Reporte Final del Ejercicio Sabático

ELABORACION DE MATERIALES, RECURSOS O AUXILIARES DIDACTICOS.

Domingo Trujillo Venegas

Desarrollo de Aplicaciones de Dispositivos Móviles

23 de Enero al 24 de Julio del 2017

CONTENIDO

INTRODUCCION

La tecnología ha impactado en todos los ámbitos de la vida de los seres humanos, pero uno de los aspectos que más ha impactado en la sociedad son los dispositivos móviles.Los primeros dispositivo estuvieron enfocados en un principio, a la telefonía celular y los mensajes de texto.

En este trabajo vamos abordar desde el inicio de la telefonía móvil hasta el desarrollo de teléfonos inteligentes o smartphones y como podremos desarrollar una aplicación para esos dispositivos.

Los temas a desarrollar son los que se encuentran dentro de la materia de “Desarrollo de aplicaciones de dispositivos móviles”, por lo mismo, esta obra esta diseñada para que los alumnos de la carrera de Ingeniería en Informática, tenga un respaldo cuando cursen dicha materia.

En el primer capítulo se analizaran la evolución de los diferente dispositivos, así como las diferentes tecnologías de redes inalámbricas en las que se apoyan estos dispositivos; además analiza cuales son las tendencias que hay en estos dispositivos.

El segundo capítulo se estudiaran los diferentes sistemas operativos, que se encuentran el mercado, además de las diferentes arquitecturas que presenta los distintos dispositivos móviles. También veremos los distintos entornos de desarrollo en los que podemos construir una aplicación y los diferentes tipos de lenguaje en los que podemos programar nuestra aplicación.

En el tercer capítulo se abordaran las distintas metodologías para el desarrollo de las aplicaciones de dispositivos móviles, así como también, la programación de estos dispositivos qe estará basada para dispositivos que funcionen con el Sistema Operativo Android.

En el cuarto capítulo se hablara de las diferentes maneras para manejar la persistencia en los equipos móviles, además de las formas de transmisión de la información a través de la red.

1.-INTRODUCCIÓN A LAS TECNOLOGÍAS DE MÓVILES

Una de los impactos más importante de la tecnología en esta década es la comunicación inalámbrica, pero quien verdaderamente ha impulsado esta tecnología ha sido la Internet y la aparición de teléfonos inteligente o Smartphone. Es tan grande el desarrollo del Internet que al final de 2016 existían 7,377 millones de suscriptores de telefonía celular en el mundo[1]; otro punto importante es el uso de Internet, ya que en el 2010 utilizaban Internet 2,014 millones de individuos mientras que en 2016 lo utilizan 3,488 millones de individuos en forma global.[2] Un aumento considerable en uso de teléfonos celulares como en el uso del Internet. En México existían 105.6 millones de suscriptores en 2015 en la telefonía celular, en el 2010 existían 91 millones de suscriptores, esto fue un aumento del 18.2%; también es importante recalcar que en el 2015 teníamos 88.9 de suscriptores por cada 100 habitantes y en el 2010 se tenía 77.32, lo que corresponde a un aumento del 14.9 % en el número de suscriptores[3] .

1.2.-Evolución de los dispositivos móviles.

La evolución de los dispositivos ha sido de gran impacto en la sociedad, ya que en tan solo 20 años. Durante este tiempo dicha tecnología ha evolucionado desde la comunicación inalámbrica hasta los dispositivos inteligentes, que de alguna manera están enfocados actualmente a la comunicación sobre Internet, dejando un poco de lado la comunicación vía telefonía celular y la mensajería.

[1] **ITU World TeleComminucation/ICT Indicators Database.**

2 ITU Statistics (http://www.itu.int/ict/statistics**)**

3 Comisión Federal de Telecomunicaciones (CFT).

Es asi como las compañías telefónicas se están enfilando sus negocios hacia la comunicación sobre este medio.

La evolución se ha dado desde las enormes terminales móviles hasta los teléfonos inteligentes, esto debido al gran avance de la tecnología. De tener teléfonos de gran tamaño, costosos y de limitadas prestaciones, a la fecha se tienen teléfonos inteligentes con grandes características como ser más baratos, baterías con una gran duración, tener acceso a Internet de alta velocidad, etc.

Estos avances están altamente relacionado con computación ubicua también conocida como computación ambiental, concepto acuñado por Mark Weiser en el año de 1991, este conocimiento se basa en cuatro paradigmas fundamentales:

- Descentralización.
- Diversificación.
- Conectividad.
- Simplicidad.

La descentralización se caracteriza por el paso de la era de los mainframe a la era de las PC´s, en este paradigma aparece el concepto cliente-servidor. Es importante mencionar que la sincronización, es el concepto que dio pie a que podamos tener nuestra información actualizada en tiempo real.

La diversificación surge de la necesidad de poder realizar los mismos servicios pero en diferentes dispositivos que tienen funcionalidades específicas diferentes, con usuarios diferentes y entornos diferentes.

La conectividad vino a resolver el problema de integración de plataformas, a través de desarrollar protocolos y estándares, dando como resultado que cualquier software se pueda ejecutarse, sobre cualquier plataforma y red.

Por último la simplicidad centrada en diseño de dispositivos con interfaces de usuario intuitivas, que permitirán la integración entre el software y el hardware, que nos dan un acceso rápido y sencillo de los datos.

Durante sus actividades diarias, un usuario que utiliza la computación ubicua lo hace a través de diversos dispositivos y sistemas computacionales simultáneamente, y generalmente lo hará sin percibirlo.

Computación ubicua es entendida como la integración de la informática en el entorno de la persona, de forma que los ordenadores no se perciban como objetos diferenciados, esta definición engloba todos los paradigmas de los que define Weiser. Esta disciplina también se conoce en inglés por otros términos como Pervasive computing, Calm technology, Things That Think y Everywhere. Desde hace unos años también se denomina inteligencia ambiental.

Actualmente el desarrollo de la telefonía móvil ha llegado a perfeccionarse, de tal maneraque, de teléfonos celulares comunes han avanzado a teléfonos inteligentes o Smartphones. Estos teléfonos tienen la capacidad de tener poder comunicarse con diferentes protocolos (Wi-Fi, Bluetooth, NFC, etc), y siendo su mayor característica el poder estar conectado a la red (Internet). Además de ser un organizador personal, posee una características de poder instalar

aplicaciones de procesamiento de datos y conectividad, ya sean del fabricante, operador o por un tercero.

La evolución de la tecnología inalámbrica data desde 1970, y ha ido evolucionado de la siguiente manera:

En 1956 esto anterior a las redes de telefonía celular, las redes de telefonía del mundo carecían de un estándar, por lo que cada empresa definía su propio modelo. Uno de los primeros equipos que se comercializaron fue SRA/Ericsson MTA (Mobile Telephone System A). Un teléfono típico de los primeros sistemas móviles, como se muestra en la Fig1.1, eran equipos, de gran tamaño y que requerían gran consumo de energía por lo que necesitaban de una instalación permanente sobre un vehículo, también era típico que el sistema de comunicación estaba definido por cada empresa que prestaba el servicio.

En esa época eran pocas las personas que utilizan ese servicio, el modelo que se muestra solo era utilizado por dos ciudades suecas, tenía un total de 125 suscriptores entre el año de 1956 a 1967.

Fig. 1.1 SRA/Ericsson MTA

En 1973 Motorola presenta su prototipo que fue el primer teléfono celular portable, que medía 2 pies de largo y pesaba casi 2 libras y tenía un costo de $3995 dólares. El tiempo de funcionamiento era de 30 minutos y se debía que recargar por 10 horas.

Fig. 1.2 Dr. Martin Cooper con su teléfono Dynatec

En 1983 el Dynatec TAC 8000K se convirtió en producto comercial que utilizaba un Sistema de Radio Teléfono, en ese año su inventor el Dr. Martin Cooper logra realizar su primera llamada, a Bell´s Laboratorios.

Fig. 1.3. Teléfono Dynatec TAC 8000K

En 1984 aparece el Nokia Mobira Talkman a diferencia del Dynatec, este teléfono podría suminístranos varias horas de continuo funcionamiento.

Fig. 1.4.-Telefono Nokia Mobira Talkman

En 1989 Motorola presenta el MicroTac 9800X, el primer teléfono totalmente portable. Desde el lanzamiento de los primeros teléfonos móviles, solo era posible el uso de estos equipos cundo eran instalados de estos dado que por su tamaño, solo era posible su instalación en vehículos. Las características de estos teléfonos es que eran flip, primer teléfono de bolsillo; teléfono celular más pequeño y ligero cuando fue presentado.

Fig.1.4.-Telefono Microtac 9800X

En 1993 IBM, presenta su PDA/teléfono, el cual fue considerado el primer Smartphone de la historia, este equipo funcionaba como un equipo de telefonía móvil, asistente digital personal (calendario, agenda, reloj mundial, calculadora, bloc de notas, correo electrónico, juegos). También contaba con una pantalla táctil, para marcar números y el texto se ingresaba a través de un teclado QWERTY

Fig 1.5. Teléfono Simon de IBM.

En 1996, Motorola lanzo al mercado una línea de teléfonos portátiles, desarrollando el Motorola StarTAC, este dispositivo permitía al usuario doblar el teléfono cuando este no era utilizado, a lo que se reconoce como “clamshell”.

Cualidades: su innovador diseño “clamshell”, era el teléfono más ligero y más pequeño al momento de salir al mercado.

Fig.1.6. Telefono StartTAC

En 1996 aparecen las primeras ayudantes personales digitales, conocidas en ese entonces como, Pilot manufacturadas por Palm Inc. Una de las características importantes de estos equipos, es que si bien no presentaban un puerto infrarrojo, luz de fondo o memoria flash, tenían un puerto de comunicación serial. Su tamaño de la RAM era de 512 Kb, y utilizaba la versión 1 de Palm OS. En versiones posteriores fue posible aumentar la RAM interna hasta 1 Mb. Esto fue posible con la compra de módulos de actualización y la sustitución de módulos internos. Esto fue viable dado que los ingenieros de diseño concibieron que estas Palm pudieran ser actualizadas en su hardware.

Fig.1.7. Asistente Personal Digital.- PalmPilot5000.

En 1997 aparece el Siemens S10, que el primer teléfono con pantalla de color, el cual desplegaba los colores, rojo, verde, azul y blanco.

Fig.1.8. Siemens S10

En el año de 1998 aparece el Nokia 8810, es el primer teléfono celular que pensó que las antenas exteriores, eran formas estéticamente desagradables. Por lo que se dio a la tarea de diseñar una antena que pudiera ser escondida dentro del teléfono, llevando al diseño antena plana, como una placa que se ocultaba dentro del equipo, este aparato marco el standard para que los siguientes diseños de teléfonos, no tuvieran antenas externas.

Fig.1.9. Nokia 8810

En el año 1999 la compañía finlandesa de Nokia lanza al mercado el Nokia 7110, que fue el primer móvil en integrar un navegador WAP limitado, que podía realizar navegación a internet. También poseía una ingeniosa tapa de teclado deslizante.

Fig.1.10. Nokia 7710

La compañía Research In Motion (RIM), es la compañía que lanza al mercado los productos Blackberry en 1999, como un paginador de datos, RIM trabajo conjuntamente con Ericcson, para desarrollar una red de datos denominada Mobitex, con característica de paginación dual y sistema inalámbrico de correo electrónico, estas características en un principio solo se podían manipular entre usuarios que poseían equipos Blackberry, posteriormente 20002 aparece el primer teléfono de esta compañía. En esa época esta marca de equipos se hizo indispensable en segmento de gente de negocios.

Fig.1.11. RIM Blackberry 5810.

En 2002 aparece el Sanyo CP-5300, que fue el primer teléfono, con cámara donde se obtenían imagenes de muy baja calidad. En un principio los usuarios pensaban que era excesivo, pero hemos visto que actualmente todos los Smartphone, han desplazado la utilización de cámaras fotográficas digitales y de video.

Fig.1.12. Sanyo SCP-5300.

En 2004 Motorola lanza al mercado el Motorola Razr V3, que presentaba las siguientes características: teléfono clamshell, con una tapa delgada que cubría la pantalla y poseía el teclado ultra delgado, tenía también una cámara integrada, con capacidades multimedia.

Fig.1.13. Motorola Razr V3.

En Junio 2007 entra al mercado de los dispositivos móviles Apple, con su teléfono iPhone, esta compañía es una de las marcas que más tecnología de consumo que han sido tendencia en el mundo de los Smartphone. Su teléfono iPhone ha sido tendencia desde su aparición hasta nuestra fecha; son los creadores de las primeras computadoras de bolsillo, una máquina de juego, dispositivo de reproducción de multimedia, además de poseer un sensor de auto-rotación de pantalla, una interfaz táctil, que sustituye el tradicional teclado QWERTY, el tener una tienda digital para la adquisición de todas las aplicaciones disponibles, para ese modelo y muchas otras características que han hecho de iPhone uno de los teléfonos más vendidos y de mayor tecnología a nivel mundial.

Fig.1.14. iPhone de APPLE

Si bien es cierto en 2007 aparece iPhone, el Smartphone que revoluciono el mercado de los teléfonos inteligentes, los siguientes modelos y las siguientes marcas, que actualmente se encuentran en el mercado han tratado por mucho querer superar a Apple con sus modelos, llámense LG, Samsung, Huawei, Sony, etc, pero lo cierto es que Apple con su teléfono iPhone 7, sigue siendo el modelo de Smartphone más comercializado en el mundo, en el primer trimestre de 2017 como se muestra en la tabla 1.1. Esto es debido a que la tecnología de este teléfono sigue siendo mayor a las demás marcas, los consumidores siguen prefiriendo esta marca, a pesar de que su costo es mucho mayor que sus competidores.

Tabla 1.1 Global Smartphone enviados para su comercialización por modelo en Q1 2017

Global Smartphone comercializados por modelo (millones de unidades)	Q1 2016	Q2 2017
Apple iPhone 7	0.0	21.5
Apple iPhone 7 Plus	0.0	17.4
OPPO R9S	0.0	8.9
Samsung Galaxy J3	0.0	6.1
Samsung Galaxy J5	0.0	5.0
Otros	330.1	294.4
Total	333.1	353.3

1.2 Introducción a las tecnologías y herramientas móviles.

Desde el desarrollo de la telefonía celular y de las nuevas herramientas tecnológicas, así como los avances en software y hardware, el gran impulso del internet, ha generado un gran mejoramiento en los negocios

y en la vida cotidiana de la sociedad, esto debido a que podemos conectarnos a cualquier cosa, nos encontremos en donde nos encontremos.

Dentro de este nuevo contexto, la tecnología móvil, ha tomado gran relevancia en más del 50% de la población mundial, dado que es posible realizar actividades varias de manera ubicua.

Estos avances han hecho que la sociedad, se apegue más y más a este modelo, donde las nuevas generaciones han podido adaptarse más rápidamente a realizar servicios antes impensables, como la comunicación personal, transacciones financieras, que han llevado a que la economía crezca de manera impensable unos cuantos años atrás, otro punto que es importante, es la de poder tener servicios donde podemos tener información fidedigna para la tomas de decisiones.

1.3 Tecnologías emergentes.

El gran avance de la industria digital ha hecho que la vida cotidiana de las personas, haya cambiado en los últimos años, de tal manera que la sociedad está impregnada en todos los aspectos de la vida en este era digital.

Cada día nos hacemos más dependientes de los servicios que nos proporciona la red, esto ha hecho que la industria avance a pasos agigantados para poder dar al usuario los servicios que están demandando.

En 2015 en la reunión anual del World Economic Forum, donde se reunieron más de 800 expertos, para revisar el presente y el futuro de la industria. De esta reunión se publicó un informe bajo el título “Cambio profundo: Puntos de inflexión tecnológicos y su impacto en la sociedad”.

En esta reunión trataron de identificar el software y los servicios que se convertirán en mega tendencias que estarán presentes entre el 2018 - 2027.

Dichas mega tendencias son las siguientes:

Personas y el internet.

Ya en esta época la gente, las empresas, se encuentra conectada a través de la tecnología, cual ha transformado a la sociedad, haciéndola en una sociedad digital, donde más y más individuos se suman a las tecnologías, esperando que para el 2015 el 80% de las personas tendrán presencia digital en internet.

Computación, comunicaciones y almacenaje en todo el mundo.

Como ha venido sucediendo, la tecnología ha avanzado tan rápidamente que el costo y el tamaño de las computadoras han descendido dramáticamente, y la facilidad para poder acceder a internet, además de todos los dispositivos móviles que tienen los recursos, para tener acceso a equipos de gran capacidad de cómputo y acceso casi ilimitado a capacidad de almacenamiento.

Se espera que para 2025 el 91% de las persona tendrán limitada y sin costo el almacenamiento de su información.

Internet de las cosas.

Se están introduciendo sensores más pequeños, más baratos y más inteligentes - en hogares, ropa y accesorios, ciudades, transporte y redes de energía, así como procesos de fabricación

Artificial intelligence (AI) and big data.

La digitalización exponencial crea exponencialmente más datos: sobre todo y todos. En paralelo, la sofisticación de los problemas que el software puede abordar, y la capacidad de software para aprender y evolucionar, está avanzando rápidamente.

Esto se basa en el aumento de los grandes datos para la toma de decisiones, y la influencia que la IA y la robótica están comenzando a tener en toma de decisiones y trabajos.

La economía colaborativa y la confianza distribuida.

Internet está impulsando un cambio hacia las redes y modelos sociales y económicos basados en plataforma. Los activos pueden ser compartido, creando no solo nuevas eficiencias sino también nuevos modelos de negocio y oportunidades para la auto organización social.

El blockchain, una tecnología emergente, reemplaza la necesidad de que las instituciones de terceros brinden confianza para actividades financieras, contractuales y de votación.

La digitalización de la materia.

Los objetos físicos se "imprimen" desde las materias primas a través de impresión aditiva o 3D, un proceso que transforma la industria fabricación, permite imprimir productos en casa y crea un conjunto completo de oportunidades de salud humana.

Todas estas megatendencias, ya están impactando en la tecnología de los dispositivos móviles, lo que hará que en cuanto más se desarrollen estas tecnologías, se podrá tener dispositivos que puedan realizar un

mayor número de tareas en ámbito financiero, tecnológico, de esparcimiento, etc.

1.4 Tecnología de clientes ligeros:

a) Tecnología inalámbrica.

La tecnología móvil ha dado paso a lo que se conoce como Internet móvil, que permite que dispositivos móviles y personas se conecten a la Red desde cualquier lugar y en cualquier momento, lo que ha facilitado la aparición de nuevos servicios y aplicaciones sobre estos dispositivos.

Por otra parte, más del 60% de la población mundial se conecta a Internet con una conexión inalámbrica.

En una red de computadores, podemos distinguir cuatro elementos importantes que intervienen en su definición.

- Protocolo de comunicación
- Topología
- Seguridad
- Medio de transmisión.

Según el alcance, podemos establecer tres grandes grupos:

- Redes de área personal inalámbrica (WPAN: wireless personal area networks).
- Redes de área local inalámbrica (WLAN: wireless local área networks).
- Redes de área extendida inalámbrica (WWAN: wireless wide área networks).

Podemos diferenciar 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).

Redes personales inalámbricas.

- 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.
- Estas pueden ser:
- Bluetooth,

  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.

- DECT: *digital enhanced cordless telecommunicatios*.

  El estándar DECT aparece oficialmente a principios de 1988 impulsado por el ETSI. Inicialmente, se centró en la definición del radioenlace entre los dispositivos inalámbricos y las estaciones fijas, y en los protocolos y estándares necesarios para desarrollar funciones de traspaso (handover) entre estaciones base.

  El estándar DECT, que originalmente admitía transferencias de datos de hasta 552 Kbps, ha evolucionado hasta permitir transferencias de 2 Mbps.

- IrDA: Infrared Data Association

El estándar IrDA utiliza el espectro de frecuencia de infrarrojo para transmitir información.

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°.

- NFC: near field comnunication

  La tecnología near field comnunication (NFC), permite la transmisión de datos de una manera simple entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda ISM de 3,56 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.

- 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.

Redes locales inalámbricas (WLAN)

WLAN es una red de cobertura geográfica limitada, velocidad de transmisión relativamente alta, bajo nivel de errores y administrada de manera privada.

Las WLAN son una extensión y/o una alternativa a las LAN con cables. Los usuarios de una WLAN pueden acceder a los recursos que les ofrece la LAN sin tener que depender de infraestructuras de red (cableado, conectores, etc.).

Las redes WLAN presentan varias ventajas sobre las redes LAN como son:

- **Movilidad**: los usuarios de una WLAN pueden acceder a información en tiempo real desde cualquier lugar de la organización.
- **Instalación simple**: no hay que preocuparse por la instalación de cables dentro del radio de cobertura.
- **Flexibilidad**: permite acceder a lugares que una LAN cableada no alcanzaría nunca.
- **Bajo coste:** aunque el coste inicial de instalación de las WLAN puede ser superior a las LAN con cable, a largo plazo puede suponer un ahorro, sobre todo en entornos con cambios frecuentes de ubicación de los dispositivos.
- Escalabilidad: Las WLAN se pueden configurar con diferentes topologías de una manera sencilla según la necesidad del entorno. Podemos tenerlas WLAN ad hoc (donde los dispositivos se van añadiendo a la red) y las WLAN con puntos de acceso conectados a la red principal.

Existen varias variantes de este tipo de redes, el protocolo de comunicación está dado por el estándar **IEEE 802.11** el cual ha

ido evolucionando, esta evolución está dada por los siguientes protocolos:

- IEEE 802.11 a
- IEEE 802.11 b(Inicialmente denominado WI-Fi)
- IEEE 802.11 g
- IEEE 802.11 i
- IEEE 802.11 n
- HiperRLAN.

Redes de gran alcance inalámbricas (WWAN).

Las WWAN permiten la conexión de redes y usuarios de zonas geográficamente distantes. Podemos distinguir dos tipos:

• WWAN fijas, que utilizan radioenlace o satélite.

• WWAN móviles, que utilizan las compañías u otros servicios públicos en la transmisión y recepción de señales.

Pero si duda las redes WWAN móviles (MWWAN) son las que han vivido una expansión más espectacular en los últimos años. Actualmente las MWWAN son el sistema de comunicación inalámbrico más utilizado, ya que es el que utilizan las operadoras de telefonía móvil.

- **WWAN móvil (MWWAN)**

  En las redes MWWAN el terminal que envía y recibe la información está en movimiento. En estas redes normalmente hay muchos usuarios conectados simultáneamente (acceso múltiple) que utilizan los servicios.

Las más importantes son las siguientes:

- **2G (segunda generación)**. Tecnología de segunda generación, utilizada para describir las redes móviles digitales, como las GSM, que sustituyeron a las redes móviles analógicas de primera generación. Básicamente estaban diseñadas para comunicaciones de voz, mensajería instantánea (SMS).
- GSM: gobal system for mobile communictions.
- Esta tecnología apareció en el año 1990 con una velocidad de transmisión de 9,6 kbps. GSM opera por comunicación de circuitos; esto quiere decir que existe una fase de establecimiento de la conexión que añade tiempo de espera circuito queda abierto hasta que la conexión sea cerrada.
- **2.5G (segunda generación y media)**. Considerada una tecnología intermedia entre 2G y 3G basada en las actualizaciones tecnológicas de las redes móviles GSM para aumentar la velocidad de transmisión de datos y su eficacia.
- La generación abarca los sistemas GPRS y EDGE:
- **GPRS**. Es una técnica de conmutación de paquetes que empezó a utilizarse en el 2001 y que se integró con la estructura actual de redes GSM.
- Esta tecnología permite una velocidad de datos de entre 56 y 115 kbps. Sus ventajas son múltiples y se aplican fundamentalmente a las transmisiones de datos que requieren tráfico discontinuo, como por ejemplo Internet y mensajería electrónica (SMS y MMS), pasan de ser conmutación de circuitos a conmutación de paquetes.

- **EDGE**. También conocida como EGPRS (Enhanced GPRS), es una tecnología que apareció en el 2003 y considerada una evolución del GPRS. EDGE proporciona un ancho de banda superior a la de GPRS, entre 236 y 384 kbps, que permite ejecutar aplicaciones que requieren una mayor velocidad de transferencia de datos, como vídeo y otros servicios multimedia.
- **3G (tercera generación)**. Las tecnologías de 3G son la respuesta a la especificación IMT-2000 de la Unión Internacional de Telecomunicaciones (ITU) para disponer de banda ancha en telefonía móvil y transmitir un volumen de datos importante mediante la red. Con la tercera generación serán posibles las videoconferencias, descargar vídeos, ver televisión en tiempo real y poder realizar la mayoría de las operaciones desde el móvil. La generación abarca el sistema UMTS:
- UMTS se comercializó por primera vez en el 2005 y su velocidad máxima de transmisión de datos es 1,92 Mbps.
- Es considerada una tecnología intermedia entre 3G y 4G, con el principal objetivo de aumentar considerablemente la velocidad de transmisión de datos por las necesidades actuales de los clientes consumidores. Es, por lo tanto, la evolución de 3G y se considera el paso previo de la cuarta generación 4G. La generación abarca los sistemas HSPA y HSDPA:
- HSPA: Teóricamente, admite velocidades de hasta 14,4 Mbps en bajada y hasta 2 Mbps en subida, dependiendo del estado o la saturación la red y de su implantación.
- **4G (cuarta generación).**
- Se define 4G como una integración de red que funciona con la tecnología de Internet donde toda la red es IP, combinándola

- con otros usos y tecnologías, como WiFi y WiMAX.
- En estos momentos, 4G no es una tecnología o estándar definido, sino una colección de tecnologías y protocolos que permiten el máximo rendimiento y con una red inalámbrica más barata. La generación abarca los sistemas LTE y WiMax:
- LTE es la clave para el despegue de Internet móvil, ya que posibilita la transmisión de datos a más de 300 Mbps en movimiento, lo que permite la transmisión de vídeos o TV de alta definición.
- **WIMAX**. Es una tecnología, entre WLAN y WWLAN, que permite hacer conexiones a grandes distancias, con grandes anchos de banda y sin necesitar línea de visión directa entre antenas. WiMAX cumple los estándares IEEE 802.16 y es compatible con otros estándares, como el IEE 802.11, para establecer sistemas de telecomunicaciones conjuntos.

II ARQUITECTURAS Y ENTORNO DE DESARROLLO.

2.1 Sistemas operativos para dispositivos ligeros.

En la actualidad existen varias posibilidades de sistemas operativos para dispositivos móviles, los sistemas más importantes de acuerdo al tamaño de su mercado son los siguientes:

Android:

El sistema operativo Android es sin duda el líder del mercado móvil en S.O, está basado en Linux diseñado originalmente para dispositivos móviles como los teléfonos inteligentes pero después tuvo modificación para ser usado en tablets como es el caso del Galaxy Tab de Samsung , actualmente se encuentra en desarrollo para usarse en netbooks y PCs, el desarrollador de este S.O. es Google, fue anunciado en el 2007 y liberado en el 2008; además de la creación de la Open Handset Alliance, compuesto por 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para celulares, esto le ha ayudado mucho a Google a masificar el S.O,

hasta el punto de ser usado por empresas como HTC, LG, Samsung, Motorola.

Tabla 2.1-Versiones de las distintas distribuciones de Android.

