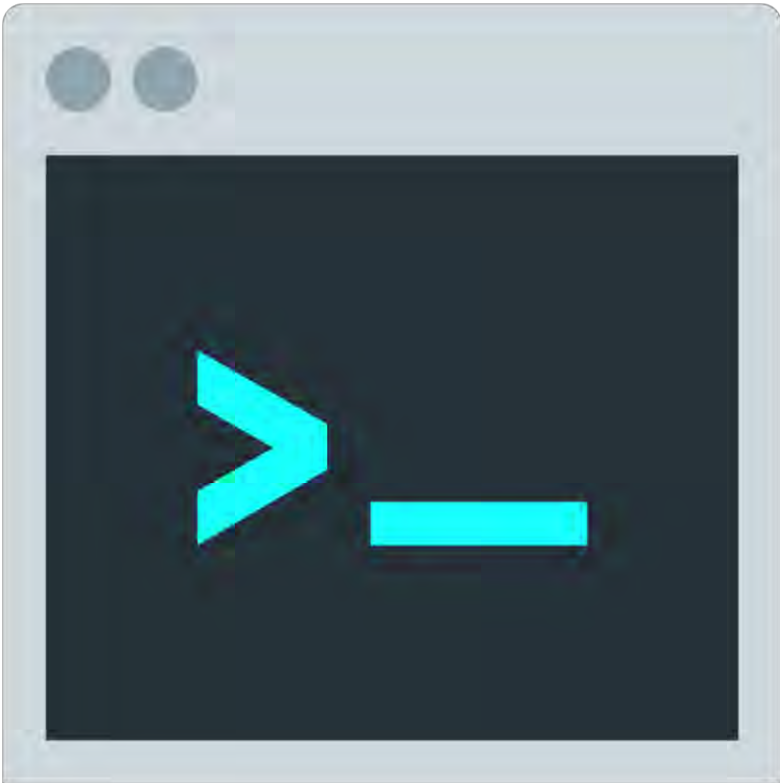


- Ericsson. (2014). *Ericsson Mobility Report*. Suecia: Rima Qureshi. Obtenido de [https://www.ericsson.com/mx/res/region\\_RLAM/pdf/2014/2014-11-18-emr-appendix-es.pdf](https://www.ericsson.com/mx/res/region_RLAM/pdf/2014/2014-11-18-emr-appendix-es.pdf)
- Espeso, P. (18 de Abril de 2014). *De cupcake a Marshmallow*. Obtenido de Xataka: <http://www.xatakamovil.com/sistemas-operativos/de-cupcake-a-marshmallow-asi-han-sido-las-versiones-de-android-a-lo-largo-de-su-historia>
- Fielding, R. T. (13 de Enero de 2016). *Architectural Styles and the Design of Network-based Software Architectures*. Obtenido de School of Information and Computer Sciences, UCI: [https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)
- Gargenta, M. (2011). *Learning Android*. United States of America: O'Reilly Media.
- Gironés, J. (2012). *El gran libro de Android*. México D.F.: Alfaomega.
- Gizmodo. (14 de Marzo de 2014). *Android Lollipop*. Obtenido de Gizmodo: <http://es.gizmodo.com/android-5-1-debuta-inesperadamente-en-los-smartphones-a-1683769621>
- GNU. (20 de Diciembre de 2015). *El proyecto GNU*. Obtenido de GNU: <https://www.gnu.org/gnu/theproject.es.html>
- Gómez, S. (26 de Abril de 2014). *Componentes de una aplicación Android*. Obtenido de Sgoliver: <http://www.sgoliver.net/blog/componentes-de-una-aplicacion-android/>
- Herrera, M. (29 de Febrero de 2014). *INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA*. Obtenido de Academia: [http://www.academia.edu/9795641/INGENIER%C3%8DA\\_DEL\\_SOFTWARE\\_METODOLOG%C3%8DAS\\_Y\\_CICLOS\\_DE\\_VIDA\\_Laboratorio\\_Nacional\\_de\\_Calidad\\_del\\_Software](http://www.academia.edu/9795641/INGENIER%C3%8DA_DEL_SOFTWARE_METODOLOG%C3%8DAS_Y_CICLOS_DE_VIDA_Laboratorio_Nacional_de_Calidad_del_Software)
- Huertas, W. (20 de Diciembre de 2015). *Comunicación inalámbrica*. Obtenido de Decimotrestec: <http://decimotrestec.blogspot.mx/2015/11/comunicacion-inalambrica.html>
- Karch, M. (2010). *Android for Work: Productivity for Professionals*. United States of America: Apress.
- LinuxZone. (20 de Diciembre de 2015). *Ubuntu*. Obtenido de Linux Zone: <http://linuxzone.es/distribuciones-principales/ubuntu/>
- López, E. (28 de Junio de 2014). *Android de 0 a 100*. Madrid, España.
- López, M. (2 de Noviembre de 2014). *GPS*. Obtenido de Unocero: <https://www.unocero.com/2014/09/28/como-funciona-un-gps/>
- López, M. (Febrero de 16 de 2014). *La historia de Android*. Obtenido de UnoCero: <https://www.unocero.com/2013/09/23/la-historia-de-android/>
- Meier, R. (2010). *Professional Android 2 application development*. Indianapolis: Wiley.
- Mena, G. (28 de Diciembre de 2015). *Rad*. Obtenido de Mena: <http://www.mena.com.mx/gonzalo/maestria/ingsoft/presenta/rad/>
- Milette, G., & Stroud, A. (2012). *Professional Android sensor programming*. Indianapolis: Wiley.
- Navarro Maset, R. (9 de Abril de 2015). *REST vs Webservices*. Obtenido de UPV: <http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf>
- Neoteo. (15 de Abril de 2014). *IBM Simon, el primer smartphone de la historia*. Obtenido de Neoteo: <http://www.neoteo.com/ibm-simon-el-primer-smartphone-de-la-historia>
- NFC. (14 de Abril de 2014). *About Near Field Communication*. Obtenido de Near Field Communication: <http://nearfieldcommunication.org/about-nfc.html>

