UNIVERSIDAD NACIONAL AUTÓNOMA

DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

DIVISIÓN DE LAS CIENCIAS FISICO-MATEMATICAS Y DE LAS INGENIERIAS.

"DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN, PARA SMARTPHONE CON SISTEMA OPERATIVO ANDROID, CON EL OBJETIVO DE REDUCIR TIEMPOS DE OPERACIÓN EN CONSULTAS DE INFORMACIÓN TEÓRICA Y TÉCNICA"

T E S I S

PARA OBTENER EL TÍTULO DE:

INGENIERO MECÁNICO ELECTRICISTA

AREA: INGENIERÍA ELÉCTRICA - ELECTRÓNICA

P R E S E N T A:

LÓPEZ GONZÁLEZ PABLO MICHELLE

TUTOR:

Dr. VEGA RAMÍREZ ALEJANDRO ANTONIO

SAN JUAN DE ARAGÓN, MÉXICO, 2015

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

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

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

Agradecimiento.

Agradezco a todas las personas que hicieron posible este logro, ya sea de una u otra forma ayudándome a terminar mis estudios profesionales, en especial a mi padre y madre así como a mis hermanos que siempre me brindaron su apoyo incondicional. Igualmente agradezco el apoyo de mis amigos que tuve en toda mi vida estudiantil ya que gracias a esas amistades logramos tener el objetivo en común de terminar una carrera y también son parte importante de este logro.

Esta obra se dedica para cualquier persona que tenga la iniciativa de realizar una herramienta de apoyo en cualquier ambiente de trabajo y también a las personas que quisieran aprender a iniciarse en el mundo de las aplicaciones móviles y en algún momento pudieran auto emplearse.

INTRODUCCION.

Esta investigación se realiza tomando en cuenta que los equipos móviles Smartphone ya forman parte de la vida cotidiana de la mayoría de personas de todo el mundo y no solo para mantenerse comunicado sino como equipo que permite el entretenimiento, información y herramienta laboral, todo entro de un mismo dispositivo.

Sabiendo que en este momento la tecnología avanza a pasos agigantados, en el área laboral también es necesario adaptarse y evolucionar para poder utilizar las herramientas que esta, la tecnología, nos brinda y poder hacer nuestra vida más útil, rápida y eficiente en cualquier ámbito.

En el terreno laboral la tecnología también se está utilizando para poder realizar las actividades más rápidas y eficientes. Por ello, es necesario tener o idear una aplicación para llevar a cabo consultas de información importante de la empresa y así no depender, por ejemplo, de estar conectado a la intranet corporativa, o bien, agilizar la operación al minimizar las consultas con algún jefe o compañero, y quizá darle una solución errónea por falta de conocimiento e información. Por esta razón es importante tener una aplicación móvil de consulta para darle salida a los problemas que surgen a cada momento en la actividad laboral.

Teniendo en cuenta esta necesidad para cualquier empresa, se decide realizar una aplicación base funcional con el objetivo de ayudar a los empleados de cualquier organización a tener la información importante de su empresa en sus manos, en cualquier momento y con toda la seguridad de que no podrá dar acceso a gente no autorizada, logrando que se den mejores soluciones al momento y en cualquier lugar.

Tomando en cuenta que una gran parte de las personas utilizan su Smartphone como herramienta de trabajo se toma la decisión de investigar los sistemas operativos y la forma de poder realizar una aplicación móvil.

Se revisará a grandes rasgos la historia de los Smartphone y sus diferentes sistemas operativos que existen a nivel mundial y así mismo dar una revisión de los programas computacionales de Windows para llevar a cabo una aplicación móvil.

Se llevara a cabo el desarrollo de una aplicación para el sistema operativo móvil más usado a nivel mundial y con el entorno de desarrollo más adecuado, tomando las mejores decisiones sobre las características importantes y necesarias que debe de llevar la aplicación, para que esta pueda ser implementada y desarrollada por cualquier persona que lea esta investigación y para cualquier organismo que desee tener una aplicación de consulta de información teórica y técnica en su Smartphone.

Objetivo General.

Desarrollar e Implementar una aplicación, totalmente funcional en el sistema operativo Android a partir de la versión 2.3, con la finalidad de reducir tiempos de operación en consultas de información teóricas y técnicas.

Objetivos particulares:

- Conocer lo que es un Smartphone, desde su historia y características hasta su estado actual que lo catalogan dentro de este rubro, así como los diferentes sistemas operativos con los que opera.
- Conocer el sistema operativo Android de acuerdo a sus ventajas, así como algunos programas que existen para desarrollar aplicaciones para este sistema.
- Conocer la plataforma Android Studio así como sus características básicas para desarrollar aplicaciones para dicho sistema.
- Desarrollar la aplicación con código de programación, mostrando las características básicas e indispensables con las que debe de contar una aplicación, con el fin de ser implementada como herramienta de trabajo de cualquier empresa para hacer consultas de información teóricas y técnicas.
- Implementar la aplicación totalmente desarrollada y validar el nivel de satisfacción obtenido por la empresa al utilizarla.

Índice

CAPITULO 1

SMARTPHONE

SMARTPHONE

Los dispositivos móviles y su incidencia en nuestra vida cotidiana.

La tendencia es irreversible. Los dispositivos móviles están ingresando a nuestra vida cotidiana en una forma más rápida que la pronosticada.

El aumento de oferta de estos dispositivos, principalmente el denominado Smartphone, teléfono inteligente, con nuevas aplicaciones que solucionan operaciones que antes nos exigían una mayor inversión de tiempo y que permiten potenciar la comunicación ya sea como destinatario de noticias y novedades específicas como generador de contenido en los medios sociales ya está modificando nuestras rutinas.

Como todos los cambios profundos en las costumbres sociales, esta nueva era de los dispositivos móviles como centro de nuestra actividad diaria, pondrá fin a muchos servicios existentes que eran eficientes mientras no existían los mismos, como por otro lado producirán el nacimiento de otros que tienen a los mismos como punto de referencia.

Ya Internet había decretado muchos cambios en varias actividades de la sociedad.

La prensa escrita fue una de los servicios más populares en la sociedad que se vio afectado en su incidencia en el consumo de la gente, debiendo adaptarse a esos cambios que provocaron la red o sucumbir.

Hoy lo que viene, o mejor dicho lo que tenemos es una vida social que tiene como centro de actividades los dispositivos móviles y asociado a los mismos la tecnología Cloud, es decir almacenamiento de datos y programas en servidores externos a nuestro dispositivo entre los que existe una interacción permanente de datos sin importar donde nos encontremos las posibilidades que se abren son infinitas.

Las aplicaciones móviles.

Las Apps son aplicaciones para dispositivos móviles que te permiten accesar, trabajar, compartir, informarte, analizar o crear lo que quieras desde tu Smartphone o tablet.

En un mundo donde lo online toma fuerza y se trabaja las 24 hrs. al día los 7 días de la semana, todos necesitamos estar conectados y localizables, no se diga de las personas responsables de la estrategia en redes sociales de una empresa. De ahí la importancia de contar con todas las herramientas para desempeñar bien nuestro trabajo desde donde estemos.

La importancia de las aplicaciones móviles para las empresas.

En nuestros días el trabajo a distancia con el móvil Smartphone también se ha convertido en una herramienta fundamental. Tecnologías como internet y herramientas como las que esta gran red ofrece son cada vez más importantes en la vida de las empresas y los trabajadores de todo el mundo, hasta el punto de que hoy sea difícil concebir las rutinas y las tareas laborales sin el enorme abanico de posibilidades que estas tecnologías y herramientas ofrecen para hacer más fáciles, rápidos y eficientes los negocios y las comunicaciones.

Y no se trata de tecnologías implementadas solo por grandes multinacionales, sino que, por el contrario, desde los más pequeños negocios han encontrado la forma de adaptarse a las alternativas más convenientes para ellos, de manera que puedan sacarles el mayor provecho y obtener excelentes resultados a partir de ellas.

Precisamente, la nueva AT&T Small Business Technology Poll, reveló que las pequeñas empresas son cada vez más dependientes de las aplicaciones móviles, las redes sociales, el trabajo a distancia de sus empleados y las tecnologías wireless. La encuesta fue realizada en línea entre más de dos mil dueños de empresas de entre dos y cincuenta empleados de diferentes ciudades de Estados Unidos.

Pequeñas empresas y herramientas de internet. Aunque algunos directivos y empleados pudieran ser escépticos frente a las tecnologías y herramientas que ofrece internet antes de atreverse a utilizarlas, tal parece que cada día son más los que se atreven y que las pequeñas empresas han incorporado en un gran porcentaje varias de las posibilidades que ofrece internet así como los móviles o equipos Smartphone.

De acuerdo con la encuesta de AT&T, 72% de los pequeños negocios afirman utilizar aplicaciones móviles en su trabajo y 38% consideran que no podrían, o que sería un reto enorme, sobrevivir sin ellas. Lo que más motiva el uso de este tipo de aplicaciones es el ahorro de tiempo, el incremento en la productividad y la reducción en los costos.

El trabajo a distancia sigue dando de qué hablar. En las pequeñas empresas también está jugando un papel fundamental: 40% de estas afirman que todos sus empleados utilizan dispositivos inalámbricos o tecnologías wireless para trabajar desde lugares diferentes a la oficina. Y las tecnologías wireless han llegado a un punto en el que casi todos las han incorporado a su vida laboral.

La importancia de tener una aplicación móvil.

Es importante tener una aplicación móvil si se quiere prosperar como empresa. Como todos sabemos al día de hoy, los dispositivos móviles tienen una continua presencia en nuestras vidas.

Nos ofrecen un gran abanico de posibilidades. Y un gran ejemplo es la aplicación (App) que todos podemos tener en nuestro Smartphone. Son muchas las aplicaciones móviles que utilizamos para nuestra vida diaria, ya sean para jugar o para que nos hagan la vida más fácil y cómoda.

Son muchas las marcas y negocios de tecnología móvil que se han dado cuenta de lo importante que es para su futuro estar presentes en el día a día de sus actuales y potenciales consumidores, el modo más efectivo de conseguirlo es a través de nuestro dispositivo móvil y sus apps móviles. El desarrollo de las aplicaciones móviles ya es un hecho, es el presente con el que cuentan ya muchos negocios y empresas.

Hace unos años y hoy en día también la empresa que no tuviera página web estaba perdiendo difusión, ahora el que no tiene presencia en una app móvil va por el mismo camino.

Ventajas de tener una aplicación móvil para tu empresa

Tu empresa estará disponible las 24 horas del día. Gracias a los Smartphone, cualquier emprendedor puede estar conectado a su empresa las 24 horas del día y, a través de las Apps, aumentar la eficiencia, reducir costes o hacer crecer las ventas y notoriedad.

Dirígete a consumidores, clientes, profesionales o inversores. Las Apps móviles ya no son solo una herramienta habitual para los consumidores, podrás crear aplicaciones adaptadas a cada grupo de personas específico que formen parte de tu negocio.

Una sola persona puede manejar todos los aspectos de la aplicación. Al limitar el tamaño de la base de usuarios, es más fácil que una sola persona maneje aspectos como el marketing, el desarrollo y la atención al cliente, asistida muchas veces por algunos trabajadores freelance.

Teniendo en cuenta que se trata de un mercado que está en pleno crecimiento, de hecho el desarrollador de aplicaciones está entre los perfiles digitales más demandados en los próximos años.

Un freelance especializado en este ámbito podrá hacer el diseño de tus aplicaciones y realizar las pruebas y correcciones necesarias para que funcionen adecuadamente.

Después podrás ser el que maneje todos los aspectos relacionados con la gestión y el seguimiento de la aplicación.

1.1 HISTORIA DEL SMARTPHONE.

El término Smartphone es un término comercial con el que se denomina a un teléfono móvil en el que se pueden instalar programas o aplicaciones con las que mejorar las características útiles del teléfono como son el procesamiento de datos o la conectividad.

El apellido inteligente hace referencia a la posibilidad de ser creador de contenido y no sólo receptor o reproductor mediante un teclado físico QWERTY o una pantalla táctil con la que interactuar con la interfaz y las aplicaciones.

La idea inicial del Smartphone o teléfono inteligente era de, básicamente, unir las funciones de un PDA (Personal Digital Assistant*)* con las de un teléfono para mayor comodidad y compactibilidad.

El primer dispositivo en cumplir con esta definición fue el IBM Simon, que tenía todas las funciones de un PDA de aquella época (1992) con capacidades telefónicas y de SMS, y una pantalla totalmente táctil la cual podía ser manipulada con el dedo, a diferencia de otros PDAs de esos tiempos que requerían un stylus.

Pero, sin embargo, este no era muy conveniente por su peso de 510 gramos y, también, que solo podía funcionar en 190 ciudades distribuidas en 15 estados de los Estados Unidos, lo cual no lo hacía conveniente para viajes.

El primer teléfono móvil en usar el término 'Smartphone' fue el Ericsson GS88 el cual era más avanzado y poseía funciones de correo electrónico, navegación web, reloj mundial, un teclado QWERTY físico, modo avión, puerto infrarrojo, conexión a PC, etc.

Es posible que el 'boom' de los Smartphone comenzó con el sistema operativo Windows Pocket PC (2000).

Los teléfonos y dispositivos que llegaron al mercado con el sistema operativo Windows Pocket PC como los de la marca HTC, los cuales tuvieron un gran auge en Europa con sus teléfonos Wallaby, Falcon e Himalaya, entre el 2002 y el 2004.

Otras compañías que tuvieron gran auge durante estos años tempranos del Smartphone fueron: Palm Inc, con su Palm OS y su gran línea de Smartphone y PDAs con este sistema operativo, y RIM (Research In Motion) con su famosa línea Blackberry y el Blackberry OS.

A continuación se muestra en la figura 1 un ejemplo de Smartphone de la marca Blackberry teniendo como característica principal un teclado querty.

Figura 1: Ejemplo de un Smartphone.

Sin duda el evento que cambio la percepción de lo que era un Smartphone fue el anuncio del iPhone y de iOS en 2007, revolucionando la industria de la telefonía móvil y de los Smartphone.

Este nuevo OS dio paso a Android OS de Google (el mayor competidor de iOS) lanzado unos meses después del anuncio del iPhone, presentando cambios en la interfaz y compitiendo también con Windows Phone OS, Blackberry OS, Symbian OS, etc.

Este último (Symbian OS, de Nokia) fue descontinuado en el 2012 a favor del Windows Phone 8 (Lumia) y de los teléfonos Nokia Lumias con este sistema operativo.

Actualmente existen muchos fabricantes y también muchos sistemas operativos, entre los más importantes están: Windows Lumia, RIM BlackBerry OS, APPLE IOS, Google Android.

1.2 CARACTERÍSTICAS DE LOS SMARTPHONE.

Este tipo de teléfonos destacan por una serie de características comunes independientemente de la marca del fabricante, aquí se resaltan las más importantes:

- Función multitarea a partir de un software específico y un hardware avanzado.
- Acceso a Internet vía Wifi o 3G (actualmente se están desplegando las redes 4G).

- GPS o sistema de posicionamiento global por satélite.

- Acelerómetros (instrumentos destinados a detectar y medir aceleraciones o cambios de posición del objeto en cuestión).

- Programas de navegación.

- Lectura y modificación de documentos a modo de procesador de texto.

- Reproductor multimedia (video, mp3, radio, fotografía).

- Cámara digital de fotos y video integrada.

Esto es un mínimo de características que todo Smartphone debería incluir, ya que el uso y el diseño de este dispositivo permiten realizar una serie de tareas que son las que aportan valor a la definición.

¿Qué es un Sistema operativo?

El sistema operativo es el principal programa que se ejecuta en toda computadora de propósito general, es un programa que se encarga de manejar los procesos básicos de un dispositivo permitiendo el uso de sus diferentes recursos.

El sistema operativo es el único programa que interactúa directamente con el hardware de la computadora.

Sus funciones primarias son:

a) **Abstracción**. Los programas no deben tener que preocuparse de los detalles de acceso a hardware, o de la configuración particular de una computadora. El sistema operativo se encarga de proporcionar una serie de abstracciones para que los programadores puedan enfocarse en resolver las necesidades particulares de sus usuarios.
   Un ejemplo de tales abstracciones es que la información está organizada en archivos y directorios (en uno o muchos dispositivos de almacenamiento).
b) **Administración de recursos**. Un sistema de cómputo puede tener a su disposición una gran cantidad de recursos (memoria, espacio de almacenamiento, tiempo de procesamiento, etc.), y los diferentes procesos que se ejecuten en él compiten por ellos. Al gestionar toda la asignación de recursos, el sistema operativo puede implementar políticas que los asignen de forma efectiva y acorde a las necesidades establecidas para dicho sistema.
c) **Aislamiento**. En un sistema multiusuario y multitarea cada proceso y cada usuario no tendrá que preocuparse por otros que estén usando el mismo sistema —Idealmente, su experiencia será la misma que si el sistema estuviera exclusivamente dedicado a su atención (aunque fuera un sistema menos poderoso). Para implementar correctamente las funciones de aislamiento hace falta que el sistema operativo utilice hardware específico para dicha protección.

Sistemas Operativos Móviles.

El Sistema Operativo (SO) móvil de un teléfono o tableta significa la interacción real con lo que podemos hacer a partir de las capacidades del hardware que conforman un equipo.

A manera de traductor, esta plataforma interpreta lo que el usuario quiere que la terminal realice y cada vez, lo ejecuta con mayor inteligencia.

Una de las cualidades más atractivas de un sistema operativo móvil es la rapidez con la que en general se desempeña.

No precisa apagar el equipo completamente, sino dejarlo en un estado de suspensión para ahorrar energía, las aplicaciones se lanzan en pocos segundos, la instalación es transparente para el usuario y muchos periféricos son actualmente compatibles con los dispositivos más comunes.

La única diferencia con una PC tradicional es que todavía no soportan aplicaciones robustas como podrían ser las enfocadas en diseño o edición de video profesional.

No todos los sistemas operativos son iguales, por lo que un programa que corre en un sistema operativo específico, probablemente no funcionará en otro. Es muy importante conocer las características de los diferentes sistemas operativos para evaluar si se ajusta o no a nuestras necesidades.

Algunos ejemplos de sistemas operativos para celulares son:

Android, Windows Mobile, IOS, BlackBerry OS y otros desarrollados específicamente para una u otra marca de celulares.

1.3 SISTEMAS OPERATIVOS DE LOS SMASTPHONE.

Los Sistemas Operativos más importantes que usan los Smartphone son los siguientes:

Symbian OS.

El sistema operativo Symbian era el más utilizado por la mayoría de los modelos del tipo Smartphone, debido a que era uno de los softwares que tenían la mayor cantidad de herramientas para este tipo de equipos, fruto de haber nacido como resultado de la alianza de importantes compañías de telefonía celular. Con los años, la empresa Nokia adquirió el total de las acciones de la compañía, convirtiéndose en el único propietario con el fin de crear la Fundación Symbian para convertir este software en un sistema operativo de código abierto. La Serie 60 de Symbian era de los más utilizados, por supuesto en móviles de la marca Nokia. Una de las características más interesantes del sistema operativo Symbian reside en que cuenta con seis interfaces de usuario, opera en ROM y ha sido creado con el fin de ahorrar batería. En su última versión lanzada, Symbian OS permitía la conectividad con diferentes dispositivos a través de Bluetooth, además había mejorado la calidad de sus gráficos 3D, incorporo nuevas funciones de seguridad para sus usuarios y agrego compatibilidad con otros sistemas de diversas cámaras digitales que tenían hasta 2 megapíxeles.

BlackBerry OS.

El sistema operativo BlackBerry OS fue desarrollado por la compañía Research in Motion (RIM), en función de los equipos del tipo Handheld, y cuya característica principal se centra en ser un software diseñado fundamentalmente para un uso profesional, gracias a sus herramientas para correo electrónico y agenda.

BlackBerry OS es un sistema operativo que brinda una gran funcionalidad en el ámbito empresarial debido a una serie de características que lo hacen más que útil para este sector.

Además de permitir la utilización del teclado QWERTY completo, facilitando así el tipeo de textos, BlackBerry OS brinda la posibilidad de tener acceso a las cuentas de correo electrónico y navegación por Internet en tiempo real. Entre otras características principales de este sistema operativo, cabe

destacar que permite la sincronización con herramientas tales como Novell GroupWise, Microsoft Exchange Server y Lotus Notes.

Microsoft Windows Mobile.

El sistema operativo Windows Mobile fue desarrollado por la compañía Microsoft con el fin de ofrecer un software similar al conocido Windows OS, incluyendo una suite de aplicaciones que fuera de uso exclusivo para dispositivos móviles, tales como teléfonos celulares, PDA y otros. En sus comienzos fue denominado Windows CE y Pocket PC y se encontraba en una escasa cantidad de celulares disponibles en el mercado, aunque con el paso de los años Windows Mobile supo ganarse un lugar de prestigio entre los fabricantes y usuarios de telefonía celular. Entre las características principales de Windows Mobile, cabe destacar que se trataba de un sistema operativo similar en su forma de uso al Windows que se utiliza en las PC, lo que lo convierte en un software con una interfaz gráfica familiar para el usuario. Lo más apreciado por los usuarios de móviles que incluyen el sistema operativo de Microsoft ha sido hasta el momento la posibilidad de utilizar importantes herramientas pertenecientes a las suites Office Mobile, Outlook Mobile e Internet Explorer.

Apple IOS.

El sistema operativo IOS es de uso exclusivo de los dispositivos iPod, Ipad y iPhone, ya que fue desarrollado por la compañía Apple para ser utilizado en sus productos, para lo cual fue diseñado en base a una variante del kernel de Mac OS X. Una de las características que destacan al IOS, en la cual han trabajado arduamente sus desarrolladores, radica en ofrecer la sensación de velocidad durante su uso, mediante algunos trucos de programación.

Obviamente IOS permite la utilización de una serie de aplicaciones que se utilizan en los sistemas Mac, hecho por el cual ha logrado ganarse una gran popularidad entre aquellos usuarios fanáticos de la firma Apple.

Por otra parte la mayoría de los dispositivos que utilizan IOS reciben de manera constante actualizaciones que mejoran el funcionamiento y la seguridad del software.

Google Android.

El sistema operativo para móviles Android fue creado por Android Inc., empresa que en 2005 fue comprada por Google, Android se desarrolla en base al kernel Linux, con el fin de ofrecer un software gratuito de código abierto, por lo que permite la utilización de gran cantidad de aplicaciones y una importante posibilidad para customizar el equipo. Google Android permite la utilización de gráficos 2D y 3D, posee soporte multimedia para imágenes, audio y video, como así también para pantallas táctiles, además de ofrecer una serie de herramientas de código abierto, como por ejemplo su navegador web integrado basado en WebKit.

Windows Phone.

Microsoft presentó el Windows Phone en el Mobile World Congress 2010 que se celebró en Barcelona. En Octubre de 2010 el sistema operativo Windows Phone 7 sale a la venta con el apoyo de 4 fabricantes que crearon 10 modelos que se pusieron a la venta en 30 países a través de 60 operadores. La versión del Windows Phone 7.5 apenas comenzaba a ser trabajada y finalmente fue presentada el 24 de Mayo de 2011 con algunos avances esperados como el necesario Multitasking, un People Hub repotenciado con Facebook y Twitter, la inclusión del Internet Explorer 9 como navegador de línea y muchas más características. Algo que hace único al Windows Phone es su interfaz de usuario, llamada “Estilo Metro” porque ha sido inspirada en la tipografía e iconografía de los sistemas de transporte globales.

La pantalla de inicio está llena de “tiles” dinámicas (mosaicos) que comparten a primera vista parte del contenido. Esto le brinda gran dinamismo a la interfaz.

CAPITULO 2

SISTEMA OPERATIVO ANDROID

SISTEMA OPERATIVO ANDROID

2.1 Historia del Sistema Operativo Android.

Android es un sistema operativo y una plataforma software, basado en Linux para teléfonos móviles. Además, también usan este sistema operativo tablets, netbooks, reproductores de música e incluso PC's. Android permite programar en un entorno de trabajo (framework) de Java, aplicaciones sobre una máquina virtual Dalvik (una variación de la máquina de Java con compilación en tiempo de ejecución).

Además, lo que le diferencia de otros sistemas operativos, es que cualquier persona que sepa programar puede crear nuevas aplicaciones, widgets o incluso, modificar el propio sistema operativo, dado que Android es de código libre, por lo que sabiendo programar en lenguaje Java, va a ser muy fácil comenzar a programar en esta plataforma.

Fue desarrollado por Android Inc., empresa que en 2005 fue comprada por Google, aunque no fue hasta 2008 cuando se popularizó, gracias a la unión al proyecto de Open Handset Alliance, un consorcio formado por 48 empresas de desarrollo hardware, software y telecomunicaciones, que decidieron promocionar el software libre. Pero ha sido Google quien ha publicado la mayor parte del código fuente del sistema operativo, gracias al software Apache, que es una fundación que da soporte a proyectos software de código abierto. Dado que Android está basado en el núcleo de Linux, tiene acceso a sus recursos, pudiendo gestionarlo, gracias a que se encuentra en una capa por encima del Kernel, accediendo así a recursos como los controladores de pantalla, cámara, memoria flash…

Arquitectura de Android.

El siguiente gráfico de la figura 2, muestra la arquitectura de Android. Como se puede ver está formada por cuatro capas. Una de las características más importantes es que todas las capas están basadas en *software* libre.