Nombre Código	Num. de versión	Fecha de Lanzamiento	Nivel API
Apple Pie	1.0	23/Septiembre/2008	1
Banana Bread	1.1	9/Febrero/2009	2
CupCake	1.5	25/Abril/2009	3
Donut	1.6	15/Septiembre2009	4
Eclair	2.0-2.1	26/Octubre/2009	5-7
Froyo	2.2-2.2.3	20/Mayo/2010	8
Gingerbread	2.3-2.3.7	6/Diciembre/2010	9-10
Honeycomb	3.0-3.2.6	22/Febrero/2011	11-13
Ice Cream Sandwich	4.0-4.0.5	18/Octubre/2011	14-15
Jelly Bean	4.1-4.3.1	9/Julio/2012	16-18
Kitkat	4.4-4.4.4	31/Octubre/2013	19-20
Lollipop	5.0-5.1.1	12/Nov/2014	21-22
Marshmallow	6.0-6.0.1	5/Octubre/2015	23
Nougat	7.0-7.1-7.1.1	15/Junio/2016	24-25
Oreo	8.0-8.1	21/Agosto/2017	26-27
Pie	9.0	6/Agosto/2018	28

Windows 10 Mobile:

Este es el sistema operativo que Microsoft utiliza en los smartphones y en otros dispositivos móviles. Este S.O. está basado en Windows CE versión 5.2. En 2010, Microsoft inauguró la nueva plataforma para smartphones conocida como Windows Phone 7. El lanzamiento de Windows Phone 8.1 llegó en 2014; La última actualización es el sistema

operativo Windows 10 Mobile. El mayor punto de venta del sistema operativo Windows 10 Mobile es Cortana y la aplicación para las búsquedas. Cortana ya está disponible en español, portugués y francés. Windows 10 Mobile aspira a dar mayor consistencia que la de sus contrapartes en computación, incluyendo:

- Un más alto nivel de **sincronización** de contenidos.
- Una innovadora **plataforma global de aplicaciones** que permite que una aplicación única opere en múltiples dispositivos que utilizan Windows 10, como dispositivos móviles, ordenadores personales y Xbox.
- La libertad de **actualizar** tu dispositivo móvil que funciona con Windows 8.1 a Windows 10 Mobile, sujeto a la aprobación y soporte del fabricante.

Firefox OS

- El S. O. de Firefox está diseñado por Mozilla para smartphones, tabletas y televisores inteligentes. Inicialmente, fue lanzado para la venta en 2013. El S.O. de Firefox OS hace enfoque en tecnología HTML5 para alinearse con capacidades tales como el SMS y el soporte de Bluetooth. Viene con características agregadas tales como:

- Altamente optimizado para hardware de bajo costo, lo que quiere decir que los fabricantes pueden usarlo gratuitamente para producir dispositivos que unen el bajo costo con un alto grado de funcionalidad.
- Está escrito en código abierto, lo que significa que está dirigido por la comunidad.
- Está basado en Linux y en la tecnología Gecko de Mozilla
- Requiere de una baja capacidad de batería, ya que el dispositivo móvil está diseñado para iniciarse en la red y para ejecutar las aplicaciones en la red.

- Viene con tres capas: la capa Giaia- UI (utiliza APIs de web abierta), Gecko- servicios de aplicación en tiempos de ejecución y Gonk- núcleo de Linux y HAL de Android.

Amazon Fire OS

- El S.O. Amazon Fire está basado en el sistema operativo de Android. Amazon es su productor y está específicamente diseñado para el Fire Phone de Amazon y el rango de smartphones y tabletas de Kindle Fire. Este sistema operativo se enfoca principalmente en el consumo de contenidos. Viene con una interfaz de usuario fortificada y está hecho a medida para hacer disponible el contenido de las tiendas y servicios de Amazon. Algunas de las más nuevas actualizaciones del S.O. incluyen:
- **Mayor compatibilidad que la que existía hasta el momento con aplicaciones actuales de Android:** Esto quiere decir que muchas de tus aplicaciones funcionarían en dispositivos Fire sin agregados de esfuerzo de ingeniería.
- **Testeo gratuito de compatibilidad dentro de los 90 segundos:** Esto significa que solo necesitas arrastrar y soltar tu aplicación de Android en el Servicio de Testeo de Aplicaciones (App Testing Service) para averiguar los resultados de la compatibilidad de la misma.
- **Amazon Rapids:** Se considera que esto cambia las reglas del juego para Amazon. Amazon Rapids es una aplicación para la lectura, específicamente dirigida a los niños. Los cuentos son contados a los niños en el estilo de sesiones de chateo.
- **Soporte de video Alex:** Es una aplicación capaz de buscar juegos, libros de audio, programas de televisión y películas en Amazon video.

Ubuntu Touch

- En perspectiva, Ubuntu Touch imita a cualquier otro sistema operativo. Mientras que Ubuntu Touch está por detrás de Android en algunos

aspectos, el último conjunto de parches podría resultar en una plataforma para una nueva confrontación entre sistemas operativos. Las nuevas actualizaciones incluyen:

- Panel mejorado de actualizaciones: Esto permite estar al tanto de las actualizaciones nuevas y ser testigo de los cambios que traen en tiempo real.
- Teclado mejorado de emoticonos: El último teclado de emoticonos era desalentador a la hora de navegarlo. La nueva versión es bastante sencilla para comprender y para hallar el emoticón correcto y apto para ese momento.
  Sincronización del calendario: El nuevo sistema operativo agrega soporte para el iCal y para el CalDAV. También te permite sincronizar varios calendarios a través de tus cuentas de Google y de tu nube propia.
- Nuevo panel de notificaciones: Trae una configuración refinada de notificaciones. La nueva configuración de notificaciones de permite elegir cuáles aplicaciones específicas te notificarán con sonido, vibración o un globo (bubble) de notificación.

iOS

- iOS de los iPhones, anteriormente denominado iPhone OS creado por Apple originalmente para el iPhone, siendo después usado en el iPod Touch e iPad. Es un derivado de Mac OS X, se lanzó en el año 2007, aumento el interés con el iPod Touch e iPad que son dispositivos con las capacidades multimedia del iPhone pero sin la capacidad de hacer llamadas telefónicas, en si su principal revolución es una combinación casi perfecta entre hardware y software, el manejo de la pantalla multitáctil que no podía ser superada por la competencia hasta el lanzamiento del celular Galaxy S I y II por parte de Samsung.
- El tiempo ha jugado a su favor y el número de iPhone (iPod Touch) que se han vendido ha hecho que la cantidad de desarrolladores sea

abrumadora, impulsados por el perfil de compradores de iPhone (que históricamente han tenido cierto poder adquisitivo), siendo Apple con su sistema operativo móvil uno de los mayores precursores de las tiendas de aplicaciones. En este caso se trata de App Store y cuenta con gran cantidad de aplicaciones (tanto de pago como gratuitas).

Tabla2.2.-Mercado de los sistemas operativos.

Operating System	4Q16 Units	4Q16 Market Share (%)	4Q15 Units	4Q15 Market Share (%)
Android	352,669.9	81.7	325,394.4	80.7
iOS	77,038.9	17.9	71,525.9	17.7
Windows	1,092.2	0.3	4,395.0	1.1
BlackBerry	207.9	0.0	906.9	0.2
Other OS	530.4	0.1	887.3	0.2
Total	**431,539.3**	**100.0**	**403,109.4**	**100.0**

Source: Gartner (February 2017)

Further information is available in the Gartner report entitled "Market Share: Final PCs, Ultramobiles and Mobile Phones, All Countries, 4Q16."

Como se observa, el mercado de los sistemas operativos es por mucho dominado por Android que tiene un 82% del mercado y seguido por iOS, el cual tiene un 17%. Los demás sistemas operativos, solo tienen un 1% del mercado mundial de los sistemas operativos.

2.2 Arquitecturas.

Básicamente la arquitectura de un teléfono celular está compuesta por los siguientes elementos:

Sistema de repetidores de celular RF

Sistema Banda Base.

Circuito integrado digital.

Circuito integrado analógico.

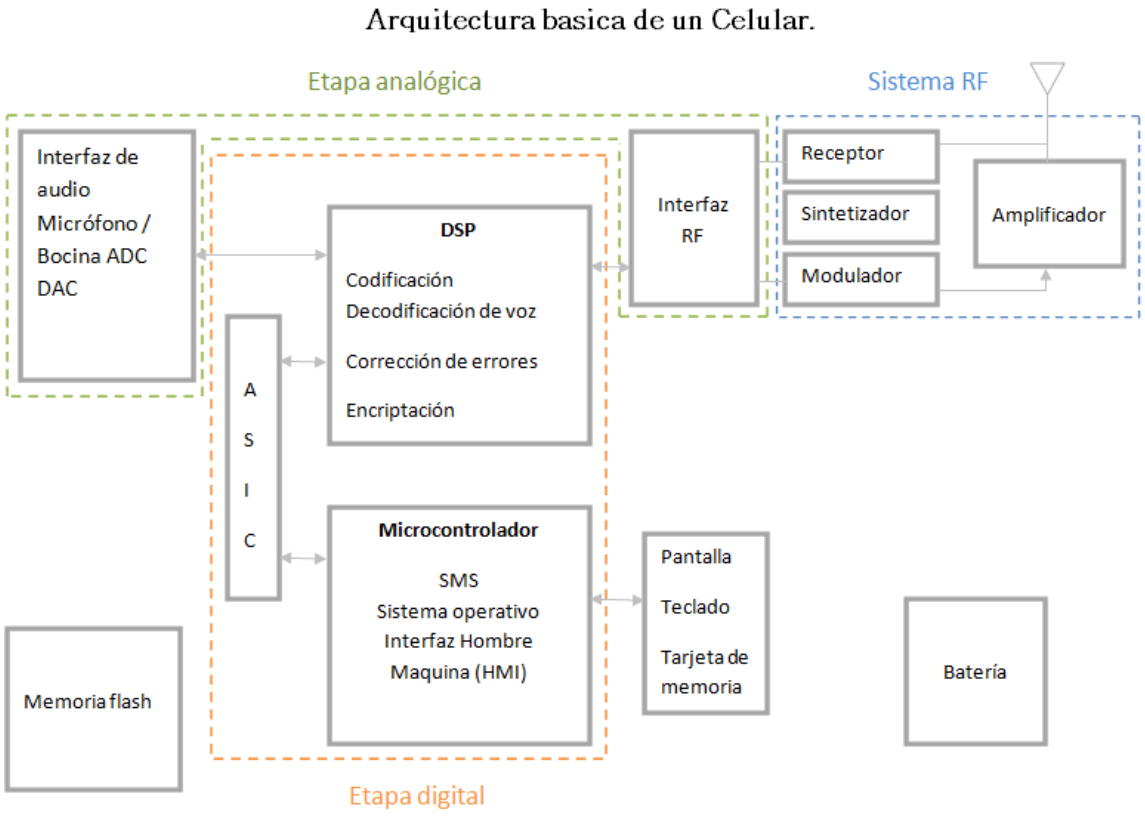


Figura 2.1 Arquitectura básica de un celular

Como podemos observar de la figura anterior, podemos observar que los teléfonos celulares se pueden descomponer en 3 módulos: el módulo de radio (Radio Frecuencia), el módulo de audio (AF audiofrecuencias) y el módulo de lógico de control (CPU).

El módulo RF es módulo encargado de administrar todas las señales que entran y salen de los dispositivos móviles.

El módulo AF es el responsable de la conversión de las señales FI (frecuencia intermedia) provenientes de módulo RF en señales de voz que el usuario pueda escuchar en dispositivo móvil.

El módulo lógico de control tiene la estructura similar a la de una computadora.

Dada la estructura del módulo lógico de control una de las partes importantes es el CPU del dispositivo móvil en donde se realizan los procesos u operaciones. En la actualidad los procesadores más utilizados, están basados en la tecnología System-on-a-chip (SoC).

SOC.-describe la tendencia cada vez más frecuente de usar tecnologías de fabricación que integran todos o gran parte de los módulos componentes de un computador o cualquier otro sistema informático o electrónico en un único circuito integrado o chip.

El diseño de estos sistemas puede estar basado en circuitos de señal digital, señal analógica o incluso de señal mixta (tanto analógica como digital), y a menudo módulos o sistemas de radio frecuencia.

La diferencia principal de un SoC con un microcontrolador tradicional no debe pasarse por alto, puesto que estos rara vez disponen de más de 100 Kilobytes de memoria RAM.

Actualmente los llamados MultiProcessor System-on-Chip (MPSoc) – son construidos con microprocesadores dotados de varios núcleos o bien más de un microprocesador.

De los procesadores más utilizados en la construcción de dispositivos móviles son los procesadores ARM (Advanced RISC Machine)que son la arquitectura que domina el mercado de los procesadores para celulares con aproximadamente un 85% del mercado, esto debido a su principal característica; el bajo consumo de energía.

ARM es una arquitectura RISC (Reduced Instruction Set Computer=Ordenador con Conjunto Reducido de Instrucciones) de 32 bits desarrollada por ARM Holdings. Posteriormente se lanzó la arquitectura de 64 bits la ARMv8-A con los modelos ARM Cortex-A53 y ARM Cortex-A57.

Otro procesador importante es el procesador Hummingbird que está basado en una tecnología ARM cortex-A8, sin embargo Samsung optimizo el rendimiento de estos procesador, mejorando el diseño lógico logrando que un mismo número de procesos puedan realizar en menos tiempo.

Cuando Apple aparece en mercado con sus dispositivos móviles lanza el procesador Apple A4 es un System on chip, que integra un microprocesador basado en la arquitectura ARM; y una GPU PowerVR 535 en un mismo encapsulado, ha esta tecnología se le denomina sistema de paquete (PoP).

Este procesador basado en Cortex A8, hace que el iPhone puede reproducir vídeos en alta definición y gracias a la GPU (Unidad de Procesamiento Grafico) ejecutar juegos fluidamente y añadir efectos gráficos a la interfaz.

Actualmente los procesadores utilizados en IPhone X son los El Apple A11 Bionic es un sistema basado en ARM de 64-bits en un chip (SoC), diseñado por Apple Inc. Tiene dos núcleos de alto rendimiento qué son 25% más rápido que el A10 Fusión y cuatro núcleos de alta eficiencia

que son un 70% más rápido que los núcleos de bajo consumo de energía en el A10.

Procesadores Medfield de Intel procesadores de un solo núcleo, pero con la posibilidad de procesar dos threads (hilos).

2.3 Entorno de desarrollo.

Un entorno de desarrollo también denominado IDE (Integated Development Enviroment) es programa informático formado por un conjunto de herramientas. Que pueden dedicarse exclusivamente a un lenguaje de programación para varios.

Es un entorno de programación en el cual nos ayuda al tener un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

En el desarrollo de aplicaciones móviles las más importantes son:

- Eclipse
- Android Studio
- Netbeans
- Visual Studio
- X-Code.
- Swift

2.4 Requerimientos de los dispositivos ligeros.

Un dispositivo de cliente ligero es aquel que trabaja con arquitectura cliente-servidor y que en su interior tiene poca o nula lógica del programa, por lo que depende del principalmente del servidor central para muchas áreas de procesamiento.

Cuando se va a desarrollar un proyecto de una aplicación para dispositivos móviles, es importante determinar qué elementos y funciones que necesita un proyecto para desarrollarse adecuadamente.

Los requerimientos son declaraciones que identifican atributos, capacidades o características y/o cualidades que necesita cumplir un sistema.

Requerimientos de hardware y software para el desarrollo de aplicaciones Android.

Windows Vista o posterior.

Linux centos 6.o Centos 7.

Java versión 8.0 (jdk 1.8.0_51)

800 Mhz en el procesador

1G de RAM.

1024 X 768 en resolución de pantalla.

Existen 2 plataformas para el desarrollo de aplicaciones para Android

Plataforma Eclipse, Eclipse es la plataforma en la que se ejecuta el plug-in

Instalar el Android SDK.

Configurar a virtualización de la memoria RAM.

Configurar el Emulador para un dispositivo en particular.

Plataforma Android Studio, en este caso es la plataforma que se ejecuta l Plug-in.

Instalar el Android SDK.

Configurar a virtualización de la memoria RAM.

Configurar el Emulador para un dispositivo en particular

Requerimientos para desarrollo de aplicaciones para dispositivos con iOS.

Los requisitos mínimos para desarrollar esta plataforma son

Mac OS X 10.6 (Snow Leopard) o posterior:

iOS SDK 5.0 o posterior.

Dispositivo móvil para pruebas (opcional).

El IDE de desarrollos se XCode para objective-C oel del lenguaje Swift.

Configuración del emulador, para un dispositivo en particular

.

2.5 Lenguajes de programación.

En la actualidad, el trabajar en el campo de la programación para dispositivos móviles se hace necesario, debido a que las empresas se deben adaptar a las tendencias del mercado y a las necesidades de sus clientes. Por lo que se debe pensar en la posibilidad de tener acceso a la información en cualquier lugar y en cualquier instante, a través de distintos dispositivos móviles.

Los lenguajes de programación para los dispositivos móviles dependen en gran parte del dispositivo en el que se quiera trabajar, sin embargo tienen en común que se pueden crear sistemas visuales robustos con mayor facilidad independiente del lenguaje de programación que se esté aplicando.

Pero al momento de decidir qué tipo de desarrollo vamos a utilizar, tendremos que tomar en cuenta si realizamos desarrollo nativo, web o hibrido.

Aplicaciones nativas son las que utilizan el lenguaje específico de cada plataforma. Por ejemplo, Java para Android y Objetive-C o Swift para IOS

Aplicaciones híbridas es un enfoque de programación para dispositivos móviles que combina las fortalezas de la programación nativa con otras tecnologías (Como tecnologías web o algún lenguaje de programación diferente al de la plataforma) para desarrollar aplicaciones multiplataforma que se ejecuten de forma nativa en Android y IOS.

Aplicación móvil web es básicamente el desarrollo de aplicaciones web comunes, pero optimizadas para ser visualizadas desde un dispositivo móvil. Por ende, estas aplicaciones son ejecutadas mediante el navegador del dispositivo.

Programación para Windows Phone.

La plataforma Windows Phone soporta los lenguajes de programación **C#** y **Visual Basic .NET**. Esto se debe a que ambos son entendidos por el Framework .NET. Por la parte de diseño se utiliza el lenguaje **Silverlight** también conocido como **XAML**. En el caso de las aplicaciones de tipo video juegos, utilizan **XNA** para generar gráficas de tipo 2D y 3D e igualmente pueden hacer uso de Silverlight al mismo tiempo.

Para programar nuestro código, se necesita del ambiente de desarrollo Visual Studio 2015 Service Pack 1 en cualquiera de sus versiones, de igual forma se puede descargar la versión Visual Studio Express la cual es totalmente gratuita.

Programación para IPhone OS o IOS.

Programar en iOS implica utilizar OSX, el sistema operativo de Apple. Es necesario utilizar un equipo con OSX instalado.

Apple proporciona su propio entorno de desarrollo, Xcode, de forma gratuita a través de iTunes Store. El primer paso para iniciar es crear una cuenta de iTunes para poder descargarse el IDE Xcode.

Sin entrar en detalles, Objective-C es en esencia C, pero con la capacidad de programación orientada a objetos.

Para crear aplicaciones de iOS, Apple proporciona el API (Application Programming Interface) de Cocoa Touch, basado en el API de Cocoa que se utiliza en OSX; se trata de un conjunto de objetos/funciones que proporcionan una capa de abstracción: esto permite manejar en código objetos de alto nivel que referencian, por ejemplo, partes del hardware del dispositivo que se vaya a usar, o bien elementos de la interfaz como pueden ser botones u otros controles.

Utilizar Cocoa Touch implica utilizar Objective-C, lo que quiere decir que es necesario saber la sintaxis de éste y cómo funciona el envío de mensajes entre objetos.

Lenguaje Swift.

Swift es un lenguaje de programación creado por Apple con el compilador LLVM para los sistemas operativos OS X e iOS presentado el 2 de junio del 2014 en **Worldwide Developers Conference** junto con su manual.

Swift es un lenguaje de programación multiparadigma creado por Apple enfocado en el desarrollo de aplicaciones para iOS y macOS. Fue presentado en WWDC 2014, y está diseñado para integrarse con los Framwors Cocoa y Cocoa Touch, puede usar cualquier biblioteca programada en Objetive-C y llamar a funciones de C. También es posible desarrollar código en Swift compatible con Objetive-C bajo ciertas condiciones. Swift tiene la intención de ser un lenguaje seguro, de desarrollo rápido y conciso. Usa el compilador LLVM incluido en Xcode 6. En el año 2015 pasó a ser de código abierto.

Programación en Android.

La gente de Google ha escogido el lenguaje Java para dar soporte a aquellas personas y empresas que deseen realizar aplicaciones de forma “nativa” en la plataforma de Android.

Basic4Android..

Basic4Android es una plataforma de programación para aplicaciones Android cuyo lenguaje base de programación es VisualBasic, el eterno rival de Java, ese lenguaje que está orientado a aquellas personas que empezamos en el mundo de la programación de una manera más gráfica y no tan abstracta. No es el mismo lenguaje de Microsoft, pero su sintaxis es la misma, lo cual tiene sus mismas ventajas como algunos de sus inconvenientes.

Mono para Android (Xamarin).

> Otro de los lenguajes que Microsoft desarrollo para hacer aplicaciones fue C# y .NET, las cuales son muy usados en diferentes ambientes, por lo que no podría faltar que estos lenguajes tan comunes y opuestos a Java llegaran a Android.
>
> Si tu ambiente de programación es Visual Studio lo único que debes instalar es el SDK de Android, la versión para Android de Mono y listo amigo, sigue desarrollando sin ningún inconveniente; además según Xamarin (la empresa creadora de Mono), trabajas con un lenguaje nativo para Android ya que no tiene un intérprete como lo tendría Basic4Android, y su aprendizaje es relativamente sencillo en un tiempo prudente si lo que buscas es hacer esa aplicación tienes ya en mente y no tienes tiempo de aprender un nuevo lenguaje.

En este caso la programación dad en Visual Studio con C# la convierte al lenguaje nativo, Java para Android y Objetive-C para iOS.

AppInventor.

No quieres Java, ni C#, ni C, ni .NET, ni VisualBasic, en resumidas cuentas, ningún programa de desarrollo tradicional. App Inventor es para ti.

Esta plataforma de desarrollo está basada en un lenguaje de desarrollo gráfico en donde no escribes ni una sola línea de código, tan solo arrastras bloques identificados con la acción que necesitas hacer y listo.

LiveCode9.0.

Plataforma en la que puedes programar tanto para Android, iOS, Windows, Linux, iPhone, iPad, Web y para Servidores con una sola plataforma de trabajo.

El lenguaje de programación que usa LiveCode se llama “Programación Orientada a Eventos”, y se basa en arrastrar elementos a un área de trabajo y programar los eventos que estén vinculados a este elemento; por ejemplo, si arrastras un botón al área de trabajo, el evento que tienes vinculados es el click sobre ese botón, o por ejemplo si arrastras una imagen, el evento vinculado es por ejemplo moverlo sobre la pantalla.

LiveCode se ejecuta en iOS, Android , Mac OS X , Windows 95 a través de Windows 7 , y varias variantes de Unix, incluyendo Linux, Solaris y BSD. Se puede utilizar para móviles, de escritorio y aplicaciones de servidor / CGI. Los iOS versión (iPhone y iPad) fue lanzado en diciembre de 2010.

PhoneGap o Apache Callbacks (Cordoba).

PhoneGap es un framework para el desarrollo de aplicaciones móviles producido por Nitobi, y comprado posteriormente por Adobe Systems. Principalmente, PhoneGap permite a los programadores desarrollar aplicaciones para dispositivos móviles utilizando herramientas genéricas tales como Java Script, HTML5 y CSS3.

Las aplicaciones resultantes son híbridas, es decir que no son realmente aplicaciones nativas al dispositivo (ya que el renderizado se realiza mediante vistas web y no con interfaces gráficas específicas de cada sistema), pero no se tratan tampoco de aplicaciones web (teniendo en cuenta que son aplicaciones que son empaquetadas para poder ser desplegadas en el dispositivo incluso trabajando con el API del sistema nativo).

Appcelerator Titanium.

El programa Appcelerator Titanium ha conseguido romper moldes en la programación de aplicaciones, al posibilitar la creación de aplicaciones nativas de gran calidad que funcionan en la gran mayoría de dispositivos móviles sin necesidad de crear versiones distintas. Por si esto fuera poco, el proceso de desarrollo de estas aplicaciones es tremendamente ágil e intuitivo y apenas se precisan conocimientos de programación.

Titanium es un programa desarrollado por la plataforma Appcelerator que sirve para crear aplicaciones móviles, al igual que otros softwares de función similar existentes en el mercado.

Lo que marca la diferencia y hace destacar a Titanium sobre otros programas es que posibilita la creación aplicaciones nativas adaptables a los sistemas operativos más comunes de smartphones y tablets.

Hasta ahora, únicamente las Web APP permitían un diseño adaptable a todos los dispositivos. Pero gracias a Titanium ya es

posible hacer lo mismo con las aplicaciones nativas, caracterizadas por su mayor calidad, precisión, rendimiento y acabado estético.

Las aplicaciones nativas están enfocadas y planificadas para ser usadas directamente en smartphones y tablets, sin tener que conectarse a través de un navegador. Su principal inconveniente era que tenían que programarse versiones diferentes para cada sistema operativo (iOS, Android, Blackberry), lo que hacía mucho más trabajoso, complejo y caro su desarrollo. Este hándicap ha sido eliminado de raíz gracias al programa Appcelerator Titanium.

La plataforma Appcelerator Titanium ofrece respuesta a todas estas necesidades con un servicio integral que incluye varias herramientas, funciones y servicios.

Las características más importantes de este programa se encuentran en sus funcionalidades o, dicho de otro modo, lo que permite hacer y la forma de hacerlo:

- Con Titanium se pueden crear aplicaciones nativas de gran calidad, válidas y adaptables en una única versión para los principales sistemas operativos de móviles y tablets.
- Se trata de una plataforma en constante desarrollo, lo que hace que sus posibilidades vayan en aumento. Por ejemplo, en sus inicios sólo permitía crear aplicaciones para los dispositivos de Apple y Android, extendiéndose después al sistema Blackberry.
- Utiliza el lenguaje de programación JavaScript, que es el propio de las páginas web, y posteriormente traduce al sistema nativo de cada aplicación.

- El interfaz o entorno de programación está basado en Eclipse (Aptana Studio) y es muy fácil, visual e intuitivo, por lo que no es necesarios tener conocimientos de programación.
- Tienen una extensibilidad ilimitada del propio framework Titanium, lo que permite que constantemente se añadan nuevos módulos con más posibilidades y recursos.
- Máxima interconexión con servicios en la nube, lográndose altísimos niveles de capacidad y rendimiento en la aplicación

In Design CS6

Es una herramienta que fue diseñada para poder publicar libros, revistas, comics, catálogos y que sus publicaciones puedan desarrollarse de una manera rápida, limpia y como se haya diseñado para dispositivos móviles con sistema operativo Android o iOS, pero no se desea utilizar los lenguajes nativos de estos sistemas.

El In Design CS6 nos da la oportunidad de poder realizar este tipo de aplicaciones, sin tener que escribir ninguna línea de código.

.

III. Desarrollo de aplicaciones móviles en Android.

3.1 Metodología de desarrollo y ejecución.

"Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo."[4]

El objetivo principal en el desarrollo de las aplicaciones de software, es la predicción de las actividades que se requieran para llevar a cabo ese proyecto, como por ejemplo:

- Predecir el costo.
- Mantener un nivel de calidad.
- Predecir el tiempo de desarrollo.