- Nudelman, G. (2013). *Android Design Patterns. Interaction Design Solutions for Developers*. Indianapolis: John Wiley & Sons, Inc.
- Outletech. (16 de Marzo de 2014). *Características básicas de una tablet*. Obtenido de Outletech: <http://outletech.com/caracteristicas-basicas-de-una-tablet/>
- Owens, M. (2014). *The definitive guide to SQLite*. USA: Apres.
- Pozo, J. D. (2011). Introducción a los dispositivos móviles. En J. Prieto Blázquez, J. D. Morillo Pozo, & M. Domingo Prieto, *Tecnología y desarrollo en dispositivos móviles* (pág. 56). Barcelona: Eureka Media, SL.
- Prieto Blázquez, J., Ramírez Vique, R., Morillo Pozo, J., & Domingo Prieto, M. (2011). *Tecnología y desarrollo en dispositivos móviles*. Barcelona: Eureka Media, SL.
- Puttonen, H. (Dirección). (2001). *The Code: Story of Linux documentary* [Película].
- Quees. (14 de Marzo de 2014). *Explicación y definición de Smartphone*. Obtenido de Que es: <http://www.quees.info/que-es-un-smartphone.html>
- Quees. (24 de Marzo de 2014). *Que es un smartwatch*. Obtenido de Qué como quien: <http://quecomoquien.republica.com/actualidad/que-es-un-smartwatch.html>
- Quees. (1 de Diciembre de 2015). *Qué es Apache*. Obtenido de Culturacion: <http://culturacion.com/que-es-apache/>
- RAD. (28 de Diciembre de 2015). *Rad*. Obtenido de Metodología RAD: <http://metodologiarad.weebly.com/>
- Reyes, I. (9 de Noviembre de 2015). *Ubuntu Linux*. Obtenido de Time rime: <http://timerime.com/es/evento/2659013/Ubuntu+Linux/>
- Rómmel, F. (2007). SQLite: La Base de Datos Embebida. *Software Gurú*, 62.
- Scoello. (13 de Abril de 2014). *Sistema Android*. Obtenido de Scoello: <https://scoello12.wordpress.com/caracteristicas/>
- SDK. (10 de Diciembre de 2014). Obtenido de Android Developer: <http://developer.android.com/sdk/index.html>
- Soriano, A. G. (12 de Agosto de 2014). *Dispositivos móviles*. Obtenido de Seguridad: <http://revista.seguridad.unam.mx/numero-07/dispositivos-moviles>
- Tatroe, K., MacIntyre, P., & Lerdof, R. (2013). *Programming PHP, Third Edition*. California, USA: O'Reilly Media.
- TICbeat. (8 de Marzo de 2014). *El crecimiento de la tecnología móvil*. Obtenido de Ticbeat: <http://www.ticbeat.com/sim/crecimiento-telefonía-movil-infografía/>
- Vaqqas. (21 de Diciembre de 2015). *Restful webservices*. Obtenido de Drdobbs: <http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069>
- Videoconsolas. (15 de Marzo de 2014). *Videoconsolas portátiles*. Obtenido de Videoconsolas: <http://videoconsolas.jimdo.com/videoconsolas-port%C3%A1tiles/>
- VIX. (20 de Diciembre de 2015). *Ventajas de ubuntu*. Obtenido de VIX: <http://www.vix.com/es/btg/tech/13080/ventajas-de-ubuntu>
- Wei, J. (2012). *Android Database Programming*. Birmingham: Packt Publishing Ltd.