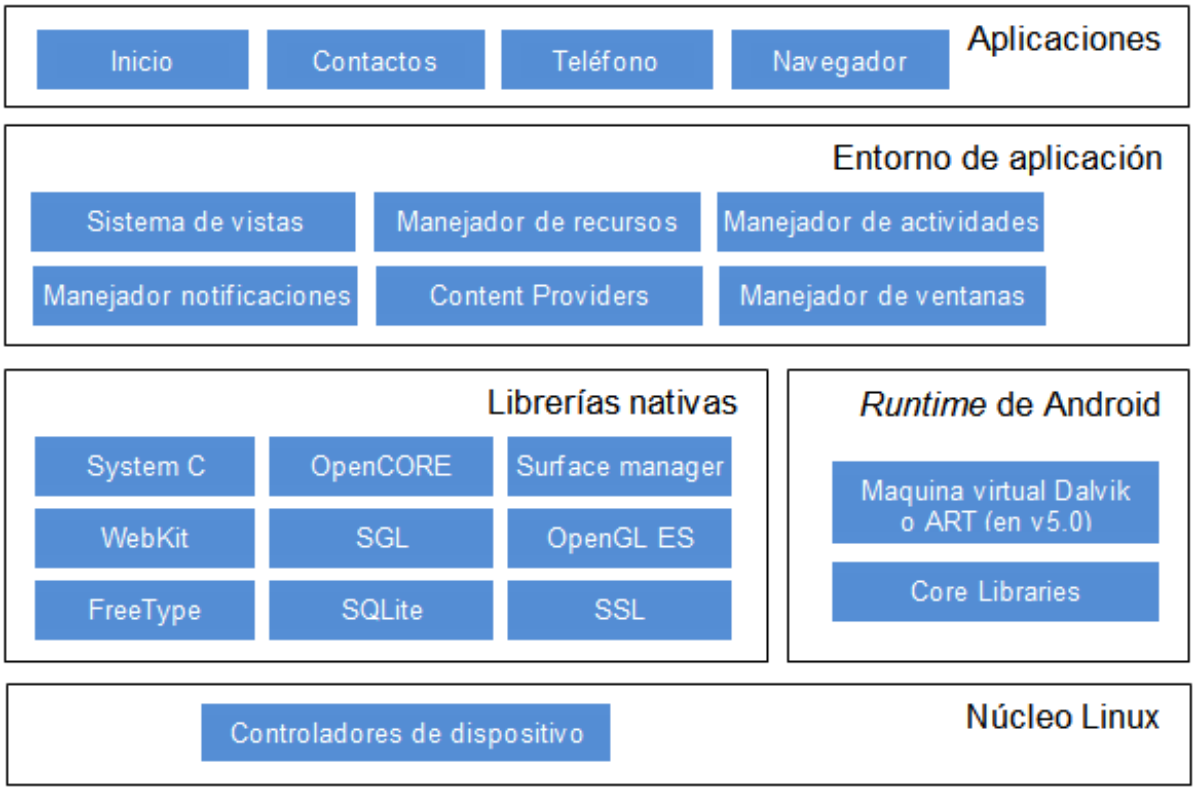

Figura 2: Arquitectura de Android

1. El Núcleo de Linux.

El núcleo de Android está formado por el sistema operativo Linux versión 2.6. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos.

Esta capa del modelo actúa como capa de abstracción entre el hardware y el resto de la pila. Por lo tanto, es la única que es dependiente del hardware.

2. Runtime de Android.

Está basado en el concepto de máquina virtual utilizado en Java. Dado las limitaciones de los dispositivos donde ha de correr Android (poca memoria y procesador limitado) no fue posible utilizar una máquina virtual Java

estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones.

Algunas características de la máquina virtual Dalvik que facilitan esta optimización de recursos son: que ejecuta ficheros Dalvik ejecutables (.dex) – formato optimizado para ahorrar memoria. Además, está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Delega al kernel de Linux algunas funciones como threading y el manejo de la memoria a bajo nivel.

A partir de Android 5.0 se reemplaza Dalvik por ART. Esta nueva máquina virtual consigue reducir el tiempo de ejecución del código Java hasta en un 33%.

También se incluye en el Runtine de Android el "core libraries" con la mayoría de las librerías disponibles en el lenguaje Java.

3. Librerías nativas.

Incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto. Algunas de estas librerías son:

- **System C library:** una derivación de la librería BSD de C estándar (libc), adaptada para dispositivos embebidos basados en Linux.
- **Media Framework:** librería basada en PacketVideo's OpenCORE; soporta codecs de reproducción y grabación de multitud de formatos de audio vídeo e imágenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.
- **Surface Manager:** maneja el acceso al subsistema de representación gráfica en 2D y 3D.
- **WebKit:** soporta un moderno navegador Web utilizado en el navegador Android y en la vista Webview. Se trata de la misma librería que utiliza Google Chrome y Safari de Apple.
- **SGL**: motor de gráficos 2D.

- **Librerías 3D:** implementación basada en OpenGL ES 1.0 API. Las librerías utilizan el acelerador harware 3D si está disponible, o el software altamente optimizado de proyección 3D.
- **FreeType:** fuentes en bitmap y renderizado vectorial.
- **SQLite:** potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.
- **SSL:** proporciona servicios de encriptación Secure Socket Layer (capa de conexión segura).

4. Entorno de aplicación.

Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones, etc.). Esta capa ha sido diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas (sujetas a las restricciones de seguridad). Este mismo mecanismo permite a los usuarios reemplazar componentes. Los servicios más importantes que incluye son:

- **Views**: extenso conjunto de vistas, (parte visual de los componentes).
- **Resource Manager:** proporciona acceso a recursos que no son en código.
- **Activity Manager:** maneja el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.
- **Notification Manager:** permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.
- **Content Providers:** mecanismo sencillo para acceder a datos de otras aplicaciones (como los contactos).

5. Aplicaciones.

Este nivel está formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema.

Normalmente las aplicaciones Android están escritas en Java. Para desarrollar aplicaciones en Java podemos utilizar el Android SDK. Existe otra opción consistente en desarrollar las aplicaciones utilizando C/C++. Para esta opción podemos utilizar el Android NDK (Native Development Kit).

2.2 Evolución y características del sistema operativo Android

El sistema operativo Android, al igual que los propios teléfonos móviles, ha evolucionado rápidamente, acumulando una gran cantidad de versiones, desde la 1.0 Apple Pie para el QWERTY HTC G1 hasta la 5.0 Lollipop.

Android 1.0 Apple Pie.

La primera versión se presenta como un sistema operativo móvil totalmente gratuito y Open Source. Estaba desarrollado sobre el kernel de Linux 2.6. Los primeros terminales con Android estarían disponibles durante el segundo trimestre del año 2008. Google realizó su propia presentación oficial del SDK el 23 de septiembre.
La primera versión comercial tenía mucho margen de mejora y apenas inquietó a la competencia, pero ya introducía algunos conceptos que años después son un estándar de los sistemas operativos móviles:

- Menú desplegable de notificaciones
- *Widgets* de escritorio
- Android Market, la tienda de Apps (no contaba con ningún sistema de pago para usuarios. Todo el catálogo era gratuito)
- Integración con Google Mail, Contactos y Calendario.
- Navegador, Mapas, Google Talk, reproductor de YouTube y soporte para cámaras-

Android 1.1 Petit Four.

En febrero de 2009 llegó la primera actualización para Android, unos tres meses después del lanzamiento del G1. La versión 1.1 fue dedicada básicamente a reparar errores y a implementar las actualizaciones “over the air” que hasta ese momento ninguna plataforma estaba haciendo.

Android 1.5 Cupcake.

Android 1.5 es más conocido por su nombre en clave, Cupcake, fue la primera versión en utilizar nombre de postres. Cada versión después de Cupcake ha sido nombrada con un nombre de postre continuando el orden alfabético.

En esta versión comenzamos a ver algunos cambios en la interfaz de usuario, por poco que se puedan apreciar, como son los cambios en la barra del buscador y en la barra del menú, también cambió el logo del navegador. Las primeras versiones de Android no contaban con un teclado virtual, ya que el G1 disponía de un teclado físico, en la 1.5 se introdujo el teclado virtual (teclado QWERTY virtual) coincidiendo con la salida del primer Android con pantalla táctil y sin teclado físico, el HTC Magic.

Desde el principio Google introdujo la posibilidad de que desarrolladores terceros pudieran crear sus propios teclados virtuales, algo que a día de hoy sigue diferenciándolo de IOS y Windows Phone.

El primer teclado virtual integrado en Android fue considerado un poco malo, cosa que provocó que fabricantes como HTC se pusieran manos a la obra para desarrollar reemplazos del mismo.

- **Widgets:** Google no se había dado cuenta todavía del potencial de los widgets ya que hasta ese momento no había colgado el SDK para desarrolladores y había muy poca variedad. Eso cambió en Cupcake permitiendo que widgets de terceros estuvieran disponibles para los usuarios.
- **Mejoras en el portapapeles:** La plataforma no ofrecía desde el principio la posibilidad de copiar y pegar en las ventanas del navegador o Gmail. Se integró por primera vez en esta versión y fue terminada en posteriores versiones.
- **Captura y reproducción de vídeo:** La primera versión de Android no ofrecía la posibilidad de grabar vídeo ni de reproducción, si no que se integró con la 1.5. Los fabricantes comenzaron a crear sus propias interfaces para la cámara añadiendo soporte para escenas adicionales, modos, opciones y la posibilidad del enfoque presionando sobre la pantalla táctil.

- **Algunos cambios más:** En versiones anteriores no se podía eliminar ni archivar mensajes de correo electrónico en múltiples cuentas. Se agrega además soporte para subir vídeos a YouTube y Picasa. Finalmente se añade la sincronización a través de múltiples cuentas. Además fue la primera versión que introdujo el copy & paste.

Android 1.6 Donut.

Con la llegada de Donut vino el soporte para redes CDMA haciendo que Android llegara a Estados Unidos y Asia.

La mejora más significativa fue la posibilidad de correr el sistema operativo en múltiples resoluciones de pantalla y relaciones de aspecto, a raíz de esta actualización es cuando podemos disfrutar hoy en día de pantallas con resolución QVGA, HVGA, WVGA, FWVGA, QHD y 720p.

Donut también introdujo la búsqueda rápida, generalmente conocida en el mundo del móvil como búsqueda universal. Antes de Donut la búsqueda se limitaba a Internet pero con las mejoras introducidas en la versión 1.6 se podría buscar además contenido propio del teléfono (contactos, aplicaciones, etcétera) incluso por voz, todo desde el mismo widget.

Otras mejoras: Nuevo diseño de Android Market al estilo de la mascota de Android en color blanco y verde, introducción de listas de aplicaciones gratuitas y de pago. Nueva interfaz de la cámara con mejor integración en la galería.

Android 2.0/2.1 Eclair.

Después de un año después del lanzamiento del G1 en noviembre del 2009. Fue ofrecido en exclusiva con Verizon y el Motorola Droid, un teléfono que marcó un antes y después para Android y con el que Motorola volvió a ser la gran marca que fue.

El Droid de Motorola fue el teléfono más potente que se había visto en el mercado hasta la fecha, con una pantalla con resolución de 854 x 480. Pero no solo fue el Droid el que impulsó las ventas de Android sino las mejoras que se introdujeron en la versión 2.0.

Después del Droid/Milestone prácticamente la mayoría de teléfonos lanzados llegaron con Android 2.1, una corrección de errores y que Google no renombró dejándola con el nombre de Eclair.

- **Apoyo de varias cuentas:** Por primera vez se podrían añadir varias cuentas en el mismo dispositivo con acceso al correo electrónico y a los contactos de cada una, además también se introdujo soporte para cuentas de Exchange. También se abrió la puerta de las sincronizaciones automáticas para los contactos gracias a la información compartida entre los tipos de cuenta; Facebook fue la primera en integrar esta funcionalidad.
- **Google Mapas Navegación:** Google Maps Navigation fue publicado junto con la versión 2.0 y fue un paso adelante para integrar un sistema de navegación de automóviles en el móvil con vistas en 3D, guía de voz e información de tráfico de forma completamente gratuita. Hoy en día sigue siendo una de las mejores opciones para tu teléfono.
- **Contacto rápido:** Eclair agregó una barra de contacto rápido, una barra de herramientas desplegable que se utiliza para realizar múltiples funciones de manera rápida (mandar email, mensajes, llamar, etc.)
- **Nuevas mejoras en el teclado:** El Droid también fue lanzado con teclado QWERTY pero Google aprovechó para mejorar un poco más el teclado virtual.
- **De voz a texto:** Los usuarios podían dictar a su teléfono y éste lo transcribía a texto (TTS). En Android 2.1 se reemplazó la tecla de la coma en el teclado en pantalla por un micrófono para utilizar dicho servicio.
- **Renovación del navegador:** Google añadió soporte HTML5, incluyendo vídeo pero solo en modo pantalla completa. Aunque seguía sin estar disponible la posibilidad de multitouch pero se agrega la posibilidad de zoom tocando dos veces.
- **Fondos de pantalla animados:** Por primera vez aparecieron los fondos de pantalla animados en Android, en lugar de usar una imagen estática.
- **Nueva pantalla de bloqueo:** Android 2.0 introdujo una nueva pantalla de bloqueo. Para desbloquear el teléfono o silenciarlo tan solo teníamos que deslizar el dedo por la misma en una dirección u otra. Android 2.1 cambió la pantalla de bloqueo y la hizo más al estilo iOS.

Android 2.2 Froyo.

Lanzado a mediados de 2010 trajo una gran cantidad de cambios. La pantalla de inicio fue rediseñada, se ampliaron los 3 paneles existentes desde el inicio a 5 con un nuevo grupo de accesos directos dedicados y se agregaron unos puntos para saber en cada momento en la pantalla donde nos encontrábamos. El Nexus One fue el primer teléfono en actualizarse a Android 2.2.

Froyo también introducía una galería completamente rediseñada con imágenes en 3D que aparecen al inclinar el teléfono. Además se introdujo soporte para hotspot móvil Wifi (compartir la conexión 3G), algo que muchas compañías decidieron desactivar o activarla con la opción de pagar un coste extra.

Se mejoró también el soporte para copiar y pegar en Gmail incorporando también Microsoft Exchange (servidor de comunicación basado en el correo electrónico), mejoras en la video-llamada y en la memoria siendo terminales más potentes.

En esta versión se agregó la posibilidad de poner una contraseña o PIN en la pantalla de bloqueo para los usuarios que no les gustaba el patrón de desbloqueo.

Android 2.3 Gingerbread.

Un año y medio después del lanzamiento de Froyo y el Nexus One (el primer teléfono de Google fabricado por HTC), Google volvió con un nuevo móvil de marca propia pero esta vez en colaboración con Samsung, el Nexus S y aprovechó para lanzar la nueva versión del sistema operativo, Android 2.3 Gingerbread. Con el Nexus S llegó la pantalla curvada y el fin del trackball. Gingerbread fue una actualización menor en muchos sentidos pero trajo algunos cambios importantes en la interfaz de usuario.

- **Mejor control en copiar y pegar:** Se añade en esta versión la posibilidad de seleccionar el texto que queremos copiar y pegar. Anteriormente solo se podía copiar el contenido de las cajas

completas. Se agregan unas pestañas para seleccionar el texto que queremos copiar.

- **Teclado mejorado:** Nuevamente Google pone su empeño en mejorar el teclado, cambios en el diseño y de coloración además del soporte multitouch.
- **Maximización de la batería y herramientas de gestión de desarrollo:** Google pecó de ser demasiado permisivo con la multitarea y esto hacía mella en la duración de la batería. Se instaló una herramienta para la gestión de la batería que informa de qué aplicaciones están consumiendo la batería.
- **Soporte para cámara frontal (video online):** Gingerbread fue la primera versión en integrar soporte para varias cámaras, aunque la opción de video chat en Google Talk no llegaría hasta mediados de 2010. El Nexus S ya dispondría de cámara frontal, aunque en un principio solo servía para tomar fotos con ella.
- **Juegos:** La nueva versión dio más libertad a los desarrolladores para poder escribir código más rápido y desarrollar juegos con gráficos en 3D que hasta entonces no disponía Android. Google estaba perdiendo la batalla de los juegos con iOS y tenía que reaccionar.
- **Otras características:** Apoyo a la tecnología NFC integrada en una antena incrustada en la tapa de la batería. En un principio es usada como si de un código QR se tratara para escanear sitios en Google Places pero más tarde Google presenta Google Wallet, una aplicación de pago utilizando la tecnología NFC integrada en el Nexus S.

Android 3.0 Honeycomb.

La versión de Android para tablets, que presentó de la mano de Motorola junto con el Xoom. Cambio de color, del verde típico de Android al azul que se utilizó para la batería, el widget del reloj, indicadores de señal y algunas otras características de la interfaz.

- **El final de los botones físicos:** Se integra una barra en la parte inferior de la pantalla con una serie de botones virtuales que hacen que no se necesiten botones dedicados. Es el fin de los botones físicos, tendencia que continuará con Android 4.0 ICS.
- **Multitarea mejorada.** La multitarea ha sido mejorada gracias al diseñador Matías Duarte, ex diseñador de web OS contratado por

Google. Así podemos cambiar de aplicación dejando las demás en espera en una columna.

- **Una nueva barra para las aplicaciones:** se introduce el concepto de barra de acción, una barra permanente situada en la parte superior de cada aplicación que los desarrolladores pueden utilizar para mostrar las opciones de acceso frecuente, menús, etc. Es como una barra de estado dedicada a cada aplicación.
- **Otras características:** soporte Flash y Divx, integra Dolphin (navegador mejorado), widgets y homepage personalizable.

Android 3.1 y 3.2 fueron versiones de mantenimiento, prueba de ello es que Google no las renombró y continuaron llamándolas Honeycomb. Aunque algunas mejoras introducidas en estas actualizaciones se han ido implementando en la mayoría de tablets con Android 3.0 del mercado, como la posibilidad de modificar el tamaño de los widgets al presionar sobre ellos.

Android 4.0 Ice Cream Sandwich.

Llegamos a la última versión del sistema operativo de Google, Android 4.0 Ice Cream Sandwich. Ha sido lanzada junto con el Galaxy Nexus, el nuevo Smartphone Google fabricado por Samsung. Ice Cream Sandwich toma prestadas muchas características de Honeycomb como los botones virtuales o la transición de tonos verdes a azules, la multitarea con una lista desplegable de miniaturas y las barras de acción dentro de las aplicaciones. Comprensiblemente sigue siendo Multiplataforma (tablets, teléfonos móviles y netbooks).

- **Teclado virtual modificado:** esta vez incluye un sistema de corrección mucho más avanzado que subraya en color rojo las palabras mal escritas e incorpora también un diccionario. Con Ice Cream Sandwich por primera vez se modifica el tipo de letra. Droid fue la fuente utilizada desde la versión 1.0 y ahora se modifica por Roboto, una fuente que ha sido diseñada para aprovechar la mayor resolución de las pantallas de hoy en día.
- **La pantalla de notificaciones también ha recibido una pequeña actualización** con las notificaciones individuales extraíbles que permiten deslizar cualquier notificación fuera de la pantalla y acceder a ella. Barra de estado

- **Pantalla de inicio:** la pantalla adopta muchos cambios de los que se introdujeron en Honeycomb pero añade además algunas características nuevas como la posibilidad de crear carpetas con solo arrastrar un icono a otro. Además la pantalla principal recibe una bandeja de favoritos que puede ser configurada por el usuario. Y mejora en el soporte 3D^.
- **NFC:** El soporte de la tecnología NFC ya había sido promocionado fuertemente con el lanzamiento de Gingerbread y el Nexus S aunque no había prácticamente ninguna aplicación. En Ice Cream Sandwich se busca potenciar el uso de NFC con una nueva característica para transferencia de datos entre dos teléfonos con solo tocarlos.
- **Desbloqueo facial:** Además del bloqueo con contraseña y con patrón de desbloqueo se ha agregado la opción del desbloqueo facial.
- **Análisis de los datos:** Se añade un gestor para el uso de los datos en el que se informa de las aplicaciones que consumen más datos, se puede ver el uso total desglosado en un periodo de tiempo configurable por el usuario.
- **Nuevo calendario y aplicaciones de correo electrónico**. El correo electrónico de Gmail ha sido revisado en Ice Cream Sandwich con nuevos diseños y con la incorporación de la barra de acción. El calendario está unificado, se pueden ver todos los eventos de todas las cuentas en el mismo calendario.

Android 4.1.2 Jelly Bean.

Las novedades son pocas, exceptuando las típicas correcciones de bugs y mejoras en la estabilidad. No obstante, en la Nexus 7 esta actualización incorpora la posibilidad de rotar la pantalla de inicio, algo que venía deshabilitado de fábrica y que solo se podía conseguir mediante aplicaciones de terceros.

Otra de las funciones que incluye esta actualización es poder expandir las notificaciones enriquecidas de Jelly Bean con un sólo dedo, y no con dos como se hacía antes, que era algo incómodo y poco natural, ahora con un sólo dedo saldrá desplegada toda la información

Android 4.4 Kitkat.

El propósito principal de KitKat es el comienzo de una estrategia de Google para llevar la última versión de Android a todos los dispositivos Android, tanto los de calidad superior como los de baja gama.

La fragmentación hace que los dispositivos más baratos parezcan ser menos atractivos incluso teniendo en cuenta sus precios bajos, y deja a muchos ofendidos por no poder obtener las últimas funcionalidades sin pagar un alto costo.

Google intenta solucionar esto con KitKat, al reducir el tamaño del sistema operativo en un 16 por ciento, permitiendo que pueda correr en dispositivos con apenas 512 MB de RAM. Esto significa que los dispositivos económicos y los teléfonos destinados a los mercados emergentes, que es exactamente lo que Google está buscando, puedan correr KitKat, en lugar tener una versión como Android Gingerbread (2.3) o Ice Cream Sandwich (4.0).

- **Llamadas**. La mejorada aplicación de llamadas telefónicas analizará los contactos con que hablas con más frecuencia y priorizará automáticamente tu directorio. Además, la integración con las aplicaciones de Google como Mapas te permitirán buscar lugares y negocios cercanos directamente desde está aplicación de marcado.
- **El identificador de llamadas también recibe mejoras.** Para las llamadas entrantes que no coinciden con un número de teléfono en tus contactos, las aplicaciones de Google escanearán y mostrarán los negocios locales que figuran en Google Maps.
- **El modo de inmersión**. Limpia el desorden en la pantalla al ocultar automáticamente todo menos la única cosa que estás viendo (como una foto, mapa, o juego). Una nueva aplicación Hangouts consolida todos sus mensajes de texto, mensajes multimedia, conversaciones y video llamadas en un solo lugar.
- **Print.** Ahora podrás imprimir fotos, documentos y páginas Web desde tu teléfono o tableta. Cualquier impresora conectada a Google Cloud Print será compatible, junto con las impresoras HP ePrint y otras impresoras con aplicaciones de Google Play.

- **Control remoto.** Si tienes un dispositivo con Android con un emisor Infrarrojo, podrás utilizar aplicaciones que hacen que tu dispositivo funcione como un control remoto de televisión.
- **Subtítulos.** Los subtítulos llegan a la mayoría de las aplicaciones.
- **NFC**. Las funcionalidades NFC ahora trabajarán con más operadores.
- **Funcionalidades en Sensores.** Los desarrolladores de aplicaciones pueden aprovechar los nuevos sensores de detección y monitoreo de pasos.
- **Otros Detalles**. Detalles más pequeños van desde un color uniforme de la barra de estado a un nuevo tipo de letra condensada.

Android 5.0 Lollipop.

Uno de sus rasgos más llamativos ha sido la inclusión Material Design, un nuevo lenguaje de diseño que unificará la experiencia de uso en cualquier tipo de dispositivo.

- Nuevo diseño basado en Material Design que logra un flujo de trabajo más fluido
- Interfaz que se adapta a cualquier tamaño de dispositivo
- Mejoras en el apartado visual con efectos y animaciones que proporcionan una interacción más real
- Renovado sistema de notificaciones inteligente
- Interesante vista multitarea que muestra capas con las diferentes aplicaciones abiertas
- Aumento del rendimiento energético (enmarcado en el Proyecto Volta), posibilitando una mayor autonomía de la batería
- Función Android Smart Lock, que permite emparejar un dispositivo Android con otro, ya sea un reloj inteligente o un automóvil.
- Modo "Invitado" para que puedas prestar el dispositivo sin que otros usuarios tengan acceso a tu información privada.
- El entorno de ejecución de aplicaciones ART (Android RunTime) pasa a ser el predeterminado
- Soporte para sistemas de 64 bits

La versión de Android más usada actualmente.

La versión de Android más usada por los usuarios de telefonía móvil con sistema operativo Android 4.1,4.2 y 4.3 llamada Jelly Bean y seguida por la versión 4.4 llamad Kitkat de acuerdo a cifras de Google. A continuación en la figura 3 se muestran las versiones del software de Android mas usados actualmente.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.4%
4.1.x	Jelly Bean	16	18.4%
4.2.x		17	19.8%
4.3		18	6.3%
4.4	KitKat	19	39.7%
5.0	Lollipop	21	1.6%

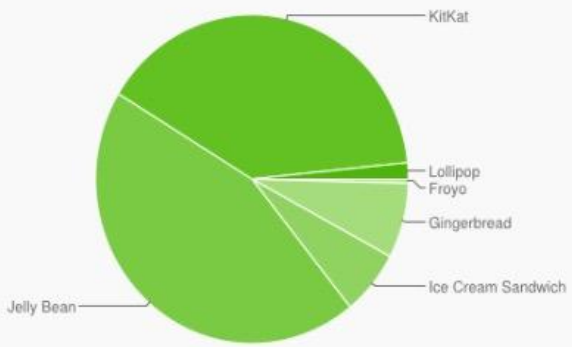

Figura 3: Imagen que muestra las versiones de Android más usadas a nivel mundial.

2.3 PROGRAMAS PARA WINDOWS PARA DESARROLLAR APLICACIONES EN ANDROID

Existen ya en el mercado varios servicios que permiten crear desde cero y paso a paso aplicaciones para las diferentes plataformas móviles, utilizando los asistentes de edición para personalizar el diseño de la interfaz y configurar las funciones de aplicaciones básicas.

Una vez terminada nuestra creación, los mismos asistentes se encargan de compilar nuestro proyecto en una aplicación nativa para la plataforma elegida, guiándonos finalmente en el proceso de publicación de la misma en la tienda de aplicaciones.

A continuación se verán algunos generadores de aplicaciones móviles existentes y que posibilidades ofrece cada uno:

Creapp.

Creapp es uno de los asistentes para crear aplicaciones móviles desde cero más sencillo de usar. Dispone además de un potente editor visual que permite crear y configurar diferentes tipos de aplicaciones en tan solo 5 pasos y sin conocimientos de programación.

Creapp ofrece un buen catálogo de plantillas prediseñadas para ayudar a los iniciados en este interesante campo a crear nuestra primera aplicación.