Cuáles son los objetivos de seguir una metodología en desarrollo de una aplicación de software:
Mejorar la calidad del producto en:

- Disminuir el número de fallos
- Bajar la severidad de los defectos
- Mejorar la reusabilidad
- Mejorar la estabilidad del desarrollo y el costo de mantenibilidad

[4] Josep Prieto Blázquez, Robert Ramírez Vique, Julián David Morillo Pozo, Marc Domingo Prieto. Tecnología y Desarrollo de dispositivos móviles. 2011.Universidad Oberta de Cataluya. Editorial : Eureca Media, SL

Mejorar la predictibilidad del proyecto en:

- La cantidad de trabajo que requiere.
- El tiempo de desarrollo que se necesita.

Generar la información adecuada para los diferentes responsables del proyecto de tal forma que se pueda hacer un seguimiento de este en forma efectiva.

En desarrollo de aplicaciones para dispositivos móviles hay que tomar en consideración el contexto en que nuestra aplicación va ser ejecutada, si esta va ejecutarse un móvil antiguo o en un móvil nuevo, un Smartphone o una Tablet, o todavía más en equipos como un televisor, un smart watch o posiblemente un smart card; se tendrá que revisar si la persistencia se dará en un servidor externos o el almacenamiento de datos se hará en nuestro mismo dispositivo.

Existen varios métodos para desarrollo de las aplicaciones móviles en este caso presentaremos uno propuesto por una investigación presentada en la revista de Tecnura en el año de 2014[5].
Esta tecnología está diseñada en cinco etapas etapa de análisis, donde se obtienen y clasifican los requerimientos y se personaliza el servicio; etapa de diseño, momento en el que se define el escenario tecnológico y se estructura la solución por medio de algún diagrama o esquema, integrando tiempos y recursos; etapa de desarrollo, cuando se implementa el diseño en un producto de software; etapa de prueba de funcionamiento, donde se emula y simula el producto ajustando

[5] Gasca Mantilla, Maira Cecilia; Camargo Ariza, Luis Leonardo; Medina Delgado, Byron. Metodología para el desarrollo de aplicaciones móviles. Tecnura, vol. 18, núm. 40, abril-junio, 2014, pp. 20-35. <http://www.redalyc.org/articulo.oa?id=257030546003>

detalles, se instala en equipos reales y se evalúa el rendimiento, y posteriormente se evalúa el potencial de éxito; y finalmente, en la etapa de entrega, se define el canal de distribución.

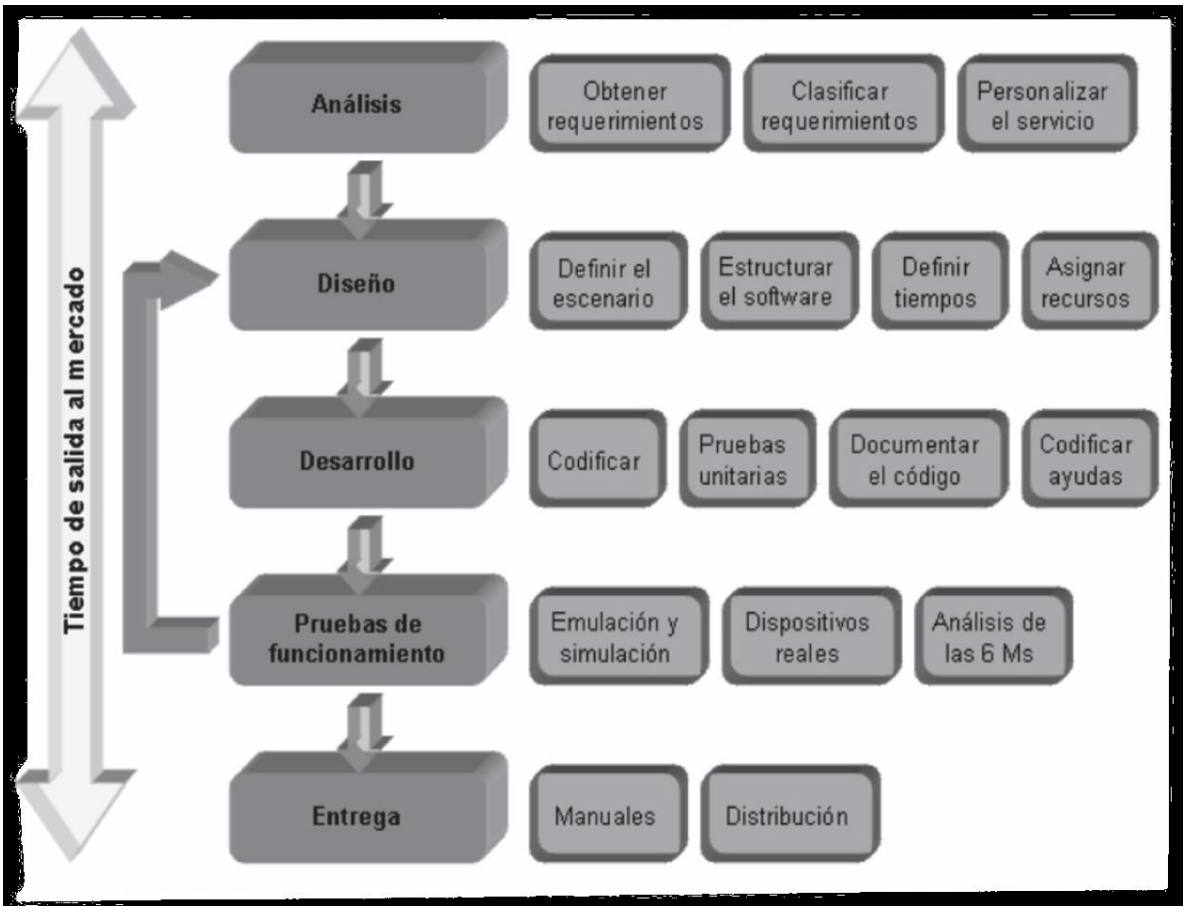


Figura 3.1.-Etapas de la metodología para el desarrollo de aplicaciones móviles

Análisis.

En esta fase se analizan las peticiones o requerimientos de las personas o entidad para la cual se desarrolla el servicio móvil “Cliente”, el

propósito es definir las características del mundo o entorno de la aplicación. Se realizan tres tareas: obtener requerimientos, clasificar los requerimientos.

Obtener requerimientos.
En este caso se realizan entrevistas con los usuarios potenciales de la aplicación, para determinar cuál es el problema a solucionar y las características que se deseen en la aplicación.

Clasificar los requerimientos.
Una vez obtenidos los requerimientos se procede a clasificarlos. Estos se pueden clasificar en entorno, mundo, funcionales y no funcionales.
El entorno se refiere a todo lo que rodea al servicio. Por ejemplo, las características técnicas del dispositivo móvil del cliente, el sistema operativo subyacente (móvil y servidores), la tecnología utilizada para la transferencia de información, el Sistema Manejador de Base de Datos, si se requiere, el formato de archivos y, otros módulos tecnológicos si se requieren.
El mundo es la forma en que el usuario interactúa con la aplicación. En este puno se revisará las interfaces gráficas con el usuario, los datos de salida y el formato con que le presentara al usuario, y todos los requerimientos que involucren la comunicación hombre-máquina.
Los requerimientos funcionales son todos aquellos que demandan una función dentro del sistema. Se deben definir claramente cada una de las tareas que debe realizar la aplicación.
Los requerimientos no funcionales son la estabilidad, la portabilidad, el rendimiento, el tiempo de salida al mercado y, el costo, entre otros.

Diseño.

En esta etapa se es donde se plasma las ideas mediante diagramas o esquemas, considerando la mejor alternativa al integrar aspectos técnicos, funcionales, sociales y económicos.

En esta etapa se realizan cuatro actividades principales: definir el escenario, estructurar el software, definir tiempos y asignar recursos.
Definir el escenario.
Las aplicaciones móviles se diseñan principalmente para ejecutarse diferentes escenarios, dependiendo del sistema de conexión y sincronización con el servidor o aplicación central.
Entre los diferentes escenarios tenemos: desconectados, semiconectado, conectado.

Estructurar el software.
En este punto se podrán utilizar algunos diagramas del Modelado del Lenguaje Unificado (UML Unified Modeling Language), según las necesidades de la aplicación, se modelara desde varias perspectivas, revisar Figura 3.2-

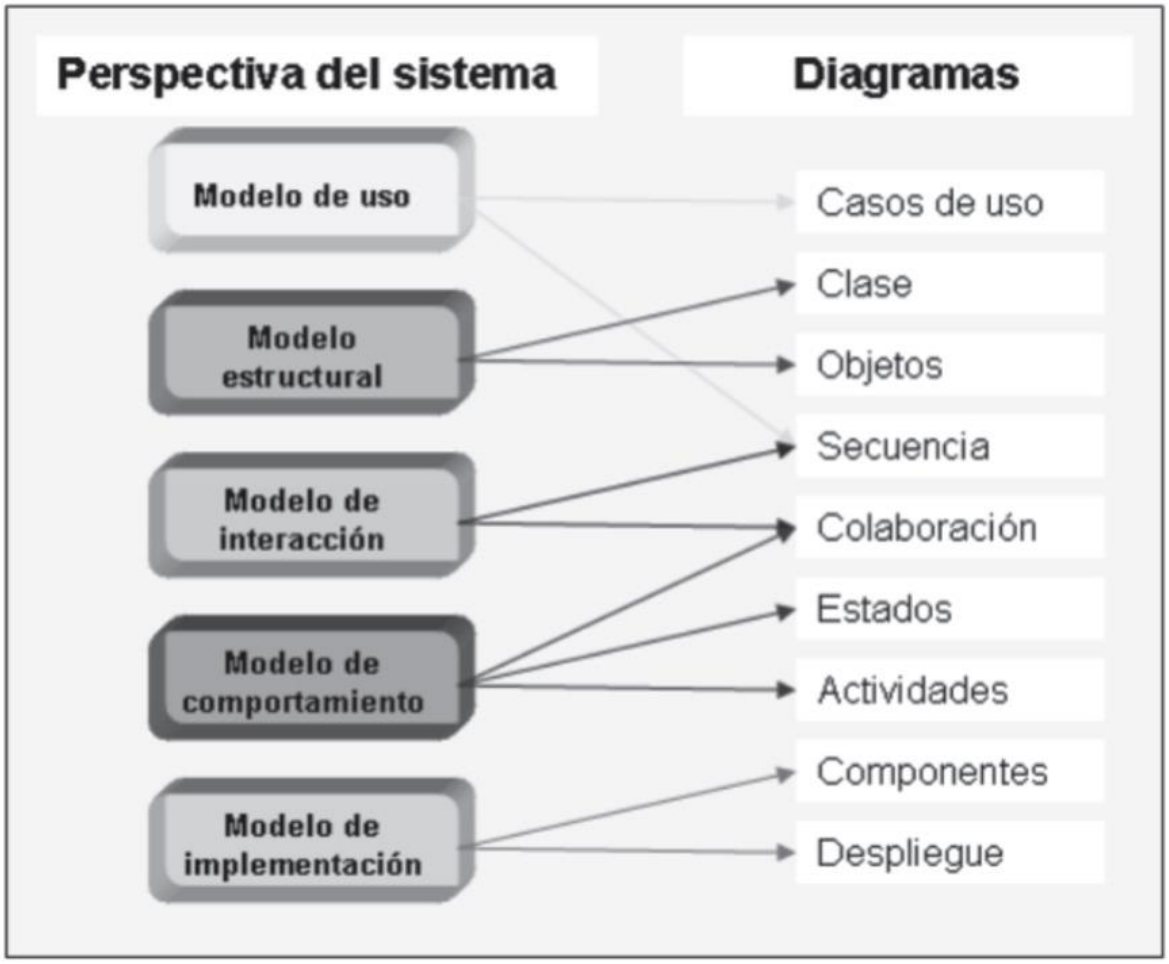


Figura 3.2.- Posibles diagramas para el desarrollo de aplicaciones para dispositivos móviles.

Se requiere que los requerimientos obtenidos en la etapa anterior queden plasmados, objetivamente en la aplicación que se va diseñar. Además es importante definir un patrón de diseño que flexibilice, module y reutilice lo desarrollado. Algunos patrones de diseño que se ajustan al desarrollo de aplicaciones móviles pueden ser el modelo vista controlador y el modelo de capas.

Definir tiempos.

Definir tiempos: se establecen los plazos para cada una de las actividades restantes, con el objetivo de terminar la aplicación a tiempo para su salida al mercado.

Asignar recursos.

Se asignan los recursos para realizar cada actividad y alcanzar los objetivos propuestos, se deben considerar recursos humanos, financieros y tecnológicos.

Desarrollo.
El objetivo de esta fase es implementar el diseño en un producto de software.
En esta etapa las actividades más importantes son:
Codificar.
Se escribe en el lenguaje de programación seleccionado, cada una de las partes definidas en la etapa de diseño.
Pruebas unitarias.
Se verifica el funcionamiento de la aplicación. En primer lugar, se comprueba la correcta operación de cada elemento desarrollado objeto, clase, actividad, documento, entre otros; en forma individual; posteriormente, se pone en funcionamiento el conjunto de elementos, comprobando la interrelación entre ellos.
Documentar el código.
Conforme se codifica y se prueba cada elemento de nuestro software se va redacto la documentación de lo desarrollado.

Pruebas de funcionamiento.
El objetivo de esta fase es probar el funcionamiento de nuestra aplicación en diferentes escenarios y condiciones; para esto se deben de realizar estas tareas:

Emulación y simulación.
Se realizan pruebas en condiciones similares en las que el dispositivo estaría funcionando, revisando todas las funcionalidades e introduciendo datos, inclusive erróneos, para medir la funcionalidad y la robustez del software.

Dispositivos reales.
En esta etapa se probara nuestro software en teléfonos reales, para medir el rendimiento y desempeño de la aplicación. En este caso deberemos de observar si en tiempo real la aplicación funciona adecuadamente y el cliente no tiene observaciones al trabajo desarrollado.

Análisis de las 6M's.
Una forma de valorar la calidad de nuestra aplicación, nos sugiere buscar un grupo de expertos desarrolladores en dispositivos móvil que utilice el método de evaluación de las 6M's y califiquen la presencia de los seis atributos en aplicación desarrollada.

Atributo	Definición	Calificación	Justificación
Momento	Un servicio que cuente con este atributo debe de estar disponible en cualquier instante de tiempo en que el usuario desee usar dicho servicio		
Movilidad	Un servicio móvil debe de ser "móvil" por naturaleza la ubicación debe ser una parte integral del servicio.		

Dinero	Como cualquier acción comercial, un servicio móvil tiene un fin lucrativo, ya sea para el operador, para el proveedor del servicio o para el usuario.		
Yo	Se refiere el nivel de personalización del servicio.		
Máquina	La tecnología (terminal o redes) siempre es el factor que posibilita o limita, atributo máquina busca añadir potencia a los dispositivos de última generación que cada vez tienen mayores prestaciones a nivel hardware y software.		
Multiusuario	Busca extenderse dentro de la comunidad, que el servicio sea interactivo y que pueda utilizarse por múltiples usuarios d manera simultánea.		

Tabla3.1.-.-Evaluación de las 6M´s de una aplicación móvil

Entrega.
Terminadas las pruebas, atendidos los requerimientos del cliente, se da por terminada la aplicación y se procede a la entrega del ejecutable, el código fuente, la documentación y el manual del sistema.

Si bien es cierto que muchas aplicaciones están diseñadas para un cliente específico, también existen aplicaciones que son diseñadas para resolver un problema en general y que se distribuirá en un canal de comercialización, estos canales de distribución actualmente son las llamadas tiendas de aplicaciones como por ejemplo Apple Store o Google play.

3.2 Uso de formularios Web móvil.

Las webs móviles son aquellas webs que ya existen actualmente y que son adaptadas específicamente para ser visualizadas en los dispositivos móviles. Adaptan la estructura de la información a las capacidades del dispositivo, de manera que no saturan a los usuarios y se pueden usar correctamente desde estos dispositivos.
Las ventajas de las webs móviles son las siguientes:
• Fácil implementación, testeo y actualización. Incluso se puede realizar gran parte del desarrollo sin necesidad de utilizar dispositivos móviles ni emuladores, hasta llegar a las fases finales del desarrollo.
• Lenguaje conocido y estándar. Los lenguajes de marcas son muy conocidos hoy en día por la mayoría de los desarrolladores, y en la mayoría de los casos se trata de subconjuntos de lenguajes conocidos.
• Pueden soportar múltiples dispositivos con un único código fuente. Para soportar fragmentación entre dispositivos, es necesario utilizar técnicas especiales (como el WURLF).
WURLF es un repositorio de información que identifica, a partir de la metainformación de una petición web de un dispositivo móvil, sus capacidades y limitaciones.

3.3 Uso de controles.

Como se ha revisado en los capítulos anteriores el Sistema Operativo Android, es el sistema más demando por los proveedores de dispositivos móviles, es por eso que los ejemplos que se verán en este documento estarán basados sobre este sistema.
La IDE en que se hará toda la programación es Android Studio 3.1.4; también es necesario tener el lenguaje Java instalado, dado que es el lenguaje nativo para el desarrollo de estas aplicaciones.

Para conoce a dtalle la instalación de Java y Android Studio referirse al Anexo 1.

Android es un sistema operativo para dispositivos móviles que se basa en Kernel del Sistema Linux.
Este sistema fue desarrollado originalmente por la empresa Android Ic.En el año de 2005 como parte de su estrategia comercial, Google compra Android, y se hizo cargo del desarrollo de este sistema y de los desarrolladores del mismo.
Android es un software de código abierto y gratuito; por lo que la mayor parte del código está protegido bajo la licencia de Apache de código abierto, lo que significa que cualquier persona puede utilizarlo siempre utilice el código completo.
Es importante resaltar que los vendedores tienen la posibilidad de agregarle librerías propietarias para enriquecer el software, así personalizar el Android de otros vendedores.
Al momento de aparecer iOS de iPhone de Apple, que fue un producto muy exitoso que vino a revolucionar el mercado de los teléfonos inteligentes; muchas compañías que tenían un sistema operativo propietario, tuvieron que emigrar a un sistema operativo que le diera un plus a sus marcas, ellas vieron en Android una solución, para continuar diseñando su propio hardware y usando este sistema operativo como base.

Arquitectura de Android.
El sistema operativo Android se divido en cinco secciones dentro de las cuatro secciones principales.

Kernel de Linux: este es el kernel en el que se basa Android. Esta capa contiene todos los controladores de dispositivo para los diversos componentes de hardware de un dispositivo Android.

Bibliotecas: contienen todo el código que proporciona las funciones principales de un sistema operativo Android.

Por ejemplo, la biblioteca SQLite proporciona soporte de base de datos para que una aplicación pueda usarlo para almacenamiento de datos. La biblioteca WebKit proporciona funcionalidades para la navegación web.

R Android runtime: en la misma capa que las bibliotecas, el tiempo de ejecución de Android proporciona un conjunto de bibliotecas centrales que permiten a los desarrolladores escribir aplicaciones de Android utilizando la programación de Java. El tiempo de ejecución de Android también incluye la máquina virtual Dalvik, que habilita cada Aplicación de Android para ejecutar en su propio proceso. Dalvik es un especialista máquina virtual diseñada específicamente para Android.

Aplication framework: expone las diversas capacidades del sistema operativo Android a los desarrolladores para que puedan hacer uso de ellos en sus aplicaciones.

Aplicaciones: en esta capa superior, encontrará las aplicaciones que se encuentran nativas en el dispositivo Android (como Teléfono, Contactos, Navegador, etc.), así como las aplicaciones que descargan e instalan desde Play Store. Cualquier aplicación que escriba se encuentra en esta capa.

Creando nuestra primera aplicación.

Para esto tenemos debemos de tener configurado e instalado nuestro programa Android Studio.

1.-Para crear un proyecto utilizando Android Studio será File ->New –>New Project, el cual aparecerá la siguiente figura.3.3

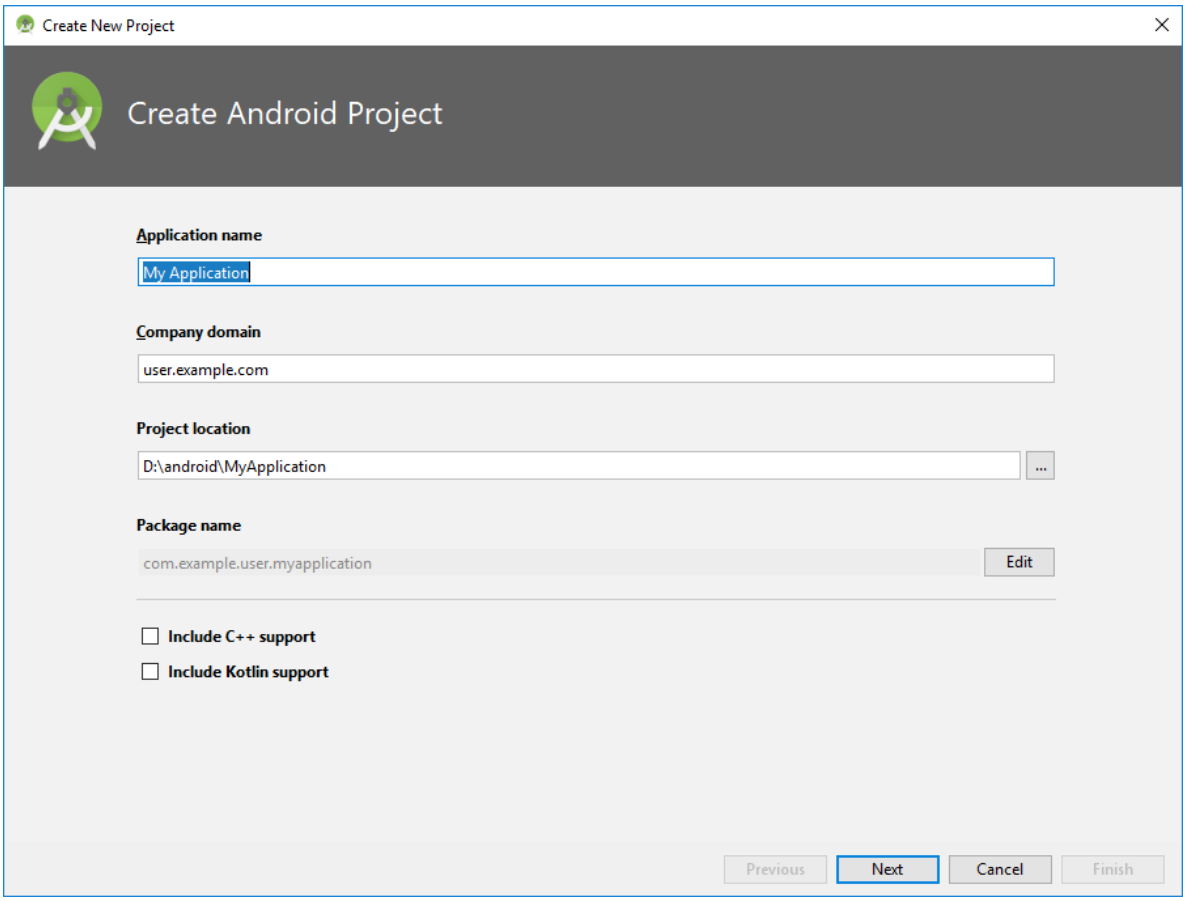


Figura 3.3 Creación del u proyecto en Java

2.-Dar el nombre al proyecto en este caso Ejemplo, oprimimos el botón Next.

3.-Seleccionamos el dispositivo donde se aplicara esta aplicación, para esto es Smartphone y tablets y el mínimo sdk, figura 3.4

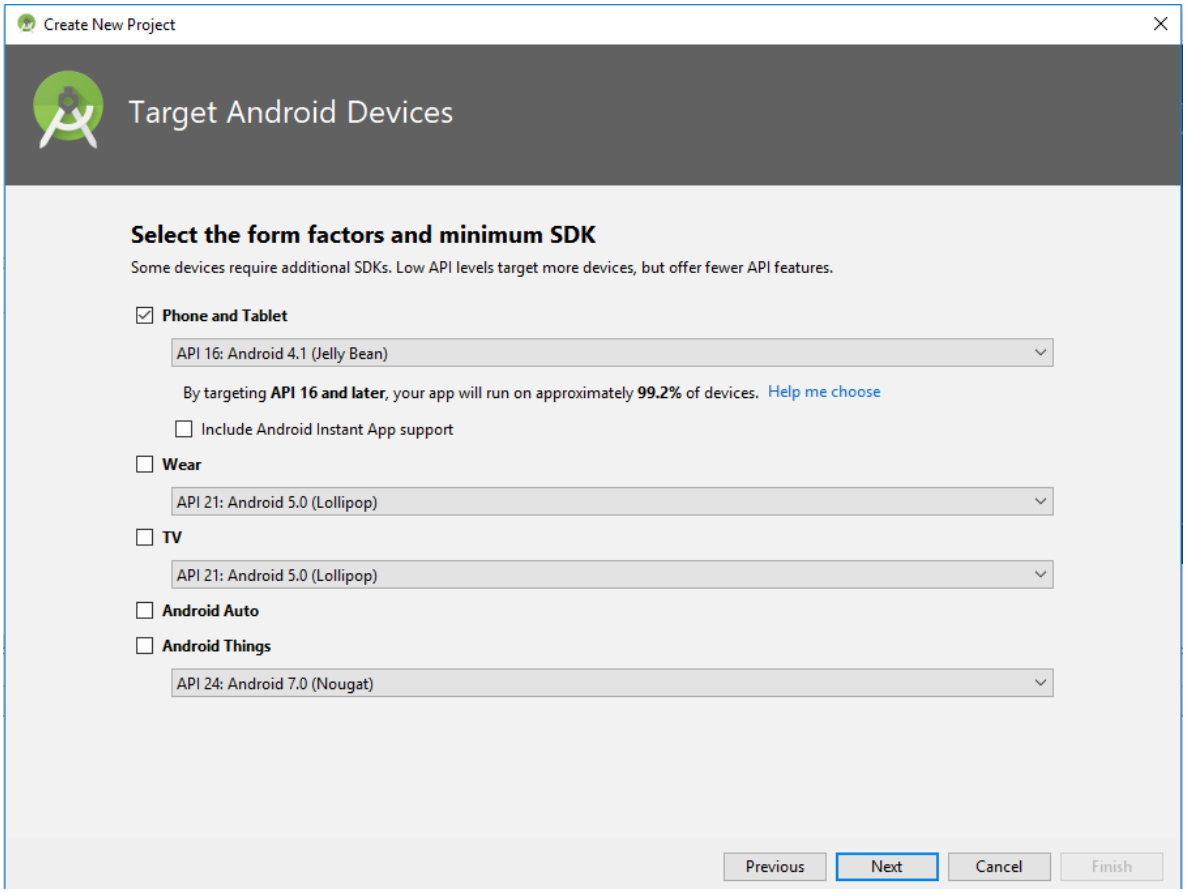


Figura 3.4 Selección de dispositivo y el mínimo SDK.

4.-Añadimos una Actividad al dispositivo en la que vamos a trabajar, figura 3.5.