# ANEXOS



# ANEXOS

## A. Index.php

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <title> CRUD con Webservice, PDO, Json</title>
  <script src="js/jquery-2.1.0.js"></script>
  <script src="js/json.js"></script>
  <script src="js/materialize.js"></script>
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
  <link href="css/materialize.min.css" rel="stylesheet" type="text/css"
/>
  <script>
    function Nuevo() {
      window.location="NewUser.php";
    }
  </script>
<script>
$(document).ready(function() {
  function $_GET(param)
  {
    url = document.URL;
    url = String(url.match(/\?+.+/));
    url = url.replace("?", "");
    url = url.split("&");
    x = 0;
    while (x < url.length)
    {
      p = url[x].split("=");
      if (p[0] == param)
      {
        return decodeURIComponent(p[1]);
      }
    }
  }
}
```

```

    }
    x++;
}
}
Id = ($_GET("id"));
alert(Id);
var idObjeto = new Object();
idObjeto.Id = Id;
var DatosJson = JSON.stringify(idObjeto);
$.post("ListaUser.php",
{
    IdPost: DatosJson },
function(objetosRetorna) {
    var rows = JSON.parse(objetosRetorna);
    console.log(rows);
    console.log("MAP function");
    var objRows = rows.map (function(objeto) {
    objRturn = Object.keys(objeto);
    console.log(objeto);
        if (objeto.Propiedad_Msg == 'Null') {
            return "<tr>" +
                "<td><a href='NewUser.php?a=" + objeto.Prop_id + "'><button
type='button' class='btn btn-default light-green lighten-1'>Editar
</button></a> <button type='button' onclick='Eliminar("+objeto.Prop_id+")'
class='red lighten-1 btn btn-danger '>Eliminar</button></td>"+
                "<td>"+objeto.Prop_titulo+"</td>"+
                "<td>"+objeto.Prop_propiedad+"</td>"+
                "<td>"+objeto.Prop_categoria+"</td>"+
                "<td>"+objeto.Prop_direccion+"</td>"+
                "<td>"+objeto.Prop_colonia+"</td>"+
                "<td>"+objeto.Prop_coordenadas+"</td>"+
                "<td>"+objeto.Prop_superficie+"</td>"+
                "<td>"+objeto.Prop_recamaras+"</td>"+
                "<td>"+objeto.Prop_imagenes+"</td>"+
                "<td>"+objeto.Prop_precio+"</td>"+
                "<td>"+objeto.Prop_antiguedad+"</td>"+
                "<td>"+objeto.Prop_fecha+"</td>"+
                "<td>"+objeto.Prop_descripcion+"</td>"+

```

```

        "<td>" + objeto.Prop_prop_id + "</td>" +
        "</tr>";
    }
    return "<tr>" +
        "<td colspan='5'><center><font
color='red'>" + objeto.Propiedad_Msg + "</font></center></td>" +
        "</tr>";
    });
    $("#tabla tbody").html(objRows.join(""));
}
);
})
</script>
</head>
<body>
    <center>
        <h2>
            Webservice Bienes Raíces
        </h2>
        <div class="fixed-action-btn" style="bottom: 45px; right: 24px;">
            <a class="btn-floating btn-large orange accent-4" onclick="Nuevo()">
                <i class="large material-icons">mode_edit</i>
            </a>
        </div>
        <!-- <button type="button" onclick="Nuevo()" class="btn btn-
success">Agregar Propiedad</button> -->
        <table width="70%">
            <tr>
                <td>
                    <div id="mensaje"></div>
                </td>
            </tr>
            <tr>
                <td>
                    <table class="display table bordered striped" id="tabla">
                        <thead>
                            <tr>
                                <th>Herramienta</th>

```