A partir de la plantilla elegida el servicio permite personalizarla para adaptarla a nivel gráfico y visual a nuestra imagen corporativa.

Creapp permite la posibilidad de integrar interesantes prestaciones en algún proyecto como sistemas de pedidos online, pasarela de pago para tiendas virtuales y otro tipo de servicios de gran utilidad para medianas y pequeñas empresas.

Finalmente, sólo queda exportar el proyecto a las diferentes plataformas existentes como Android, iOS o HTML5.

Apps Builder.

Apps Builder es otra interesante herramienta para diseñar y publicar tu propia aplicación móvil con cero conocimientos de programación y de forma rápida y sencilla.

La diferencia de este servicio con respecto al resto de asistentes es que su editor puede capturar los contenidos de nuestra página web para utilizarlo como base con la que desarrollar aplicaciones para los diferentes sistemas operativos móviles.

También dispone de una galería de 37 plantillas disponibles para el diseño de la interfaz de la aplicación, además de la posibilidad de configurar paso a paso y sin tocar una sola línea de código las características de tu aplicación añadiendo módulos como: Noticias, Foto, Vídeo, Podcasting, Tienda, Canal RSS, Facebook, Twitter, Sitio web, Radio, PDF, Texto, Código, Muro de chat, Contactos, Mapas, Mensaje de texto, Llamada y Correo.

Mobincube App Generador

Quizá el Mobincube App Generador sea uno de los generadores de aplicaciones móviles multiplataforma más completos dado el grado de personalización que permite. Ofrece un servicio muy completo de asistencia paso a paso a neófitos en el campo de la programación, con una buena galería de plantillas prediseñadas para los diferentes contenidos.
A nivel de edición gráfica y visual de la interfaz ofrece un mayor grado de libertad a la hora de decidir la imagen corporativa de nuestra aplicación.
En cuanto a integración de las funcionalidades en la aplicación, resulta bastante intuitiva y práctica y ofrece la posibilidad de exportar algún proyecto a las principales plataformas móviles como iOS, Android, HTML5, BlackBerry o Windows Phone.

App Inventor.

Es una nueva herramienta de programación visual de Google para dispositivos móviles con Android. Su método está basado en bloques visuales drag and drop, como los juegos de construcción infantiles, haciendo posible crear aplicaciones sin tener ninguna experiencia en programación. Con este libro podrá desarrollar aplicaciones capaces de detectar la ubicación del dispositivo, almacenar datos o tomar decisiones lógicas. En la segunda parte encontrará El manual del inventor, que le ayudará a comprender las bases del desarrollo y de la programación. Podrá inventar juegos, programas educativos, aplicaciones SMS o de control de robots, herramientas

totalmente personalizadas y mucho más. Tanto si es un programador novel como un desarrollador experimentado, con App Inventor construir aplicaciones es un juego de niños.

Android Studio

Es un entorno de desarrollo integrado para el sistema operativo Android lanzado por Google, diseñado para ofrecer nuevas herramientas para el desarrollo de aplicaciones, es un entorno de desarrollo integrado (IDE), basado en IntelliJ IDEA de la compañía JetBrains. Android Studio utiliza una licencia de software libre Apache 2.0, está programado en Java y es multiplataforma. Gracias a que cuenta con su sistema de emulación integrado, Android Studio permite ver los cambios que realizamos en nuestra aplicación en tiempo real, pudiendo además comprobar cómo se visualiza en diferentes dispositivos Android con distintas configuraciones y resoluciones de forma simultánea. Android Studio Ofrece:

-Un entorno de desarrollo claro y robusto.

-Facilidad para testear el funcionamiento en otros dispositivos.

-Asistentes y plantillas para los elementos de programación en Android.

-Un completo editor con muchas herramientas extra para agilizar el desarrollo de nuestras aplicaciones.

CAPITULO 3

ANDROID STUDIO

COMO ENTORNO DE DESARROLLO

3.1 CARACTERÍSTICAS DE ANDROID STUDIO

Android Studio es un entorno de desarrollo integrado (IDE), basado en IntelliJ IDEA de la compañía JetBrains.

Android Studio se ha mantenido en versión beta, pero desde el 8 de diciembre de 2014, en que se liberó la versión estable de Android Studio 1.0, Google ha pasado a recomendarlo como el IDE para desarrollar aplicaciones para su sistema operativo.

Las principales características que incluye Android Studio:

a) Soporte para programar aplicaciones para Android Wear (sistema operativo para dispositivos corporales como por ejemplo un reloj).
b) Herramientas Lint (detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador) para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.
c) Utiliza ProGuard para optimizar y reducir el código del proyecto al exportar a APK (muy útil para dispositivos de gama baja con limitaciones de memoria interna).
d) Integración de la herramienta Gradle encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de testing, compilación o empaquetado.
e) Nuevo diseño del editor con soporte para la edición de temas.
f) Nueva interfaz específica para el desarrollo en Android.
g) Permite la importación de proyectos realizados en el entorno Eclipse, que a diferencia de Android Studio (Gradle) utiliza ANT.
h) Posibilita el control de versiones accediendo a un repositorio desde el que poder descargar Mercurial, Git, Github o Subversión.

i) Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
j) Vista previa en diferentes dispositivos y resoluciones.
k) Integración con Google Cloud Platform, para el acceso a los diferentes servicios que proporciona Google en la nube.
l) Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo xml.

Ventajas de Android Studio.

Las ventajas de usar Android Studio son:

- Android Studio ha pasado a ser el entorno recomendado para el desarrollo de aplicaciones en Android, al tratarse de un IDE oficial de Google en colaboración con JetBrains (compañía de desarrollo software especializada en diseño de IDEs).
- Android Studio permite la creación de nuevos módulos dentro de un mismo proyecto, sin necesidad de estar cambiando de espacio de trabajo para el manejo de proyectos, algo habitual en Eclipse.
- Con la simple descarga de Android Studio se disponen de todas las herramientas necesarias para el desarrollo de aplicaciones para la plataforma Android.
- Su nueva forma de construir los paquetes .apk, mediante el uso de Gradle, proporciona una serie de ventajas más acorde a un proyecto Java:

    - Facilita la distribución de código, y por lo tanto el trabajo en equipo.
    - Reutilización de código y recursos.
    - Permite compilar desde línea de comandos, para aquellas situaciones en las que no esté disponible un entorno de desarrollo.
    - Mayor facilidad para la creación de diferentes versiones de la misma aplicación, que proporciona numerosas ventajas como puede ser la creación de una versión de pago y otra gratuita, o por ejemplo diferentes dispositivos o almacén de datos.

3.2 INSTALANDO EL ENTORNO DE DESARROLLO ANDROID STUDIO.

Descarga e Instalación de Android Studio.

Los pasos necesarios para la instalación son los siguientes:

- En primer lugar se deberá descargar Android Studio para la plataforma acorde a nuestras necesidades. La descarga de Android Studio se puede realizar a través del siguiente enlace oficial tal y como se puede ver en la figura 4:

http://developer.android.com/sdk/index.html#top

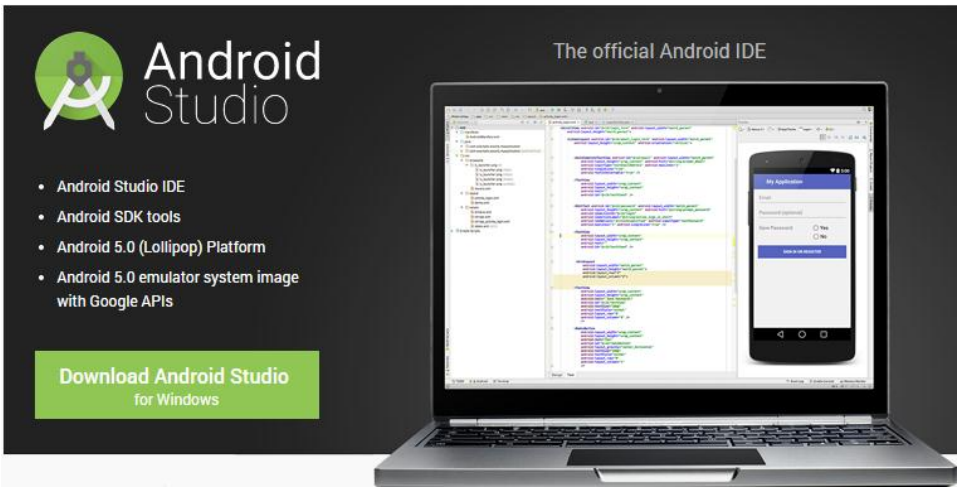

Figura 4: Enlace oficial de descarga de Android Studio

- Una vez descargado, se deberá localizar el archivo ejecutable, al que se podrá acceder a través de la ruta: Directorio local pc > android-studio > bin y seleccionaremos la arquitectura del microprocesador (studio.exe correspondiente a la arquitectura de 32 bits o studio64.exe perteneciente a 64 bits).

- Además de la descarga de Android Studio, también será necesario descargar Java SDK 7 o superior. A través del siguiente enlace es posible descargar Java SDK 8 para cualquier plataforma (puedes verlo en el video de esta serie).

- Tras la instalación del SDK de Java, y la localización del ejecutable de Android Studio, simplemente se deberá realizar doble click sobre el archivo con extensión .exe para acceder a la pantalla de bienvenida de Android Studio. En la figura 5 se puede observar la pantalla de bienvenida que al mismo tiempo es el menú de inicio del software de Android Studio.

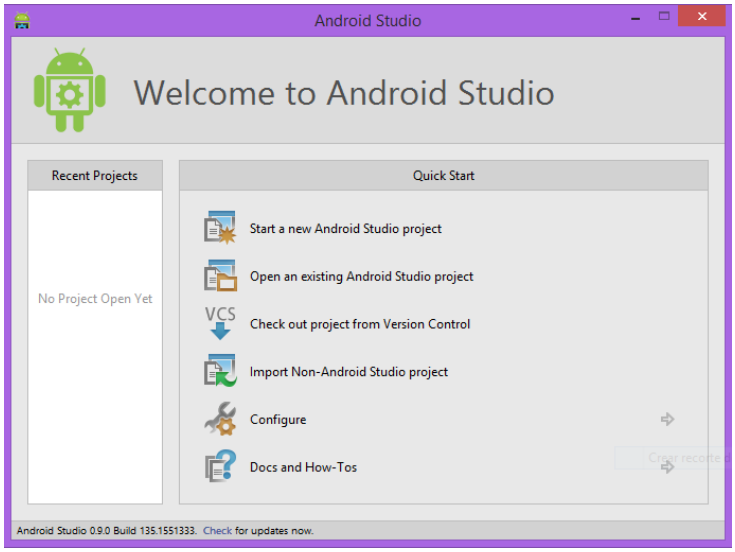

Figura 5: Pantalla de bienvenida de Android Studio.

Solución al error común de Android Studio en Windows 7 y 8.

En la figura 6 se muestra la imagen de la ventana de error que aparece al ejecutar Android Studio en las versiones de Windows 7 y 8.

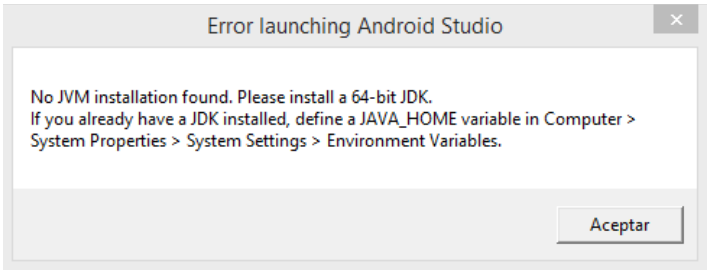

Figura 6: Mensaje de error al ejecutar Android Studio.

En el supuesto de mostrar el siguiente mensaje al ejecutar Android Studio que indica que no ha sido posible encontrar la instalación de Java (se da por

hecho que la instalación del software JDK se ha realizado correctamente), será necesaria la creación de una variable de entorno, donde se indicará el directorio de instalación. En la figura 7 se muestran las ventanas e configuración y así tener un arranque correcto de Android Studio. Para esta tarea simplemente debemos seguir los siguientes pasos:

- Se accederá a la opción "Equipo", y tras mostrar la nueva ventana se seleccionará la opción "Propiedades del sistema".
- Se accederá a una nueva ventana, donde se debe seleccionar la opción "Configuración avanzada del sistema" situada en la parte superior izquierda.
- Aparecerá una nueva ventana denominada "Propiedades del sistema", donde se seleccionará la opción "Variables de entorno".
- Posteriormente se creará una nueva variable de sistema, seleccionando el botón de "Nueva…", para finalmente completar los campos de esta nueva variable con los siguientes valores:
  - En "Nombre de la variable:" se indicará el valor de JAVA_HOME.
  - En "Valor de la variable:" se indicará la ruta o directorio de instalación de Java en nuestro pc.

Al pulsar en "Aceptar", ya será posible iniciar Android Studio sin problemas.

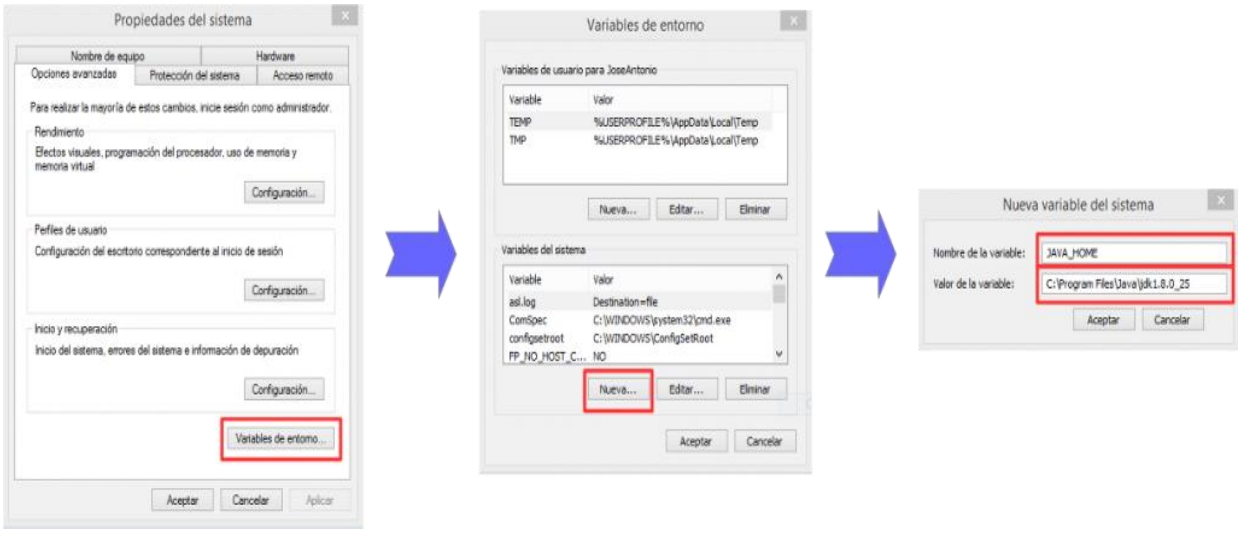

Figura 7: Configuración de parámetros para ejecutar Android Studio.

Nota: Por razones de seguridad, y para evitar posibles errores de versiones recientes, es recomendable modificar la localización de la carpeta SDK

situada dentro de la carpeta de instalación “android-studio”. Una vez se ha situado en otra localización externa a la de la instalación, bastará con indicar la nueva ruta, a través de la opción Configure > Project Defaults > Project Structure > Android SDK Location. Una vez se ha indicado la nueva ruta, ya estará disponible la opción SDK Manager.

3.3 HERRAMIENTAS DE ANDROID STUDIO

Editor de código inteligente.

El núcleo de Android Studio es un editor de código inteligente capaz de completar, re factorizar y analizar código avanzado. El poder del editor de código te ayuda a ser más productivo como desarrollador de Apps Android. A continuación se muestra la figura 8 donde se muestra la pantalla de editor de código en Android Studio.

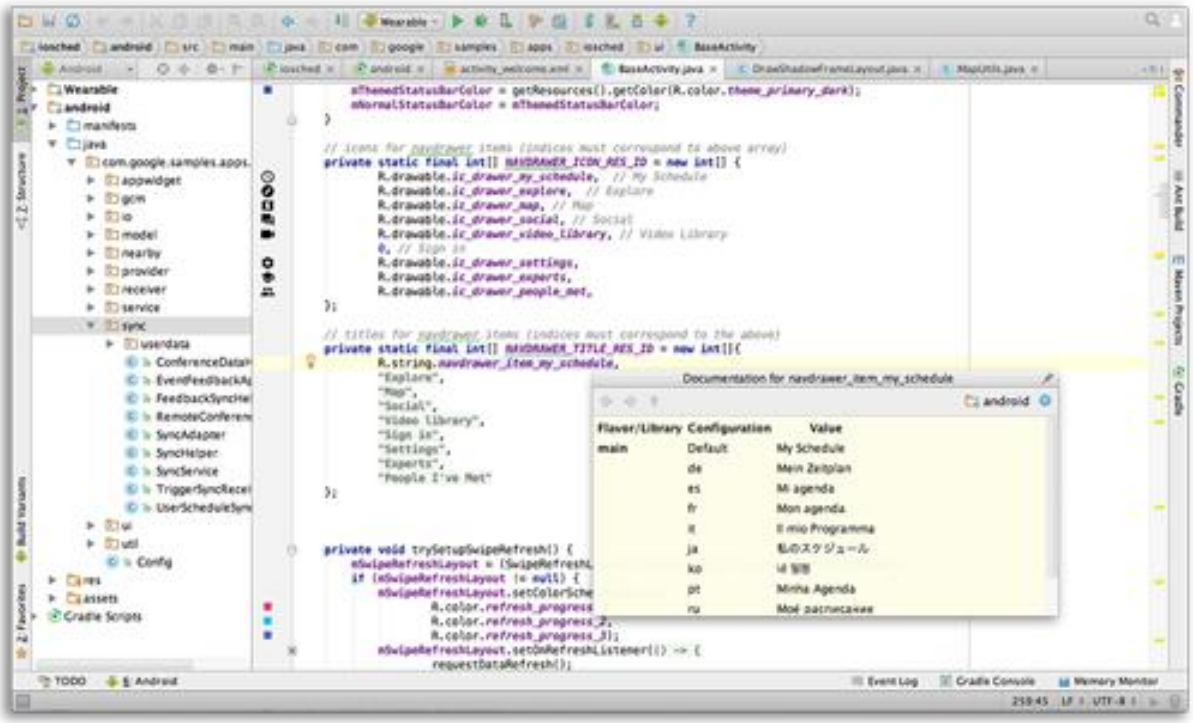

Figura 8: Pantalla de Android Studio.

Plantillas con código e integración con GitHub.

El nuevo asistente de instalación facilita crear un nuevo proyecto. Comienza proyectos utilizando plantillas con código con pautas establecidas como un cajón de navegación o una vista con páginas, o incluso importar ejemplos de Google directamente desde GitHub. En la figura 9 se muestra la pantalla de plantillas o ejemplos ya sea para editar o solo inspeccionar un código y su funcionamiento. GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

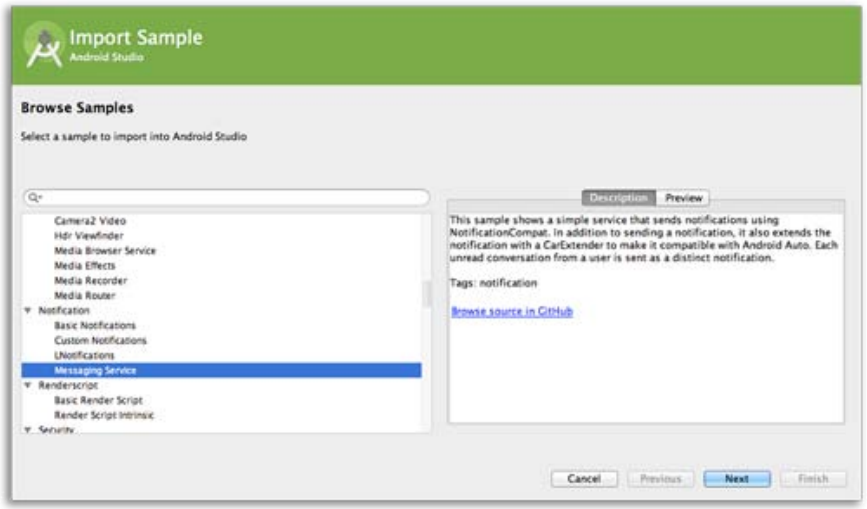

Figura 9: Pantalla para importar ejemplos de aplicaciones.

Desarrollo de Apps para varias pantallas.

Crea Apps para teléfonos y tabletas Android, Android Wear, Android TV, Android Auto y Google Glass. Con la nueva Vista de Proyecto Android y el módulo de soporte en Android Studio, es más fácil administrar los proyectos y los recursos. EN la figura 10 se muestra la pantalla donde se muestra el ejemplo de la visualización del proyecto y así validar si funciona correctamente con varios dispositivos diferentes.

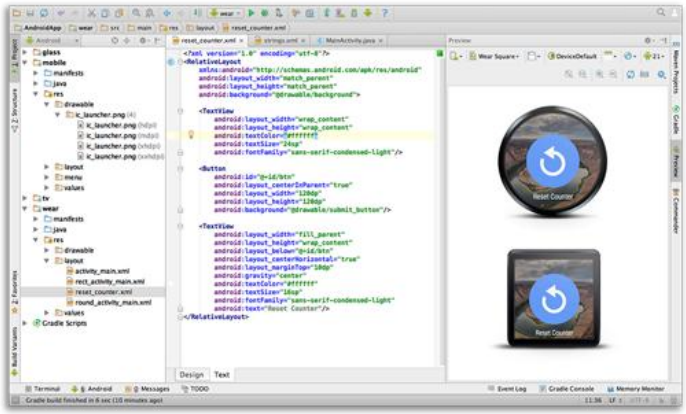

Figura 10: Pantalla de Previews de la aplicaciones en diferentes Smartphone.

Dispositivos virtuales para todas las formas y tamaños de pantalla.

Android Studio viene pre configurado con una imagen optimizada para el emulador. El actualizado Administrador de Dispositivos Virtuales proporciona perfiles de dispositivo predefinidos para los dispositivos Android más comunes. En la figura 11 se puede observar la pantalla de creación y configuración de dispositivos virtuales que se usaran para realizar pruebas. Estos dispositivos son emuladores del dispositivo original o se puede crear uno con las características y especificaciones que sean deseados por el programador.

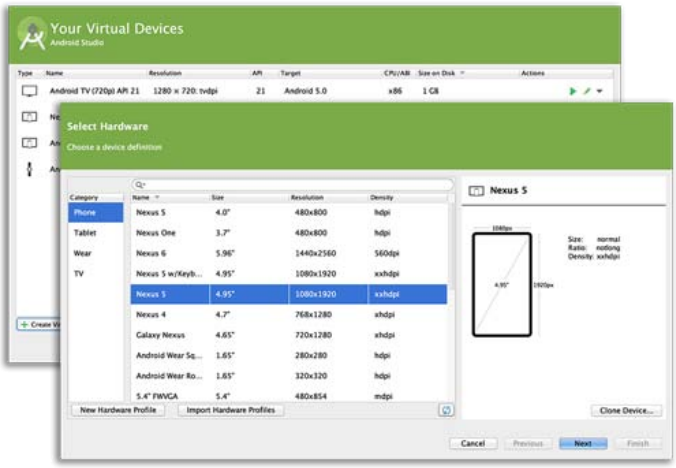

Figura 11: Pantalla de creación de un emulador de Smartphone Virtual.

CAPITULO 4

DESARROLLO DE LA APLICACIÓN

4.1 PLANEANDO LA APLICACIÓN.

Para desarrollar una aplicación para el sistema operativo Android es indispensable tener claro que tipo de aplicación se va crear. En este caso se va a desarrollar una aplicación para uso empresarial por lo que tienen que tener ciertas características que la hagan segura y útil para el sector a quien está destinada.

La aplicación de consulta de información teórica y técnica debe llevar las siguientes características:

- **Funcionalidad**. Para que una aplicación sea funcional debe de ser fluida y rápida.
- **Seguridad**. La seguridad que pueda brindar la aplicación es uno de los puntos más importantes a tomar en cuenta durante el desarrollo de la aplicación.
- **Práctica**. La aplicación será practica cuando realmente se le pueda dar un uso y no sea una aplicación que al final nadie la use por su falta de contenido práctico.
- **Interfaz**. Una buena interfaz hará el uso de la aplicación más sencillo y fácil, así como graficas amigables no solo para darle un toque profesional al diseño sino para que se pueda tener una buena experiencia de uso.
- **Compatibilidad**. Es muy importante que la aplicación se desarrolle para el mayor número de dispositivos con el sistema operativo Android que hay en el mercado.
- **Contenido**. El contenido es lo más importante ya que se debe seleccionar el mejor contenido que se desea tener en el dispositivo móvil para que la aplicación sea funcional y se pueda usar en la práctica en las labores diarias.
- **Extras**. Es importante agregar algún extra para poder asegurar que la aplicación cumpla con su objetivo se der utilizada por los integrantes a quien está destinada.

Teniendo en cuenta que la aplicación debe de tener las características anteriores se puede comenzar a ver las características más importantes como son la interfaz gráfica, la seguridad y el contenido.

Diferentes tamaños de pantallas.

Existen diferentes tipos de tamaño de pantalla, los cuales nos brindan una mejor resolución de la imagen, mientras más grande sea el número de pixeles a lo largo y ancho de la pantalla mejor será la calidad de imagen. En la figura 12 se puede observar un ejemplo de un dispositivo con sus características de tamaño de pantalla.

HDPI = (High Dots Per Inch) Mas puntos por pulgada. Otra forma de indicar la calidad de la pantalla.

Figura 12: Imagen que muestra cómo se mide la pantalla de un Smartphone.

Las imágenes que se utilicen en los proyectos tienen que estar en formato png y tienen que guardarse en la carpeta drawable-hdpi, drawable-mdpi, drawable-xhdpi y drawable-xxhdpi.