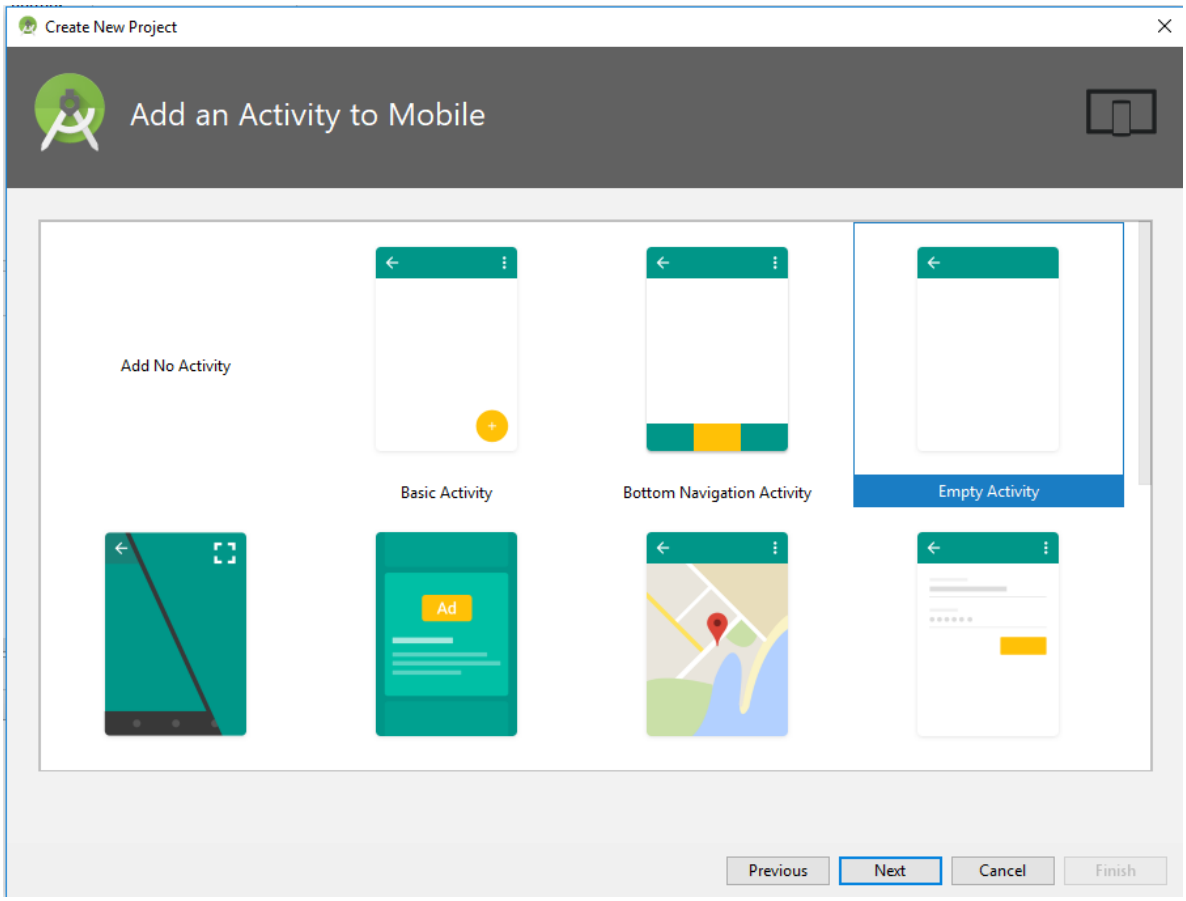


Figura 3.5 Seleccionar Actividad.

5.-Creamos la Actividad asignándole un nombre Figura 3.6. y oprimimos el Finish

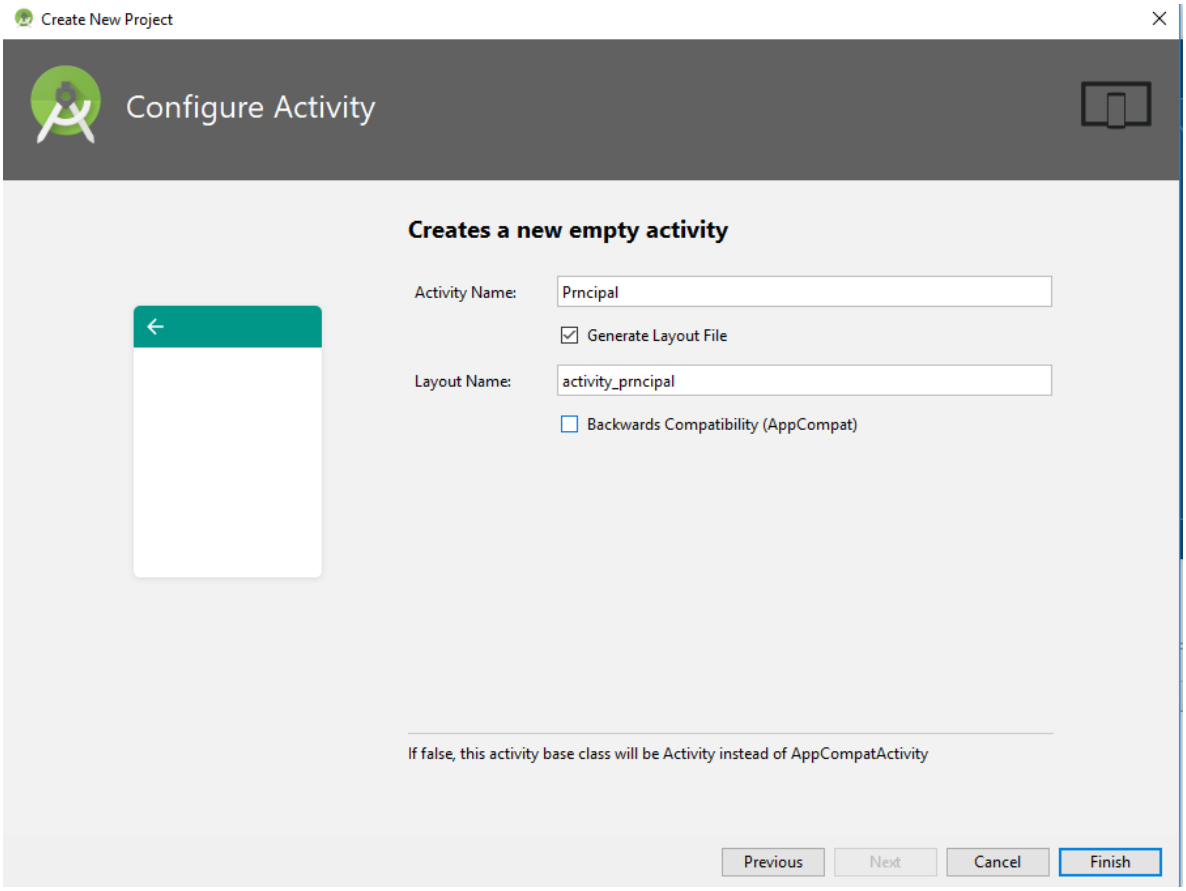


Figura 3.6 Signando un nombre a la actividad.

6.-En este momento tendremos nuestra interface de desarrollo 2 los archivos principales: Principal.java y activity_principal.xml.

El archivo activity_principal.xml será el archivo donde con los comando específicos, diseñaremos la interface que vera nuestro usuario (UI); en el archivo Principal.java tendremos el programa en java, donde se programara la lógica de nuestra aplicación.

Activity_principal.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#f9ae99"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/texto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="25dp"
        android:text="Hola Mundo"
        android:textSize="24sp"
        android:textColor="#2b2a2a"/>
</RelativeLayout>
```

Principal.java.

```
package com.example.dtrujill.ejemplo;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
```

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

7.- ya podremos ejecutar nuestra aplicación en el emulador que se ha instalado previamente. Anexo 1.

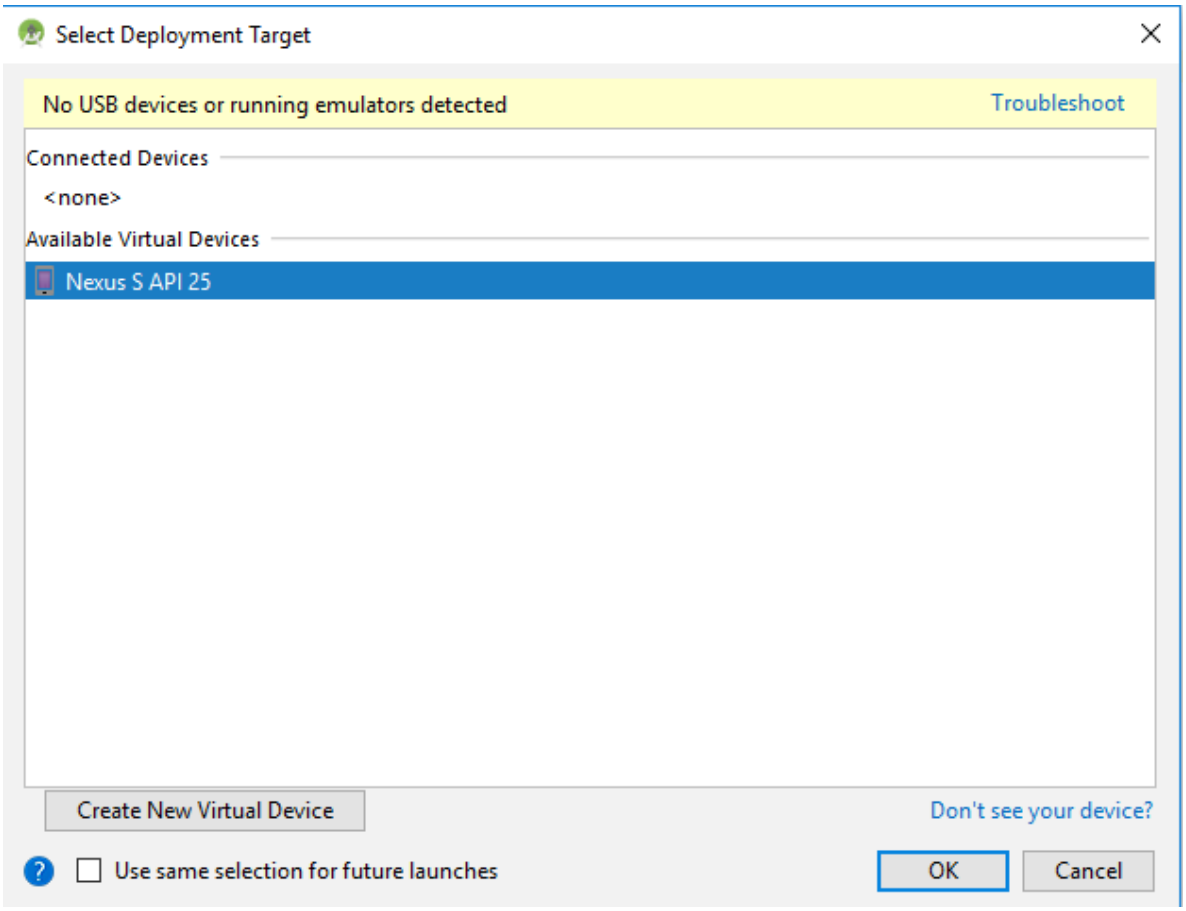


Figura 3.7 Seleccionar el emulador donde se ejecuta nuestra aplicación.

Seleccionamos el emulador sobre el que se va ejecutar nuestra aplicación, para este caso seleccionamos será Nexus 5 API 25, según Figura 3.8.

Figura 3.8 Ejecutando la aplicación.

Una vez creado nuestro proyecto, se crean una configuración de directorios que tienen una funcionalidad específica como se muestra en la Figura 3.9.

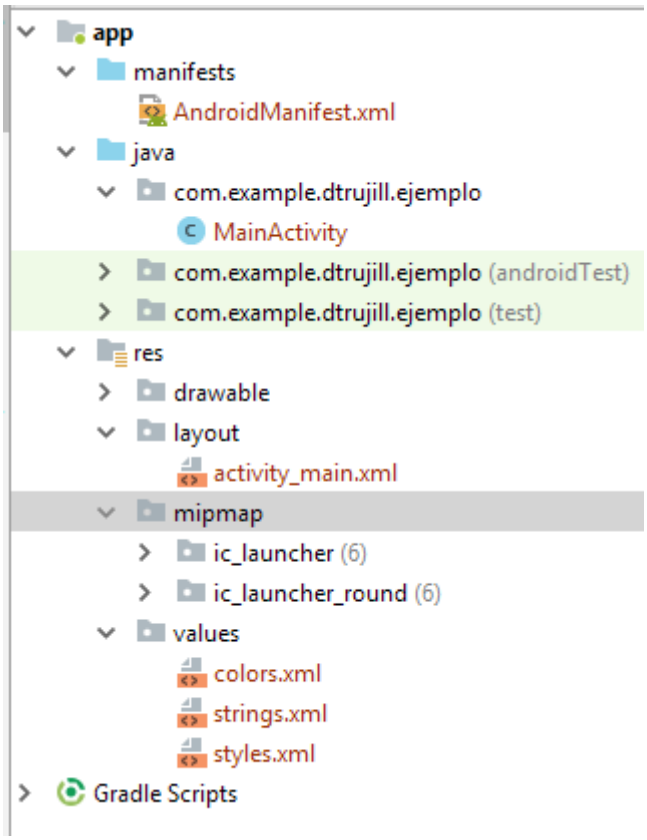


Figura 3.9 Directorios creados por el proyecto Ejemplo.

Los directorios y archivos creados son los siguientes:

- El directorio manifest con el archivo andoidManifest.xml, en este archivo se encuentra la configuración que necita nuestra aplicación. Aquí se especifican los permisos que requiere la aplicación para un adecuado funcionamiento (Internet, Bluetooh,).
- El directorio java en este se encuentra los programas en java (MainActivity) de nuestro proyecto, estos programas se encuentran bajo el package en este caso com.example.dtrujill.ejemplo.
- El directorio res es donde se encuentran los recursos utilizados por nuestra aplicación, dentro de ella se encuentran el

subdirectorio drawable donde tenemos las imágenes que utiliza el proyecto; el subdirectorio layout donde tenemos un archivo xml que define la Interface del Usuario (UI) como lo veremos más adelante. El directorio mimap donde se encuentran las imágenes con diferentes densidades, que representan la aplicación cuando esta se instala en nuestro dispositivo. El directorio values donde se tiene 3 diferentes archivos xml como son: colors.xml donde se declara la definición de los colores que vamos utilizar en nuestra aplicación (<color name="colortecnolo">#3F51B5</color>; al momento de utilizarlo en la definición de nuestra UI lo podemos hacer de la siguiente forma “android:textColor=”@color/colortecnolo”. El otro archivo es strings.xml que es donde se define las cuerdas de caracteres que se utilizan en nuestra aplicación (<string name="nombre">Tecnológico de México</string> y al momento de utilizarlo lo hacemos de la siguiente manera android:text=”@string/nombre. El archivo styles.xml, podemos declarar un estilo especifico que utilizaremos l momento de diseñar nuestra UI.

Componentes de una aplicación Android.

Activity.

Es el componente principal de la interface con el usuario en una aplicación de Android.

View.

Las vistas son los componentes con los que se construyen las interfaz gráfica de nuestra aplicación, existen dos tipos de views; los contenedores o View´s Groups y lo widgets.

Los widgets son vistas que nos una apariencia en la pantalla como puede ser etiquetas, botones, cuadro de texto, etc., y estas derivan de la clase android.view.View. En cambio los View´s Group, nos proporcionan los layouts que nos dan la apariencia y la secuencia de nuestros widgets, un ejemplo pueden ser los LinearLayout, RelativeLayout, ConstraintLayout, etc.

Service

Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (p.ej. actividades) si se necesita en algún momento la interacción con del usuario

Content Provider

Un content provider es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación.

Broadcats Receiver

Un broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: "Batería baja", "SMS recibido", "Tarjeta SD insertada") o por otras aplicaciones.

Intent

Un intent es el elemento básico de comunicación entre los distintos componentes Android que hemos descrito anteriormente. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones

Ahora se verán los principales componentes de una interface gráfica, como hemos mencionado una interface gráfica está compuesta por los contenedores o View´s Grup y los widgets.

Android puede manejar los siguientes View´s Group:

- LinearLayout
- RelativeLayout
- FrameLayout
- TableLayout
- ConstraintLayout

LinearLayout.

Este layout acomoda los widgets que contenga en forma horizontal o vertical dependiendo del atributo android:orientation: también maneja los atributos android:layout_width y atributo android:layout_heigth, los cuales podrán tener dos valores “match_parent(para que el control tomo la dimensión de su layout contendor) o wrap_content(para que el control tome las dimensiones de su contenido); además del atributo android:layout:weigth nos permite darle proporcionalidad a los elementos que contenga el contenedor, como lo veremos en el siguiente ejemplo:

Archivo activity_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".Principal">
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Instituto Tecnológico de Hermosillo" />

    <EditText
        android:layout_width="276dp"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:hint="Nombre del alumno"
        android:inputType="text" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:text="Oprimir" />
</LinearLayout>
```

Como se muestra el la Figura 3.8, podemos observar que los dos widgets(TextView y EditText y Button) que se tienen en el Layout se encuentran ocupando toda la interface,en una proporción del doble debido al atributo android:layout_weigth.

Tabla 3.1 Atributos más comunes utilizados widgets o View´sGroup

Atributo	Descripción
layout_width	Especifica el ancho del widget o View´sGroup

layout_heigth	Especifica la altura del widget o View´sGroup
Layout_x	Especifica la coordenada x del widget o View´sGroup
Layout_y	Especifica la ccordena y del widget o View´sGroup.

.

Figura 3.10 Corrida de la Aplicación con LinearLayout

RelativeLayout

Este layout permite especificar la posición de cada elemento de forma relativa a su elemento padre o a cualquier otro elemento incluido en el propio layout.

Activity_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Principal">

    <TextView
        android:id="@+id/texto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="26dp"
        android:textColor="#547af8"
        android:textSize="19sp"
        android:text="@string/encabezado"/>

    <EditText
        android:id="@+id/edtexto"
        android:layout_width="289dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/texto"
```

```
        android:layout_marginStart="25dp"
        android:layout_marginTop="30dp"
        android:hint="Nombre Alumno"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/edtexto"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:text="Oprimir"/>
</RelativeLayout>
```

Como se demuestra en la Figura 3.9 tendremos que los widgets son colocados en referencia a su contenedor y a otros widgets que se encuentran en la interface; observemos que con el atributo android:layout_below=”@id=texto, indicamos que este widget(EdiText) estará por debajo de widget TextView. El atributo android:layout_marginStart=”25dp” estará a una distancia de 25 dp del inicio del lado izquierdo de su contenedor.

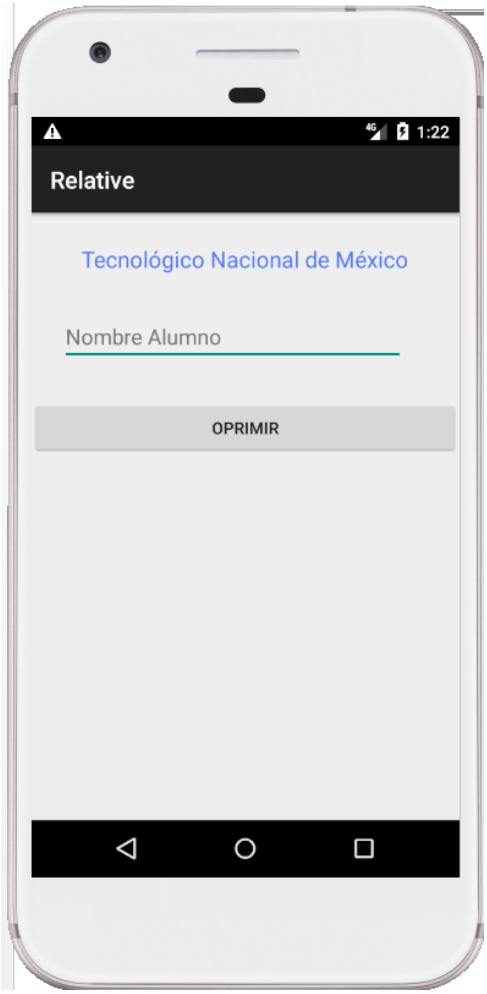


Figura 3.11 ejecución de layout RelativeLayout.

Existe un sinfín de atributos que podemos utilizar para poder diseñar una interface gráfica de dentro de un contenedor del tipo RelativeLayout como las que se indican:

Tabla 3.2 Atributos utilizados en layout RelativeLayout

Tipo	Propiedades
Posición relativa a otro control	android:layout_above android:layout_below

	android:layout_toLeftOf android:layout_toRightOf android:layout_alignLeft android:layout_alignRight android:layout_alignTop android:layout_alignBottom android:layout_alignBaseline
Posición relativa al layout padre	android:layout_alignParentLeft android:layout_alignParentRight android:layout_alignParentTop android:layout_alignParentBottom android:layout_centerHorizontal android:layout_centerVertical android:layout_centerInParent
Opciones de margen (también disponibles para el resto de layouts)	android:layout_margin android:layout_marginBottom android:layout_marginTop android:layout_marginLeft android:layout_marginRight
Opciones de espaciado o padding (también disponibles para el resto de layouts)	android:padding android:paddingBottom android:paddingTop android:paddingLeft android:paddingRight

FrameLayout

Éste es el más simple de todos los layouts de Android. Un FrameLayout coloca todos sus controles hijos alineados con su esquina superior izquierda, de forma que cada control quedará oculto por el control

siguiente (a menos que éste último tenga transparencia). Por ello, suele utilizarse para mostrar un único control en su interior.

Activity_principal.xml

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</FrameLayout>
```

Figura 3.12 Ejecución del layout FrameLayout.

TableLayout

Este layout nos permite distribuir los elementos hijos en forma tabular, definiendo las filas y columnas necesarias, y la posición de cada componente dentro de la tabla.
La estructura de la tabla está definida por n objeto denominado TableRow, que nos indica las filas que contendrá nuestra tabla, cada fila tendrá un número determinado de columnas que serán dados por los elementos ya sean widgets o inclusive por contenedores. De esta forma, el número final de filas de la tabla se corresponderá con el número de elementos TableRow insertados, y el número total de columnas quedará determinado por el número de componentes de la fila que más componentes contenga.
Por norma general, el ancho de cada columna se corresponderá con el ancho del mayor componente de dicha columna, pero existen una serie de propiedades que nos ayudarán a modificar este comportamiento:

android:stretchColumns. Indicará las columnas que pueden expandir para absorber el espacio libre dejado por las demás columnas a la derecha de la pantalla.

android:shrinkColumns. Indicará las columnas que se pueden contraer para dejar espacio al resto de columnas que se puedan salir por la derecha de la pantalla.

android:collapseColumns. Indicará las columnas de la tabla que se quieren ocultar completamente.
Todas estas propiedades del TableLayout pueden recibir una lista de índices de columnas separados por comas (ejemplo: android:stretchColumns="1, 2, 3") o un asterisco para indicar que debe aplicar a todas las columnas (ejemplo: android:stretchColumns="*").

Otra característica importante es la posibilidad de que una celda determinada pueda ocupar el espacio de varias columnas de la tabla (análogo al atributo colspan de HTML). Esto se indicará mediante la propiedad android:layout_span del componente concreto que deberá tomar dicho espacio.

Acivity_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Principal">

    <TableRow>
        <TextView android:text="Celda 1.1" />
        <TextView android:text="Celda 1.2" />
        <TextView android:text="Celda 1.3" />
```

```
    </TableRow>

    <TableRow>
        <TextView android:text="Celda 2.1" />
        <TextView android:text="Celda 2.2" />
        <TextView android:text="Celda 2.3" />
    </TableRow>

    <TableRow>
        <TextView android:text="Celda 3.1"
            android:layout_span="2" />
        <TextView android:text="Celda 3.2" />
    </TableRow>
    </TableLayout>
```

El layout activity_principal.xml se ejecuta de la siguiente manera:

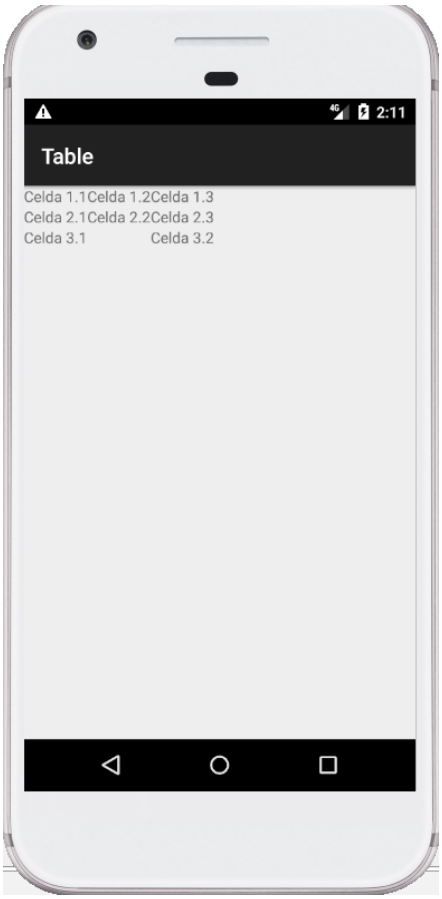


Figura 3.13 ejecución del layout TableLayout.

GridLayout

Las características son similares al TableLayout, ya que se utiliza igualmente para distribuir los diferentes elementos de la interfaz de forma tabular, distribuidos en filas y columnas. La diferencia entre ellos estriba en la forma que tiene el GridLayout de colocar y distribuir sus elementos hijos en el espacio disponible. En este caso, a diferencia del TableLayout indicaremos el número de filas y columnas como propiedades del layout, mediante android:rowCount y android:columnCount. Con estos datos ya no es necesario ningún tipo

de elemento para indicar las filas, como el elemento TableRow del TableLayout, sino que los diferentes elementos hijos se irán colocando ordenadamente por filas o columnas (dependiendo de la propiedad android:orientation) hasta completar el número de filas o columnas indicadas en los atributos anteriores. Adicionalmente, igual que en el caso anterior, también tendremos disponibles las propiedades android:layout_rowSpan y android:layout_columnSpan para conseguir que una celda ocupe el lugar de varias filas o columnas.

Existe también una forma de indicar de forma explícita la fila y columna que debe ocupar un determinado elemento hijo contenido en el GridLayout, y se consigue utilizando los atributos android:layout_ row y android:layout_column. De cualquier forma, salvo para configuraciones complejas del grid no suele ser necesario utilizar estas propiedades.

Activity_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Principal"
    android:rowCount="3"
    android:columnCount="3"
    android:orientation="horizontal" >

    <TextView android:text="Celda 1.1" />
    <TextView android:text="Celda 1.2" />
    <TextView android:text="Celda 1.3" />
    <TextView android:text="Celda 2.1" />
```

```
    <TextView android:text="Celda 2.2" />
    <TextView android:text="Celda 2.3" />
    <TextView android:text="Celda 3.1"
        android:layout_columnSpan="2" />
    <TextView android:text="Celda 3.2" />
</GridLayout>
```

La ejecución se observa en la Figura 3.14.

Figura 3.14 Ejecución de la actividad con GridLayout.

ConstraintLayout.