```

        <th>Titulo</th>
        <th>Propiedad</th>
        <th>Categoria</th>
        <th>Direccion</th>
        <th>Colonia</th>
        <th>Coordenadas</th>
        <th>Superficie</th>
        <th>Recamaras</th>
        <th>Imágenes</th>
        <th>Precio</th>
        <th>Antigüedad</th>
        <th>Fecha</th>
        <th>Description</th>
        <th>Id Usuario</th>
    </tr>
</thead>
</center>
</body>
</html>

```

## B. BienesRaicesDbHelper.java

```

public class BienesRaDbHeper extends SQLiteOpenHelper {
    private static int VERSION_db = 1;
    private static String NOMBRE_db = "BienesRaices" ;
    private static SQLiteDatabase.CursorFactory FACTORY_db = null;
    public static final String TABLA_USUARIOS = "USUARIOS";
    public static final String TABLA_PROPIEADAES = "PROPIEADAES";
    public static final String ID_COLUMNA = "_id";
    public static final String NOMBRE_COLUMNA = "nombre";
    public static final String APELLIDO_COLUMNA = "apellido";
    public static final String EMAIL_COLUMNA = "email";
    public static final String USUARIO_COLUMNA = "usuario";
    public static final String CONTRASENA_COLUMNA = "contrasena";
    public static final String ID_COLUMNA_P = "_id";
    public static final String TITULO_COLUMNA_P = "titulo";
    public static final String PROPIEDAD_COLUMNA_P = "propiedad";

```

```

public static final String CATEGORIA_COLUMNNA_P = "categoria";
public static final String COLONIA_COLUMNNA_P = "colonia";
public static final String DIRECCION_COLUMNNA_P = "direccion";
public static final String COORDENADAS_COLUMNNA_P = "coordenadas";
public static final String SUPERFICIE_COLUMNNA_P = "superficie";
public static final String RECAMARAS_COLUMNNA_P = "recamaras";
public static final String IMAGENES_COLUMNNA_P= "imagenes";
public static final String PRECIO_COLUMNNA_P = "precio";
public static final String ANTIGUEDAD_COLUMNNA_P = "antiguedad";
public static final String FECHA_COLUMNNA_P = "fecha";
public static final String ACTUALIZACION_COLUMNNA_P= "actualizacion";
public static final String EDO_SYNC_COMLUMNA_P = "edo_sync";
public static final String DESCRIPCIONC_COLUMNNA_P = "descripcion";
public static final String USR_PROP_ID = "prop_id";

public static final String CREAR_TABLA_USUARIOS = "CREATE TABLE "
    + TABLA_USUARIOS + "(" + ID_COLUMNNA + " INTEGER PRIMARY KEY
    AUTOINCREMENT, "+ NOMBRE_COLUMNNA + " TEXT NOT NULL, " +
    APELLIDO_COLUMNNA + " TEXT NOT NULL, "+ EMAIL_COLUMNNA + " TEXT
    NOT NULL, " + USUARIO_COLUMNNA + " TEXT NOT NULL, "+
    CONTRASENA_COLUMNNA + " TEXT NOT NULL )";

public static final String CREAR_TABLA_PROPIEDADES = "CREATE TABLE "
    + TABLA_PROPIEDADES + "(" + ID_COLUMNNA_P + " INTEGER PRIMARY
    KEY,"+ TITULO_COLUMNNA_P + " TEXT NOT NULL," +
    PROPIEDAD_COLUMNNA_P + " TEXT NOT NULL,"+ CATEGORIA_COLUMNNA_P + "
    TEXT NOT NULL," + COLONIA_COLUMNNA_P + " TEXT NOT NULL,"+
    DIRECCION_COLUMNNA_P + " TEXT NOT NULL," + COORDENADAS_COLUMNNA_P
    + " TEXT NOT NULL,"+ SUPERFICIE_COLUMNNA_P + " TEXT NOT NULL,"+
    RECAMARAS_COLUMNNA_P + " TEXT NOT NULL,"+ IMAGENES_COLUMNNA_P + "
    TEXT NOT NULL," + PRECIO_COLUMNNA_P + " TEXT NOT NULL,"+
    ANTIGUEDAD_COLUMNNA_P + " TEXT NOT NULL," + FECHA_COLUMNNA_P + "
    TEXT NOT NULL,"+ ACTUALIZACION_COLUMNNA_P+ " TIMESTAMP DEFAULT
    CURRENT_TIMESTAMP,"+ EDO_SYNC_COMLUMNA_P + " TEXT NOT NULL, "+
    DESCRIPCIONC_COLUMNNA_P + " TEXT NOT NULL, " + USR_PROP_ID + "
    INT,"+ "FOREIGN KEY(" + USR_PROP_ID + ") REFERENCES " +
    TABLA_USUARIOS + " (_id) " + ")";