Cuando se va a utilizar una imagen de fondo de pantalla, esta debe de estar previamente editada con los siguientes pixeles en ancho y largo:

Hdpi = 720px X 1200px

Mdpi = 480px X 800px

Xhdpi = 950px X 1300px

Xxhdpi = 1280px X 1920px

Las imágenes se guardan en la carpeta Drawable con su descripción de tamaño. Al realizar esto se indica al programa que cuando se instale ya sea en un equipo de pantalla con pocos o muchos pixeles se adapte el fondo de pantalla en automático.

Cuando se van a utilizar imágenes varias ya sea para botones o solo mostrar imágenes, estas se deben de poner en las diferentes carpetas Drawables, al invocar a una imagen, en el editor visual se podrá dimensionar su tamaño y este quedara colocado con un valor y con la descripción de dp (densidad de pixeles), lo que logra que al utilizar cualquier dispositivo se adapte la imagen a la pantalla en automático.

4.1 HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES

Estructura del proyecto.

En esta parte del programa nos aparece la estructura del proyecto donde podremos ver y añadir activitys (nuevas pantallas de visualización), Classes y demás herramientas que nos ayuden a crear nuestro proyecto.

En la figura 13 se puede observar un ejemplo de la estructura de un proyecto, esta herramienta nos permite acceder fácilmente a las partes del proyecto y nos facilita la visualización de los componentes de la aplicación para poder abrir rápidamente algún archivo que forme parte de este proyecto o crearlo.

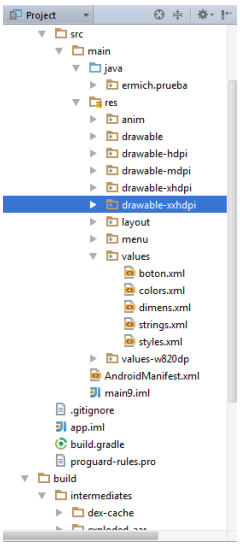

Figura 13: Imagen que muestra la estructura del proyecto en Android Studio.

Dentro de las posibilidades de edición que tiene Android Studio se tienen muchas herramientas. Existen posibilidades de edición gráfica, es decir, de una paleta de herramientas se pueden ir arrastrando elementos como son botones, cuadros de texto, botones de imágenes, Switch, etc. En la figura 14 se puede observar la herramienta de edición grafica de los Layouts la cual nos ayudara a tener una imagen más real de lo que deseamos mostrar con nuestra aplicación.

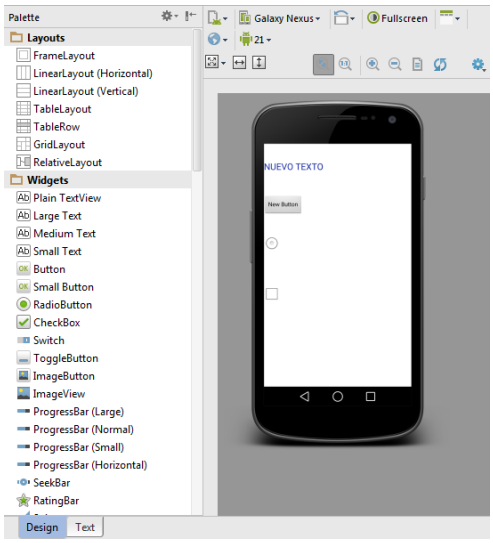

Figura 14: Imagen que muestra las opciones de edición gráfica.

Y también existen posibilidades de edición directamente en código de programación, esto se logra al darle click en el botón que dice texto en la paleta de herramientas. En la figura 15 se pude observar un ejemplo de la pantalla de edición de un Layout haciendo programación en código XML para poder realizar ediciones mas especificas a cada parte de lo que se quiera mostrar al usuario.

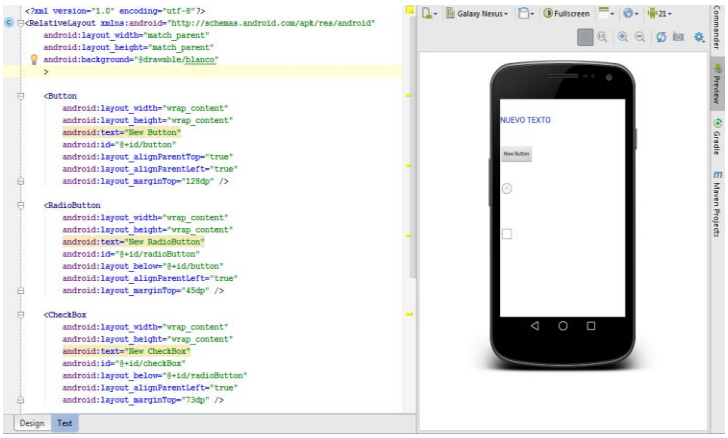

Figura 15: Imagen que muestra la edición con código XML.

En la aplicación necesitaremos agregar texto (TextView), cuadros para ingresar texto (Text Fields), botones (Button), imágenes botón (Image Button), desplazamientos verticales (ScrollView), animaciones, entre otros.

Al iniciar y crear un nuevo proyecto nos mostrara una pantalla (Activity) vacía y ese será nuestro primer contacto con la aplicación la cual ya se podrá modificar para crear de acuerdo a lo planeado nuestra aplicación.

Emulador de Teléfono móvil para realizar pruebas.

EL Programa Android Studio nos da la posibilidad de configurar un emulador de un teléfono móvil y poder realizar pruebas virtualmente y ver cómo va quedando desarrollada nuestra aplicación. En la figura 16 se puede ver un ejemplo de un emulador que viene pre configurado por Android Studio para realizar pruebas.

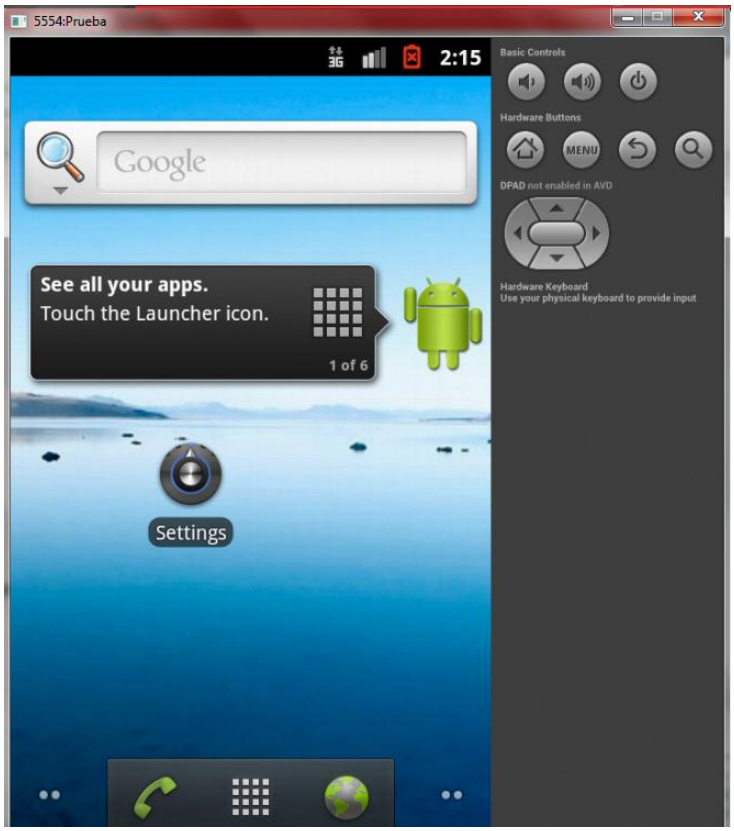

Figura 16: Ventana que muestra un emulador de Smartphone para pruebas.

Teléfono móvil Smartphone en modo desarrollador para pruebas.

EL Smartphone se puede configurar como un dispositivo de prueba, así se pueden realizar pruebas directamente en él y ver si el resultado de lo programado es el que se quiere obtener. Es muy útil ya que es más rápido que el emulador virtual.

Hay equipos con software menor al 4.1 de Android y la opción viene disponible y en equipo de versión 4.1 y superiores vienen ocultos así que es necesario activar la opción para que se muestren las opciones. En la figura 17 se puede observar las pantallas de configuración del Smartphone para poder activar el modo desarrollador y poder realizar pruebas con nuestro propio Smartphone.

La configuración es la siguiente:

Nos dirigiremos a Ajustes> Información del Teléfono > (nos desplazaremos hasta el final)> pulsaremos varias veces sobre la opción número de compilación y nos aparecerá un mensaje anunciando que ya somos un desarrollador.

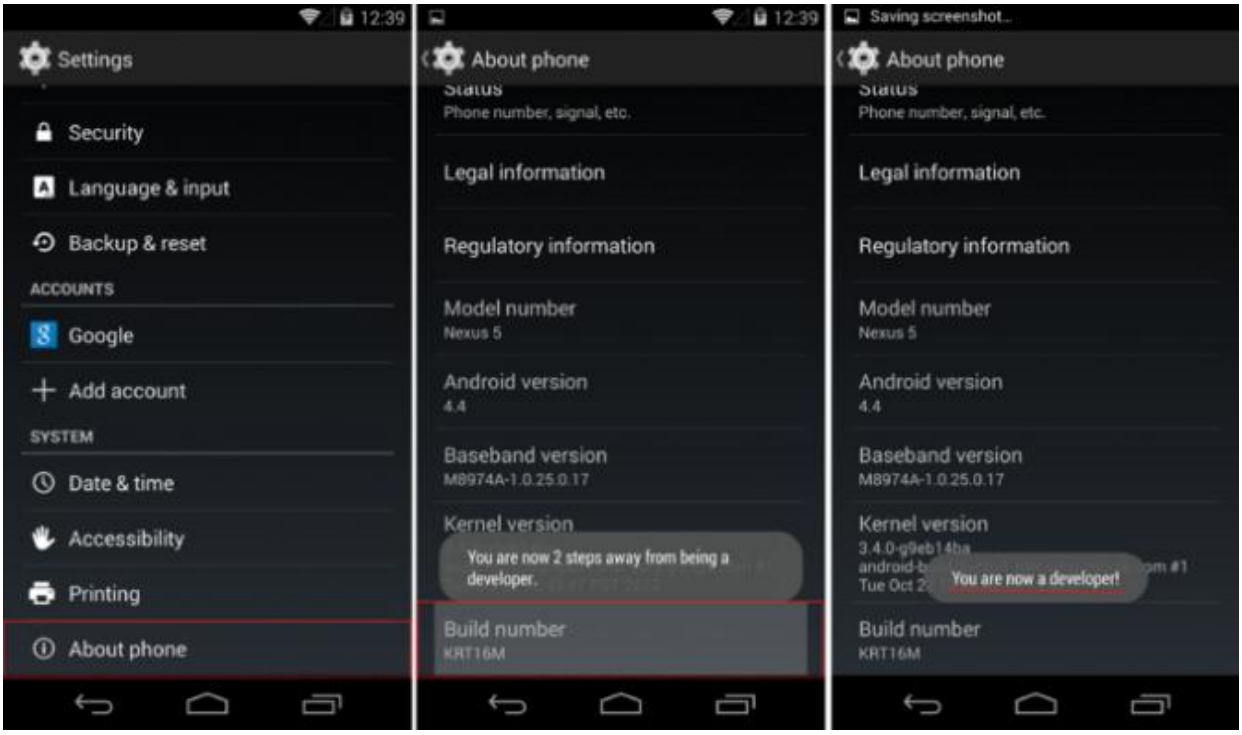

Figura 17: Configuración en el Smartphone para activar el modo desarrollador.

Volveremos atrás y veremos que ha aparecido una nueva opción titulada “Opciones de Desarrollo”. Entraremos y marcaremos la opción “Depuración USB”. Le daremos a Aceptar en el mensaje que nos aparecerá.

En la figura 18 se puede observar los parámetros a configurar en el modo desarrollador dentro de las configuraciones de nuestro Smartphone y así poder realizar pruebas del proyecto en un Smartphone físico.

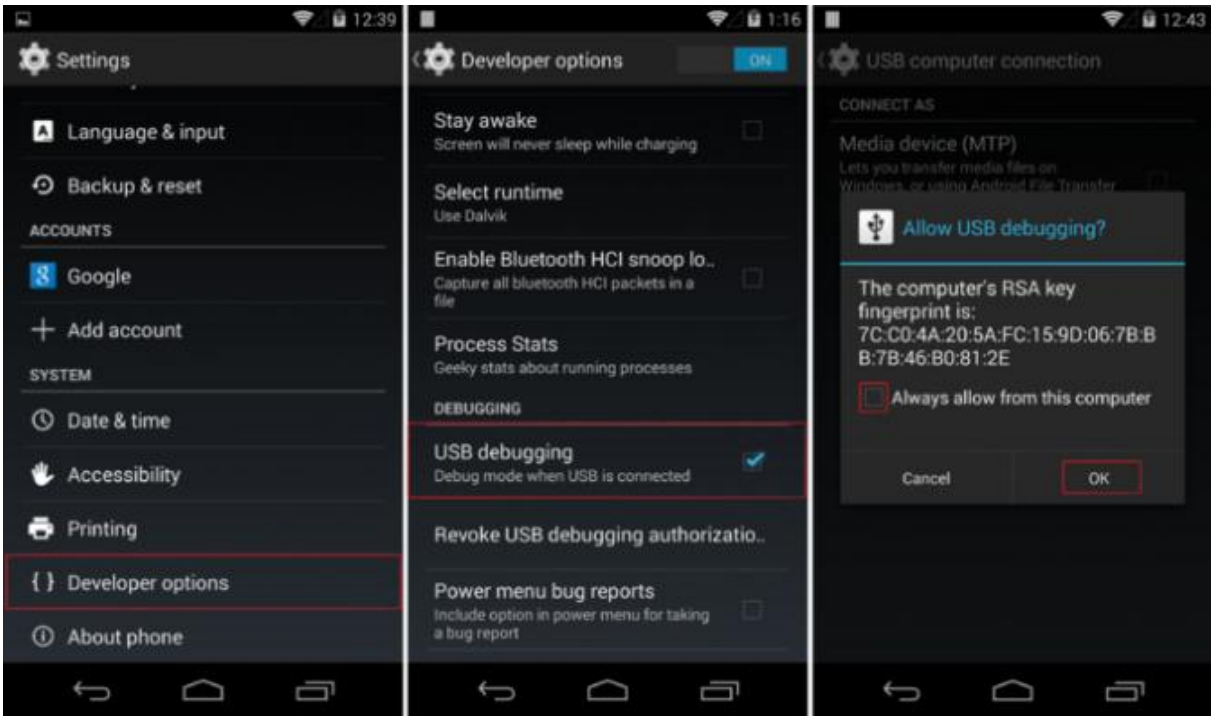

Figura 18: Configuración del modo depuración por USB para poder realizar pruebas con ese Smartphone.

Podremos activarlo o desactivarlo siempre que queramos repitiendo el mismo proceso.

Partes de la Aplicación.

Las aplicaciones en Android básicamente están conformadas por pantallas llamadas Activitys. Cada pantalla o Activity tiene una parte lógica llamada Class y una parte grafica llamada Layout.

4.2 DESARROLLO DE LA APLICACION

Crear un proyecto con Android Studio.

Al lanzar el ejecutable de Android Studio, mostrara la ventana de bienvenida, que se mostrará la primera vez que iniciemos el IDE. En la figura 19 se puede observar la pantalla del menú de inicio de Android Studio para crear un nuevo proyecto.

Seleccionaremos la opción **“Start a new Android Studio Project”:**

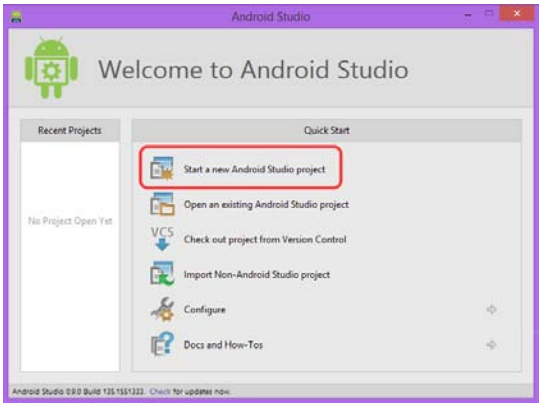

Figura 19: Ventana que muestra cómo crear un nuevo proyecto.

Posteriormente se mostrarán las ventanas para indicar determinados datos de la aplicación que se desea desarrollar.

En primer lugar nos solicitará datos relacionados con el nombre de la aplicación y dominio de la compañía, con el que se construirá el nombre del paquete de la aplicación. En la figura 20 se puede observar la pantalla de asignación de valores que llevara nuestro proyecto, tales como el nombre de la aplicación en el cual usaremos “prueba” y en nombre de empresa “ermich”.

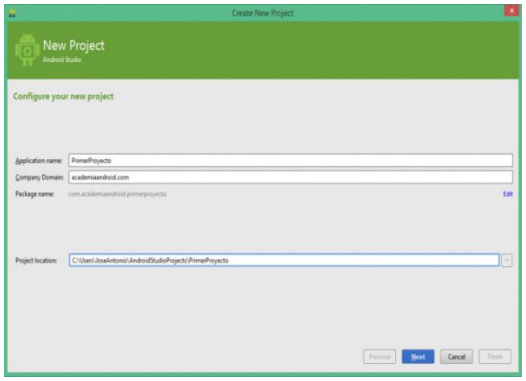

Figura 20: Ventana de asignación de nombre para la aplicación.

También será posible modificar la localización del proyecto (donde se desea almacenar).

Tras pulsar en “Next”, solicitará los datos referentes a la versión mínima de Android donde se podrá instalar la aplicación. Seleccionaremos la versión 2.3 para poder instalar la aplicación en la mayoría de equipo con Sistema Operativo Android que existen en el mercado.

Previamente se debe seleccionar la plataforma de destino (teléfono y Tablet, Android TV, Android Wear o Google Glass). En la figura 21 se observa la siguiente pantalla de configuración que nos pide configurar la versión mínima de Android en la que funcionara la aplicación y el tipo de dispositivo para el que se desea crear la aplicación ya sea teléfono o Tablet.

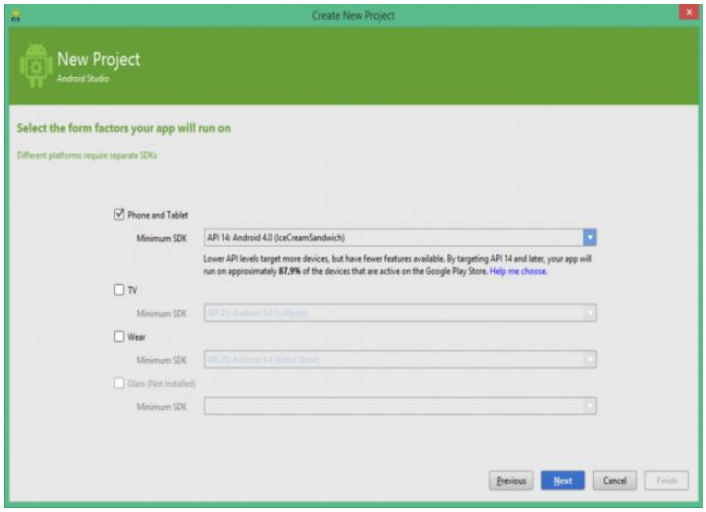

Figura 21: Ventana que muestra la selección de plataforma de la aplicación.

Una vez se aceptan los datos del formulario del proyecto, se configurará el tipo de actividad que definirá la actividad principal. Las opciones disponibles serán, entre otras, las de ventana normal, ventana a pantalla completa, ventana de login, ventana de preferencias, etc. En la figura 22 se puede observar la pantalla de configuración donde decidiremos qué tipo de Activity se creara de acuerdo a el propósito de cada aplicación.

Por defecto se dejará seleccionada la ventana normal, que será posible modificar más adelante:

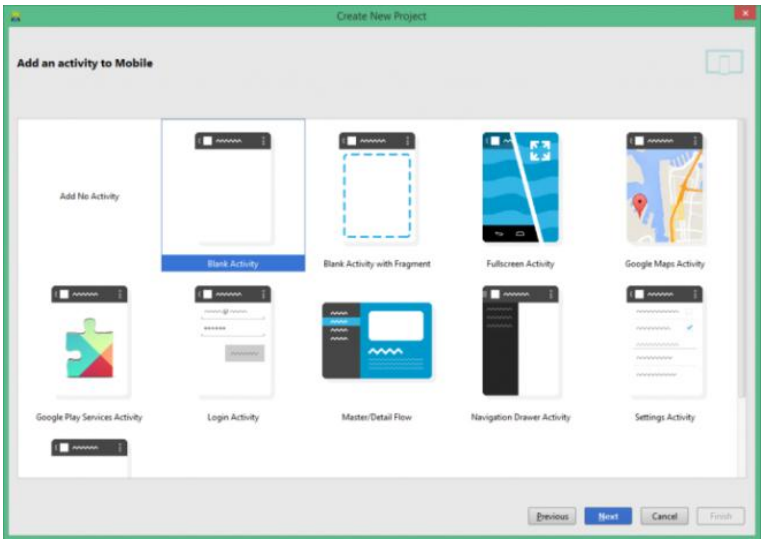

Figura 22: Ventana que muestra la selección de Activity creada desde un inicio.

Tras pulsar en "Next"**,** se deberá establecer el nombre de la actividad principal. En la figura 23 se observa la pantalla de asignación del nombre de nuestra Activity principal. En este ejemplo se dejarán los datos por defecto para mayor simplicidad:

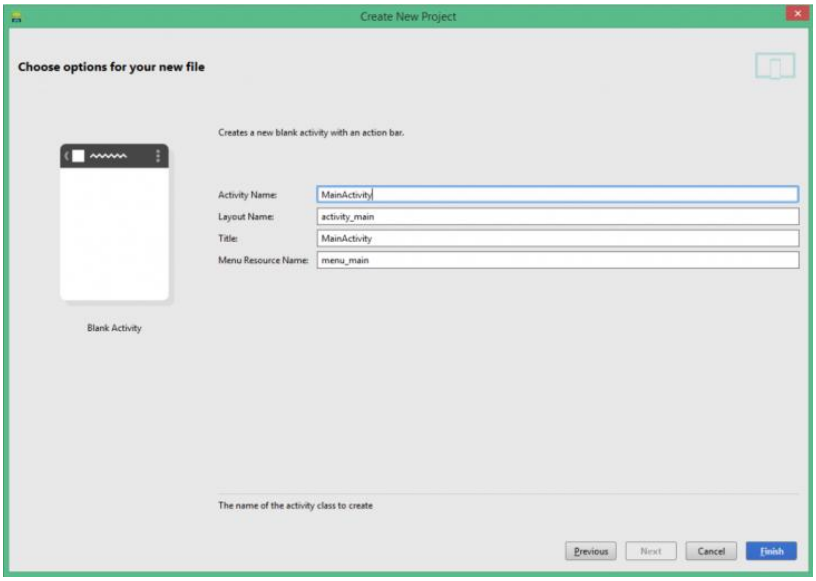

Figura 23: Ventana que muestra la asignación del nombre de la Activity creada desde un inicio.

Para finalizar, pulsaremos sobre el botón “Finish”**,** y se mostrará una ventana que indica el proceso de construcción de la aplicación, dando lugar a la ventana principal del IDE una vez terminado el proceso, así como se puede observar en la figura 24 y en la figura 25 se puede observar el proyecto ya creado y listo para programar nuestra aplicación.

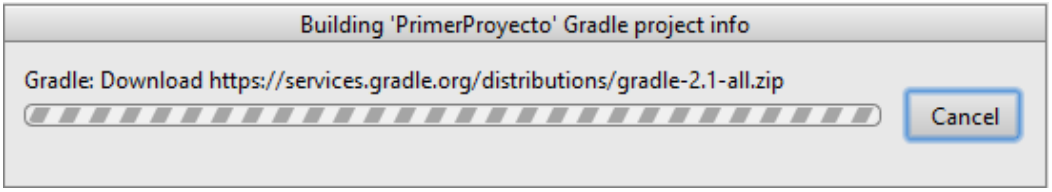

Figura 24: Ventana de creación del proyecto.

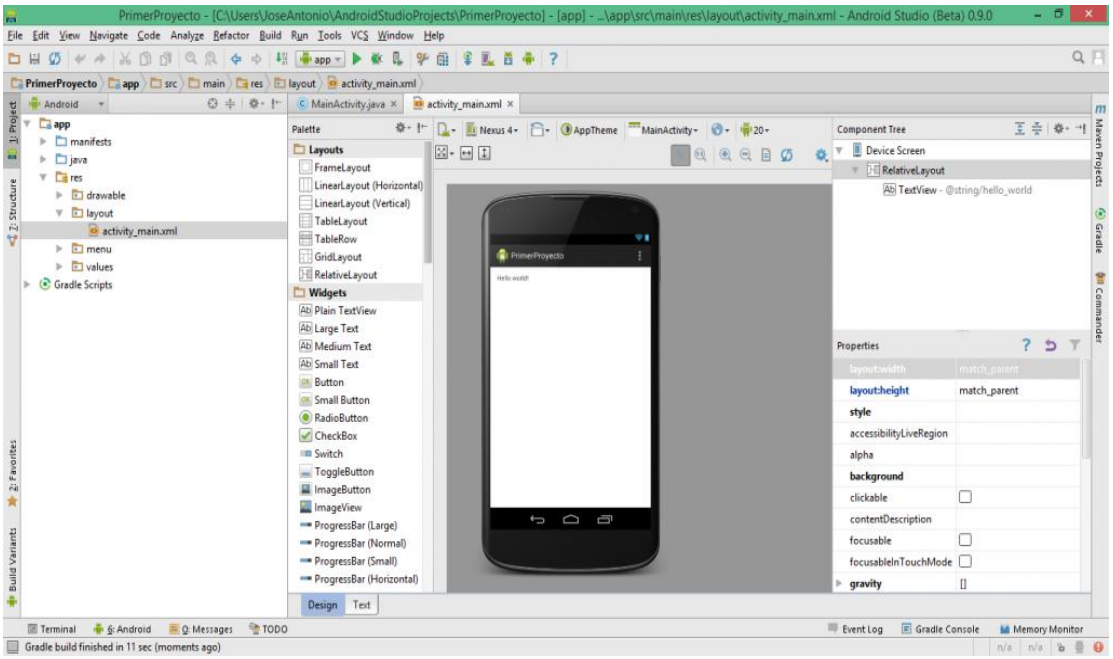

Figura 25: Ventana que muestra el proyecto recién creado.