El editor de diseño de Android Studio con ConstraintLayout, un nuevo tipo de diseño disponible en el repositorio de soporte de Android para crear diseños flexibles y eficientes. El Editor de diseño usa restricciones para determinar la posición de un elemento de UI dentro del diseño. Una restricción representa una conexión o alineación con otra vista, el diseño principal o una línea invisible. Como se visualiza en la Figura 3.12

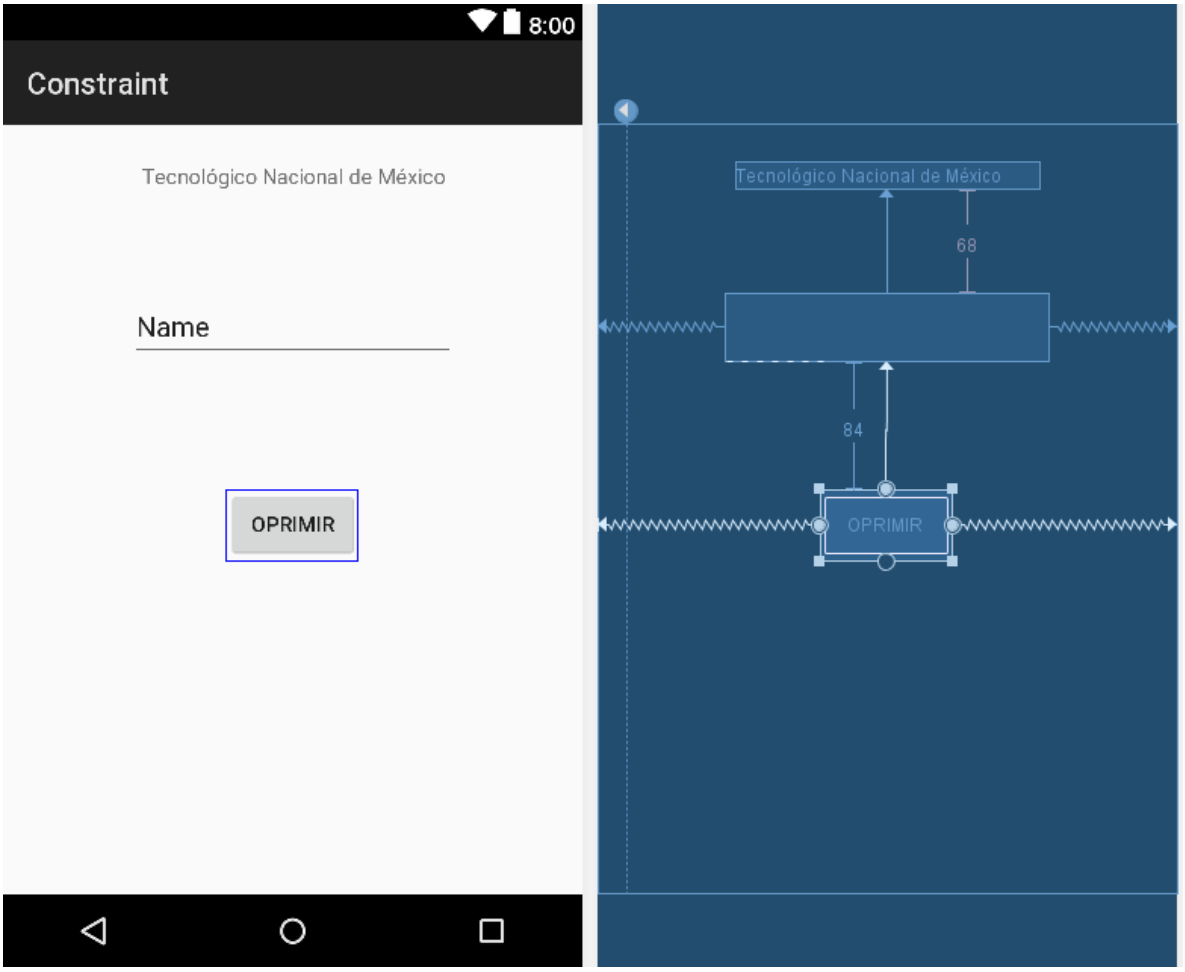


Figura 3.15 Editor de ConstraintLayout

Como se ve en el editor de ConstraintLayout, podemos arrastrar los widgets que deseamos y poner las restricciones necesarias para el diseño de nuestra Interface gráfica.
El archivo activity_principal.xml se verá así.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Principal"
    tools:layout_editor_absoluteY="81dp">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="88dp"
        android:text="Tecnológico Nacional de México"
        app:layout_constraintStart_toStartOf="parent"
        tools:layout_editor_absoluteY="31dp" />
    <EditText
        android:id="@+id/editText"
        android:layout_width="215dp"
        android:layout_height="46dp"
        android:layout_marginTop="68dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="84dp"
        android:text="Oprimir"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editText" />

    <android.support.constraint.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_begin="20dp" />
</android.support.constraint.ConstraintLayout>
```

Y en la ejecución de nuestra aplicación con ConstraintLayout tenemos:

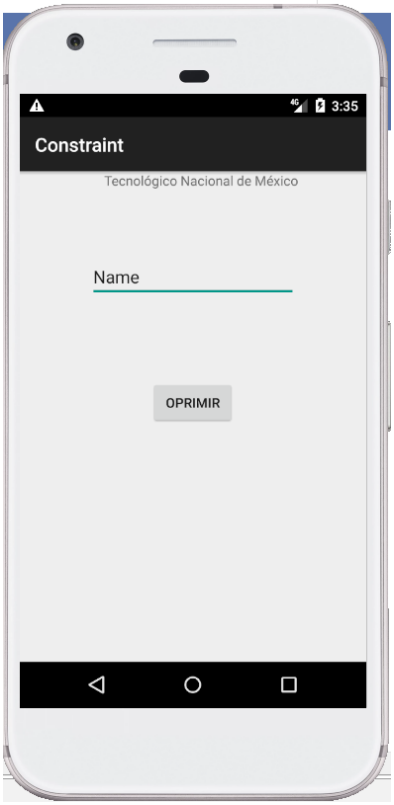


Figura 3.16 Ejecución de la aplicación con ConstraintLayout.

Widgets

Como vimos anteriormente sobre los diferentes diseños, que podemos utilizar para posesionar los diferentes widgets en una interface (Actividad).

Veremos los diferentes widgets que se pueden utilizar para diseñar una interfaz de usuario, de una aplicación.

Veremos los siguientes widgets.

➤ Vistas básicas: vistas de uso común como TextView, EditText y Vistas de botones

➤ Vistas de selector: vistas que permiten a los usuarios seleccionar de una lista, como TimePicker y vistas de DatePicker

➤ Vistas de lista: vistas que muestran una larga lista de elementos, como ListView y Vistas de SpinnerView
Fragmentos especializados - Fragmentos especiales que realizan funciones específicas.

Vistas Básicas.
Estos son los widgets más básicos con los que podemos desarrollar una UI de una aplicación.Android.

➤ TextView
➤ EditText
➤ Button
➤ ImageButton
➤ CheckBox
➤ ToggleButton
➤ RadioButton
➤ RadioGroup

Estas vistas básicas son capaces de desplegar una etiqueta, asi como realizar unas selecciones o bien desplegar un evento.

TextView.
Cuando se crea un proyecto en Android Studio, también se crea un archivo denominado activity_principal.xml, el cual contiene el layout y un widget como TextView.

Activity_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".Principal">

<TextView
    android:id="@+id/texto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="73dp"
    android:layout_marginTop="35dp"
    android:text="Tecnológico Nacional de México!" />

<TextView
    android:id=”@+id/texto1”
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="98dp”
    android:text="Tecnológico de Hermosillo" />
```

La ejecución de este programa será como se muestra en la Figura 3.17

Figura 3.17 ejecución de vistas básicas.

Además de la vista TextView, que probablemente usará con más frecuencia, hay algunos otros widgets básicos que encontrará con frecuencia.

➤ Button — Representa un widget push-button.

➤ ImageButton — Similar a Button excepto que también despliega una imagen.

➤ EditText — Es una subclase de TextView view que permite al usuario introducir y editar un texto contenido por el widget.

➤ CheckBox — Un especial tipo de Button, que tiene dos estados: checked o no checked

➤ RadioGroup and RadioButton — The RadioButton tiene dos estados esta checado o no checado. El RadioGroup es usado para agrupar uno o más RadioButton permitiendo que solo un RadioButton pueda ser marcado, entro del RadioGrup.

➤ ToggleButton muestra los estados marcados / no marcados con un indicador de luz.
Nuestro archivo de nuestra UI será de la siguiente forma:

Activity_principal.xml-

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".Principal">

    <Button
        android:id="@+id/btnSave"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="save" />

    <Button
        android:id="@+id/btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open" />

    <ImageButton
        android:id="@+id/btnImg1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="OprimirImagen"
```

```
    android:src="@drawable/ith" />

<EditText
    android:id="@+id/txtName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<CheckBox
    android:id="@+id/chkAutosave"
    android:layout_width="match_parent"
    android:onClick="onboxclicked"
    android:layout_height="wrap_content"
    android:text="Autosave" />

<CheckBox
    android:id="@+id/star"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onboxclicked"
    android:text="AutoStar"/>
<RadioGroup
    android:id="@+id/rdbGp1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
    <RadioButton
        android:id="@+id/rdb1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onradiobutton"
        android:text="Option 1" />
    <RadioButton
        android:id="@+id/rdb2"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onradiobutton"
        android:text="Option 2" />
</RadioGroup>

<ToggleButton
    android:id="@+id/toggle1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

La ejecución de la aplicación está dada en la Figura 3.18

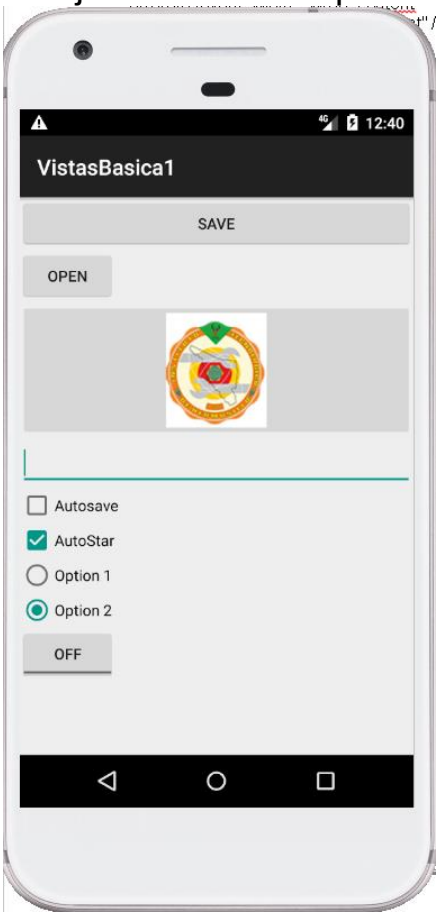


Figura 3.18.Ejecución de las Vistas

Nuestro archivo activity_principal.xml, como lo habíamos indicado es donde se diseña la interfaz de usuario en este caso, tenemos que se encuentra dentro de un contenedor LInearLayout, mismo que le indicamos que los widgets estarán dispuestos de manera vertical con el atributo android:orientation=”vertical”. El primer widget es de tipo Button como vemos tenemos el atributo android:layout_width="match_parent", lo que hace que el botón se distribuya a lo ancho de contenedor. El siguiente Button tiene los atributos android:layout_width="wrap_content"
android:layout_height="wrap_content, lo que hace que el botón se ajuste a las dimensiones del texto que se encuentra dentro botón.
El ImagenButton despliega botón con una imagen. Dicha imagen tendrá que estar almacenada en el directorio drawable y poner el atributo android:src=”@drawable/ith” del widget ImagenButton.
El EditText es un widget que nos ayuda a introducir información, el atributo android:layout_heigth o android:layout_width, nos da la referencia de lo ancho o alto de la información que está introduciendo el usuario.
CheckBox nos muestra una casilla de verificación que el usuario puede tocar para marcar o desmarcar; en este caso es posible marcar una o varias opciones.
En este caso el RadioGroup contiene dos RadioButtons. Los botones del RadioButton son utilizados generalmente para presentar múltiples opciones al usuario para su elección. Cuando un RadioButton es seleccionado en un RadioGroup los demás RadioButton se desactivan automáticamente. Si observaos los RadioButton se encuentran dispuestos verticalmente, debido que en el RadioGroup el atributo android:orientation=”vertical”, si deseamos que los RadioButton se dispongan en formación horizontal, tendremos que cambiar el atributo de la siguiente forma: android:orientation=”horizontal”.

Por último tenemos el widget ToogleButton, que nos muestra un botón con la leyenda “OFF” o “ON”. Que cambiara cada vez que el botón sea tocado.

Manejos de Eventos en los widget.
Una parte fundamental al momento de manejos de los eventos que se puedan producir cuando se oprime un botón, o bien cuando tocamos una casilla, que se producirá cuando esto sucede.
Todos estos eventos tendremos que manejarlo en archivo de java, para este caso su nombre es Principal.java, es importante realizar en primera instancia la relación del widget que se encuentra en activity_principal.xml, con alguna variable que pueda manejar los eventos. Esta relación se da con los siguientes instrucciones en programa .java.

Declaramos una variable del tipo del widget que queremos relacionar de la siguiente forma:

Button botn botn1

botn= findViewById(R.id.btnSave);, si observamos le asignamos a la variable botn el widget denominado btnSave en activity_principal.xml.
botn1=findViewById(R.id.btnOpen); y la variable botn1 al widget denominado btnOpen en activity_principal.xml.

Nuestro programa Principal.java se vería de la siguiente forma:

```
package com.example.user.eventos;

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

```
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.RadioButton;
import android.widget.Toast;
import android.widget.ToggleButton;

public class Principal extends Activity {
    Button boton, boton1;
    ToggleButton toogle;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);
        boton= findViewById(R.id.btnSave);
        boton1=findViewById(R.id.btnOpen);
        boton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(),"Oprimiste la tecla
Save", Toast.LENGTH_LONG).show();
            }
        });
        boton1.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "Oprimi la boton
Open", Toast.LENGTH_LONG).show();
            }
        });
        toogle = findViewById(R.id.toggle1);
        toogle.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
```

```
        public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
            if(isChecked)
               Toast.makeText(getApplicationContext(),"La casilla se
encuentra en ON",Toast.LENGTH_LONG).show();
            else
               Toast.makeText(getApplicationContext(),"La casilla se
encuentra en OFF",Toast.LENGTH_LONG).show();
         }
      });
   }

   public void OprimirImagen(View view) {
      Toast.makeText(getApplicationContext(), "Oprimi la boton
imagen", Toast.LENGTH_LONG).show();
   }

   public void onboxclicked(View view) {
      boolean checked = ((CheckBox)view).isChecked();
      switch(view.getId()) {
         case R.id.chkAutosave:
            if(checked)
            Toast.makeText(getApplicationContext(),"Orpimi la casilla
AutoSave",Toast.LENGTH_LONG).show();
            break;
         case R.id.star:
            if(checked)
               Toast.makeText(getApplicationContext(),"Orpimi la casilla
AutoStar",Toast.LENGTH_LONG).show();
            break;
      }
   }
```

```
    public void onradiobutton(View view) {
        boolean checked = ((RadioButton) view).isChecked();

        switch(view.getId()) {
            case R.id.rdb1:
                if (checked)
                    Toast.makeText(getApplicationContext(),"Oprimi la opción
1",Toast.LENGTH_LONG).show();
                    break;
            case R.id.rdb2:
                if (checked)
                    Toast.makeText(getApplicationContext(),"Oprimi la opción
2",Toast.LENGTH_LONG).show();
                    break;

        }
    }
}
```

Los eventos que se dan para los widgets de Button, CheckBox y RadioButton y ToggleButton los detallaremos en las siguientes líneas.

Eventos de Button.

Como lo analizamos anteriormente lo primero que tenemos que realizar es relacionar el id del widget que tenemos en archivo xml con las variables que vamos a manejar dentro del programa de java de nuestra aplicación. Por ejemplo:

```
boton= findViewById(R.id.btnSave);
boton1=findViewById(R.id.btnOpen);
```

Existen dos formas de poder programar los eventos dentro del widget Button, la primera es utilizando el método setOnClickListener() con la siguiente sintaxis.

```
boton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           Toast.makeText(getApplicationContext(),"Oprimiste la tecla
Save", Toast.LENGTH_LONG).show();
        }
    });
```

El método setOnClickListener () registra un callback que se invocará más adelante cuando se haga clic en el botón. El método onClick es llamado cuan do el botón es presionado.
En este ejemplo tenemos la instrucción Toast que nos servirá para desplegar comentarios hacia el usuario, por un tiempo corto.
Toast.makeText(getApplicationContext(), “Comentario a desplegar”, Tiempo).show();. Donde tiempo podrá ser Toast.LENGTH_LONG, para tiempos largos y Toast.LENGTH_SHORT para tiempos cortos.

Otra forma de poder manejar los eventos en los eventos es utilizando el atributo android:onClick=”onclick”, el onclick será el nombre del procedimiento que vamos a llamar y que se encuentra declarado en el programa de java. La estructura de la declaración del widget en activity y el programa de java será la siguiente:

```
<Button
    android:id="@+id/btnOprime"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Oprimir"
    Android:onClick=”onclick” />
```

Y la parte de la declaración del procedimiento será:

```
public void onclick(View v)
{
   Toast.makeText(getApplicationContext(),”Oprimiste el botón
Operime”,Toast.LENGTH.LONG).show();
}
```

En caso de los eventos de los botones ImageButton, este maneja los mismos eventos que los botones que se estudiaron anteriormente.

Evento de CheckBox.
Al igual que en cualquier widget, tendremos que relacionar el nombre de nuestro widget con la variable que lo va a manejar en el programa java y la declaración delos widgets dentro dl archivo xml. Como se ve a continuación:

Archivo xml.

```
<CheckBox
        android:id="@+id/chkAutosave"
        android:layout_width="match_parent"
        android:onClick="onboxclicked"
        android:layout_height="wrap_content"
        android:text="Autosave" />

    <CheckBox
        android:id="@+id/star"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onboxclicked"
        android:text="AutoStar"/>
```

La programación del evento se da en el método onboxclicked, donde verificamos si tenemos marcadas las casillas de nuestro CheckBox y cual ellas se encuentran checadas.

```
public void onboxclicked(View view) {
      boolean checked = ((CheckBox)view).isChecked();
      switch(view.getId()) {
         case R.id.chkAutosave:
            if(checked)
            Toast.makeText(getApplicationContext(),"Orpimi la casilla
AutoSave",Toast.LENGTH_LONG).show();
            break;
         case R.id.star:
            if(checked)
               Toast.makeText(getApplicationContext(),"Orpimi la casilla
AutoStar",Toast.LENGTH_LONG).show();
            break;
      }
   }
```

Evento RadioGroup RadioButton.
En este widget a diferencia de los CheckBoxy dado que se encuentran agrupados por el RadioGroup, solo es posible marcar una sola casilla de los todas las opciones que tiene el usuario (RdioButton). De la misma manera que tenemos en los demás widgets los declaramos en nuestro archivo xml y su programación en el archivo java.

Archivo xml.

```
<RadioGroup
      android:id="@+id/rdbGp1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
```

```
    android:orientation="vertical" >
    <RadioButton
        android:id="@+id/rdb1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onradiobutton"
        android:text="Option 1" />
    <RadioButton
        android:id="@+id/rdb2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onradiobutton"
        android:text="Option 2" />
</RadioGroup>
```

La implementación del manejo de los eventos está sustentado en indicar con el atributo Android:onClick=”onradiobutton“ en cada uno de la declaración de los RadioButton y declarar el procedimiento en archivo de java como se muestra enseguida.

```
public void onradiobutton(View view) {
    boolean checked = ((RadioButton) view).isChecked();

    switch(view.getId()) {
        case R.id.rdb1:
            if (checked)
                Toast.makeText(getApplicationContext(),"Oprimi la opción
1",Toast.LENGTH_LONG).show();
                break;
        case R.id.rdb2:
            if (checked)
                Toast.makeText(getApplicationContext(),"Oprimi la opción
2",Toast.LENGTH_LONG).show();
```

```
                break;

        }
    }
}
```

Evento ToggleButton.
Este botón nos indica, si se encuentra apagado o encendido, o bien si lo oprimimos o le cambiamos su estado. Este widget como los anteriores se declara en la interface de usuario o archivo activity_principal.xml y en archivo Principal.java.

En el archivo xml tendríamos

```
    <ToggleButton
      android:id="@+id/toggle1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
```

En el archivo de java tendremos la implementación del evento a través del método setOnCheckedChangeListener().
Principal.java

```
toogle = findViewById(R.id.toggle1);
      toogle.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
          @Override
          public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
            if(isChecked)
               Toast.makeText(getApplicationContext(),"La casilla se
encuentra en ON",Toast.LENGTH_LONG).show();
            else
```

```
                Toast.makeText(getApplicationContext(),"La casilla se
encuentra en OFF",Toast.LENGTH_LONG).show();
            }
        });
    }
```

EditText.

Este widget nos apoyara introducir datos a nuestro dispositivo, para esto haremos uso de los aprendido hasta este momento, por lo que haremos un pequeño calculador que suma, resta, multiplica y divida 2 operandos. Para lo cual diseñaremos la interface de usuario y la programación de la calculadora.

Archivo xml.

```
 <?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Principal">

    <TextView

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="22dp"
        android:textSize="25sp"
```

```
    android:text="Simple Calculadora" />

<EditText
    android:id="@+id/editText4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentTop="true"
    android:layout_marginRight="24dp"
    android:layout_marginEnd="24dp"
    android:layout_marginTop="68dp"
    android:ems="10"
    android:inputType="numberDecimal" />

<EditText
    android:id="@+id/editText5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignStart="@+id/editText4"
    android:layout_alignLeft="@+id/editText4"
    android:layout_marginTop="148dp"
    android:ems="10"
    android:inputType="numberDecimal" />

<EditText
    android:id="@+id/editText6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignStart="@+id/editText4"
    android:layout_alignLeft="@+id/editText4"
```

```
    android:layout_marginTop="209dp"
    android:ems="10"
    android:inputType="numberDecimal" />

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginBottom="196dp"
    android:onClick="onclick"
    android:layout_marginLeft="11dp"
    android:layout_marginStart="11dp"
    android:text="+" />

<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignTop="@+id/button2"
    android:onClick="onclick"
    android:layout_marginLeft="113dp"
    android:layout_marginStart="113dp"
    android:text="-" />

<Button
    android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignTop="@+id/button2"
    android:onClick="onclick"
    android:layout_marginRight="91dp"
    android:layout_marginEnd="91dp"
    android:text="*" />

<Button
    android:id="@+id/button5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:onClick="onclick"
    android:layout_alignTop="@+id/button2"
    android:text="/" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/editText4"
    android:layout_alignRight="@+id/button2"
    android:layout_alignEnd="@+id/button2"
    android:text="Operando1" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/editText5"
    android:layout_alignRight="@+id/button2"
    android:layout_alignEnd="@+id/button2"
```

```
        android:text="Operando2" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/button2"
        android:layout_alignParentTop="true"
        android:layout_marginTop="217dp"
        android:text="Resultado" />
</RelativeLayout>
```

En la programación es importante destacar dos métodos que nos ayudan a recoger la información del EditText y posteriormente desplegarla, como lo veremos en nuestra aplicación. Estos métodos son el getText() y setText(), el primero de ellos tomara el valor que se inserte en nuestro EditText como un valor de tipo String, lo cual lo tendremos que convertir en un valor numérico, posteriormente convertirla en tipo String para desplegarla sobre nuestra activdad, nuestra aplicación quedaría de la siguiente manera.

```
package com.example.user.edittext;

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

public class Principal extends Activity {
    EditText edtext, edtext1, edtext2;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_principal);
    edtext = findViewById(R.id.editText4);
    edtext1 = findViewById(R.id.editText5);
    edtext2 = findViewById(R.id.editText6);

}

public void onclick(View v) {
    String num = edtext.getText().toString();
    String num1 = edtext1.getText().toString();
    double oper = Double.parseDouble(num);
    double oper1 = Double.parseDouble(num1);
    Button b = (Button) v;
    if (b.getId() == R.id.button2) {
        double resul = oper + oper1;
        edtext2.setText(String.valueOf(resul));
    } else if (b.getId() == R.id.button3) {
        double resul = oper - oper1;
        edtext2.setText(String.valueOf(resul));
    } else if (b.getId() == R.id.button3) {
        double resul = oper * oper1;
        edtext2.setText(String.valueOf(resul));
    } else if (b.getId() == R.id.button4) {
        double resul = oper * oper1;
        edtext2.setText(String.valueOf(resul));
    } else if (b.getId() == R.id.button5) {
        double resul = oper / oper1;
        edtext2.setText(String.valueOf(resul));
    }
```

```
    }
}
```

Intents Explicitos e Implicitos.

Intents Explicitos.
Estos especifican qué componente se debe iniciar mediante su nombre (el nombre de clase completamente calificado). Usualmente, el usuario usa una intent explícita para iniciar un componente en su propia aplicación porque conoce el nombre de clase de la actividad o el servicio que desea iniciar
Típicamente una aplicación está compuesta por cero o más actividades que no son otra cosa que las interfaces que interactúan con los usuarios. Además de las actividades, otro concepto único en Android es intent. Un intent es lo que permite que varias aplicaciones puedan trabajar juntas sin problema alguno, como si se tratase de una sola aplicación.
Una actividad es creada desde una clase que se hereda de la superclase Activity, esta clase genera la actividad cargando todos los componentes que se encuentra en archivo activity_principal.xml. En nuestro archivo principal.java tendremos la instrucción que genera la actividad que es: setContentView(R.layout.activity_principal);, quedando nuestro programa de la forma siguiente:

```
package com.example.user.intent;

import android.app.Activity;
import android.os.Bundle;

public class Principal extends Activity {

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);
    }
}
```

Cada actividad debe de estar registrada en archivo de configuración denominado AndroidManifest.xml como se muestra.

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

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

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

Cuando exista la necesidad de crear una actividad, lo podamos crear abriendo nuestro menú en app click derecho y escoger Activity, click derecho y escoger Empty Activity como se muestra en Figura 3.19

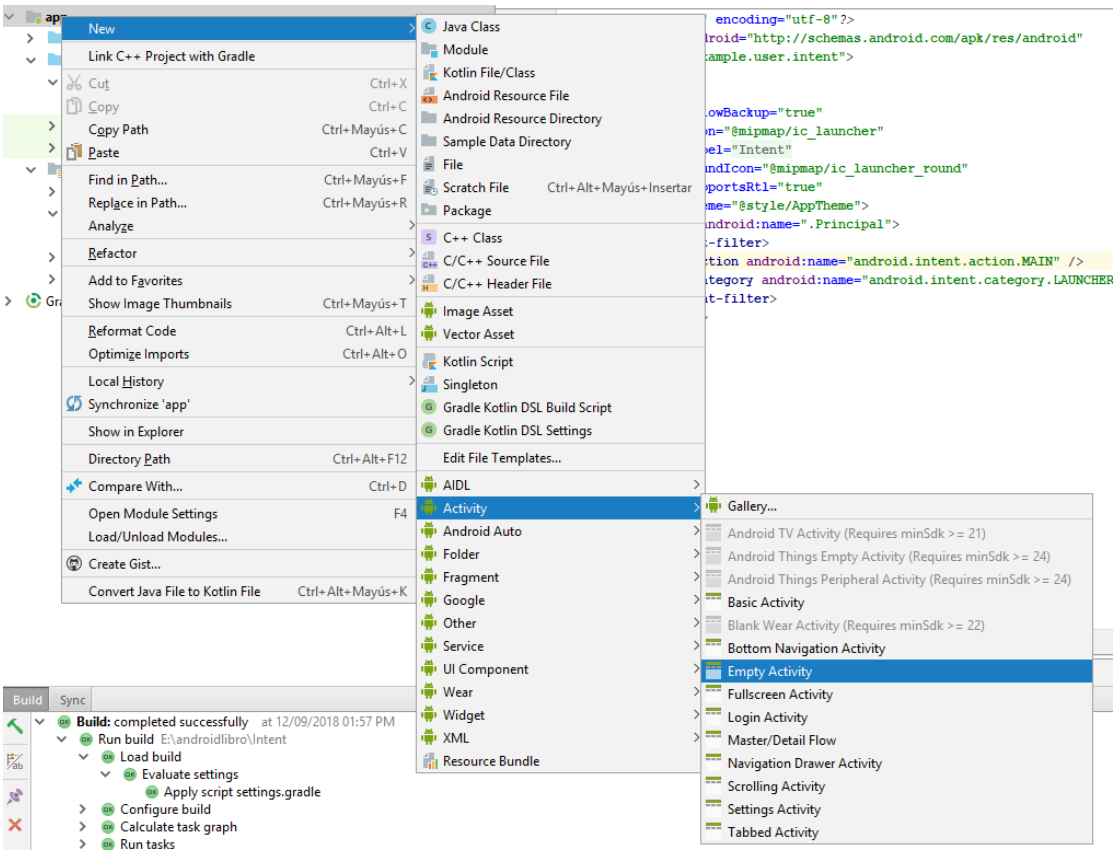


Figura 3.19 Desarrollando una segunda Actividad

Posteriormente aparece una siguiente pantalla donde nos solicita el nombre como vamos a identificar nuestra activity y además nos pide si esta estará acompañado por una layout como aparece en la Figura 3.20

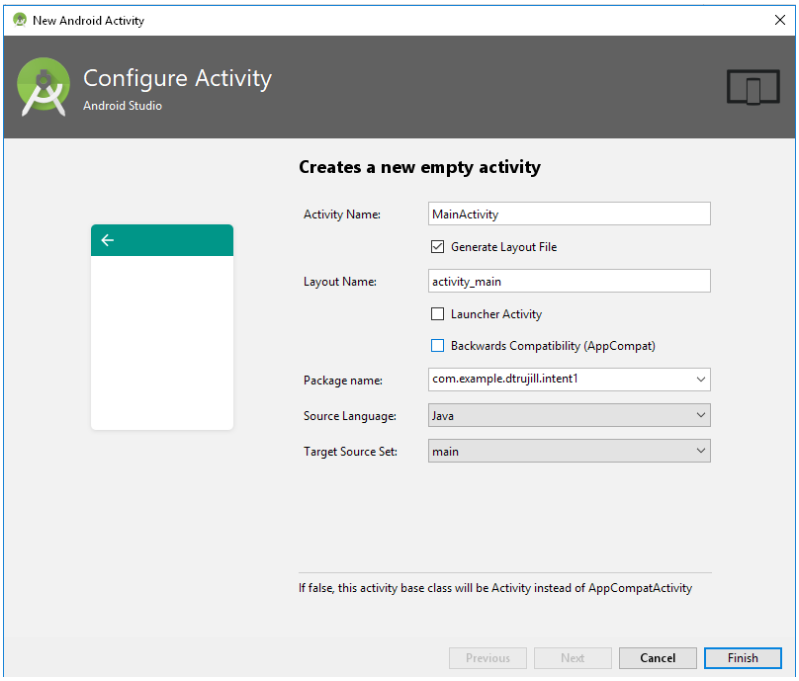


Figura 3.20 Creando una nueva actividad.

Para esto vamos a tener que diseñar dos layout como se indica:

activity:principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#f9c5ce"
    tools:context=".Principal">

    <TextView
        android:id="@+id/texto"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:text="PrimeraActividad"
        android:textSize="27sp"/>

    <Button
        android:id="@+id/button"
        android:layout_marginTop="30dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/texto"
        android:onClick="onclick"
        android:text="Pasar a la segunda actividad" />
```

El siguiente layout estará diseñado de la siguiente forma

activity_segunda.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Segunda">
    <TextView
        android:id="@+id/texto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
```

```
        android:text="Segunda Actividad"
        android:textSize="27sp" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:onClick="finaliza"
        android:layout_marginTop="76dp"
        android:text="Terminar Activity" />
</RelativeLayout>
```

Los layout se verían según la Figura 3.21

En la programación tendremos una clase denominada Intent que nos crea una instancia con los argumentos de la actividad donde se encuentra y cuál es la actividad que queremos trabajar. Como lo demuestra el siguiente código cuando oprimimos el botón

```
package com.example.dtrujill.intent1;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Principal extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_principal);

    }
        public void onclick(View v) {
            Intent inten = new Intent(Principal.this, Segunda.class);
            startActivity(inten);
        }
}
```

Figura 3.21 Layouts de las actividades

Como vemos después de hacer la instancia de la clase esta es ejecuta por un método denominado startActivity(intent);, que nos lanza hacia la segunda actividad; cual contiene el código que hace que finalice la actividad.

```
package com.example.dtrujill.intent1;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;

public class Segunda extends Activity {

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

    public void finaliza(View view) {
        finish();
    }
}
```

El método finish(); hace que la actividad finalice y devuelva el control a la primera actividad.

Pasó de Información de una actividad a otra.

En muchas de nuestras aplicaciones será necesario intercambiar datos entre nuestras actividades, por lo que Android implemento varios métodos para poder realizarlos de acuerdo al tipo de dato que

deseamos pasar, así como los métodos que se necesitan para capturar esos datos en la actividad que los recibe.

La actividad que envía los datos queda de la manera siguiente:

```
package com.example.dtrujill.intent1;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Principal extends Activity {
    int entero= 236;

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

    }
       public void onclick(View v) {
           Intent inten = new Intent(Principal.this, Segunda.class);
           inten.putExtra("int", entero);
           startActivity(inten);
       }
}
```

En este código observamos el método inten.putExtra("int", entero);, donde estamos enviando un valor entero a la actividad Segunda.class. La información se recibirá en la Segunda.class de la siguiente manera:

```
package com.example.dtrujill.intent1;

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

public class Segunda extends Activity {

    TextView tex;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_segunda);
        tex = findViewById(R.id.texto1);
        int act1 = getIntent().getIntExtra("int", -1);
        tex.append("\n" + Integer.toString(act1));
    }

    public void finaliza(View view) {
        finish();
    }
}
```

Como observamos el código int act1 = getIntent().getIntExtra("int", -1); donde explícitamente con getIntent().getIntExtra(“int”,-1) estamos aceptando un valor de tipo entero.

public Intent putExtra (String name, double[] value)	getIntDoubleArray(String, int)
public Intent putExtra (String name, char value)	getCharExtra(String, char)
public Intent putExtra (String name, float value)	getFloatExtra(String, float)
public Intent putExtra (String name, long[] value)	getLongArrayExtra(String)
public Intent putExtra (String name, String value)	getStringExtra(String)
public Intent putExtra (String name, char[] value)	getCharArray(String)
public Intent putExtra (String name, short[] value)	getShortArray(String)
public Intent putExtra (String name, int value)	getIntExtra(String, int)
public Intent putExtra (String name, int[] value)	getIntArrayExtra(String)
public Intent putExtra (String name, byte[] value)	getByteArrayExtra(String)
public Intent putExtra (String name, float[] value)	getFloatArrayExtra(String)
public Intent putExtra (String name, double value)	getDoubleExtra(String)
public Intent putExtra (String name, byte value)	getByteExtra(String)

public Intent putExtra (String name, boolean value)	getBoolean(String)

Tabla 3.2 Métodos para enviar y recibir información entre actividades.

La corrida de nuestra aplicación en la Segunda actividad nos dara como resultado, Figura 3.22

Figura 3.22 Paso de información de una actividad a otra.

Intent Implícitos

Intents implícitas: no se nombra el componente específicamente; pero, en cambio, se declara una acción general para realizar, lo que permite que un componente de otra aplicación la maneje. Por ejemplo, si desea

mostrar al usuario una ubicación en un mapa, puede usar una intent implícita para solicitar que otra aplicación capaz muestre una ubicación específica en un mapa.

Tenemos el siguiente proyecto que nos mostrara una url, hacer una llamada y visualizar un mapa.

Archivo activity_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".Principal">

    <TextView
        android:layout_weight="2"
        android:textSize="19sp"
        android:layout_width="match_parent"
        android:gravity="center"
        android:layout_height="wrap_content"
        android:text="Intent Explícito" />

    <Button
        android:layout_weight="2"
        android:id="@+id/boton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onclick"
        android:text="Lanzar Browse"/>
```


```
<Button
    android:layout_weight="2"
    android:id="@+id/boton1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="onclick"
    android:text="Marcar a un usuario"/>

<Button
    android:layout_weight="2"
    android:id="@+id/boton2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="onclick"
    android:text="Ver un mapa"/>

</LinearLayout>
```

Figura 3.23

Nuestra interface se muestra en la Figura 3.23.

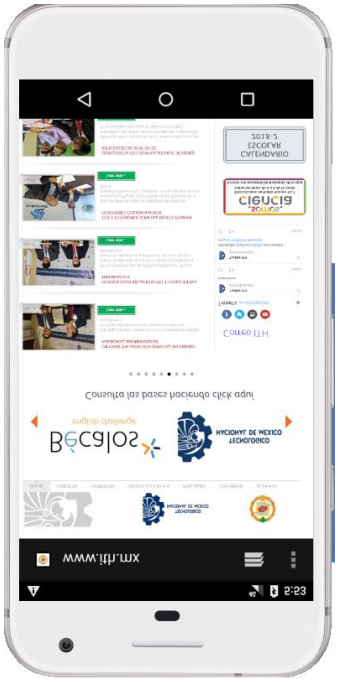
Como observamos en nuestra archivo xml, en nuestros botones tenemos el atributo android:onClick=”onclick”, esto hará referencia al método que se encuentra declarado en nuestro archivo de Java(Principal.java).

La declaración de nuestro método onclick se encuentra de la siguiente manera.

```
public void onclick(View view) {

    switch(view.getId()){
        case R.id.boton:
            Uri urluri =Uri.parse("http://www.ith.mx");
```

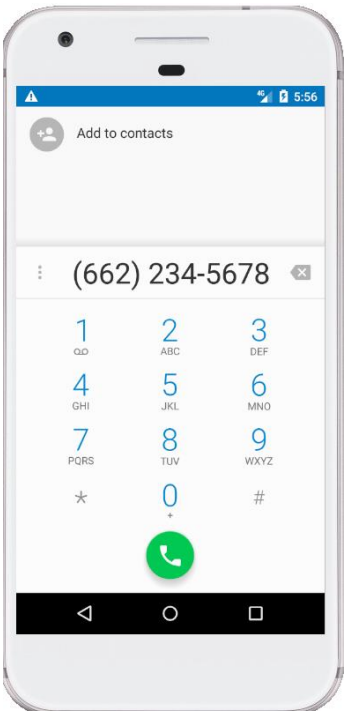
```
            Intent intent = new Intent(Intent.ACTION_VIEW, urluri);
            startActivity(intent);
            break;
        case R.id.boton1:
            intent = new Intent(Intent.ACTION_DIAL,Uri.parse("tel:"+
"6622345789" ));
            startActivity(intent);
            break;
        case R.id.boton2:
            Uri IntentUri = Uri.parse("geo:29.126000,-110.9773200?z=10");
            intent = new Intent(Intent.ACTION_VIEW, IntentUri);
            intent.setPackage("com.google.android.apps.maps");
            startActivity(intent);
            break;
        default:
            break;
    }
}
```



Como vemos en nuestra aplicación , cuando es oprimido oprimido el botón Lanzar Browse no se ejecuta una aplicación si no que Android, lanza la aplicación, un otra actividad donde pueda resolver dicho lanzamiento, si observamos no le indicamos explícitamente donde ejecute dicha instrucción.

En el caso de lanzar una url el Intent ejecutado es l el ACTION_VIEW, en este caso la url es http://ww.ith.mx y la ejecución se observa en la Figura 3.24

Figura 3.24



La otra opción que se presenta es cuando se oprime el botón Marcar a un usuario, el cual se ejecuta la instrucción ACTION_DIAL, conjuntamente con el número de teléfono del usuario al llamamos, la ejecución se observa en la Figura 3.25.

Figura 3.25 Función ACTION_DIAL

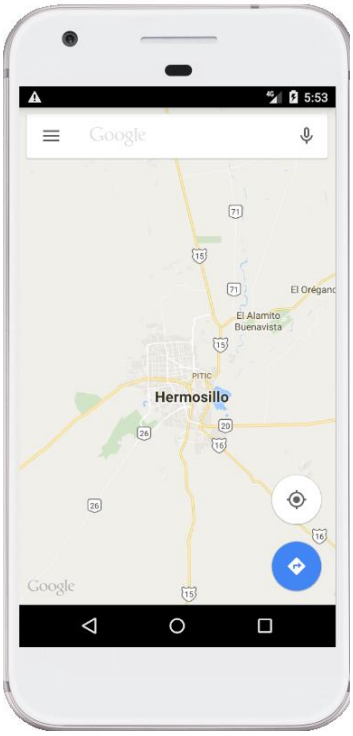


Figura3.26 Ejecución de u Intent Implicito ACTION_VIEW

La última opción nos da la oprotunidad de desplegar una mapa en una localización especifica, cuando se oprime el botón Ver un mapa. Para este caso la instrucción que se ejecuta, es

```
Uri IntentUri = Uri.parse("geo:29.126000,-110.9773200?z=10");
intent = new Intent(Intent.ACTION_VIEW, IntentUri);

 Intent.setPackage (“com.google.android.apps.maps);

startActivity(intent);
```

Donde damos las coordenadas en decimal y el zoom que deseamos observar en este caso z=10, también es posible buscar alguna referencia cerca del lugar donde estamos ubicados, o de las coordenas que se le asigne, para esto se tendría que hacer es modificar la instrucción.

```
geo:latitud,longitud?q=query
geo:0,0?q=my+street+address(“geo:0,0?q=Instituto+Tecnológico+de+Hermosillo”)

geo: 29.126000,-110.9773200?q=restaurants(Me buscara los restaurantes en Hermosillo,Sonora)
```

IV. ADMINISTRACIÓN DE DATOS EN DISPOSITIVOS MÓVILES.

4.1 Introducción.

Como se ha estado mencionado en las secciones anteriores, las aplicaciones de dispositivos móviles, funcionan con el esquema cliente-servidor, esto hace que nuestra aplicación sea el cliente y donde se tenga la información sea nuestro servidor. La persistencia de la información en este tipo funcionamiento es importante, el servidor se encuentra en forma remota y la aplicación frecuentemente se utiliza solo para recuperar dicha información. Pero Android además tiene otros esquemas para manejar la información que necesite nuestra aplicación, como se mencionaran más adelante.

4.2 Modelo de objetos de acceso a datos.

Muchas de las aplicaciones móviles, necesariamente en algún momento necesitan persistir datos, ya sea esto serializándolos, guardarlos en una base de datos o en archivos; y esto lo hace a través de la capa de persistencia que es la encargada de interactuar con el medio que guardara nuestra información.

Un modelo de encargado de crear la capa de persistencia es el DAO(Data Access Object), es importante separar la capa lógica de negocios y la capa de persistencia debido a que si por cualquier motivo cambiáramos la base de datos y tuviéramos la capa de lógica de negocio y la capa de persistencia como una sola, los accesos a la base de datos los haría directamente nuestros programas y dichos programas enfocadas a una base de datos específica, en cambio sí las capas se encuentran separados, solo se tendría que modificar la capa de persistencia.

El modelo DAO encapsula el acceso a la base de datos. Las dos capas interactúan a través de una API; generalmente esta API ofrece los métodos CRUD(Create, Read, Update, Delete). Por lo que cuando la capa lógica de negocios desee guardar un registro llamara al método create(), para almacenar la información. Aquí lo importante es que la capa lógica de negocios no le interesa como se implemente el método create(), solo sabe que debe de llamarlos para guardar la información.

4.3 Manipulación de datos.

Como lo manejamos en las secciones anteriores Android puede manejar la persistencia de sus datos de manera diferentes, estas pueden ser como bases datos remotas, bases de datos en nuestro dispositivo, en archivo que se manejan en el dispositivo, o bien información que se utiliza en nuestra aplicación durante la ejecución del programa y que guardada en nuestro dentro de él.

SharedPreferences.

También conocida como preferencias, que no son más que datos almacenados en el dispositivo, que se hace para personalizar la experiencia del usuario. La información puede ser almacenada en una base de datos o en un archivo, pero existe la forma de shared preferences, esta información se guarda en forma clave-valor es decir que cada una de ellas tiene un identificador asociado a este un valor. A diferencia de los tipos de almacenamiento que se almacena en forma binaria estos se guardan en ficheros xml.

Tendremos una aplicación que nos ejemplificara la manera en que podemos manejar las shared preferences.

Tendremos el diseño de la interface, de la siguiente manera:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:context=".Principal">

<TextView
    android:gravity="center"
    android:layout_marginTop="10dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Shared Prefrences" />

<TextView
    android:layout_marginTop="10dp"
    android:text="Usuario"
    android:textSize="12sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<EditText
    android:id="@+id/edtext"
    android:layout_marginTop="6dp"
    android:ems="16"
    android:hint="Usuario"
    android:inputType="text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView
    android:layout_marginTop="10dp"
    android:text="Password"
```

```
        android:textSize="12sp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/edtext1"
        android:layout_marginTop="6dp"
        android:ems="13"
        android:inputType="textPassword"
        android:hint="Password"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button
        android:text="Guardar"
        android:onClick="onclick"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:layout_marginTop="12dp"
        android:text="Recuperar"
        android:onClick="recuperar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

La pantalla se verá tal como lo muestra la Figura 4.1

Figura 4.1

En esta aplicación utilizaremos dos botones que con uno de ellos llamaremos al método onclick, donde guardaremos la información utilizando la clase SharedPreferences.Editor para poder guardar la información al instanciarla podemos tomar utilizando la instancia acompañado de por el método put, y el tipo de dato que se almacena, por ejemplo si deseamos guardar un string el método será putString(clave, valor),

En caso de recuperar la información recurriremos al botón recuperar en cual llamara al método del mismo nombre, como vemos tendremos que hacer la referencia de la clase SharedPreferences.Editor, pero este caso utilizaremos el método getString(clave, valor por default), si el valor fuera de tipo string.

```
package com.example.user.sharedp;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.preference.PreferenceManager;
import android.widget.Toast;

public class Principal extends Activity {
   EditText edtext,edtext1;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_principal);
      edtext = findViewById(R.id.edtext);
      edtext1 = findViewById(R.id.edtext1);

   }
   // En este punto guardamos la información
   public void onclick(View view) {

      SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(getBaseContext()).
edit();
      editor.putString("usuario",edtext.getText().toString());
      editor.putString("password",edtext1.getText().toString());
      editor.apply();
      Toast.makeText(Principal.this, "Datos guardados",
Toast.LENGTH_LONG).show();
```

```
        edtext.setText(" ");
        edtext1.setText(" ");
    }
    // En este punto recuperams la información.
    public void recuperar(View view){
        SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        edtext.setText(sp.getString("usuario","NA"));
        edtext1.setText(sp.getString("password","NA"));
```

SQLIte

SQLite es un Manejador de Base de datos Es un ligero motor de bases de datos de código abierto, que se caracteriza por mantener el almacenamiento de información persistente de forma sencilla.

A diferencia de otros Sistemas gestores de bases de datos como MySQL, SQL Server y Oracle DB, SQLite tiene las siguientes ventajas:

- No requiere el soporte de un servidor: SQLite no ejecuta un proceso para administrar la información, si no que implementa un conjunto de librerías encargadas de la gestión.
- No necesita configuración: Libera al programador de todo tipo de configuraciones de puertos, tamaños, ubicaciones, etc.
- Usa un archivo para el esquema: Crea un archivo para el esquema completo de una base de datos, lo que permite ahorrarse preocupaciones de seguridad, ya que los datos de las aplicaciones Android no pueden ser accedidos por contextos externos.

- Es de Código Abierto: Está disponible al dominio público de los desarrolladores al igual que sus archivos de compilación e instrucciones de escalabilidad.
  Es por eso que SQLite es una tecnología cómoda para los dispositivos móviles. Su simplicidad, rapidez y usabilidad permiten un desarrollo muy amigable.
  Para ejemplificar esta base de datos y como la maneja Android, tenemos una estructura de una base de datos que se llama bd_usuarios que está conformada por una tabla denominada usuarios y los siguientes campos:
  Id Integer primary key
  nomb TEXT
  direc TEXT
  ciudad TEXT

  Y nuestra aplicación desarrolla el conocido Alta, Baja, Consulta y Modificaciones.

  Tendremos una clase que nos ayuda a realizar la conexión y a generar nuestra base de datos que le llamamos ConexionSQLiteHelper.java y otra clase donde tendremos la programación de los métodos de Alta, Baja, Consulta y Modificaciones.
  La interface de desarrollo, se muestra en la Figura 4.2

Figura 4.2

La clase ConexionSQLiteHelper.java

```
package com.example.dtrujill.sqlite;

importa android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import com.example.dtrujill.sqlite.utilidades.Utilidades;

public class ConexionSQLiteHelper extends SQLiteOpenHelper {

    final String CREAR_TABLA_USUARIO="CREATE TABLE
usuarios(id INTEGER, nomb TEXT, direc TEXT, ciudad TEXT)";
```

```
    public ConexionSQLiteHelper(Context context, String name,
SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(Utilidades.CREAR_TABLA_USUARIO);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS usuarios");
        onCreate(db);
    }
}
```

En esta clase creamos la clase ConexionSQLiteHelper y el constructor, donde tenemos como parámetros el contexto de la aplicación, el nombre de la base de datos, el Cursor.Facory que es null por defecto, y la versión de la base de datos. Existen dos métodos que se sobreescriben el onCreate como parámetro la referencia a la base de datos y ejecutamos la sentencia de SQL para crear la tabla de la base de datos, y el método Upgrade, hace que si modificación la versión eliminara la tabla de usuarios si existe y la posteriormente la crea de nuevo.

```
package com.example.dtrujill.sqlite.utilidades;

public class Utilidades {
    public static final String TABLA_USUARIO ="usuarios";
    public static final String CAMPO_ID ="id";
    public static final String CAMPO_NOMB ="nomb";
```

```
    public static final String CAMPO_DIREC ="direc";
    public static final String CAMPO_CIUDAD ="ciudad";
    // create table usuarios(id INTEGER,nomb TEXT,direc TEXT,ciudad
TEXT);
    public static final String CREAR_TABLA_USUARIO="CREATE
TABLE " + " " +TABLA_USUARIO + " (" + CAMPO_ID + " "+
            "INTEGER, " + CAMPO_NOMB + " TEXT, " +
CAMPO_DIREC+ " TEXT," + " "+  CAMPO_CIUDAD +" TEXT)";
}
```

Existe otra clase denominada Utilidades que contiene el nombre de la tabla, el nombre de cada uno de los campos que conforma la tabla, de aquí hacemos las referencias a estos campos; así mismo la instrucción SQL necesaria para crear la tabla. Como veremos en el programa Principal las referencias se hacen de la manera siguiente Utilidades.CAMPO_NOMB, refiriéndose al campo nomb de la tabla usuarios.

Nuestra clase Principla.java

```
package com.example.dtrujill.sqlite;
import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import com.example.dtrujill.sqlite.utilidades.Utilidades;

public class Principal extends Activity {
    EditText campo_id,campo_nomb,campo_direc,campo_ciudad;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);
        campo_id=findViewById(R.id.editText);
        campo_nomb=findViewById(R.id.editText2);
        campo_direc=findViewById(R.id.editText3);
        campo_ciudad=findViewById(R.id.editText4);
    }

    public void Alta(View view) {
        ConexionSQLiteHelper conn = new ConexionSQLiteHelper(this,
"bd_usuarios",null,1);
        SQLiteDatabase db = conn.getWritableDatabase();
        ContentValues valor = new ContentValues();
        valor.put(Utilidades.CAMPO_ID,campo_id.getText().toString())
valor.put(Utilidades.CAMPO_NOMB,campo_nomb.getText().toString()
valor.put(Utilidades.CAMPO_DIREC,campo_direc.getText().toString());
valor.put(Utilidades.CAMPO_CIUDAD,campo_ciudad.getText().toStrin
g());
        long idResultante =
db.insert(Utilidades.TABLA_USUARIO,Utilidades.CAMPO_ID,valor);
        Toast.makeText(getApplicationContext(),"El Id registro:" +
idResultante,Toast.LENGTH_LONG).show();
        campo_id.setText(" ");
        campo_nomb.setText(" ");
        campo_direc.setText(" ");
        campo_ciudad.setText(" ");
        db.close();
    }
```

```
    public void Modficar(View view) {
        ConexionSQLiteHelper conn = new ConexionSQLiteHelper(this,
"bd_usuarios",null,1);
        SQLiteDatabase db = conn.getWritableDatabase();
        String [] parametros = {campo_id.getText().toString()};
        ContentValues valor = new ContentValues();
valor.put(Utilidades.CAMPO_NOMB,campo_nomb.getText().toString())
valor.put(Utilidades.CAMPO_DIREC,campo_direc.getText().toString());
valor.put(Utilidades.CAMPO_CIUDAD,campo_ciudad.getText().toStrin
g());
db.update(Utilidades.TABLA_USUARIO,valor,Utilidades.CAMPO_ID+"
=?",parametros);
        Toast.makeText(this,"Se actualizo el
usuario",Toast.LENGTH_LONG).show();
        db.close();
        campo_id.setText(" ");
        campo_nomb.setText(" ");
        campo_direc.setText(" ");
        campo_ciudad.setText(" ");
    }
    public void Consulta(View view) {
        ConexionSQLiteHelper conn = new ConexionSQLiteHelper(this,
"bd_usuarios",null,1);
        SQLiteDatabase db = conn.getReadableDatabase();
        String [] parametros = {campo_id.getText().toString()};
        String []
campos={Utilidades.CAMPO_NOMB,Utilidades.CAMPO_DIREC,Utilid
ades.CAMPO_CIUDAD};
        try
        {
```

```
            Cursor cursor =
db.query(Utilidades.TABLA_USUARIO,campos,Utilidades.CAMPO_ID
+"=?",parametros,null,null,null);
            cursor.moveToFirst();
            campo_nomb.setText(cursor.getString(0));
            campo_direc.setText(cursor.getString(1));
            campo_ciudad.setText(cursor.getString(2));
            cursor.close();
        }catch (Exception e)
        {
            Toast.makeText(this,"El usuario no
existe",Toast.LENGTH_LONG).show();
        }
        db.close();
    }
    public void Baja(View view) {
        ConexionSQLiteHelper conn = new ConexionSQLiteHelper(this,
"bd_usuarios",null,1);
        SQLiteDatabase db = conn.getWritableDatabase();
        String [] parametros = {campo_id.getText().toString()};

db.delete(Utilidades.TABLA_USUARIO,Utilidades.CAMPO_ID+"=?",pa
rametros);
        Toast.makeText(this,"Se elimino el
usuario",Toast.LENGTH_LONG).show();
        campo_id.setText(" ");
        campo_nomb.setText(" ");
        campo_direc.setText(" ");
        campo_ciudad.setText(" ");
        db.close();
    }
}
```

Como vemos en cada uno de los métodos tenemos que abrir la conexión con la instrucción,

```
ConexionSQLiteHelper conn = new ConexionSQLiteHelper(this, "bd_usuarios",null,1);
    SQLiteDatabase db = conn.getWritableDatabase();
```

Donde le indicamos con los parámetros en que contexto se encuentra, el nombre de la base de datos, el factory que es null y la versión en que se trabaja.

Demás indicamos de qué modo vamos a abrir mi base de datos; que podrá ser SQLiteDatabase db = conn.getWritableDatabase(), para modo lectura-escritura o si deseamos solo lectura tendremos SQLiteDatabase db = conn.getReadableDatabase();

La clase ContentValues es utilizada para recolectar un conjunto de datos que posteriormente se almacenaran, esto es a través de la instrucción valor.put(Nomb_del_campo, valor), y posteriormente insertar los valores en el manejador SQLite, a través de la instrucción

long idResultante = db.insert(Utilidades.TABLA_USUARIO, Utilidades.CAMPO_ID,valor);, donde idResultante es un valor de retorno si es -1 existió un error y un número mayor que cero tendremos la fila donde se guardó nuestro registro. En db.insert estará la tabla donde insertara el registro, el campo llave , ContestValues valor que es donde se encuentra la información que se almacenar.