private static BienesRaDbHeper instance;
public BienesRaDbHeper (Context context){
    super(context, NOMBRE_db, FACTORY_db, VERSION_db);
}

public static synchronized BienesRaDbHeper getHelper (Context context){

```



```

        if(instance == null)
            instance = new BienesRaDbHeper(context);
        return instance;
    }
    public void onOpen(SQLiteDatabase db){
        super.onOpen(db);
        if(!db.isReadOnly()){
            db.execSQL("PRAGMA foreign_keys=ON;");
        }
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREAR_TABLA_USUARIOS);
        db.execSQL("CREATE UNIQUE INDEX usuario ON USUARIOS(usuario ASC)");
        db.execSQL(CREAR_TABLA_PROPIEDADES);
    }
}

```

### C. CursorRecyclerViewAdapter.java

```

public abstract class CursorRecyclerViewAdapter<VH extends
RecyclerView.ViewHolder> extends RecyclerView.Adapter<VH> {

    private Context mContext;
    private Cursor mCursor;
    private boolean mDataValid;
    private int mRowIdColumn;
    private DataSetObserver mDataSetObserver;
    public CursorRecyclerViewAdapter(Context context, Cursor cursor) {
        mContext = context;
        mCursor = cursor;
        mDataValid = cursor != null;
        mRowIdColumn = mDataValid ? mCursor.getColumnIndex("_id") : -1;
        mDataSetObserver = new NotifyingDataSetObserver();
        if (mCursor != null) {
            mCursor.registerDataSetObserver(mDataSetObserver);
        }
    }
}

```

```

public Cursor getCursor() {
    return mCursor; }

@Override
public int getItemCount() {
    if (mDataValid && mCursor != null) {
        return mCursor.getCount();
    }
    return 0;
}

@Override
public long getItemId(int position) {
    if (mDataValid && mCursor != null &&
mCursor.moveToPosition(position)) {
        return mCursor.getLong(mRowIdColumn);
    }
    return 0;
}

@Override
public void setHasStableIds(boolean hasStableIds) {
    super.setHasStableIds(true);
}

public abstract void onBindViewHolder(VH viewHolder, Cursor cursor);

@Override
public void onBindViewHolder(VH viewHolder, int position) {
    if (!mDataValid) {
        throw new IllegalStateException("this should only be called when
the cursor is valid");
    }
    if (!mCursor.moveToPosition(position)) {
        throw new IllegalStateException("couldn't move cursor to
position " + position);
    }
    onBindViewHolder(viewHolder, mCursor);
}

public void changeCursor(Cursor cursor) {
    Cursor old = swapCursor(cursor);
    if (old != null) {
        old.close(); } }

```

```

public Cursor swapCursor(Cursor newCursor) {
    if (newCursor == mCursor) {
        return null;
    }
    final Cursor oldCursor = mCursor;
    if (oldCursor != null && mDataSetObserver != null) {
        oldCursor.unregisterDataSetObserver(mDataSetObserver);
    }
    mCursor = newCursor;
    if (mCursor != null) {
        if (mDataSetObserver != null) {
            mCursor.registerDataSetObserver(mDataSetObserver);
        }
        mRowIdColumn = newCursor.getColumnIndexOrThrow("_id");
        mDataValid = true;
        notifyDataSetChanged();
    } else {
        mRowIdColumn = -1;
        mDataValid = false;
        notifyDataSetChanged(); }
    return oldCursor;
}

private class NotifyingDataSetObserver extends DataSetObserver {
    @Override
    public void onChanged() {
        super.onChanged();
        mDataValid = true;
        notifyDataSetChanged();
    }
    @Override
    public void onInvalidated() {
        super.onInvalidated();
        mDataValid = false;
        notifyDataSetChanged();
    }
}
}

```