Características de la Aplicación.

El programa a desarrollar tendrá las siguientes características para que cumpla con el objetivo de ser una aplicación que se pueda usar en alguna empresa para consultas teóricas y técnicas:

Características de interfaz:

- La aplicación contara con una pantalla de inicio que mostrara por unos segundos el nombre o logotipo de la empresa.
- Pantalla de bienvenida después de la Activity Login.
- Fondo de pantalla homologado para todas las Activitys en donde se muestre el nombre de la empresa.
- Colores homologados y tamaño para el texto dentro de la aplicación.
- Edición de botones haciéndolos estilizados y homologados en color y del mismo tamaño en cada Activity.

Características de funcionalidad:

- Pantalla de Login con una base de datos de passwords codificados que al ingresar un password este se convierta en texto codificado en Hash 256 y este se compare con la base de datos y de ingreso al contenido de la App.
- Actividades o pantallas con contenido el cual se podrá modificar por cada persona de acuerdo a las necesidades de consulta de información teórica y técnica a utilizar por su empresa.
- Extras a implementar.

Información de las Pantallas Activitys.

Cada pantalla Activity está compuesta de dos archivos, una que es un Layout que es un archivo Xml que es la parte visual y un Class Java el cual es el código de programación que le da funcionalidad a la pantalla.

Es importante dejar claro que al hacer una aplicación se deberán crear Activitys y la forma de crearlas es la siguiente:

En la estructura del proyecto la carpeta principal tiene el nombre de “app”, dentro de ella la carpeta que nos interesa lleva el nombre de “src”, dentro de esta carpeta se encuentra otra de nombre “main” y dentro de esta hay dos carpetas que llevan por nombre “java” y “res”.

Para crear un nuevo Layout accederemos dentro de la pantalla “res” y encontraremos la carpeta “Layout” dentro de la cual ya que estará el “activity_main” que será la pantalla principal de la aplicación (si es que al crear el proyecto seleccionamos que este tenga una primera actividad), hay que darle click derecho a la carpeta de Layout y seleccionar “New” y después “Layout Resource file”, nos aparecerá una ventana nueva donde se seleccionara el nombre de la actividad el cual tendrá que estar en minúsculas y para este proyecto se trabajara casi siempre con Relative Layout y Linear Layout. En la figura 26 se puede observar la venta para crear un nuevo Layout el cual editaremos y será la pantalla que visualizara y con la que va a interactuar el usuario

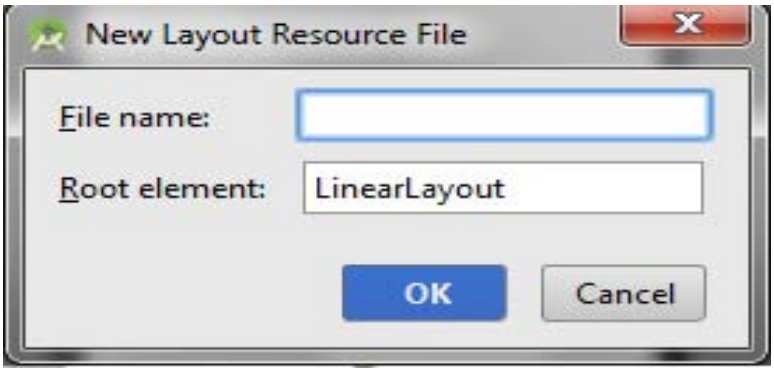

Figura 26: Ventana de creación de un Layout.

Para crear un nuevo archivo Java Class que le dará funcionalidad a nuestro Layout, dentro de la carpeta Java encontraremos una carpeta con el nombre de nuestra aplicación el cual en este caso podrá ser ermich.prueba (al igual que con el Layout, si se creó una actividad al inicio del proyecto aquí estará el archivo MainActivity el cual le dará funcionalidad a nuestra pantalla principal), se le dará click derecho a la carpeta con el nombre de nuestra aplicación y se seleccionara “New” y después “Java Class”, aparecerá una ventana en la que tendremos que asignarle un nombre y el cual deberá empezar con una mayúscula. La figura 27 muestra la ventana de creación de un archivo Class de programación Java donde programaremos toda la lógica que seguirá nuestra aplicación para poderle dar funcionamiento a cada layout con el que cuente nuestra aplicación.

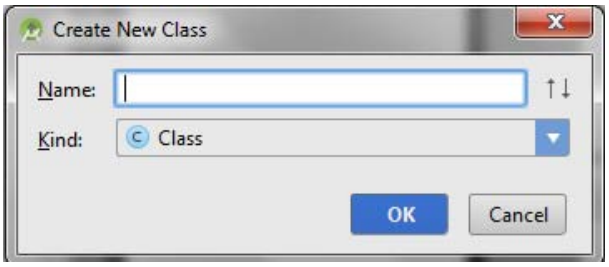

Figura 27: Ventana de creación de archivo de Class de programación Java.

Dentro de la carpeta res también nos encontramos con carpetas importantes en las cuales se podrán crear archivos de tipo Xml en los que se agregaran los siguientes elementos:

- En la carpeta “anim” se agregaran el código para efectos de difuminación de Activitys (fade_in y fade_out) tanto de entrada como salida.
- En la carpeta “drawable” agregaremos algunos códigos de homologación que se utilizaran para estilizar los botones.
- En las carpetas” drawable- xxxx” son (las carpetas con distintos tamaños de pantallas hdmi, mdpi, xhdpi y xxhdpi) van las imágenes que se mostraran en nuestra aplicación así como los fondos de pantalla.
- Y en la carpeta “values” agregaremos códigos para estilos, colores, declaraciones, etc.

Las animaciones

Las animaciones se crearan en la carpeta “anim”, será un archivo xml para el código de entrada y otro para el código de salida. Ambos códigos servirán para después solo invocarlos con un código en específico. Para este proyecto se utilizara una animación de entrada y salida para cada cambio de pantalla, la animación será una difuminación y el código se encuentra en el ANEXO 2.

Información importante para antes de comenzar con el proyecto.

Antes de comenzar con el proyecto es necesario mencionar algunos conocimientos básicos y son los siguientes:

- Al crear un proyecto nuevo con una activity o que creemos un archivo nuevo Layout o Class, cada archivo vendrá con un código en el que se especifica a que proyecto se está enlazando. El código que se crea en automático para cada Layout y casa archivo de tipo Class se puede consultar en el ANEXO 1.
- También es importante aclarar que cuando vamos agregando código en la lógica de programación java en los archivos de tipo Class es necesario importar bibliotecas para hacer referencia a ciertas actividades a realizar, esto se logra de la siguiente manera: ctrl + enter. Al realizar esto se importan las librerías necesarias para que el proyecto reconozca las funciones a realizar y no marque errores al desconocer a que hace referencia cierto código programado.
- Un ejemplo de cómo se importan estas librerías se encuentra en el anexo 1.

También es importante mencionar que a lo largo de proyecto se utilizaran imágenes con diferentes tipos de resolución, esto se lograra utilizando la herramienta de Paint que trae el sistema operativo Windows. Un ejemplo seria tener la imagen de fondo de pantalla en diferentes resoluciones. Para realizar esta imagen de fondo realizamos lo siguiente:

- Primero abrimos el programa Paint.
- Enseguida en las herramientas superiores damos un click en cambiar tamaño.
- En la ventana que nos abre nos aparece como primera opción cambiar tamaño por pixeles o porcentaje.
- Seleccionamos que se realizara por pixeles.
- Colocamos el ancho y largo en pixeles.
- Quitamos la selección de mantener relación aspecto.

Con esto logramos tener una plantilla con los pixeles que necesitamos y en la cual agregaremos las imágenes que necesitemos y al final lo guardaremos como un archivo con extensión png.

Un ejemplo de la edición de imágenes con el software Paint de Windows es el de la figura 28 donde se podrán realizar cambios en las dimensiones de la imágenes y dejarlas adecuadas para poderlas utilizar n nuestro proyecto.

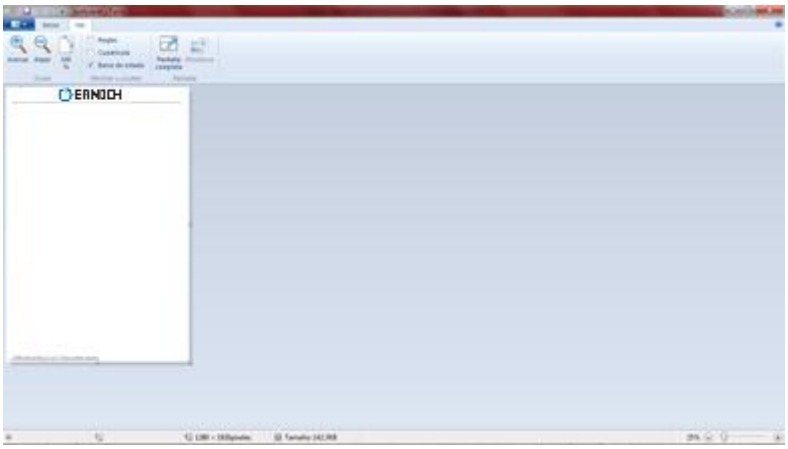

Figura 28: Ventana de Paint editando en tamaño una imagen.

Creando el proyecto

Antes de comenzar con la pantalla de inicio la cual será un Splash que mostrara la imagen de la empresa por unos segundos y después nos mandara a la pantalla de acceso, es necesario indicarla ubicación de los archivos que serán necesarios para poder generar en seguida cada una de las pantallas de nuestra aplicación como seve en la figura 29.

Lo primero es indicar que se tendrán que generar varias carpetas en la carpeta principal SRC las cuales llevaran el nombre de: mdpi (480x800px), hdpi (720x1280px), xhdpi (950x1300px), xxhdpi (1280x1920px). Dentro de estas carpetas se agregaran las imágenes a mostrar en la aplicación tanto los fondos de pantalla así como las imágenes que se utilicen como botones o solo ilustrativas. Hay que hacer las imágenes con Paint que serán los fondos de pantalla con sus respectivos tamaños y el logotipo de la empresa, así como los fondos de pantalla que serán utilizados como una imagen homologada para todas las pantallas en la aplicación.

Hay que crear los archivos que harán que los botones tengan una apariencia mejorada, estos archivos deben ir dentro de la carpeta drawable y serán los colores de cada uno de sus estados normal y al ser presionado; así como el archivo que va a indicar esta actividad del botón.

También hay que crear una carpeta llamada “anim” dentro de la que estarán los archivos de animación de entrada y salida (fade_in y fade_out) de cada pantalla haciéndolo lucir mejor que la animación que viene por default con el software Android Studio.

También hay que configurar el color del tecto (textColor el cual utilizaremos #ff00159c, tamaño 25dp y negrita) y guardarlo con un nombre “Aazul” para que sea utilizado siempre el mismo color, es decir homologar el color y letra a usar. Y por último hay que crear los archivos de programación java llamados class, los cuales serán utilizados en las bases de datos que se usaran en la aplicación, y estos son el dbHelper y LoginDataBaseAdapter, y tendrán que ir donde van todos los archivos de programación de este tipo en la careta “java”.

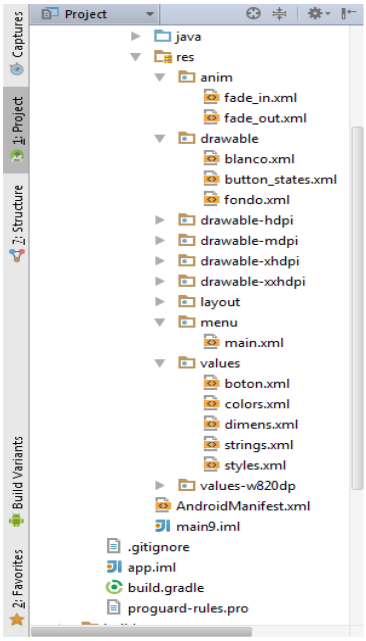

Figura 29: Ejemplo de pantalla con las carpetas y archivos de la aplicación.

Pantalla de Inicio.

La pantalla de inicio será una pantalla que tendrá una animación de difuminación de entrada y otra de salida, así como un tiempo de duración denominado Splash, esta pantalla tendrá el logotipo de la empresa.

Creando el Layout de la pantalla de inicio. Lo primero será crear un Layout de tipo linear llamado en este caso “splash.xml”, cabe destacar que es una buena opción en lugar de usar código de programación utilizar la herramienta de edición grafica en la que se puede arrastrar y pegar, así como dar dimensiones de la imagen y ver el resultado en diferentes dispositivos.

Creando el Class de la pantalla de inicio. El código java donde se le dan instrucciones a la pantalla de inicio contiene tanto el código de animación de difuminación de entrada y salida así como el código del Splash para darle un tiempo de duración a la Activity y también que Activity es la siguiente en mostrar en pantalla.

En la figura 30 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuerdo a las necesidades planteadas anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. El código del Layout y del archivo Class se encuentran en el ANEXO 6.

Figura 30: Ejemplo de la pantalla de Inicio (Splash).

Pantalla de Login.

La pantalla de Login es la principal Actividad y de la cual depende si esta es aceptada por la empresa en la cual se pretende implementar esta aplicación.

Uno de los puntos más importantes es la seguridad, en esta aplicación se usa una base de datos para guardar el número suficiente de passwords codificados con un sistema de codificación SHA-256.

Hash.

Es una secuencia de bits que tiene como objetivo identificar un archivo. Esto significa que si hacemos una representación en un archivo generará un hash que es único, y así lograr la garantía de la integridad. Utilizando Hash también tienen la propiedad de un solo punto en el que el camino de regreso no es posible. Además, el uso de Hash no hay necesidad de llaves y tenemos que lograr la coherencia, ya que si se introduce el mismo mensaje Hash tendrá exactamente el mismo hash está generando.

Entre las familias de Hash están el MD (Message Digest) que se compone de MD2 (lento y con una producción de 128 bits), MD3 , MD4 , MD5, que es actualmente el más utilizado y, por último, la familia SHA (Secure Hash Algorithm) diseñado por la Agencia Nacional de Seguridad (NSA), publicado como un estándar del gobierno de Estados Unidos consiste en la algoritmos SHA-1 , SHA-224 , SHA-256 , SHA-384 y SHA-512 . Cada uno de los algoritmos se diferencian por el tamaño de la tamaño admitido mensaje de entrada de bloque, tamaño de la palabra, el tamaño de la Message Digest algoritmo y la seguridad.

Hay básicamente dos tipos de algoritmos de cifrado:

Algoritmos de dos vías: una en la que es posible descifrar el mensaje cifrado;

Los algoritmos de una manera: es una en la que no hay manera para descifrar el mensaje porque el cálculo matemático hecho no lo hace, desde el extremo de valor, volver al valor inicial.

Los algoritmos de una pista son llamados Mensaje Digest, y son extremadamente eficientes para almacenar contraseñas.

Unas buenas paginas en la web para crear passwords de palabras en codificación Sha-256 son las siguientes:

- http://www.puertosabiertos.com/es/sha256-online.htm
- http://hash.online-convert.com/es/generador-sha256

Base de Datos.

Una base de datos (cuya abreviatura es BD) es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible.

SQLite.

Es un ligero motor de bases de datos de código abierto, que se caracteriza por mantener el almacenamiento de información persistente de forma sencilla. 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**: Esta 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.

LoginDataBaseAdapter como herramienta para la base de datos de los passwords.

Antes de crear la lógica de programación del la Activuty Login será necesario crear un archivo de tipo Class llamado LoginDataBaseAdapter que nos ayudara con el funcionamiento de la base de datos de los passwords. Este archivo se encargara del funcionamiento de la base de datos. El código se encuentra en el Anexo 3.

DbHelper como archivo de funcionalidad de la base de datos.

Antes de empezar a crear la Activity de Login hay que crear un archivo en el cual se le va a crear la base de datos donde se guardaran los passwords así como donde se podrán ir agregando nuevos passwords. Esto se lograra creando un archivo de tipo Class llamado DbHelper. El código está en el Anexo 4.

Botón.

También se utilizaran botones editados para que se vean más estilizados y homologados que harán la apariencia de la aplicación más profesional.

Al poner un botón, dentro del código xml en el Layout se le agrega una línea de código invocando un estilo (en este caso llamado botón azul) de la siguiente manera:

style="@style/botonazul"

Editar un botón es la unión de varios archivos de tipo xml que se encuentran en la carpeta “values” y “drawable”.

En este caso botonazul es el nombre asignado en un archivo guardado en el “values” en este caso con el nombre de boton.xml, en la primera parte se

puede ver que se especifican detalles del botón en especial que al ponerle texto al botón este se encuentre centrado. En la segunda parte se menciona que se toman características del archivo llamado en este caso "button_states.xml" que se encuentra en la carpeta "drawable". Los códigos se encuentran en el ANEXO 5.

El archivo "button_states..xml" es un código que nos indica el color del botón al tenerlo presionado o al estar sin presionar.

En este caso se mencionan dos colores, el "blanco" y "fondo" que también se encuentran en la carpeta "drawable". Los códigos están en el ANEXO 5.

En el archivo llamado "fondo.xml" tenemos declaradas las características que tendrá el botón en color, donde inicia y termina el color dentro del botón y sus equinas redondeadas, el código está en el ANEXO 5:

En el archivo llamado "blanco.xml" tenemos declaradas las características que tendrá el botón en color, donde inicia y termina el color dentro del botón y sus equinas redondeadas, el código está en el ANEXO 5.

Color del Texto.

También se guardara el color del texto para homologar y darle estilo a la aplicación. Esto se logra al poner un textview en el cuadro de propiedades hay un elemento llamado "textColor" y al final aparece un recuadro con tres puntos que al darle click nos aparece una pantalla la cual es una pantalla de recursos.

En la pantalla de recursos, en la primer pestaña donde nos aparecen los colores guardados, para guardar un color en la tercer pestaña de la misma ventana e recursos se llama "color" y ahí se puede seleccionar un color de una paleta de colores y al guardar solo bastara con seleccionarla e la ventana de recursos y así homologar el color del texto en la aplicación. En la figura 31 se puede observar la pantalla de creación de recursos que utilizaremos durante la creación del proyecto.

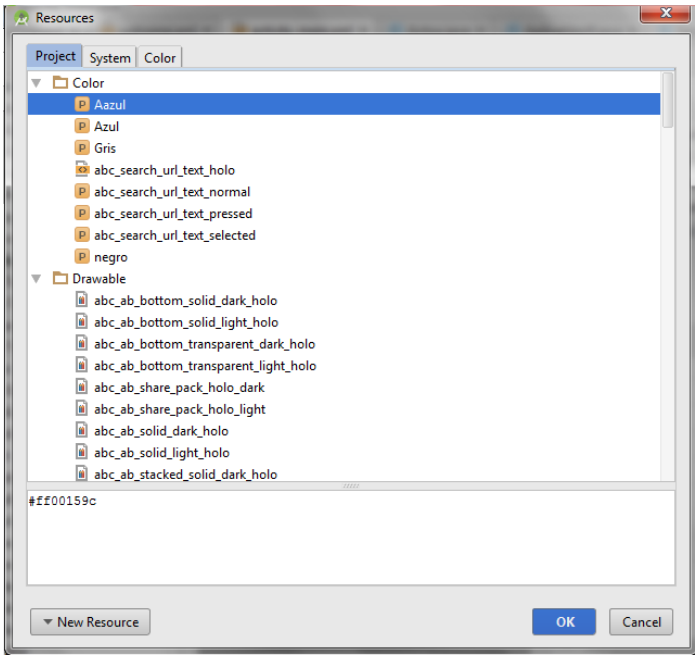

Figura 31: Ventana de creación de recursos como son los colores.

Pantalla de Login en el Layout.

La pantalla de Login es una pantalla que contara con un TextView que mencionara que es una pantalla de ingreso, Un EditText que será donde se podrá ingresar el password, un botón que iniciara el proceso de verificación de datos y dos TextView que nos indicaran los intentos restantes para poder accesar con el password correcto. Al igual que la Activity anterior se recomienda la herramienta de edición gráfica.

En este caso se utilizara el layout principal creado al inicio del proyecto el cual es llamado “activity_main” y será un Layout relativo. El código está en el ANEXO 7.

En el código anterior se puede apreciar es que primero se ocupa un Relative Layout con la finalidad de colocar los elementos en el lugar que más nos agrade. También se ocupa un wallpaper el cual se encuentra en la carpeta Drawable y el cual ocuparemos en casi todas las Activitys para homologar la aplicación.

Enseguida se puede apreciar que se agrega un Texto el cual está declarado y guardado en un documento de recurso de la aplicación llamado String y en el cual Titulo significa Ingreso.

El siguiente elemento es un cuadro de textos y que internamente tiene un texto temporal que dice password para indicar que ahí se debe de colocar el password, este texto temporal desaparecerá al darle click al cuadro de texto y poner una letra o número del teclado que aparece.

El siguiente elemento es un botón que dice ingresar el cual no ayudara para poder ingresar a la App.

Y por último tenemos dos Textos ocultos, el primero al igual que el String título tiene un texto declarado en y en cual "intentos" significa "Numero de intentos restantes" y el segundo texto es un texto oculto en el cual se asignara un valor de los intentos restantes de acuerdo a la lógica de la aplicación.

Pantalla de Login creando el archivo Class.

En la parte lógica de esta activity es la más importante por las características que debe de tener y que son las siguientes:

- Se ve que tipo de fondo tiene, se quita la barra de notificaciones
- Lee los últimos datos guardados de la aplicación sobre los intentos restantes para dar acceso.
- Crea una base de datos con los password encriptados en SHA-256 si es que no existe, si ya existe solo actualiza los nuevos passwords no existentes en su base de datos.
- Se establece el número máximo de intentos para dar acceso, se crea una condición para que en caso de llegar a 0 intentos de acceso la aplicación se bloquee y la aplicación no vuelva abrir nunca más.
- Se establece el código para la lectura de los datos insertados como paswword, su conversión a hexadecimal y su codificación en SHA-256, así como su posterior lectura y comparación con los passwords que

hay en la base de datos de passwords, se ve el código para que en caso de ser una contraseña valida brinde el acceso a la siguiente actividad declarada y si no es correcto el password reste un intento de acceso.

- Se ve el código para que siempre se guarden los datos y la seguridad en el número de intentos siempre sea la real y si se bloquea la aplicación no exista forma de acceso de nuevo.

En la figura 32 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuerdo a las necesidades planteadas anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. La pantalla obtenida es la siguiente:

Figura 32: Ejemplo de pantalla de Login

Pantalla de Bienvenida

La pantalla de bienvenida es una pantalla que al igual que la pantalla de inicio será un Splash que durar unos segundos en la cual se mostrara el texto de bienvenida después de ingresar el password correcto en la pantalla de Login. Esta Activity se puede utilizar u omitir dependiendo del gusto del desarrollador.

La parte visual Layout tendría el siguiente código aunque se puede desarrollar con Android Studio con la herramienta de arrastrar y pegar acomodando al gusto el texto de bienvenida.

Lo primero es crear un nuevo Layout y esto se hace dando click derecho en la carpeta Layout y seleccionando un nuevo Layout de tipo Relative y dándole un nombre con la primera letra en minúsculas, en este caso será “welcome”. El código de la pantalla de bienvenida está en el ANEXO 8.

Creando el archivo Class de lógica de programación.

Este archivo será como el de la Activity de inicio donde se utiliza un Splash para mostrar una imagen de bienvenida durante unos segundos. Se crea un nuevo archivo Class en este caso con el nombre de “WelcomeActivity”. El código se encuentra en el Anexo 8.

En la figura 33 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuedo a las necesidades planteadas anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. La pantalla es la siguiente:

Figura 33: Ejemplo de la Activity de bienvenida.

Pantalla del menú principal.

Esta pantalla será el menú principal al cual siempre podremos regresar y de aquí partir a la información que queramos consultar.

La pantalla constara de varios botones, tantos como queramos y sean necesarios que al presionarlos nos envíen a una página con esa información.

En la siguiente página existirá una imagen con función de botón que al presionarla nos regresara a la pantalla anterior en este caso sería la del menú principal.

En esta Activity de menú se mostrara el código para una pantalla con muchos botones y se verá como fueron diseñados en la parte grafica de Layout.

En la parte lógica se verá el código para varios botones en el apartado de Class.

Creando el Layout de la Activity de Menu.

Lo primero es crear un nuevo Relative Layout el cual en este caso llevara por nombre “menu.xls” y al igual que en los Layouts se recomienda la edición con la herramienta de edición grafica para acomodar los elementos de una mejor manera y ya después anexarle código para estilizar nuestros botones.

El código se encuentra en el Anexo 9.

Creando la parte lógica con el archivo Class del menú principal.

La parte lógica de esta aplicación se realizara creando un archivo Class, en este caso se llamara “MenuActivity”, el cual contiene el código para hacer funcionar cada uno de los botones que tengamos en el menú principal, dentro de la función de cada botón se le asigna la funcionalidad de poder ir a otra Activity al presionarlo y un botón de salir que cerrara la aplicación.

El código se encuentra en el ANEXO 9.

En la figura 34 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuerdo a las necesidades planteadas

anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. La Activity quedaría así de forma visual:

Figura 34: Ejemplo de la Activity de Menú principal.

La Activity de contenido técnico o teórico.

En esta Activity se va el contenido que sería de importancia a tener en la palma de tus manos y la importancia en sí de la aplicación. Lo que nos lleva a que en este ejemplo se pondrá un texto cualquiera y no algo que afecte a la empresa con el logotipo aquí presentado.

Esta Activity al igual que las otras Activitys tendrá un Layout que tendrá en la parte visual un título, un ScrollView el cual tendrá dentro un Linear

Creando la parte visual el Layout de Contenido.

Para este ejemplo que se llamara “almacen.xls” se usara un botón y será el de nombre Almacén. Este Layout tendrá un Titulo, también un ScrollView el

cual servirá por si ponemos mucho texto ya sea en un solo párrafo o varios este se encuentre dentro de ese scrollView que parecerá una caja de texto y al deslizar el dedo de abajo hacia arriba el texto de abajo ira apareciendo.