En nuestro método Baja tenemos la instrucción:

```
String [] parametros = {campo_id.getText().toString()};
db.delete(Utilidades.TABLA_USUARIO,Utilidades.CAMPO_ID+"=?",parametros);
```

La variable parámetros que viene siendo un arreglo, que en este caso solamente presenta un solo elemento, que si hubiera la necesidad podremos manejar más de un elemento solamente separados por una coma; este parámetro se utilizara en la instrucción db.delete(tabla,campo llave, la opción where), la opción where nos ayudara a encontrar el campo que deseamos eliminar..

En nuestro método también tenemos la declaración de un parámetro que me dará la condición para recuperar nuestro registro, además tenemos otro arreglo denominado campos donde se recuperara la información de nuestra base de datos String[]campos={Utilidades.CAMPO_NOMB,Utilidades.CAMPO_DIREC,Utilidades.CAMPO_CIUDAD}; ; en este método tenemos un clase denominada Cursor, la instancia recupera la información cada vez que se lee un registro de la base de datos.

```
Cursor cursor =
db.query(Utilidades.TABLA_USUARIO,campos,Utilidades.CAMPO_ID
+"=?",parametros,null,null,null);
        cursor.moveToFirst();
        campo_nomb.setText(cursor.getString(0));
        campo_direc.setText(cursor.getString(1));
        campo_ciudad.setText(cursor.getString(2));
        cursor.close();.
```

Cuando se ejecuta la instrucción db.query(nomb_tabla, , variable donde se guardara, clausula where, arguamentos de where, groupBy, Having, orderBy), la información es almacenada en cursor y esta es recuperada a través de curso.getString(0), el get dependerá del tipo de dato que se desea recuperar.

Conexiones a Bases de Datos Remotas.

Como lo hemos comentado anteriormente la persistencia en cualquier aplicación es importante, es por eso que en dispositivos móviles generalmente esta se da en bases de datos remotas, misma que se conectan con el protocolo HTTP(Hipertext Transfer Protocol), este protocolo. Este protocolo consiste en reglas sencillas de transferencia de recursos o archivos entre equipos interconectados a una red. Aquí es donde el cliente hace una petición al servidor, existe varios tipos de peticiones dependiendo si estas son para petición de datos o

publicación hasta se les conoce como GET o POST. Las peticiones GET son aquellas que recupera cualquier información (en forma de una entidad) identificada por el Request-URI. Si el Request-URI se refiere a un proceso de producción de datos.
Las peticiones POST son aquellas que se utiliza para enviar información al servidor como, por ejemplo, un archivo de actualización, información de formulario, etc. En otras palabras, éste método se usa cuando se necesita enviar una entidad para algún recurso determinado. Para esto tenemos un ejemplo donde utilizaremos una pequeña base de datos en mysql, un web services en PHP y nuestra apalicación en Android. La interfce de nuestra es la kostrada en la Figura 4.3

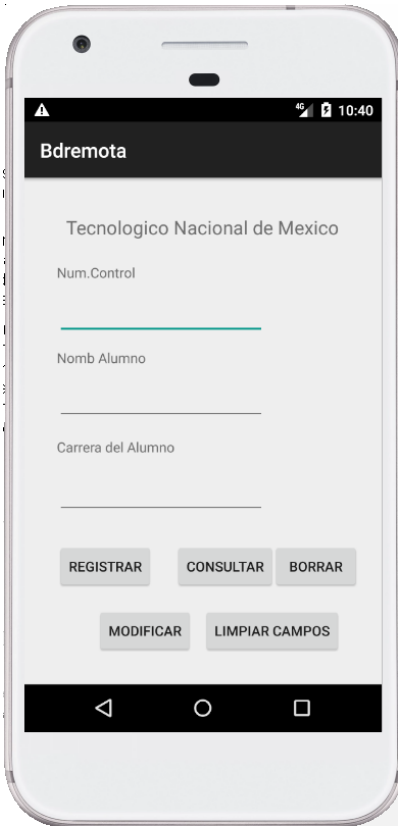


Figura 4.3

Como se observa en nuestra base de datos tiene tres campos que son los que vamos manipular.
En primera instancia debemos de revisar si nuestra red esta funcionando lo ue lo haremos con la clase ConectivityMnager y la clase NetworkingInfo, ns darán información para determinar si nuestra red se encuentra funcionando.

```
ConnectivityManager connMgr = (ConnectivityManager)
   getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
if (networkInfo != null && networkInfo.isConnected()) {
 // Operaciones http }
 else
{ // Mostrar errores }
```

Generalmente las peticiones a la base de datos, se realizan en forma Asíncrona , para lo cual se utilizatremos la clase AsyncTask<String….String>. esta clase maneja cuatro métodos los cuale se mencionaran mas adelante.

```
 Clase Tarea extends AsynTask<String..String> {
@override
protected void onPreExecute(){

    ……..

}
@override
protected resul doInBackgound(Parametros ……..paramet) {


}
```

```
@override
Proteced void onProgressUpdate(Progreso……..progre) {


}

@override
 Protected void onPostExecute(Resultado.resul)
{



}
```

Los métodos se pueden sobreescribir y son los pasos que sigue Async. El método onPreExecute realizara los trabajos previos a la realización de a tarea. Se utiliza normalmente configurar la tarea y para mostrar en el la interfaz de usuario que empieza la tarea.

doInBackground(Parametros….) Es llamado cuando termina onPreExecute(). Es la parte más importante donde tenemos que realizar la tarea propiamente dicha. Es el único método de los cuatro que no se ejecuta en el hilo del interfaz de usuario. Lo va a hacer en un hilo nuevo creado para este propósito. Como hemos visto la clase AsyncTask ha de ser paramétrizada con tres tipos de datos. Es decir, cuando creas un AsyncTask la clase Parametros ha de ser reemplazada por una clase concreta que será utilizada para indicar la información de entrada a la tarea.

onProgressUpdate(Progreso….)Este método se utiliza para mostrar el progreso de la tarea al usuario. Se ejecuta en el hilo interfaz de usuario,

por lo que podremos interactuar con las vistas. El progreso de una determinada tarea ha de ser controlado por el programador.

onPostExecute(Resultado): Este método se usa para mostrar en el interfaz de usuario el resultado de la tarea. El parámetro de entrada (de la clase Result) corresponde con el objeto devuelto por el método doInBackground()

Para hacer la conección a la base de datos es necesario utilizer el web services que es el intermediario entre nuestra base de datos y nuestra aplicación; las peticines que se realizan son del tipo POST, por lo que tendremos que inidcarlo explicitamente.

```
Class async extends Asyntask<String,String> {

String num_ctl;

protected void onPreExecute() {
        //para el progress dialog
        pDialog = new ProgressDialog(MainActivity.this);
        pDialog.setMessage("Autenticando....");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(false);
        pDialog.show();
    }

Protected String doInBackground(String… params) {
 num_ctl=params[0];
```

```
HttpClient cliente =new DefaultHttpClient();
        HttpContext contexto = new BasicHttpContext();
        HttpPost httpPost = new HttpPost(
                "http://I.P del servidor /Escuela/GetData.php");
        String resultado=null;
        try {
            List<NameValuePair> params = new
ArrayList<NameValuePair>(1);
            params.add(new BasicNameValuePair("num_ctrl", num_ctrl));
            httpPost.setEntity(new UrlEncodedFormEntity(params));
            HttpResponse response = cliente.execute(httpPost,contexto);
            HttpEntity entity = response.getEntity();
            resultado = EntityUtils.toString(entity, "UTF-8");
        } catch (Exception e) {
            // TODO: handle exception
        }
        return resultado;

protected void  onPostExecute(Sring resultado){
  pDialog.dismiss();//ocultamos progess dialog.
            Log.e("onPostExecute=", "" + result);
getresponse();
getjasonarray();

public void getpostresponse(){
```

```
//Convierte respuesta a String
Try{
BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
 StringBuilder sb = new StringBuilder()
 String linea = null;
while ((linea = reader.readLine()) != null) {
 sb.append(linea+ "\n");
}
is.close();
 result=sb.toString()
Log.e("getpostresponse"," result= "+sb.toString())
}catch(Exception e){
 Log.e("log_tag", "Error converting result "+e.toString());
}
 }
 public JSONArray getjsonarray(){
//parse json data
try{
JSONArray jArray = new JSONArray(result)
 return jArray
}Catch(JSONException e)
Log.e("log_tag", "Error parsing data "+e.toString());
    return null;
}
```

Como observamos en el listado de nuestro programa la coneccion se da a hacia el web services, el nos contestara enviando la información, codificada como JSON, que es la forma mas adecuada para la transmision de la información. Entonces ya que tenemos la información podemos desplegara en neustro dispositivo.

Como vemos el web services nos ayuda a conectarnos a la base de datos y al servidor donde se encuentra; posterioemente se ejecuta la instrucción que recuperara la información que es transformada en JSON para enviarse al cliente.

Web services GetData.php

```
<?php
$clave_esc=$_REQUEST["num_ctrl"];
 $con = mysql_connect(“IP de la base de datos","root","") or die("Sin conexion");
  mysql_select_db("escuela");
 sql="select id,num_ctrl, nomb_Alum, carr_Alum from alumno";
  $datos=array();
  $rs=mysql_query($sql,$con);
  while($row=mysql_fetch_object($rs)){
      $datos[] = $row;
  }
  echo json_encode($datos);
```

4.4 XML.

Siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información. (Bases de datos Silberschatz).

XML no ha nacido sólo para su aplicación para Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

VENTAJAS:

· Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.

· El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones.

· Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.

· Transformamos datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.

ESTRUCTURA DE XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
    <Mensaje>
        <Remitente>
            <Nombre>Nombre del remitente</Nombre>
            <Mail> Correo del remitente </Mail>
        </Remitente>
        <Destinatario>
            <Nombre>Nombre del destinatario</Nombre>
```

```
        <Mail>Correo del destinatario</Mail>
    </Destinatario>
    <Texto>
        <Asunto>
            Este es mi documento con una estructura muy sencilla
            no contiene atributos ni entidades...
        </Asunto>
        <Parrafo>
            Este es mi documento con una estructura muy sencilla
            no contiene atributos ni entidades...
        </Parrafo>
    </Texto>
  </Mensaje>
</Edit_Mensaje>
```

4.5 JSON.

SON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en

este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

EJEMPLO DE JSON

En teoría, es trivial analizar JSON en JavaScript usando la función eval() incorporada en el lenguaje. Por ejemplo:

```
miObjeto = eval('(' + json_datos + ')');
```

En la práctica, las consideraciones de seguridad por lo general recomiendan no usar eval sobre datos crudos y debería usarse un analizador JavaScript distinto para garantizar la seguridad. El analizador proporcionado por JSON.org usa eval() en su función de análisis, protegiéndola con una expresión regular de forma que la función sólo ve expresiones seguras.

DIFERENCIA ENTRE XML Y JSON:

A continuación se muestra un ejemplo simple de definición de barra de menús usando JSON y XML.

JSON:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
```

```
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
 }
}
```

XML:

```
 <menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
 </menu>
```

Bibliografía.-

Agar, John. *CONSTANT TOUCH A Global History of Mobile Phone*. Icon Books Ltd: London UK, 2013.

Stark, J. (2010). Building iPhone apps with HTML, CSS, and JavaScript. Editorial O'Reilly. ISBN:978-0-596-80578-4.

The Evolution of Cell Phone Design Between 1983-2009. WebdesignerDepot. 22 Mayo del 2009. 27 de Marzo del 2017. <http://webdesignerdepot.com/2009/05/the-evolution-of-cell-phone-design-between-1983-2009>.

Cassavoy, Liane. In Pictures: A History of Cell Phones. PCWorld . 7 de Mayo del 2007. 27 de Marzo del 2007. <http://www.pcworld.com/article/131450/in_pictures_a_history_of_cell_phone.html>.

Edwards, Benj. Evolution of Cell Phone. PcWorld. 4 de Octubre de 2009. 27 de Marzo del 2017.

<http://www.pcworld.com/article/173033/evolution_of_cell_phone.html>

Barrett,Brian. Sprints´HTC Evo, the First Ever 4g Phone: Meet the New Terrific. GIZMODO. 23 de Marzo del 2010. 20 de Marzo del 2017.

http://gizmodo.com/5500343/prints-htc-evo-the-firts-ever-4g-phone-meet-the-new-terrific

Las tendencias en tecnología móviles en 2016.1 de Enero del 2016. El Universal/cartera/negocios. 22 de Marzo del 2017. http://www.eluniversal.com.mx/articulo/cartera/negocios/2016/01/1/las-tendencias-en-tecnologia-movil-para-el-2016.

Mathew,David. Examinando las tendencias de tecnología móvil para 2016. SearchDataCenter. 2016. 22 de Marzo del 2017. <http://searchdatacenter.techtarget.com/es/cronica/Examinando-las-tendencias-de-tecnologia-movil-para-2016>

Mawston, Neil. "Apple iPhone 7 was world´s best-selling Smartphone model in Q1 2017". Srategy Analytics Press Releases. 10 de Mayo 2017. 25 de Mayo 2017.

Josep Prieto Blázquez, Robert Ramírez Vique, Julián David Morillo Pozo, Marc Domingo Prieto. Tecnología y Desarrollo de dispositivos móviles. 2011.Universidad Oberta de Cataluya. Editorial : Eureca Media, SL

D. E. Avison y G. Fitzgerald, Information system development. Maidenhead: McGraw-Hill Education, 2006

<https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-press-releases/strategy-analytics-press-release/2017/05/10/strategy-analytics-apple-iphone-7-was-world's-best-selling-smartphone-model-in-q1-2017#.WSzbXGh96Un>

Commited to improving the state of world, Deep Shift Technology Tipping Points and Societal Impacts, Septiembre del 2015, World

Forum Economic , 2 de Mayo del 2017, <http://www3.weforum.org/docs/WEF_GAC15_Technological_Tipping_Points_report_2015.pdf >.

Melki Reyes. Los 5 mejores sistemas operativos para celulares. iPhone web tecnologías.6 de Marzo del 2013. 27 de Mayo del 2017. <http://iphoneandord.com/los-5-mejores-sistemas-operativos-para-celulares/>

http://androideity.com/2012/07/16/5-lenguajes-para-programar-en-android/

htttp://www.startcapps.com/blog/tutorial-objective-c/

http://blogs.msdn.com/b/ricardoj/archive/2012/07/12/desarrollo-de-aplicaciones-para-windows-phone-herramientas.aspx.

Gasca Mantilla, Maira Cecilia; Camargo Ariza, Luis Leonardo; Medina Delgado, Byron. Metodología para el desarrollo de aplicaciones móviles. Tecnura, vol. 18, núm. 40, abril-junio, 2014, pp. 20-35. <http://www.redalyc.org/articulo.oa?id=257030546003

Nielsen, J., and Molich, R. (1990). Heuristic evaluation of user interfaces, *Proc. ACM CHI'90 Conf.*(Seattle, WA, 1-5 April), 249-256.

Griffiths, Dawn y Griffiths, David. Head First Android Development. 1005Gravenstein Highway North, Sebastopol, CA 95472 : O'Reilly, 2015.

MacLean, Dave, Komatineni, Satya y Allen, Grant. Pro Android 5. New York :Apress, 2015.

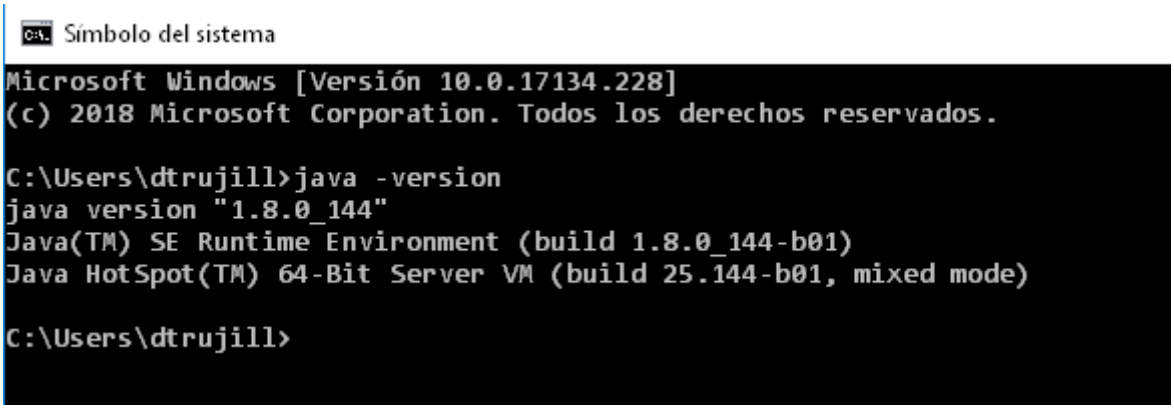
Murphy, Mark L. The Busy Coder's Guide to Android Development. United States of America : CommonsWare, LLC., © 2008-2017.

Wei-Meg Li. Beginning Android™ Application Development. Wiley Publishing, Inc. 2011.

Anexo 1.Instalación de Java y Android Studio.

Instalación de Lenguaje Java.
Verificar sis se encuentra instalado el JDK (Java Development Kit) en nuestra computadora. Una forma de verificar sis se encuentra instalado Java será

a. Entrar a cmd de Windows y teclear a instrucción java –versión como lo muestra la figura.



b. Si no se encuentra entonces tendremos que instalarlo de la siguiente forma:

En un navegador busque “java jdk download”. Lo cual le mostrará varias ligas, de las cuales le mostrará una que diga “Descargas de Java SE - Oracle” y nos muestra una imagen como:

Java SE Development Kit 8u162

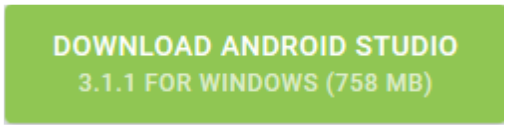
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.93 MB	jdk-8u162-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u162-linux-arm64-vfp-hflt.tar.gz
Linux x86	169.01 MB	jdk-8u162-linux-i586.rpm
Linux x86	183.81 MB	jdk-8u162-linux-i586.tar.gz
Linux x64	166.13 MB	jdk-8u162-linux-x64.rpm
Linux x64	181.02 MB	jdk-8u162-linux-x64.tar.gz
macOS	247.12 MB	jdk-8u162-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.98 MB	jdk-8u162-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.3 MB	jdk-8u162-solaris-sparcv9.tar.gz
Solaris x64	140.68 MB	jdk-8u162-solaris-x64.tar.Z
Solaris x64	97.03 MB	jdk-8u162-solaris-x64.tar.gz
Windows x86	198.57 MB	jdk-8u162-windows-i586.exe
Windows x64	206.76 MB	jdk-8u162-windows-x64.exe

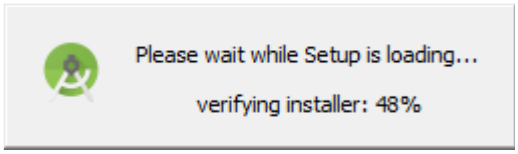
Antes de empezar a descargar deberos verificar si nuestro procesador es de 32 o 64 bits, lo ejecutamos y damos next hasta que finalice la instalación.

Descargar e instalar el software Android Studio.

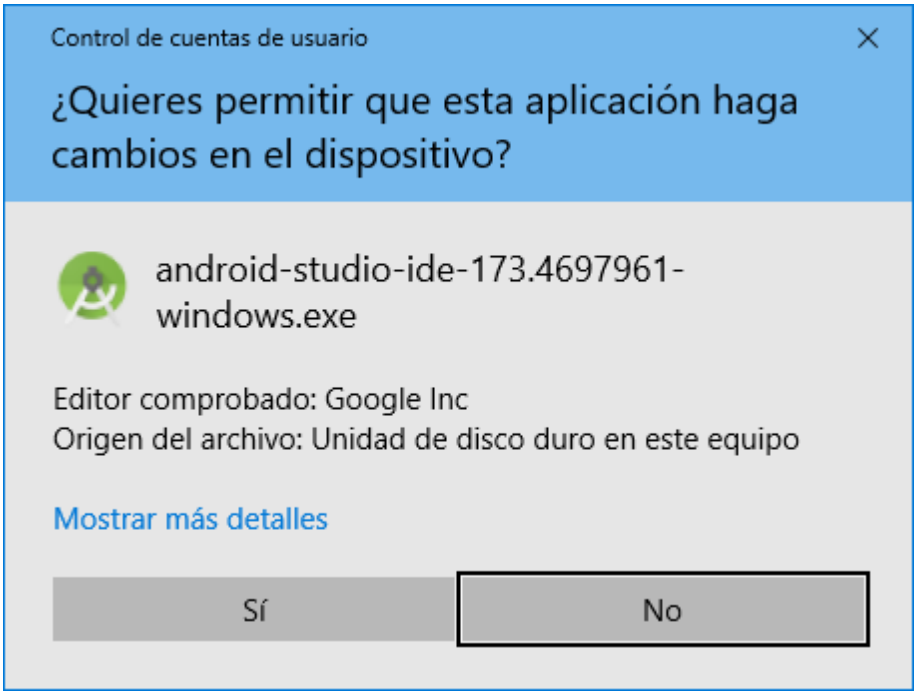
En un navegador busque “android studio download”. Nos va a llevar a la liga: https://developer.android.com/studio/index.html; y la abrimos. Damos clic en la liga siguiente.



Al bajar el software obtenemos un archivo similar a “android-studio-ide-173.4697961-windows.exe”. Lo ejecutamos e inmediatamente se comienza a instalar de la siguiente, manera.

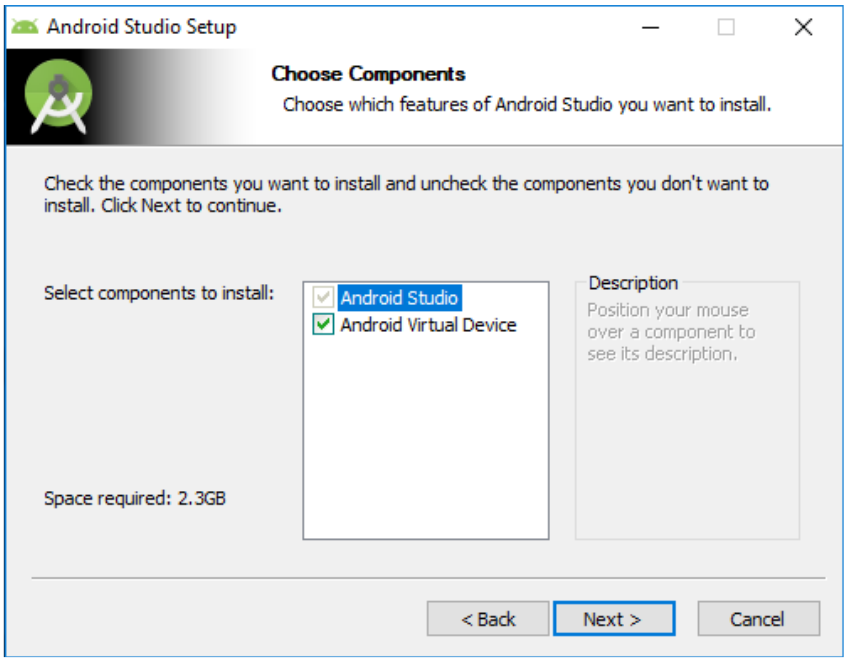


Se verifica el software que se va instalar.Nos pide permiso para instalar Androis Stuio de la siguiente manera:

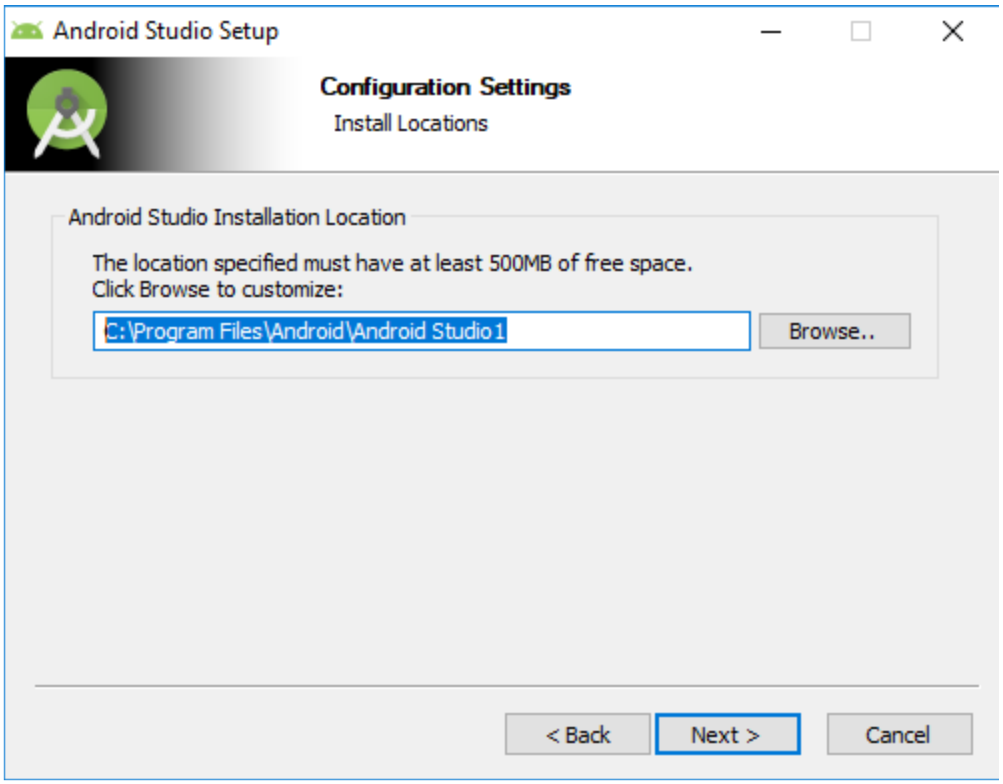


Posteriormente nos muestra la pantalla inicial y oprimimos la tecla .Next.