El ScrollView tendrá dentro un Linear Layout y dentro los TextView necesarios por párrafo de texto y también se le pueden agregar imágenes.

Agregaremos al ejemplo también una imagen de una flecha en la parte inferior derecha que servirá de botón y al presionarla nos regresara a la Activity en este caso será el menú principal. El código de este archivo se encuentra en el ANEXO 10.

Creando el archivo Class de lógica de la Actividad de Contenido

La parte Lógica de esta Activity a la que llamaremos “Almacén” nos ayudara a darle funcionalidad al Layout de esta aplicación. En este archivo Class se incluirá el código de solo se le dirá que nos muestre el Layout relacionado a esta Activity y se le dará funcionabilidad al botón con imagen de flecha de regreso. En la figura 35 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuerdo a las necesidades planteadas anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. La Activity quedara así:

Figura 35: Ejemplo de una Activity con contenido Teórico o Técnico.

Actividad de Extras

Esta actividad en este ejemplo solo tendrá dos botones pero se le pueden asignar tantos como queramos para añadirle plus a nuestra aplicación.

En esta aplicación los extras a agregar al ser para una empresa de comunicaciones se agregara una mini aplicación que será un conversor de Kb a Mb y Gb en automático, y también se agregara una mini aplicación de creador y editor de notas, para poder guardar y editar información que será importante para cada miembro de la empresa.

Es importante dejar claro que se pueden agregar otras pantallas de extras que podrán incluir otras mini aplicaciones dentro de esta aplicación y la forma de agregarlas es la misma, la parte visual estará programada en los archivos de tipo layout y la lógica para darle funcionalidad a los elementos que forman estas pantallas se tendrán que programar en los archivos Class de programación en Java, integrando ambos archivos como ya se ha mencionado anteriormente.

Creando el Layout, la parte grafica de la Actividad Extras.

Se crea un nuevo Layout para este ejemplo será “extras.xls” y el código se encuentra en el ANEXO 11.

Creando la lógica de programación Class para la Activity Extras.

La lógica de programación solo nos incluirá la funcionabilidad del botón imagen de regreso a la actividad anterior que en este caso es el menú principal y los dos botones hacia nuevas Actividades las cuales son el conversor el creador y editor de notas.

Primero creamos un nuevo archivo Class que llamaremos “Extras.java”, y el código se encuentra en el ANEXO 11.

En la figura 36 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuerdo a las necesidades planteadas

anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. La Activity quedaría así:

Figura 36: Ejemplo de un submenú para abrir otras Activitys.

Activity del Conversor de Kb a Mb y Gb.

La Activity de conversor es una mini aplicación dentro de la aplicación. Se encargara de que al ingresar un número de Kb y presionar un botón nos haga una conversión en Mb y Gb con la finalidad de hacer más fácil el trabajo diario en una empresa de comunicaciones en la que se tienen que hacer muchas conversiones de este tipo.

Creando la parte visual Layout del Conversor.

Este Layout Contara con un título también se agregara un cuadro de texto donde solo se podrán ingresar números, también un botón, dos TextView donde se agregaran los resultados de la conversión y dos TextView que nos indicaran si el resultado es en Mb o en Gb.

Creamos un nuevo Layout que en este caso lo llamaremos “converter.xls” y el código se encuentra en el ANEXO 12.

Creando la lógica Class del Conversor.

La lógica de esta Activity se encargara de darle funcionalidad al botón imagen de flecha de regreso, de que al ingresar un número este lo tome y haga dos operaciones y los resultados de estas sean asignados a dos TextViews donde se asignaran los resultados.

Crear un nuevo documento Class en este caso llamado “Converter.java” y el código se encuentra en el Anexo 12. En la figura 37 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuerdo a las necesidades planteadas anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. La Activity quedaría así:

Figura 37: Ejemplo de una Activity de conversiones de Kb a Mb y Gb.

La Activity de Notas.

Esta Activity al igual que el conversor es una mini aplicación dentro de la aplicación. Esta Activity de notas contara con 3 Activitys para su funcionamiento.

En un principio será una pantalla donde se encontraran listadas las notas que hayamos creado antes y un botón para crear una nueva nota. Esto nos da dos caminos:

- El primero es que si presionamos sobre una nota antes creada nos mandara a una pantalla donde podremos editar los datos que estaban guardados y actualizar esa información, al darle guardar nos regresara a la pantalla principal de la Activity notas.

El segundo es que si se presión el botón de nueva nota, nos mandara a una pantalla donde podremos ingresar los datos y guardarlos, nos regresara a la pantalla principal de la Activity notas y ya se encontrara en la lista nuestra nota antes creada.

Esta aplicación dentro de una aplicación tendrá su propio funcionamiento, es decir llevara los siguientes archivos:

-Un archivo especial donde se dará la funcionalidad completa a la base de datos de las notas.

-Las tres pantallas a mostrar al usuario:

- La pantalla principal donde se podrá crear o visualizar una nota.
- La pantalla de visualización donde podrá presionar un botón para acceder al modo de edición de la nota.
- La última es donde se puede editar la nota.

Ademas de lo mencionado se creara un layout que no será activity ya que no contara con archivo complementario de programación java, solo será un archivo necesario para hacer funcionar la mini aplicación de notas y su nombre será “notes_row” y su código esta en el ANEXO 17.

Y sera necesario un archivo llamado “NotesDbAdapter.java” y será de tipo class, este va a contener toda la configuración ncesaria para que esta mini aplicación pueda crear una base de datos y también la funcionalidad necesaria, su código se encuentra en el ANEXO 16.

Tendrá el contenido requerido para poder crear una nota nueva, guardarla, visualizarla y enlistarla para poder tener la información que sea importante para los usuarios dentro de la aplicación principal protegida por una contraseña de seguridad.

Creando el Layout de pantalla principal de la Activity Notas.

En este Layout se agregara un título, un botón para crear nuevas notas, una imagen botón de flecha de regreso, un ListView para enlistar las notas creadas y un TextView que indicara que no hay notas cuando no se ha creado ninguna nota.

Creamos un nuevo Layout que se llamara en este caso “notepad.xls” y el código está en el ANEXO 13.

La lógica Class de esta primer Activity de Notas.

Para darle la lógica de programación a esta primer activity que es la principal se creara un archivo de tipo Class al cual le daremos el nombre de “NotePad.java” y este código se encuentra en el ANEXO 13.

En la figura 38 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuerdo a las necesidades planteadas anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. La Activity quedaría así:

Figura 38: Ejemplo del menú principal de la Activity de Notas.

La segunda Activity de la App Notas Visualización.

La segunda Activity tendrá la función mostrar la nota guardada en la base de datos y también si se desea editar la nota existirá un botón que al presionarlo nos mandara a la Activity de Edición de nota.

Creando el Layout de Visualización de Notas.

En este Layout tendremos la misma imagen de fondo antes editada en la cual le agregaremos al centro una imagen de nota para darle más personalidad a la nota mostrada, se le agregara un TextView para ponerle título de Nota, Un botón que dirá Editar para poder ir a la Activity de edición, un botón de imagen de flecha para regresar al menú principal de las notas, y dos Text View dentro de las dimensiones de la imagen de Nota de os cuales uno será para que ahí aparezca el título de la nota guardada y el segundo sea para el contenido de la nota.

Lo primero es crear el Layout que en este caso se llamara “noteview.xls”. El código está en el ANEXO 14.

Creando la parte lógica Class de la Activity de Visualización de Notas.

La parte logia de esta Activity mostrara los datos guardados en la base de datos de notas, esto se lograra al obtener los valores previamente almacenados en las dos columnas de la base de datos y uno lo asigna al TextView de título y el otro al del contenido. También se declara un botón el cual su funcionamiento será llevarnos a la activity de edición si es presionado, así si se presiona el botón imagen de flecha nos regresara a la Activity anterior que es el menú de notas.

Lo primero será crear un archivo de tipo Class que en este caso se llamara “NoteView”. El código se encuentra en el ANEXO 14.

En la figura 39 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuerdo a las necesidades planteadas anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. La Activity quedaría así:

Figura 39: Ejemplo de visualización de notas.

La Activity de Edición de Notas.

Esta Activity se encargara de dos Actividades que desee el usuario hacer, ya sea si va a crear apenas una nota y guardar los datos o si editara una nota antes guardada se mostraran los datos que hay guardados en la base de datos y podrá modificarlos para posteriormente guardar la nota ya editada. Estos datos se guardaran en dos columnas, una de título y otra del contenido de la nota. Se agregara un botón de guardar que tomara los datos ingresados por el usuario y los guardara en las columnas que ya tenía asignadas si solo fue una edición y si se creó una nueva nota los guarda en nuevas columnas para su posterior visualización o edición.

Creando el Layout de Edición de Notas.

Este Layout solo contendrá Un TextView al que le pondremos el título de "Edita tu Nota", También se le agregaran dos TextView para poder indicar donde se agregara el título de la nota y el contenido de la misma, y también se agregaran dos EditText en los cuales el usuario podrá ingresar los datos de su nota. Por ultimo llevara un botón para guardar los datos ingresados.

Lo primero es crear un archivo de tipo Layout que en este caso se llamara "note_edit.xml". El código se encuentra en el ANEXO 15.

Creando la Lógica de la Activity Edición de Notas.

En la lógica de esta Activity se trabajara con la base de datos al tomar los valores tanto de título como de su cuerpo de la nota y los asignara a las columnas correspondientes en caso de que solo se estuviera editando la nota, pero si se está creando una nueva nota le asignara unos nuevas columnas al presionar el botón guardar.

Lo primero será crear un archivo de tipo java con el nombre "NoteEdit.java". El código de esta Activity se encuentra en el ANEXO 15.

En la figura 40 se puede observar un ejemplo de lo que se logra programando el Layout y archivo Class de acuerdo a las necesidades planteadas

anteriormente, esto lo podemos modificar y editar a nuestro gusto de acuerdo a nuestro gusto y necesidades. La Activity queda así:

Figura 40: Ejemplo de edición de notas.

Archivo AndroidManifest.xls

Este archivo concentra todo lo necesario para que funcione la aplicación, en este archivo se declara el nombre de la aplicación, la actividad inicial, todas las actividades dentro de la aplicación ya que si no se declaran no funcionara la aplicación correctamente. En la figura 41 se observa un ejemplo de la configuración del Android Manifest utilizado en esta aplicación. El código de programación se encuentra en el ANEXO 18.

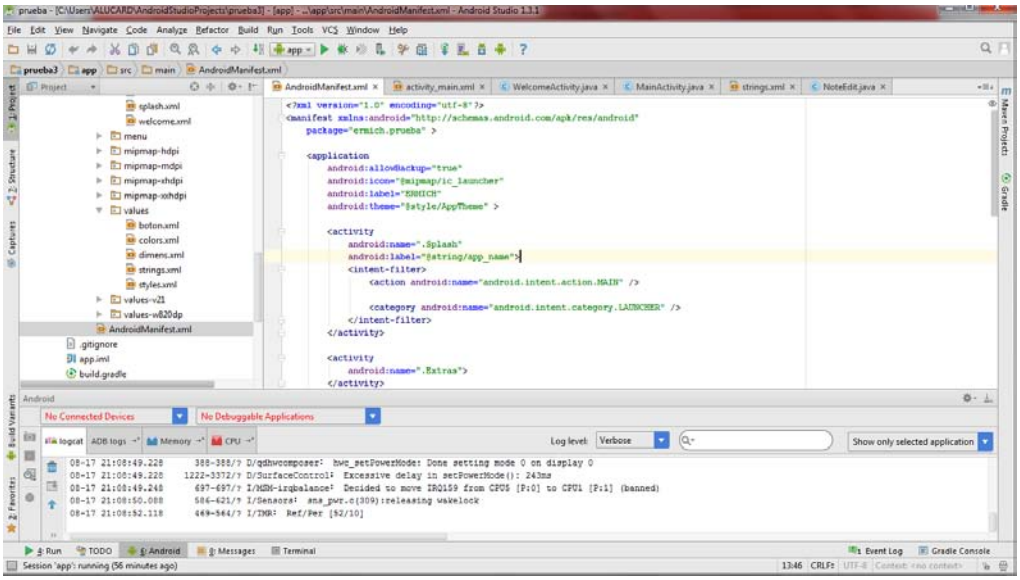

Figura 41: Ejemplo de la estructura de AndroidManifest.

Icono.

Por ultimo hay que crear un icono con el cual personalizaremos la aplicación y esto se logra con una imagen tipo png sin fondo, y se agrega si no la hay o si ya existe hay que re emplazar los iconos en las carpetas de sus respectivos tamaños: mipmap-mdpi (48x36px), mipmap-hdpi (75x54px), mipmap-xhdpi (96x73px) y mipmap-xxhdpi (144x110px). Un ejemplo de esto lo vemos en la figura 42 que muestra el icono usado en esta aplicación.

Figura 42: Muestra un ejemplo de icono parautilizar en el menú de Smarthpone Android.

CONCLUSIONES

Conclusiones.

Gracias al desarrollo de un proyecto utilizando la aplicación base vista en esta investigación y haciéndole las modificaciones necesarias, se llevó a cabo una implementación para móvil Smartphone con sistema operativo Android, dirigida al área de servicio técnico de una importante empresa de telecomunicaciones, específicamente enfocada a su área de atención a clientes, logrando reducir tiempo de operación y dar soluciones más eficaces, tanto a clientes como a compañeros de otras áreas. Los objetivos generales y particulares planteados al inicio del proyecto se cumplieron totalmente al satisfacer las necesidades requeridas por el área de servicio técnico, las cuales se detallan a continuación:

- Para comprobar que tanto beneficia una aplicación como ésta, se desarrolló un apoyo de consulta teórica y técnica para una empresa privada del sector de comunicaciones en telefonía móvil, en específico para el área y gerencia de Servicio Técnico. Se propuso la idea y fue considerada como una buena alternativa no solo para los Analistas de Servicio Técnico sino para los Supervisores y Gerente de esta área, ya que es bueno tener la información en la palma de la mano para cualquier momento que se necesite.
- Sabiendo que la información es poder, se realizó la aplicación con la información que se considera más importante y de más utilidad para poder hacerla útil en las labores diarias del personal de esta área de la empresa. Se le dio seguridad para poder dar tranquilidad a los pilares de la empresa y que sepan que su información estará segura y sin problemas de filtraciones de información.
- Además de la información, se investigó sobre algunas herramientas que podrían ser útiles para poder agilizar las operaciones de los integrantes de esta área y así reducir tiempos de operación, haciendo más productivo su trabajo. Por ello, se integró a la información un par de elementos como son un convertidor de Kb a Mb y GB, haciendo más rápidas y precisas las respuestas cuando un usuario tiene dudas acerca de este tipo de conversiones y con el consumo de datos de sus líneas

telefónicas. También se implementó un editor y visualizador de notas, logrando que el personal con acceso a esta aplicación concentre la información de esta empresa y también anexe la información que considere importante, ya que cada integrante personaliza su información y la hace más útil.

- Se llegó a la conclusión de que si se pudo cumplir el objetivo de reducción de tiempo de operación, ya que en ocasiones los Analistas de Servicio Técnico tienen a varios Asesores de Atención a Clientes realizándoles preguntas y se tenían que retirar de la ventanilla donde les proporcionan la atención para desplazarse a sus lugares y hacer una consulta de su información de intranet empresarial. Ahora solo tenían que sacar su Smartphone, con Sistema Operativo Android, abrir la aplicación que se realizó y solo ingresar su password. Al hacerlo ya tenían la información en sus manos, así como los extras que fueron bien recibidos, ya que cuando los Analistas necesitan una conversión rápida de Kb a Mb y Gb se pueda realizar al momento utilizando la tecnología. También el extra de notas gustó mucho por la información adicional que se puede guardar en esta, como contraseñas de cuentas de paquetería para envíos de equipos a reparación y demás información importante para cada Analista.
- Por parte de la gerencia se obtuvo una buena aceptación por la seguridad que brinda a la empresa, ya que esta aplicación si la llega a querer usar alguien no autorizado se bloquea después de tres intentos y ya no es posible usarla, solo reinstalándola y esto solo se podría hacer accediendo a la intranet corporativa de la empresa, haciendo que solo el personal autorizado le de uso a esta aplicación. Les agradó que la información se clasificara y se pusiera lo más importante, así como los extras, haciéndola más útil, ya que cada integrante agregaría información personal, logrando que sea personalizada por cada uno y con esto se logró que si se le diera uso y no se quedara como una aplicación mas. También es importante no perder de vista que toda la información está segura y no habrá filtraciones de información.

FUENTES

INFORMÁTICAS

Fuentes Informáticas.

/academiaandroid. (s.f.). Recuperado el 08 de junio de 2015, de Android Studio: instalación y primer proyecto.: http://academiaandroid.com/android-studio-instalacion-y-primer-proyecto/

¿Qué es un smartphone?. (jun-nov 2012). *Revista de ciencias sociales Prismasocial - Nº 8*.

978-607-02-6544-0, I. D. (2015). *Sistemas Operativos.* D. R. © UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO: Impreso y hecho en México.

academiaandroid. (s.f.). Recuperado el 09 de julio de 2015, de Android Studio v1.0: características y comparativa con Eclipse: http://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/

academiaandroid. (s.f.). Recuperado el 10 de julio de 2015, de Android Studio: instalación y primer proyecto.: http://academiaandroid.com/android-studio-instalacion-y-primer-proyecto/

Andrea. (s.f.). *smartupmarketing*. Recuperado el 25 de junio de 2015, de Aplicaciones Móviles Esenciales Para PYMES: Apps.: http://www.smartupmarketing.com/aplicaciones-moviles-esenciales-para-pymes-apps/

desarrollador-android. (s.f.). Recuperado el 16 de junio de 2015, de Descargar Android Studio.: http://desarrollador-android.com/desarrollo/herramientas/descargar/

formanet.ikaroo. (s.f.). Recuperado el 20 de junio de 2015, de La importancia de las aplicaciones móviles para las empresas.: http://formanet.ikaroo.es/noticias/la-importancia-de-las-aplicaciones-moviles-para-las-empresas-id-24113.htm

GmbH, v. (s.f.). *vogella*. Recuperado el 20 de julio de 2015, de Using the Android SQLite Database.: http://www.vogella.com/tutorials/AndroidSQLite/article.html

Gross, M. (s.f.). *manuelgross.bligoo*. Recuperado el 07 de junio de 2015, de Android: Origen, evolución y liderazgo del Sistema Operativo para Smartphones.: http://manuelgross.bligoo.com/20121026-android-origen-evolucion-y-liderazgo-del-sistema-operativo-para-smartphones

hermosaprogramacion. (s.f.). Recuperado el 12 de julio de 2015, de Tutorial De Bases De Datos SQLite En Aplicaciones Android.: http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos.html.

Hernández., H. (s.f.). *malavida*. Recuperado el 02 de julio de 2015, de La historia de Android: de Apple Pie 1.0 a Lollipop 5.0.: http://www.malavida.com/post/la-historia-de-android

HOY, E. (s.f.). *altonivel*. Recuperado el 11 de junio de 2015, de Sistemas operativos para móviles.: http://www.altonivel.com.mx/sistemas-operativos-para-moviles.html

kioskea. (s.f.). Recuperado el 13 de junio de 2015, de Introducción - Bases de datos.: http://es.kioskea.net/contents/66-introduccion-bases-de-datos.

LanceTalent. (s.f.). *lancetalent*. Recuperado el 15 de julio de 2015, de Las 8 ventajas de tener una aplicación móvil para tu empresa.: http://www.lancetalent.com/blog/las-8-ventajas-de-una-aplicacion-movil-para-tu-empresa/

Marker, G. (s.f.). *informatica-hoy*. Recuperado el 05 de junio de 2015, de Sistemas Operativos para Móviles.: http://www.informatica-hoy.com.ar/soluciones-moviles/Sistemas-Operativos-para-Moviles.php

Marker, G. (s.f.). *informatica-hoy*. Recuperado el 06 de junio de 2015, de Sistemas Operativos para Móviles II.: http://www.informatica-

hoy.com.ar/soluciones-moviles/Sistemas- Operativos-para-Moviles-II.php

Medeiros, H. (s.f.). *devmedia*. Recuperado el 03 de julio de 2015, de Como funciona la Criptografia Hash en Java.: http://www.devmedia.com.br/como-funciona-a-criptografia-hash-em-java/31139

Penella. (s.f.). *Abalit Technologies*. Recuperado el 21 de julio de 2015, de Importancia de tener una aplicación móvil.: http://blog.abalit.org/viewpost/55/importancia-de-tener-una-aplicacion-movil

PINGBACK. (s.f.). *ticweb*. Recuperado el 15 de Junio de 2015, de Los dispositivos móviles y su incidencia en nuestra vida cotidiana: http://www.ticweb.es/los-dispositivos-moviles-y-su-incidencia-en-nuestra-vida-cotidiana/

Prismasocial isdfundacion revista de ciencias sociales. (junio de 2012). Recuperado el 08 de junio de 2015, de ¿Qué es un smartphone?: http://www.isdfundacion.org/publicaciones/revista/numeros/8/secciones/tematica/pdf/04-smartphone-nueva-incertidumbre.pdf

Rivera, A. (s.f.). *pcworld*. Recuperado el 10 de julio de 2015, de Sistemas Operativos Móviles: Comunicación en tiempo real: http://www.pcworld.com.mx/Articulos/20734.htm

roshanpeter. (s.f.). *stackoverflow*. Recuperado el 19 de junio de 2015, de Android Limiting signup for one user.: http://stackoverflow.com/questions/19152998/android-limiting-signup-for-one-user

smartphoneavancetecnologico. (s.f.). Recuperado el 06 de Julio de 2015, de Historia y evolución del smartphone.: http://smartphoneavancetecnologico.blogspot.mx/p/historia-y-evolucion-del-smartphone.html

Subirats., J. (s.f.). *fandroides* . Recuperado el 20 de julio de 2015, de Qué es y como habilitar la depuración usb en android: http://www.fandroides.com/que-es-y-como-habilitar-la-depuracion-usb-en-android/

Tomás, J. (s.f.). *androidcurso*. Recuperado el 08 de junio de 2015, de Arquitectura de Android.: http://www.androidcurso.com/index.php/tutoriales-android/31-Unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android.

ANEXOS

ANEXO 1

CODIGO DE LAYOUT Y CLASS AL CREARSE UN NUEVO PROYECTO Y COMO SE IMPORTA UNA LIBRERIA.

Al crear un nuevo Layout tendremos el siguiente código dándole el nombre de ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>

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

   android:layout_width="match_parent"

   android:layout_height="match_parent">

</RelativeLayout>
```

Al crear un nuevo archivo Class con programación en java tendremos el siguiente código dándole el nombre de ejemplo:

```
package prueba;

public class ejemplo { }
```

Un ejemplo de código con librerías en los archivos Class importadas es el siguiente:

```
package ermich.prueba;

import android.content.Intent;

public class NoteView extends Activity {
```

En el código anterior se aprecia que se importan varias librerías para que la Activity NoteView pueda reconocer el código y realice su correcto funcionamiento, además se ve que el importar librerías se realiza al inicio del código en los archivos Class.

ANEXO 2

CODIGO DE ANIMACIONES FADE

Fade_in.xml El siguiente código llamado “fade_in” se utiliza para realizar efectos de difuminación de entrada:

```
<?xml version="1.0" encoding="UTF-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
   android:duration="400"
   android:fromAlpha="0.0"
   android:interpolator="@android:anim/accelerate_interpolator"
   android:repeatCount="0"
   android:toAlpha="1.0" />
```

fade_out.xml El siguiente código llamado “fade_out” se utiliza para realizar efectos de difuminación de salida:

```
<?xml version="1.0" encoding="UTF-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
   android:duration="400"
   android:fromAlpha="1.0"
   android:interpolator="@android:anim/accelerate_interpolator"
   android:repeatCount="0"
   android:toAlpha="0.0" />
```

ANEXO 3

LOGINDATABASEADAPTER COMO HERRAMIENTA PARA BASES DE DATOS

Lo primero es crear un nuevo archivo Class que llamaremos LoginDataBaseAdapter.java. El código es el siguiente:

```
package ermich.prueba;

import java.util.HashMap;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase;

public class LoginDataBaseAdapter {

/*Aquí se establecen los parámetros sobre los que se creara la base de datos, en las siguientes líneas se establece un nombre a la base de datos, versión y creación de la tabla de datos con una columna llamada login donde irán los passwords*/

   static final String DATABASE_NAME = "LOGIN";

   static final int DATABASE_VERSION = 1;

   static final String DATABASE_CREATE = "create table "+"LOGIN"+

         "( " +"ID integer primary key autoincrement,"+ "PASSWORD text) ";

   public SQLiteDatabase db;
```

```
   private final Context context;

   private DbHelper dbHelper;

/*Aquí nos indica que si hay una base de datos vea los datos de la columna
login*/

   public LoginDataBaseAdapter(Context _context)

   {

      context = _context;

      dbHelper = new DbHelper(context, "LOGIN", null, 1);

   }

/*Aquí se crea una flecha que seleccionara una dato de la columna de login*/

   public LoginDataBaseAdapter open() throws SQLException

   {

      db = dbHelper.getWritableDatabase();

      return this;

   }

   public void close()

   {

      db.close();

   }

/*Aqui se prepara a la base de datos para ingreso de valores*/
```

```
public SQLiteDatabase getDatabaseInstance()
{
    return db;
}
```

*/*Aquí se indica que se agregaran nuevos passwors o valores a la columna de login*/*

```
public ContentValues generarContentValues(String password){
    ContentValues valores = new ContentValues();
    valores.put("PASSWORD", password);
    return valores;
}
public void insertEntry(int id, String password)
{
    if (db != null){
    ContentValues valores = new ContentValues();
    valores.put("PASSWORD", password);
    valores.put("ID",id);
    db.insert("LOGIN", null, valores);
    db.close();}
}
```

```
/*Aquí se indica por si se desea borrar un valor dentro de la tabla*/

   public int deleteEntry(String password)

   {

      String where="PASSWORD=?";

      int numberOFEntriesDeleted= db.delete("LOGIN", where, new
String[]{password}) ;

      return numberOFEntriesDeleted;

   }
```

*/*Aquí se le dan instrucciones al cursor para que sea el encargado de buscar valores dentro de la base de datos*/*

```
   public String getSinlgeEntry(String password)

   {

      Cursor cursor=db.query("LOGIN", null, " PASSWORD=?", new
String[]{password}, null, null, null);

      if(cursor.getCount()<1) // UserName Not Exist

      {

         cursor.close();

         return "NOT EXIST"; }

      cursor.moveToFirst();

      cursor.close();

      return password; } }
```

ANEXO 4

DBHELPER COMO HERRAMIENTA PARA BASES DE DATOS.

Esto se lograra creando un archivo de tipo Class llamado DbHelper.java. El código es el siguiente:

```
package ermich.prueba;

import android.content.ContentValues;

import android.content.Context;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.util.Log;

public class DbHelper extends SQLiteOpenHelper {

/*Aquí se crea una base de datos indicándole que debe crear una base de datos con los datos del nombre versión, etc.*/

  public DbHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {

    super(context, name, factory, version);

  }

  @Override

  public void onCreate(SQLiteDatabase db) {

    db.execSQL(LoginDataBaseAdapter.DATABASE_CREATE);

  }

/*Aquí se indica que debe de actualizar los datos o valores guardados en la base de datos*/
```

```
   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
      db.execSQL("delete table if exist LOGIN");
      db.execSQL("create table LOGIN(ID integer primary key
autoincrement,"+ "PASSWORD text)");
   }
}
```

ANEXO 5

HERRAMIENTAS PARA UN BOTON EDITADO GRAFICAMENTE.

boton.xml. Aquí e declara las características del botón y el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>

<resources>

<style name="boton">

   <item name="android:gravity">center</item>

   <item name="android:textColor">#ff000000</item>

   <item name="android:textSize">15dp</item>

   <item name="android:layout_height">40dp</item>

   <item name="android:layout_width">140dp</item>

   <item name="android:textStyle">bold|italic</item>

   <item name="android:paddingLeft">10dp</item>

   <item name="android:paddingRight">10dp</item>

   <item name="android:paddingBottom">5dp</item>

   <item name="android:paddingTop">5dp</item>  </style>

   <style name="botonazul"

      parent="@style/boton">

      <item name="android:background">@drawable/button_states
```

```
</item> </style>  </resources>
```

button_states.xml. Aquí se declara que hara el botón al estar sin presionar y al presionarlo, el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android">

   <!-- Estado Pulsado -->

   <item android:drawable="@drawable/blanco"

       android:state_pressed="true"/>

   <!-- Estado Normal -->

   <item android:drawable="@drawable/fondo">

   </item> </selector>
```

Fondo.xml. Es el color que tendrá el botón es su estado normal al no ser presionado, el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android">

      <stroke

         android:width="0dp"

         android:color="#ff060bff"/>

      <gradient

         android:startColor="#ff060bff"
```

```
        android:centerColor="#ffffffff"
        android:endColor="#ff060bff"
        android:angle="270"
        />
    <padding
        android:left="1dp"
        android:top="2dp"
        android:right="1dp"
        android:bottom="2dp"/>
    <corners
        android:radius="25dp"/>  </shape>
```

blanco.xml. Es el color que tendrá el botón al estar presionado y al estar sin presionar, el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <stroke
    android:width="0dp"
    android:color="#ff060bff"/>
  <gradient
```

```
        android:startColor="#fff8fff7"
        android:centerColor="#ff060bff"
        android:endColor="#fff9fff6"
        android:angle="270"
        />
    <padding
        android:left="1dp"
        android:top="2dp"
        android:right="1dp"
        android:bottom="2dp"/>
    <corners
        android:radius="100dp"/>
</shape>
```

ANEXO 6

CODIGO DE UN SPLASH

splash.xml. El código del Layout del Splash es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".SplashScreenActivity">
   <ImageView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:contentDescription="@string/app_name"
      android:scaleType="fitXY"
      android:src="@drawable/logo" />
</LinearLayout>
```

En el código anterior se ve que solo se agregó una imagen (ImageView) con el logotipo de la empresa, la cual esta guardada en la carpeta Drawable con sus diferentes tamaños de pantalla y el cual se llama logo.

Splash.java. Lo primero es crear una nueva Class en este caso se llama "Splash". El código así con la descripción de esté es el siguiente:

```
package ermich.prueba;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
public class Splash extends Activity {
   // Aquí se configura la duración del Splash.
   private static final long SPLASH_SCREEN_DELAY = 2000;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
/* Al iniciar la activity realizara una animación de difuminación de entrada y
Salida*/

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
// Aquí se oculta la barra de notificaciones.
      requestWindowFeature(Window.FEATURE_NO_TITLE);
/*Aquí le indicamos que nos muestre en pantalla lo que tengamos guardado
en el Layout Splash.*/
```

```
        setContentView(R.layout.splash);
        /* Al iniciar la activity realizara una animación de difuminación de
entrada y Salida*/
        overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
//Aquí se indica que Actividad iniciara después que será la actividad principal.
                Intent mainIntent = new Intent().setClass(
                        Splash.this, MainActivity.class);
                startActivity(mainIntent);
// Aquí se indica que si se oprime el botón regresar se salga de la App.
                finish();
            }
        };
        // Aquí se indica que existe un reloj que toma el tiempo de duración.
        Timer timer = new Timer();
        timer.schedule(task, SPLASH_SCREEN_DELAY);
    }
    @Override
    public void onDestroy(){
        super.onDestroy();
        finish();
    }
}
```

ANEXO 7

PANTALLA DE LOGIN. CODIGOS DE LAYOUT Y CLASS

activity_main.xml. Será un Layout relativo. El código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="match_parent"
   android:background="@drawable/wallpaper">
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceMedium"
      android:text="@string/Titulo"
      android:id="@+id/ingreso"
      android:textColor="@color/Aazul"
      android:textStyle="bold|italic"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="159dp"
      android:textSize="23dp"
```

```
    android:gravity="center" />
<EditText
    android:layout_width="220dp"
    android:layout_height="50dp"
    android:inputType="textPassword"
    android:ems="10"
    android:id="@+id/password"
    android:hint="     Password"
    android:hapticFeedbackEnabled="false"
    android:layout_below="@+id/ingreso"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="10dp"
    android:textAlignment="center" />
<Button
    android:id="@+id/login"
    style="@style/botonazul"
    android:layout_marginTop="38dp"
    android:layout_below="@+id/password"
    android:text="Ingresar"
    android:onClick="login"
    android:layout_centerHorizontal="true" />
```

```
  <TextView
     android:layout_width="wrap_content"
     android:layout_height="match_parent"
     android:textAppearance="?android:attr/textAppearanceLarge"
     android:textStyle="bold|normal"
     android:id="@+id/intentos"
     android:textColor="@color/Aazul"
     android:layout_below="@+id/login"
     android:layout_centerHorizontal="true"
     android:layout_marginTop="63dp" />
  <TextView
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:id="@+id/tv"
     android:layout_alignTop="@+id/intentos"
     android:layout_centerHorizontal="true" />
</RelativeLayout>
```

MainActivity.java. En este archivo se dará la funcionalidad a la pantalla de login y el código es el siguiente:

```
package ermich.prueba;
import android.content.Intent;
import android.graphics.Color;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.app.Activity;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {
   /*Aquí se declaran las componentes del Layout así como el número de
intentos máximo.*/
   private TextView attempts;
   private Button login;
   private EditText password;
   static int counter = 3;
```

```
   SharedPreferences app_preferences;
   LoginDataBaseAdapter loginDataBaseAdapter;
   private String passwd;
   @Override
/*Aquí se indica que lea los datos guardados de número de intentos
restantes.*/
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
//Aquí se indica que no se vea la barra de notificaciones.

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
      // Aquí se oculta la barra de notificaciones.
      requestWindowFeature(Window.FEATURE_NO_TITLE);
/*Aquí se indica el Layout de la Activity a la que se la está dando la lógica de
programación.*/
      setContentView(R.layout.activity_main);
      /* Al iniciar la activity realizara una animación de difuminación de
entrada y Salida*/
      overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
/*Aquí se declaran todas las componentes con la que se trabajara la lógica de
programación.*/
      attempts = (TextView) findViewById(R.id.intentos);
      attempts.setText(Integer.toString(counter));
      login = (Button) findViewById(R.id.login);
      password = (EditText) findViewById(R.id.password);
```

```
        attempts = (TextView) findViewById(R.id.intentos);
        attempts.setText(Integer.toString(counter));
        password = (EditText) findViewById(R.id.password);
        login = (Button) findViewById(R.id.login);
/*Aquí se indica que tendrá una animación de difuminación de entrada y
salida.*/
        overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
/*Creando la base de datos de passwords con codificación SHA-256 si es la
primera vez que se ejecuta, si ya existe la base de datos solo actualiza los
passwords no existentes.*/
        loginDataBaseAdapter = new
LoginDataBaseAdapter(getApplicationContext());
        loginDataBaseAdapter.open();
        LoginDataBaseAdapter manager = new LoginDataBaseAdapter(this);
        manager.open();
/*Se asignaran 2 passwords, el primero será la palabra"ejemplo1" y la
segunda "ejemplo2" ambos códificados en SHA-256*/
        manager.insertEntry(1,
"531647a0776ccd779c5333f0945e6cccaf8c9f4cd1c85bf755a1bbe444ae9866
");
        manager.close();
        manager.open();
        manager.insertEntry(2,
"e1f9bece32ddf8071b8565af942fe1ba362691869a2b4a54b0eb9136b4e413c
4");
```

```
        manager.close();

//Leyendo los datos guardados en el número de intentos para acceder.

        app_preferences = this.getSharedPreferences("myPrefscount",
MODE_PRIVATE);
        counter = app_preferences.getInt("counter", 3);
        attempts.setText(Integer.toString(counter));
/*Poniendo una condición para que en caso de que el contador llegue a cero
se bloquee y se cierre la App.*/
        if (counter <= 0) {
            Toast.makeText(getApplicationContext(), "APLICACION BLOQUEADA",
                    Toast.LENGTH_LONG).show();
            finish();
        }
    }
    //Lectura del código insertado y codificación en código de 256 bits SHA 1.
    public void login(View view) {
        passwd= password.getText().toString();
        MessageDigest digest=null;
        String hash= "";
        try {
            digest = MessageDigest.getInstance("SHA-256");
            digest.update(passwd.getBytes());
            hash = bytesToHexString(digest.digest());
```

```
        }
        catch (NoSuchAlgorithmException e1) {
            e1.printStackTrace();
        }
//Abriendo la base de datos donde se encuentran los passwords codificados.
        LoginDataBaseAdapter manager = new LoginDataBaseAdapter(this);
        manager.open();
        String storedPassword = loginDataBaseAdapter.getSinlgeEntry(hash);
/*Comparando el password insertado con el guardado en la base de datos, si
es correcto inicia la siguiente Activity, si es incorrecto se descuenta un número
del contador.*/
        if (hash.equals(storedPassword))
        {
/*Aquí se indica que si el password es correcto con alguno de los guardados
se accederá a la siguiente Activity declarada, en este caso es la que tiene un
Class llamado Welcome.*/
            Intent i = new Intent(MainActivity.this, WelcomeActivity.class);
            startActivity(i);
            SharedPreferences.Editor editor = app_preferences.edit();
            counter = 3;
            editor.putInt("counter", counter);
            editor.commit();
            finish();
        } else {
            password.setText("");
```

```
        Toast.makeText(getApplicationContext(), "Password Incorrecto",
              Toast.LENGTH_LONG).show();
        attempts.setBackgroundColor(Color.WHITE);
        counter--;
        attempts.setText(Integer.toString(counter));
        if (counter == 0) {
           login.setEnabled(false);
           password.setEnabled(false);
           SharedPreferences.Editor editor = app_preferences.edit();
           editor.putInt("counter", counter);
           editor.commit();
/*Si el contador llega a cero se cierra la aplicación y aparecerá un texto
emergente que dice “Aplicación Bloqueada” y no se podrá abrir nunca más.*/
           Toast.makeText(getApplicationContext(), "APLICACION
BLOQUEADA",
                 Toast.LENGTH_LONG).show();
           finish();
        }
     }
     manager.close();
  }
  // Función para convertir en Hexadecimal.
  private static String bytesToHexString(byte[] bytes) {
     StringBuffer sb = new StringBuffer();
     for (int i = 0; i < bytes.length; i++) {
```

```
        String hex = Integer.toHexString(0xFF & bytes[i]);
        if (hex.length() == 1) {
            sb.append('0');
        }
        sb.append(hex);
    }
    return sb.toString();
}
// Al cerrarse por completo la aplicación, guardamos las preferencias.
@Override
public void onDestroy() {
    super.onDestroy();
    guardarPreferencias();
}
// Al abrir la aplicación, cargamos preferencias.
@Override
protected void onStart() {
    super.onStart();
    cargarPreferencias();
}
// Guardar configuración aplicación Android usando SharedPreferences.
public void guardarPreferencias() {
    SharedPreferences.Editor editor = app_preferences.edit();
    editor.putInt("counter", counter);
    editor.commit();    }
```

```
   // Cargar configuración aplicación Android usando SharedPreferences.
   public void cargarPreferencias() {
      app_preferences = this.getSharedPreferences("myPrefscount",
MODE_PRIVATE);
      counter = app_preferences.getInt("counter", 3);
      attempts.setText(Integer.toString(counter));
   }
   /*Indicando que cargue preferencias cuando se regrese a la App después de
ponerse en pause*/
   @Override
   protected void onResume() {
      super.onResume();
      cargarPreferencias();

   }
   /*Se indica que guarde las preferencias al ponerse en pause*/
   @Override
   protected void onPause() {
      super.onPause();
      guardarPreferencias();
   }
   /*Aquí se indica que se guarden las preferencias cuando se detenga la
aplicación*/
   @Override
   protected void onStop() {
```

```
        super.onStop();
        guardarPreferencias();
    }
    /*Aquí se indica que se lean las preferencias guardadas al reiniciar la
aplicación*/
    @Override
    protected void onRestart() {
        super.onRestart();
        cargarPreferencias();
    }
}
```

ANEXO 8

PANTALLA DE BIENVENIDA

welcome.xml será un Layout relativo. El código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:background="@drawable/wallpaper"
   >
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="Welcome !!"
      android:id="@+id/txWelcome"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true"
      android:textStyle="bold|italic"
      android:textAllCaps="false"
      android:textColor="@color/Aazul"
      android:textSize="30dp"
```

```
    android:textIsSelectable="true"
    android:gravity="center|center_vertical|center_horizontal" />
</RelativeLayout>
```

WelcomeActivity.java. La programación en java de este archivo es la siguiente:

```
package ermich.prueba;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.Window;
public class WelcomeActivity extends Activity {
   // Aquí se declara la duración del Splash.
   private static final long SPLASH_SCREEN_DELAY = 1200;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
//Aquí se declara que solo se puede ver en forma vertical.

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
      //Aquí se declara que no muestre la barra de notificaciones.
      requestWindowFeature(Window.FEATURE_NO_TITLE);
```

```
//Aquí se declara a que layout se le esta dando la logica de programacion.

        setContentView(R.layout.welcome);
        //Aquí se declara que hay animación d entrada y salida.
        overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
//Aquí se da de alta un timer.
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
//Aquí se declara cual es la siguiente actividad cuando termine el Splash.
                Intent mainIntent = new Intent().setClass(
                        WelcomeActivity.this, MenuActivity.class);
                startActivity(mainIntent);
// Aquí se indica que si se presiona el botón de regreso se cierre la App.
                finish();
            }
        };
        // Se simula el proceso de carga al dejar avanzar el timer.
        Timer timer = new Timer();
        timer.schedule(task, SPLASH_SCREEN_DELAY);
    }
}
}
```

ANEXO 9

PANTALLA DEL MENU PRINCIPAL

menu.xls. Este archivo será un Relative Layout y el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:background="@drawable/wallpaper"
   >
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="SERVICIO TECNICO R9"
      android:textColor="@color/Aazul"
      android:textStyle="bold|italic"
      android:id="@+id/intentos"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="42dp"
      android:textSize="25dp"
```

```
    android:textIsSelectable="false"
    android:hapticFeedbackEnabled="false"
    android:gravity="center" />
<Button
    android:id="@+id/btButton1"
    style="@style/botonazul"
    android:text="Garantias"
    android:layout_below="@+id/intentos"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="51dp"
    android:layout_marginLeft="10dp"
    android:layout_marginBottom="10dp" />
<Button
    android:id="@+id/btButton2"
    style="@style/botonazul"
    android:text="Envios"
    android:layout_alignTop="@+id/btButton1"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp" />
```

```
<Button
   android:id="@+id/btButton3"
   style="@style/botonazul"
   android:text="SAP"
   android:layout_below="@+id/btButton1"
   android:layout_alignLeft="@+id/btButton1"
   android:layout_alignStart="@+id/btButton1"
   android:layout_marginTop="10dp"
   android:layout_marginBottom="10dp" />

<Button
   android:id="@+id/btButton4"
   style="@style/botonazul"
   android:text="ECAC y SERTEC"
   android:layout_below="@+id/btButton2"
   android:layout_alignParentRight="true"
   android:layout_alignParentEnd="true"
   android:layout_marginRight="10dp"
   android:layout_marginBottom="10dp"
   android:layout_marginTop="10dp" />
<Button
```

```
    style="@style/botonazul"
    android:text="Hard Resets"
    android:id="@+id/btButton30"
    android:layout_below="@+id/btButton3"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="10dp" />
<Button
    style="@style/botonazul"
    android:text="Act de SW"
    android:id="@+id/btButton31"
    android:layout_below="@+id/btButton4"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginTop="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp" />
<Button
    android:text="Almacen"
    style="@style/botonazul"
```

```
    android:id="@+id/btButton32"
    android:layout_below="@+id/btButton30"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="10dp" />
<Button
    style="@style/botonazul"
    android:text="Extras"
    android:id="@+id/btButton33"
    android:layout_below="@+id/btButton31"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginTop="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp" />
<Button
    style="@style/botonazul"
    android:text="Salir"
    android:layout_marginBottom="130dp"
    android:id="@+id/btButton34"
    android:layout_alignParentBottom="true"
```

```
    android:layout_centerHorizontal="true" />
</RelativeLayout>
```

MenuActivity.java. Este archivo es la programación de la lógica del menú principal y el código es el siguiente:

```
package ermich.prueba;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.net.PasswordAuthentication;
public class MenuActivity extends Activity{
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      //Aquí se indica que la solo se usara en modo vertical
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
```

```
//Aquí se indica que no llevara barra de notificaciones
        requestWindowFeature(Window.FEATURE_NO_TITLE);
//Aquí se indica que llevara animacion de entrada y salida de difuminación.
        overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
//Aquí se indica a cual Layout se le está dando funcionalidad.
        setContentView(R.layout.menu);

//Iniciamos el Botón número almacen.
        Button boton7 = (Button) findViewById(R.id.btButton32);
        boton7.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MenuActivity.this,
                        Almacen.class);
                startActivity(intent);
                finish();
            }
        });

        //Iniciamos el Botón número extras.
        Button boton8 = (Button) findViewById(R.id.btButton33);
        boton8.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MenuActivity.this,
```

```
                Extras.class);
            startActivity(intent);
            finish();
        }
    });

//Iniciamos el Botón el botón "Salir" el cual cerrara la aplicación.
    Button boton10 = (Button) findViewById(R.id.btButton34);
    boton10.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            finish();
        }
    });
  }
}
```

ANEXO 10

LA ACTIVITY DE CONTENIDO TÉCNICO O TEÓRICO.

almacen.xls. Este archivo será de tipo Relative Layout y el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>

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

   android:layout_width="match_parent"
android:layout_height="match_parent"

   android:background="@drawable/wallpaper">

   <TextView

      android:layout_width="match_parent"

      android:layout_height="wrap_content"

      android:textAppearance="?android:attr/textAppearanceLarge"

      android:text="Almacen de Blackberry"

      android:textColor="@color/Aazul"

      android:textStyle="bold|normal|italic"

      android:id="@+id/titulo_envios"

      android:textSize="25dp"

      android:textIsSelectable="false"

      android:hapticFeedbackEnabled="false"

      android:gravity="center"

      android:layout_marginTop="40dp"
```

```
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp" />
<ImageButton
    android:id="@+id/imageButton30"
    android:background="@drawable/flecha"
    android:layout_gravity="bottom|right"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp"
    android:layout_width="50dp"
    android:layout_height="50dp" />
<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/scrollView"
    android:layout_below="@+id/titulo_envios"
    android:layout_above="@+id/imageButton30">

    <LinearLayout
        android:orientation="vertical"
```

```
android:layout_width="fill_parent"

android:layout_height="fill_parent">

<TextView

  android:layout_width="wrap_content"

  android:layout_height="wrap_content"

  android:textAppearance="?android:attr/textAppearanceMedium"

   android:text="El almacén es un lugar especialmente estructurado y planificado para custodiar, proteger y controlar los bienes de activo fijo o variable de la empresa, antes de ser requeridos para la administración, la producción o al venta de artículos o mercancías."

  android:id="@+id/textView200"

  android:layout_below="@+id/titulo_envios"

  android:layout_alignParentLeft="true"

  android:layout_alignParentStart="true"

  android:textColor="@color/Aazul"

  android:textStyle="bold|normal|italic"

  android:layout_marginTop="20dp"

  android:layout_marginLeft="10dp"

  android:layout_marginRight="10dp"

  android:layout_marginBottom="10dp" />

<ImageView

  android:layout_width="wrap_content"

  android:layout_height="wrap_content"
```

```
            android:id="@+id/imageView2"
            android:background="@drawable/devices"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:layout_marginBottom="15dp" />
    </LinearLayout>
  </ScrollView>
</RelativeLayout>
```

Almacen.java. Este archivo será la lógica de programación de una actividad que llevara contenido para realizar una consulta de información teórica o técnica y su código es el siguiente:

```
package ermich.prueba;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.ImageButton;
public class Almacen extends Activity {
  //Aquí se declara que tenemos un botón de tipo imagen.
  ImageButton imageButton;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
//Aquí le indicamos que el Layout de esta Actividad es almacen.xls.
        setContentView(R.layout.almacen);
        overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
//Aquí se declara que el boton tendra fincinalidad al precionarlo.
        addListenerOnButton();
    }
    /*Aquí se le indica que el botón al ser presionado ir a a otra Actividad en
este caso regresara a la Actividad de Menú.*/
    private void addListenerOnButton() {
        imageButton = (ImageButton) findViewById(R.id.imageButton30);
        imageButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(Almacen.this,
                        MenuActivity.class);
                startActivity(i);
                finish();
                  }
              }
         ) ;
     }
}
```

ANEXO 11

ACTIVIDAD DE EXTRAS

extras.xls. Este archivo será un Relative Layout y el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@drawable/wallpaper"
   android:weightSum="1">
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="Extras"
      android:textColor="@color/Aazul"
      android:textStyle="bold|normal|italic"
      android:id="@+id/titulo_envios"
      android:textSize="25dp"
      android:textIsSelectable="false"
      android:hapticFeedbackEnabled="false"
      android:gravity="center"
```

```
    android:layout_marginTop="40dp"/>
<ImageButton
    android:id="@+id/imageButton8"
    android:background="@drawable/flecha"
    android:layout_gravity="bottom|right"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp"
    android:layout_width="50dp"
    android:layout_height="50dp" />
<Button
    style="@style/botonazul"
    android:text="Convertir KB a MB y GB"
    android:id="@+id/button150"
    android:layout_marginTop="50dp"
    android:textSize="12dp"
    android:layout_below="@+id/titulo_envios"
    android:layout_alignLeft="@+id/button151"
    android:layout_alignStart="@+id/button151" />
<Button
```

```
    style="@style/botonazul"
    android:text="Notas"
    android:id="@+id/button151"
    android:layout_below="@+id/button150"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="20dp" />
</RelativeLayout>
```

Extras.java. Este archive tendrá programado en java la lógica de la actividad extras la cual es solo un menú con dos botones cada uno con su actividad a realizar. El código es el siguiente:

```
package ermich.prueba;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.ImageButton;
public class Extras extends Activity {
   //Aqui se declara que hay un boton de imagen.
   ImageButton imageButton;
```

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

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
//Aquí se indica a que Layout se hace referencia.
      setContentView(R.layout.extras);
      overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
      addListenerOnButton();
   }
   //Aquí se da funcionalidad al botón de imagen de fleche de regreso.
   private void addListenerOnButton() {
      imageButton = (ImageButton) findViewById(R.id.imageButton8);
      imageButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            Intent i = new Intent(Extras.this,
                  MenuActivity.class);
            startActivity(i);
            finish();  }
      });
//Aquí se indica que el que actividad iniciara al presionar el botón de Notas.
      Button boton2 = (Button) findViewById(R.id.button151);
      boton2.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
           Intent intent = new Intent(Extras.this,
                 NotePad.class);
           startActivity(intent);
           finish();
        }
     });
//Aquí se indica que el que actividad iniciara al presionar el botón de
Convertir KB a MB y GB.
     Button boton3 = (Button) findViewById(R.id.button150);
     boton3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           Intent intent = new Intent(Extras.this,
                 Converter.class);
           startActivity(intent);
           finish();   }
             }
         );
     }
}
```

ANEXO 12

ACTIVITY DEL CONVERSOR DE KB A MB Y GB.

converter.xls. Sera un archivo de tipo Relative Layout. Y el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>

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

   android:layout_width="match_parent"

   android:layout_height="match_parent"

   android:background="@drawable/wallpaper"

   >

   <TextView

      android:layout_width="match_parent"

      android:layout_height="wrap_content"

      android:textAppearance="?android:attr/textAppearanceLarge"

      android:text="Convertidor KB a MB y GB"

      android:textColor="@color/Aazul"

      android:textStyle="bold|normal|italic"

      android:id="@+id/titulo_almacen"

      android:textSize="25dp"

      android:textIsSelectable="false"

      android:hapticFeedbackEnabled="false"
```

```
    android:gravity="center"
    android:layout_marginTop="40dp"/>
<ImageButton
    android:id="@+id/imageButton153"
    android:background="@drawable/flecha"
    android:layout_gravity="bottom|right"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp"
    android:layout_width="50dp"
    android:layout_height="50dp" />
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/editText50"
    android:layout_below="@+id/titulo_almacen"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
```

```
    android:layout_marginTop="60dp"
    android:hint="Ingresa KB a Convertir"
    android:layout_marginLeft="10dp" />
<Button
    android:layout_width="60dp"
    style="@style/botonazul"
    android:text="GO"
    android:id="@+id/button154"
    android:layout_alignTop="@+id/editText50"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginRight="20dp" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView50"
    android:textAlignment="center"
    android:textSize="20dp"
    android:textColor="@color/Aazul"
    android:layout_below="@+id/editText50"
    android:layout_marginTop="20dp"
    android:layout_toLeftOf="@+id/textView51"
```

```
    android:layout_toStartOf="@+id/textView51"
    android:layout_alignLeft="@+id/editText50"
    android:layout_alignStart="@+id/editText50" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MB"
    android:id="@+id/textView51"
    android:textSize="20dp"
    android:layout_alignTop="@+id/textView50"
    android:layout_alignRight="@+id/editText50"
    android:layout_alignEnd="@+id/editText50"
    android:textColor="@color/Aazul"
    android:layout_alignBottom="@+id/textView50" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView52"
    android:textAlignment="textEnd"
    android:textSize="20dp"
    android:textColor="@color/Aazul"
    android:layout_toStartOf="@+id/textVie53"
```

```
    android:layout_alignLeft="@+id/textView50"
    android:layout_alignStart="@+id/textView50"
    android:layout_below="@+id/textView50"
    android:layout_alignRight="@+id/textView50"
    android:layout_alignEnd="@+id/textView50"
    android:layout_marginTop="20dp" />
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="GB"
    android:id="@+id/textVie53"
    android:layout_alignTop="@+id/textView52"
    android:layout_alignRight="@+id/textView51"
    android:layout_alignEnd="@+id/textView51"
    android:textSize="20dp"
    android:textColor="@color/Aazul" />
</RelativeLayout>
```

Converter.java. Este archive llevara la lógica de programación en java del conversor. El código es el siguiente:

```
package ermich.prueba;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
```

```
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
public class Converter extends Activity {
   //Aquí se declaran los elementos que tendrán funcionabilidad.
   private Button calcular;
   private EditText ingreso;
   private TextView resultado;
   private TextView resultado2;
   ImageButton imageButton;
   @Override
   protected void onCreate (Bundle savedInstanceState){
      super.onCreate(savedInstanceState);

      //Aquí se indica que no se vea la barra de notificaciones.