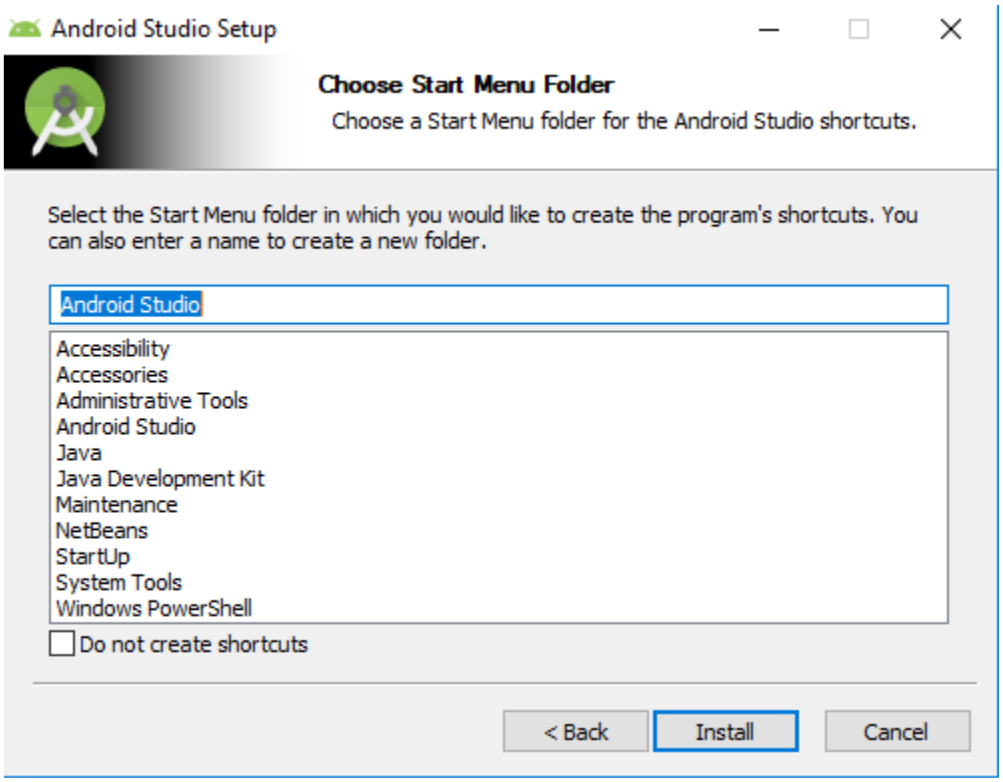
Luego instalaremos los componentes por default y oprimimos la tecla Next.



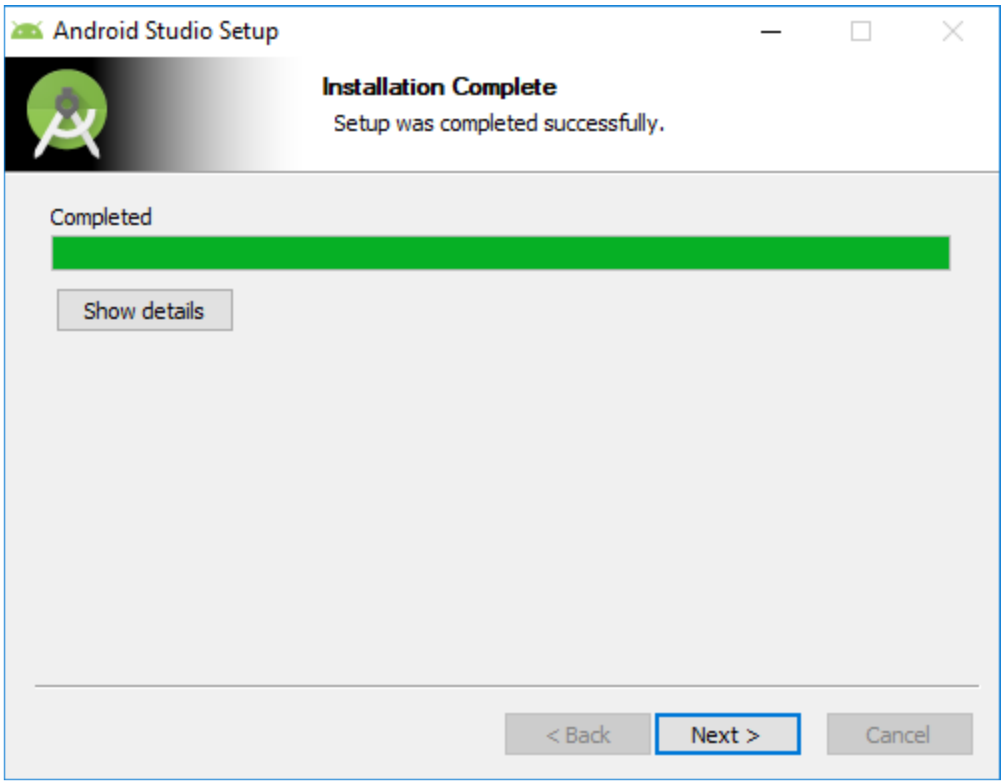
Luego le indicamos en que trayectoria queremos guardar nuestro programa.



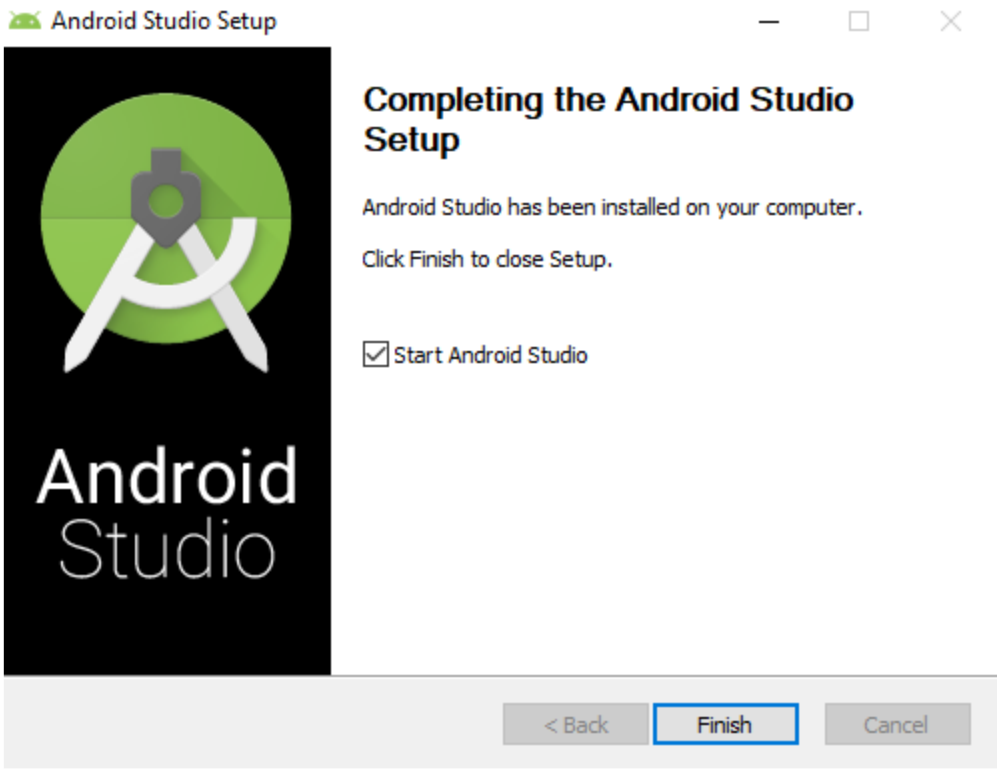
Oprimimos la tecla Install y esperamos que finalice la instalación.



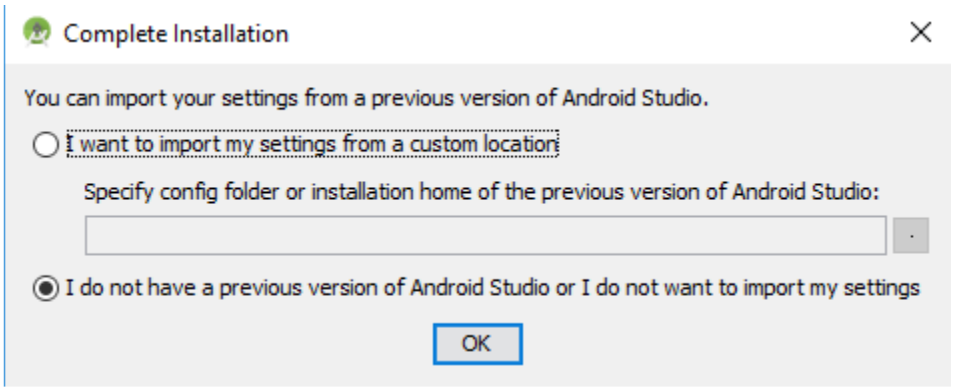
Nos muestra que la instalación se ha completado y oprimimos la tecla Next.



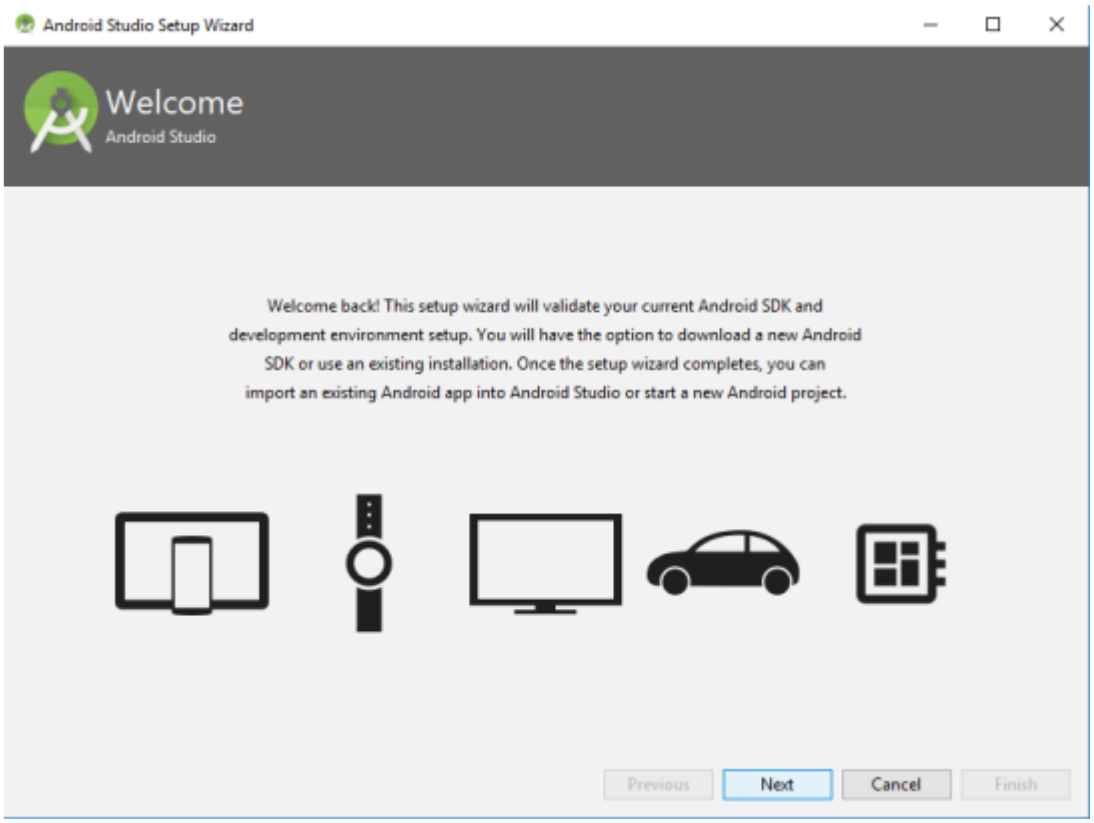
Al finalizar la instalación nos muestra una última pantalla. En este momento no pregunta si deseamos ejecutar Android Studio. Dejamos seleccionado la ejecución de Android Studio y damos “Finish”.



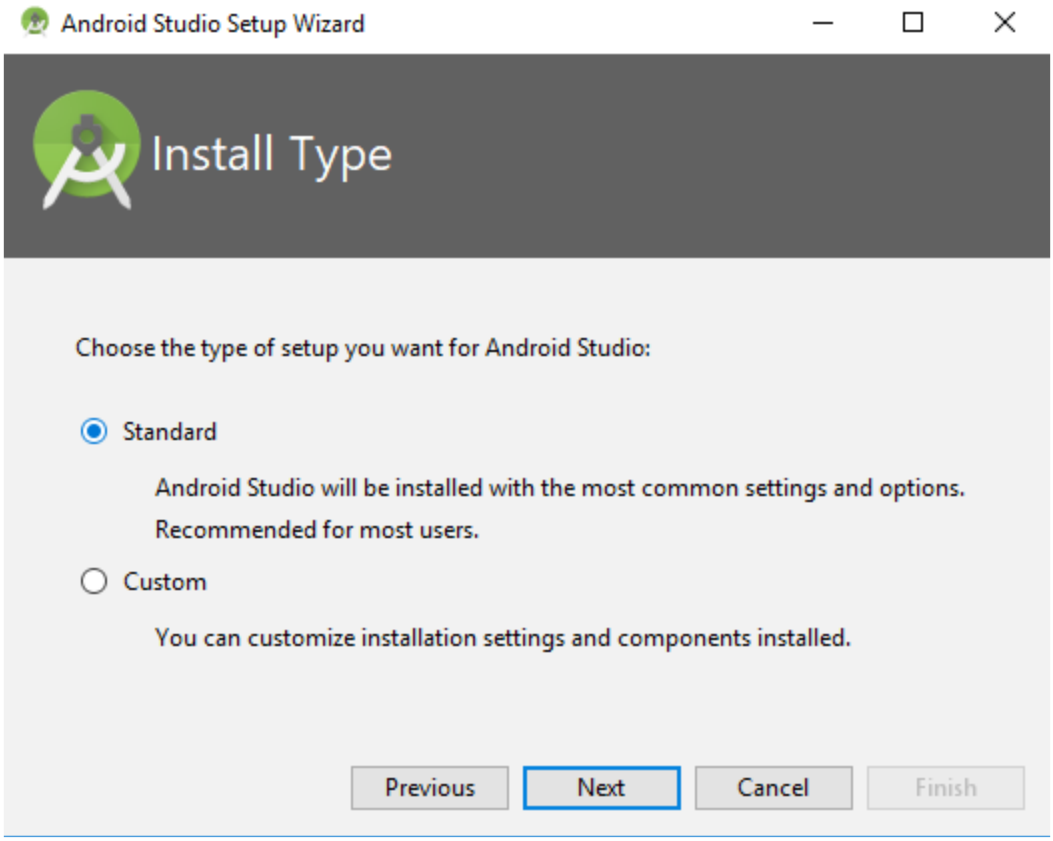
Al oprimir la tecla Finish y antes que se ejecute Android Studio nos pregunta si queremos importar una configuración de una software que una versión antigua de Android Studio de la siguiente manera, a lo cual indicaremos que no deseamos hacerlo.



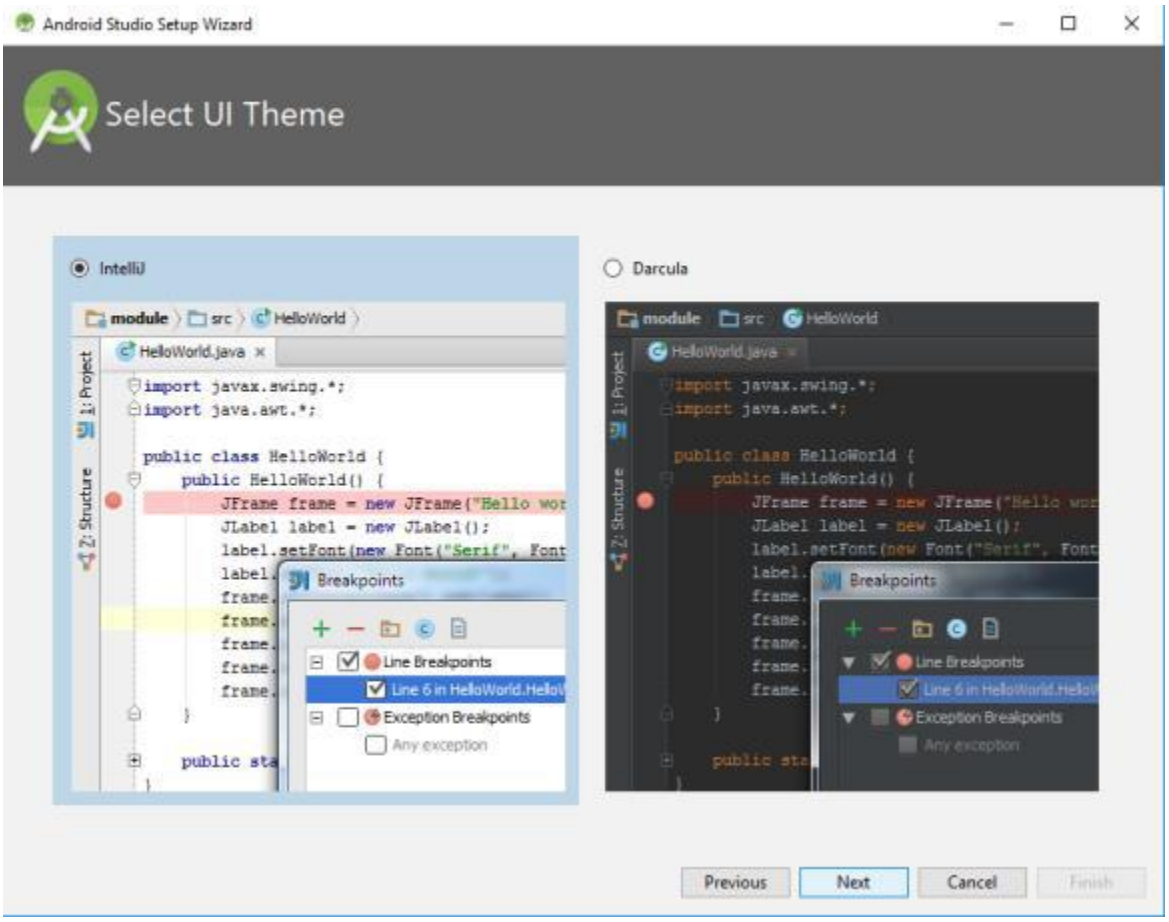
Seguidamente nos muestra una pantalla, donde se carga el IDE de Android Studio.



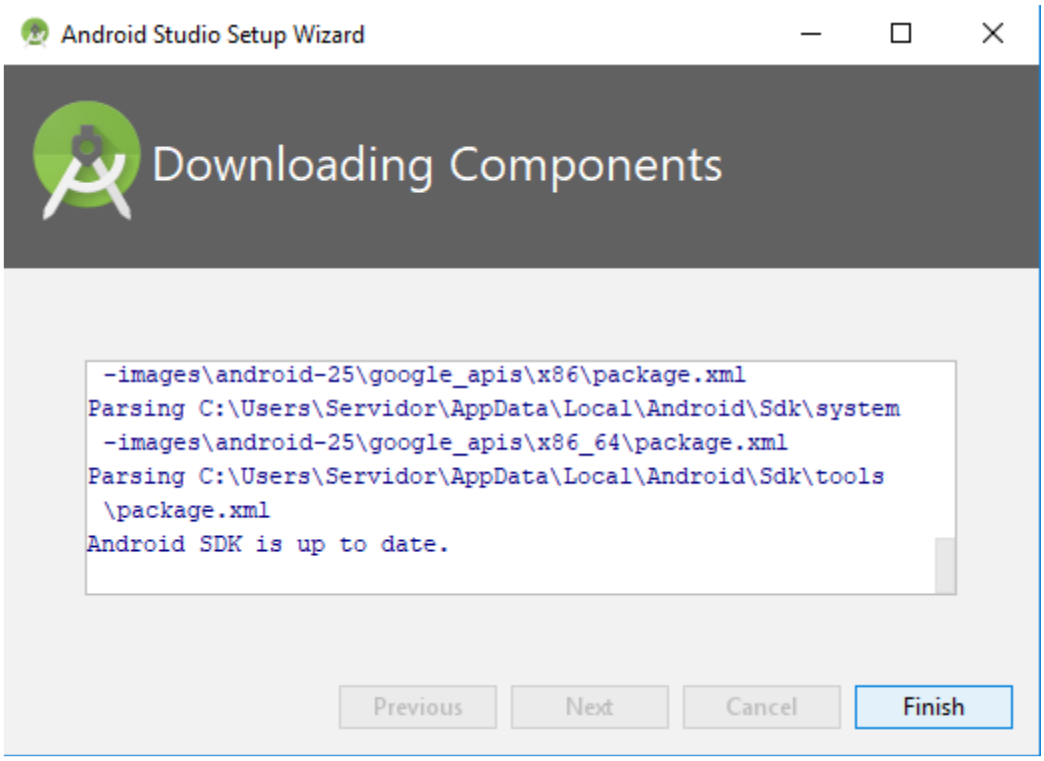
En la pantalla siguiente nos pregunta que tipo de instalación deseamos hacer, si la vamos hacer personalizada o si será standard. En este caso escogeremos la opción standard.



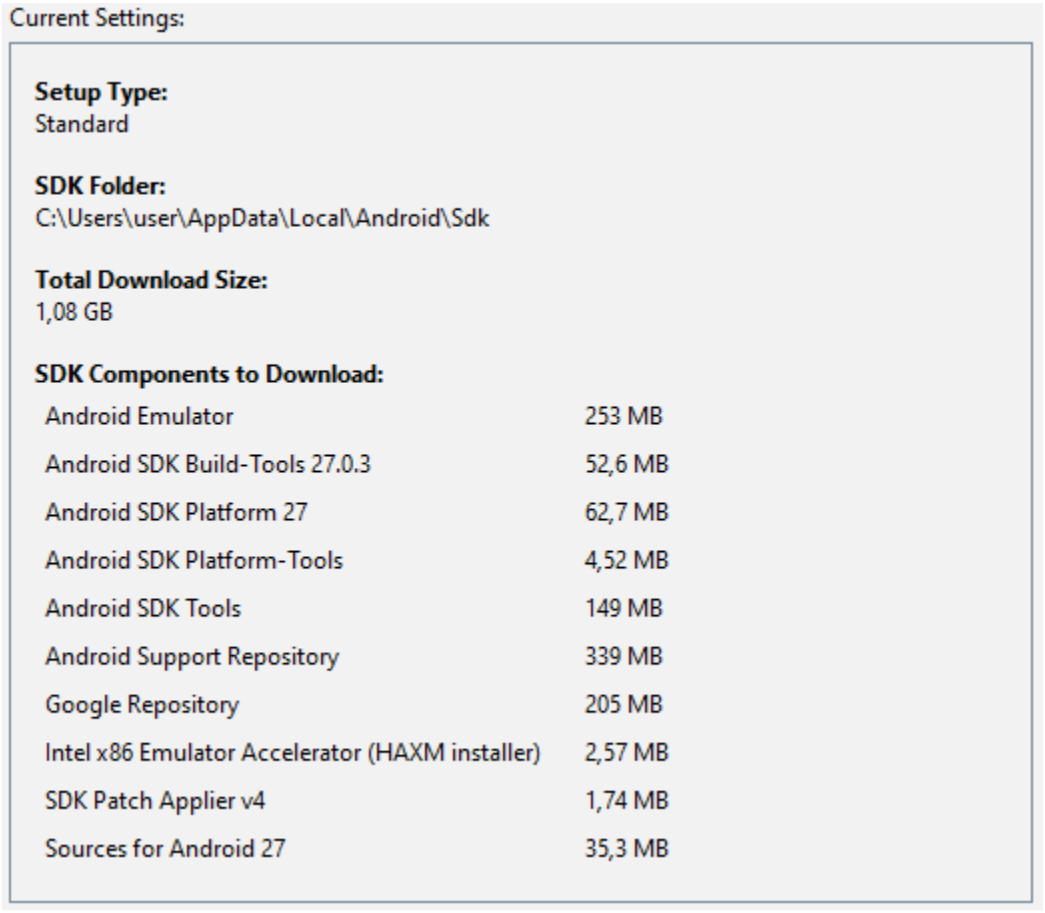
La siguiente nos muestra las interfaces que podemos configurar en Android Studio, que puede ser la IntelliJ o la Darcula, para este caso seleccionamos ItelliJ.



Al seleccionar muestra interface, enseguida empieza algunos componentes que necesita para funcionar adecuadamente, al finalizar nos indica que el SDK se ha actualizado.



Posteriormente nos mostrara cuales han sido los componentes que se encuentran instalados.


Current Settings:

Setup Type:
Standard

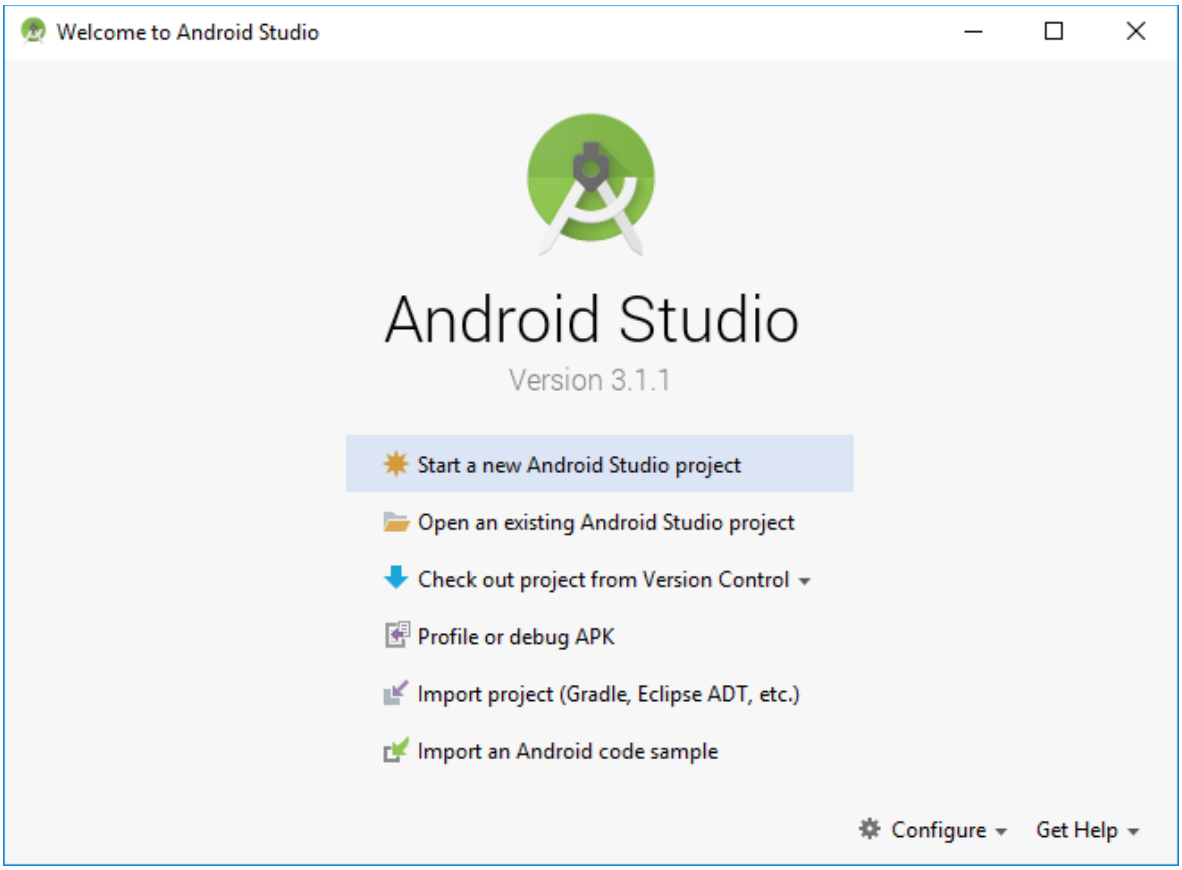
SDK Folder:
C:\Users\user\AppData\Local\Android\Sdk

Total Download Size:
1,08 GB

SDK Components to Download:

Android Emulator	253 MB
Android SDK Build-Tools 27.0.3	52,6 MB
Android SDK Platform 27	62,7 MB
Android SDK Platform-Tools	4,52 MB
Android SDK Tools	149 MB
Android Support Repository	339 MB
Google Repository	205 MB
Intel x86 Emulator Accelerator (HAXM installer)	2,57 MB
SDK Patch Applier v4	1,74 MB
Sources for Android 27	35,3 MB

Ya finalizada la instalación aparece la pantalla principal de Android Studio, como se muestra en la figura de la parte inferior.


Welcome to Android Studio
Android Studio
Version 3.1.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help