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
      // Aquí se oculta la barra de notificaciones.
      requestWindowFeature(Window.FEATURE_NO_TITLE);
/*Aquí se indica el Layout de la Activity a la que se la está dando la lógica de
programación.*/
```

```
        setContentView(R.layout.converter);
    //Aquí se indica que el botón.
        calcular = (Button) findViewById(R.id.button154);
    //Aquí se indica que existe un botón de imagen que será la fleche de regreso.
        imageButton = (ImageButton) findViewById(R.id.imageButton153);
        overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
        addListenerOnButton();
    //Aqui se declara los componentes que tendra esta pantalla.
        calcular.setOnClickListener(new View.OnClickListener() {
            @Override
    //Aquí se indican las componentes y se declara un texto de visualización.
            public void onClick(View agr0) {
                ingreso = (EditText) findViewById(R.id.editText50);
                resultado = (TextView) findViewById(R.id.textView50);
                resultado2 = (TextView) findViewById(R.id.textView52);
    //Aquí se indican las operaciones a realizar por cada conversor.
                try {
                    float kb = Float.parseFloat(ingreso.getText().toString());
                    float mb = (kb / 1024);
                    float gb = (mb / 1024);
    //Aquí se indica donde se deben de asignar los resultados obtenidos.
                    resultado.setText("" + mb);
                    resultado2.setText(""+ gb);
                } catch (Exception e) {
                }
```

```
            ingreso.setText("");
        }
    });
  }
  /*Aquí se le da funcionalidad al botón imagen de fleche indicándole que
Actividad iniciara al ser presionado en este caso regresara a la Actividad
Extras.*/
  private void addListenerOnButton() {
    imageButton = (ImageButton) findViewById(R.id.imageButton153);
    imageButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Intent i = new Intent(Converter.this,
            Extras.class);
        startActivity(i);
        finish();
      }
    });
  }
  @Override
  public void finish() {
    super.finish();
  }
}
```

ANEXO 13

ACTIVITY DE NOTAS

Layout de pantalla principal de la Activity Notas.

notepad.xls. Este archivo será de tipo Relative Layout y será la pantalla principal de la actividad de notas ya que esta es una mini aplicación dentro de la aplicación principal. El código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@drawable/wallpaper"
   android:weightSum="1">
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="Notas Rapidas"
      android:textColor="@color/Aazul"
      android:textStyle="bold|normal|italic"
      android:id="@+id/titulo_envios"
      android:textSize="25dp"
      android:textIsSelectable="false"
```

```
    android:hapticFeedbackEnabled="false"
    android:gravity="center"
    android:layout_marginTop="40dp"/>
<ImageButton
    android:id="@+id/imageButton150"
    android:background="@drawable/flecha"
    android:layout_gravity="bottom|right"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp"
    android:layout_width="50dp"
    android:layout_height="50dp" />
<ListView
    android:id="@+id/android:list"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textFilterEnabled="false"
    android:textColor="@color/Aazul"
    android:textSize="20dp"
    android:layout_below="@+id/android:empty"
```

```
    android:layout_alignLeft="@+id/android:empty"
    android:layout_alignStart="@+id/android:empty"
    android:layout_alignRight="@+id/button153"
    android:layout_alignEnd="@+id/button153"
    android:layout_above="@+id/imageButton150"
    android:layout_marginTop="20dp"
    android:cacheColorHint="@color/Aazul" />
<TextView
    android:id="@+id/android:empty"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="no_notes"
    android:textColor="@color/Aazul"
    android:textSize="20dp"
    android:layout_below="@+id/button153"
    android:layout_alignLeft="@+id/button153"
    android:layout_alignStart="@+id/button153"
    android:layout_marginTop="20dp" />
<Button
    style="@style/botonazul"
    android:text="Nueva nota"
    android:id="@+id/button153"
```

```
    android:layout_below="@+id/titulo_envios"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginLeft="15dp"
    android:layout_marginTop="20dp" />
</RelativeLayout>
```

NotePad.java. Este Archivo contiene la programación java de la Actividad del menú principal de las Notas. EL código es el siguiente:

```
package ermich.prueba;
import android.app.Activity;
import android.app.ListActivity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.database.Cursor;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Window;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ListView;
```

```
import android.widget.SimpleCursorAdapter;
import android.widget.AdapterView.AdapterContextMenuInfo;
public class NotePad extends ListActivity {
   //Aquí se declara el botón de imagen.
   ImageButton imageButton;
   private static final int ACTIVITY_CREATE=0;
   private static final int ACTIVITY_EDIT=1;
   private static final int INSERT_ID = Menu.FIRST;
   private static final int DELETE_ID = Menu.FIRST + 1;
   private NotesDbAdapter mDbHelper = null;
   @Override
/*Aquí se indica que al crear la activity nos muestre visualmente lo que existe
en el Layout llamado notepad, hacemos referencia a la herramienta
mDbHelper para la base de dato, también se agrega la instrucción para darle
función a un botón y se declara las animaciones de entrada y salida*/
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      requestWindowFeature(Window.FEATURE_NO_TITLE);

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
      setContentView(R.layout.notepad);
      mDbHelper = new NotesDbAdapter(this);
      mDbHelper.open();
      fillData();
      registerForContextMenu(getListView());
```

```
        addListenerOnButton();
        overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
    }
    private void addListenerOnButton() {
/*Se inicia declarando la función de un botón que será la fleche de regreso a
la Activity anterior*/
        imageButton = (ImageButton) findViewById(R.id.imageButton150);
        imageButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(NotePad.this,
                        Extras.class);
                startActivity(i);
                finish();
            }
        });
/*Aquí se declara que al presionar este botón que será el de nueva nota nos
envié a la Activity de editar nota*/
        Button boton1 = (Button) findViewById(R.id.button153);
        boton1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(NotePad.this,
                        NoteEdit.class);
                startActivity(intent);
```

```
            finish();
        }
    });
  }
  /*Aquí se declara que en caso de haber unas notas ya almacenadas nos ayude la herramienta mDbHeper y se enlisten en el layout de acuerdo a como fueron guardadas.*/
  private void fillData() {
    Cursor notesCursor = mDbHelper.fetchAllNotes();
    startManagingCursor(notesCursor);
/*Se crea un vector para especificar los campos que se mostraran en la lista (solo los títulos)*/
    String[] from = new String[]{NotesDbAdapter.KEY_TITLE};
    int[] to = new int[]{R.id.text1};
// Ahora se crea un cursor para seleccionar la nota a mostrar
    SimpleCursorAdapter notes =
        new SimpleCursorAdapter(this, R.layout.notes_row, notesCursor, from, to);
    setListAdapter(notes);
  }
  /*Se crean opciones en las que al agregar una nueva nota se le agrega un ID para identificarla.*/
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
```

```
    menu.add(0, INSERT_ID, 0, R.string.menu_insert);
    return true;
  }
  @Override
  public boolean onMenuItemSelected(int featureId, MenuItem item) {
    switch(item.getItemId()) {
      case INSERT_ID:
        createNote();
        return true;
    }
    return super.onMenuItemSelected(featureId, item);
  }
  //Se dan condiciones para borrar una nota que ya no se ocupe.
  @Override
  public void onCreateContextMenu(ContextMenu menu, View v,
                  ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    menu.add(0, DELETE_ID, 0, R.string.menu_delete);
  }
  @Override
  public boolean onContextItemSelected(MenuItem item) {
    switch(item.getItemId()) {
      case DELETE_ID:
        AdapterContextMenuInfo info = (AdapterContextMenuInfo)
item.getMenuInfo();
```

```
            mDbHelper.deleteNote(info.id);
            fillData();
            return true;
        }
        return super.onContextItemSelected(item);
    }
    /*Aquí se indica que si se va a crear una nota nos envié a la Activity de
NoteEdit para editarla*/
    private void createNote() {
        Intent i = new Intent(this, NoteEdit.class);
        startActivityForResult(i, ACTIVITY_CREATE);
        finish();
    }
    /*Aquí se indica que si se presiona sobre una nota de la lista se debe iniciar
una nueva Activity Llamada "NoteView" y que se deben de obtener los datos
de la base de datos para que aparezcan en esta activity*/
    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        super.onListItemClick(l, v, position, id);
        Intent i = new Intent(this, NoteView.class);
        i.putExtra(NotesDbAdapter.KEY_ROWID, id);
        startActivityForResult(i, ACTIVITY_EDIT);
        finish();
    }
    @Override
```

```
/*Aquí nos indica que se busque el archivo seleccionado y después lo muestre en la siguiente Activity correspondiente*/
   protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
      super.onActivityResult(requestCode, resultCode, intent);
      fillData();
      finish();
   }
   @Override
   public void finish()
   {
      super.finish();
   }
}
```

ANEXO 14

ACTIVITY DE NOTAS

Visualizar Notas.

noteview.xls. Este archivo es de tipo Relative Layout y será de visualización de notas así como un botón por si se desea editar la información contenida en la nota. Y el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@drawable/wallnota"
   android:weightSum="1">
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="Nota"
      android:textColor="@color/Aazul"
      android:textStyle="bold|normal|italic"
      android:id="@+id/titulo_almacen"
      android:textSize="25dp"
      android:textIsSelectable="false"
```

```
    android:hapticFeedbackEnabled="false"
    android:gravity="center"
    android:layout_marginTop="40dp"/>
<ImageButton
    android:id="@+id/imageButton155"
    android:background="@drawable/flecha"
    android:layout_gravity="bottom|right"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginRight="15dp"
    android:layout_marginBottom="15dp"
    android:layout_width="50dp"
    android:layout_height="50dp" />
<Button
    style="@style/botonazul"
    android:text="Editar"
    android:id="@+id/button160"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginLeft="15dp"
```

```
    android:layout_marginBottom="20dp" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView200"
    android:layout_below="@+id/titulo_almacen"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="52dp"
    android:textColor="@color/Aazul"
    android:textSize="20dp"
    android:textIsSelectable="false" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView201"
    android:layout_below="@+id/textView200"
    android:layout_alignLeft="@+id/button160"
    android:layout_alignStart="@+id/button160"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="40dp"
```

```
    android:textColor="@color/Aazul"
    android:textSize="20dp"
    android:layout_marginRight="50dp" />
</RelativeLayout>
```

NoteView.java. Este archive contiene la lógica de programación de la actividad para visualizar las notas guardadas en la base de datos. El código es el siguiente:

```
package ermich.prueba;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.TextView;
public class NoteView extends Activity {
   ImageButton imageButton;
   private Long mRowId;
   private NotesDbAdapter mDbHelper;
   private TextView Titulo;
   private TextView Nota;
   @Override
```

*/*Aquí se indica que al crear la activity nos muestre visualmente lo que existe en el Layout llamado noteview, hacemos referencia a la herramienta mDbHelper para la base de datos, también se agrega la instrucción para darle función a un botón y se declara las animaciones de entrada y salida, así como dos TextView que mostraran el contenido de las bases de datos en Titulo y en Nota*/*

```
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      requestWindowFeature(Window.FEATURE_NO_TITLE);

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
      setContentView(R.layout.noteview);
      overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
      mDbHelper = new NotesDbAdapter(this);
      mDbHelper.open();
      addListenerOnButton();
      populateFields();
      Titulo = (TextView) findViewById(R.id.textView200);
      Nota = (TextView) findViewById(R.id.textView201);
/*Aquí se declara que se usara con flecha con ID para indicar donde se
colocara el contenido a mostrar que estará guardado en la base de datos*/
      mRowId = (savedInstanceState == null) ? null :
            (Long)
savedInstanceState.getSerializable(NotesDbAdapter.KEY_ROWID);
      if (mRowId == null) {
```

```
        Bundle extras = getIntent().getExtras();
        mRowId = extras != null ?
extras.getLong(NotesDbAdapter.KEY_ROWID)
                : null;
      } }
   /*Aquí se declara el funcionamiento del botón imagen de fleche para
regresar a la Activity anterior*/
   private void addListenerOnButton() {
      imageButton = (ImageButton) findViewById(R.id.imageButton155);
      imageButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            Intent i = new Intent(NoteView.this,
                  NotePad.class);
            startActivity(i);
            finish();
         }  });
/*Aquí se de funcionalidad al botón que servirá para editar la nota en caso de
no estar correcta la información antes guardada*/
      Button boton2 = (Button) findViewById(R.id.button160);
      boton2.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Intent intent = new Intent(NoteView.this,
                  NoteEdit.class);
```

```
            intent.putExtra(NotesDbAdapter.KEY_ROWID, mRowId);
            startActivity(intent);
            finish();
        }
    });
}
/*Aquí se indica que si hay contenido guardado en las columnas de la base de datos los mande a una parte que será el título de la nota y el otro a el cuerpo de la nota.*/
private void populateFields() {
    if (mRowId != null) {
        Cursor note = mDbHelper.fetchNote(mRowId);
        startManagingCursor(note);
        Titulo.setText(note.getString(
                note.getColumnIndexOrThrow(NotesDbAdapter.KEY_TITLE)));
        Nota.setText(note.getString(
                note.getColumnIndexOrThrow(NotesDbAdapter.KEY_BODY)));
    }
}
/*Aquí se declara que si se para la aplicación se guarden los datos de la base de datos*/
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    saveState();
```

```
        outState.putSerializable(NotesDbAdapter.KEY_ROWID, mRowId);
    }
    @Override
    protected void onPause() {
        super.onPause();
        saveState();
    }
    @Override
    protected void onResume() {
        super.onResume();
        populateFields();
    }
    private void saveState() {
    }
}
```

ANEXO 15

ACTIVITY DE NOTAS

La activity de edición de notas.

note_edit.xml. Este archivo será de tipo Relative Layout. Y el código es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@drawable/wallpaper"
   >
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="Edita Tu Nota"
      android:textColor="@color/Aazul"
      android:textStyle="bold|normal|italic"
      android:id="@+id/titulo_envios"
      android:textSize="25dp"
      android:textIsSelectable="false"
```

```
    android:hapticFeedbackEnabled="false"
    android:gravity="center"
    android:layout_marginTop="40dp"
    android:layout_marginBottom="20dp" />
<TextView android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/body"
    android:id="@+id/textView200"
    android:textColor="@color/Aazul"
    android:textSize="20dp"
    android:layout_below="@+id/title"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="10dp" />
<EditText android:id="@+id/title"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:scrollbars="vertical"
    android:layout_toEndOf="@+id/textView200"
    android:layout_below="@+id/titulo_envios"
```

```
    android:layout_alignLeft="@+id/body"
    android:layout_alignStart="@+id/body"
    android:layout_alignRight="@+id/body"
    android:layout_alignEnd="@+id/body" />
<Button
    style="@style/botonazul"
    android:id="@+id/confirm"
    android:text="@string/confirm"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/body"
    android:layout_alignRight="@+id/body"
    android:layout_alignEnd="@+id/body"
    android:layout_marginTop="20dp" />
<EditText android:id="@+id/body"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:layout_toEndOf="@+id/textView200"
    android:layout_alignTop="@+id/textView200"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
```

```
    android:layout_marginRight="20dp"
    android:layout_marginLeft="70dp" />
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/title"
    android:layout_toStartOf="@+id/title"
    android:textColor="@color/Aazul"
    android:textSize="20dp"
    android:textIsSelectable="false"
    android:layout_below="@+id/titulo_envios"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:id="@+id/textView"
    android:layout_marginLeft="10dp"
    android:layout_toLeftOf="@+id/title" />
</RelativeLayout>
```

NoteEdit**.java. Este archivo tendrá la lógica de programación de la edición de notas donde se podrá generar nuevo contenido y guardarlo en la base de datos ya sea creando la nota o guardando la nueva información si ya existía la nota en la base de datos. El código es el siguiente:**

```
package ermich.prueba;
import android.app.Activity;
```

```
import android.app.ListActivity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
public class NoteEdit extends Activity {
   /*Aquí se declaran los elementos que se usaran en esta Activity como son
los EditText para agregar los textos de nuestra nota, la herramienta
mDbAdapter y mRowId para el uso de las bases de datos y unos TextView
para indicarnos si lo que ingresamos será el titulo o el cuerpo de la nota.*/
   private EditText mTitleText;
   private EditText mBodyText;
   private Long mRowId;
   private NotesDbAdapter mDbHelper;
   ImageButton imageButton;
   private TextView Titulo;
   private TextView Nota;
   @Override
/*Aquí se indica que al crear la activity nos muestre visualmente lo que existe
```

en el Layout llamado note_edit, hacemos referencia a la herramienta mDbHelper para la base de datos, también se agrega la instrucción para darle función a un botón y se declara las animaciones de entrada y salida, así como dos EditText que servirán para ingresar los datos que queremos guardar y dos TextView que mostraran el contenido de las bases de datos en Titulo y en Nota/*

```
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      //Aquí se indica que no se vea la barra de notificaciones.

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
      // Aquí se oculta la barra de notificaciones.
      requestWindowFeature(Window.FEATURE_NO_TITLE);
/*Aquí se indica el Layout de la Activity a la que se la está dando la lógica de programación.*/
      mDbHelper = new NotesDbAdapter(this);
      mDbHelper.open();
      overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
      setContentView(R.layout.note_edit);
      mTitleText = (EditText) findViewById(R.id.title);
      mBodyText = (EditText) findViewById(R.id.body);
      Titulo = (TextView) findViewById(R.id.textView200);
      Nota = (TextView) findViewById(R.id.textView201);
      Button confirmButton = (Button) findViewById(R.id.confirm);
/*Aquí se declara que se usara con flecha con ID para indicar donde se
```

```
colocara el contenido a mostrar que estará guardado en la base de datos*/
      mRowId = (savedInstanceState == null) ? null :
            (Long)
savedInstanceState.getSerializable(NotesDbAdapter.KEY_ROWID);
      if (mRowId == null) {
         Bundle extras = getIntent().getExtras();
         mRowId = extras != null ?
extras.getLong(NotesDbAdapter.KEY_ROWID)
               : null;
      }
      populateFields();
/*aquí se indica que el botón de guardar al ser presionado mandara a la
página principal el menú de notas*/
      confirmButton.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            setResult(RESULT_OK);
            Intent i = new Intent(NoteEdit.this, NotePad.class);
            startActivity(i);
            finish();
         }
      });
   }
   /*Aquí se indica que lo que se capture en los EditText sea guardado en la
base de datos para su posterior consulta con el nombre de KEY_TITLE y
KEY_BODY para guardar el título y cuerpo o contenido de la nota
```

```
respectivamente*/
   private void populateFields() {
      if (mRowId != null) {
         Cursor note = mDbHelper.fetchNote(mRowId);
         startManagingCursor(note);
         mTitleText.setText(note.getString(
               note.getColumnIndexOrThrow(NotesDbAdapter.KEY_TITLE)));
         mBodyText.setText(note.getString(
               note.getColumnIndexOrThrow(NotesDbAdapter.KEY_BODY)));
      }
   }
   /*Aquí se indica que si se sale de la aplicación se guarden los datos de la
base de datos para su consulta posterior*/
   @Override
   protected void onSaveInstanceState(Bundle outState) {
      super.onSaveInstanceState(outState);
      saveState();
      outState.putSerializable(NotesDbAdapter.KEY_ROWID, mRowId);
   }
   /*Aquí se indica que guarde el estado de la aplicación si se pausa*/
   @Override
   protected void onPause() {
      super.onPause();
      saveState();
   }
```

```
    @Override
/*Aquí se indica que si se regresa después de ponerse en pause la aplicación
se resuma en lo que se había quedado*/
    protected void onResume() {
        super.onResume();
        populateFields();
    }
    /*Aquí se indica específicamente que los valores ingresados en los EditText
sean guardados en como unos nuevos datos en una columna de la base datos
asignándoles un ID y si no hay datos entonces la nota queda igual y no se
edita nada*/
    private void saveState() {
        String title = mTitleText.getText().toString();
        String body = mBodyText.getText().toString();
        if (mRowId == null) {
            long id = mDbHelper.createNote(title, body);
            if (id > 0) {
                mRowId = id;
            }
        } else {
            mDbHelper.updateNote(mRowId, title, body);
        }
    } }
```

ANEXO 16

NOTESDBADAPTER

Archivo necesario para poder crear y gestionar las bases de datos de las notas.

```
package ermich.prueba;
//Aquí se importan las librerias que se usaran.
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class NotesDbAdapter {
//Aquí se indian componentes la aplicación.
   public static final String KEY_TITLE = "title";
   public static final String KEY_BODY = "body";
   public static final String KEY_ROWID = "_id";
   private static final String TAG = "NotesDbAdapter";
   private DatabaseHelper mDbHelper;
   private SQLiteDatabase mDb;

   //Aquí se crean los datos asignandos que llevara la base de tados.
   private static final String DATABASE_CREATE =
```

```
        "create table notes (_id integer primary key autoincrement, "
            + "title text not null, body text not null);";

    private static final String DATABASE_NAME = "data";
    private static final String DATABASE_TABLE = "notes";
    private static final int DATABASE_VERSION = 2;
    private final Context mCtx;
    private static class DatabaseHelper extends SQLiteOpenHelper {

        DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

//Aquí se crea la base de datos.
        @Override
        public void onCreate(SQLiteDatabase db) {

            db.execSQL(DATABASE_CREATE);

        }
//Aquí se validan los datos ya guardados y si hay cambios los actualiza.
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
            Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
```

```
                + newVersion + ", which will destroy all old data");
            db.execSQL("DROP TABLE IF EXISTS notes");
            onCreate(db);
        }
    }
  //Aquí se abre la base de datos y se validan los daos que existen en ella.
    public NotesDbAdapter(Context ctx) {
        this.mCtx = ctx;
    }

    public NotesDbAdapter open() throws SQLException {
        mDbHelper = new DatabaseHelper(mCtx);
        mDb = mDbHelper.getWritableDatabase();
        return this;
    }

    public void close() {
        mDbHelper.close();
    }
 //Aquí se indica donde guardara los datos si se crea una nota nueva
    public long createNote(String title, String body) {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_TITLE, title);
        initialValues.put(KEY_BODY, body);
```

```
        return mDb.insert(DATABASE_TABLE, null, initialValues);
    }
    //Aquí se indica cuando se quiere borrar un datos dentro de la base de
datos.
    public boolean deleteNote(long rowId) {

        return mDb.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) >
0;
    }
    //Aquí se indica el cursor que sera el encargado de buscar los datos en la
base de datos.
    public Cursor fetchAllNotes() {

        return mDb.query(DATABASE_TABLE, new String[] {KEY_ROWID,
KEY_TITLE,
                KEY_BODY}, null, null, null, null, null);
    }

    public Cursor fetchNote(long rowId) throws SQLException {

        Cursor mCursor =

                mDb.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
                        KEY_TITLE, KEY_BODY}, KEY_ROWID + "=" + rowId, null,
                    null, null, null, null);
```

```
        if (mCursor != null) {
            mCursor.moveToFirst();
        }
        return mCursor;

    }
//Si se actualizan datos de una nota ya creada aquí se indicara donde
guardarlos para que solo se actualice la información.
    public boolean updateNote(long rowId, String title, String body) {
        ContentValues args = new ContentValues();
        args.put(KEY_TITLE, title);
        args.put(KEY_BODY, body);

        return mDb.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId,
null) > 0;
    }

}
```

ANEXO 17

NOTES_ROW

Archivo necesario para hacer funcionar las notas que que funciona como archive para agregar el nombre de la nota en la lista de notas guardadads en la base de datos, el nombre es notes_row.xls.

```
<?xml version="1.0" encoding="utf-8"?>


<TextView android:id="@+id/text1"
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
 android:layout_height="match_parent"/>
```

ANEXO 18

ANDROIDMANIFEST

Archivo que gestiona todas las partes de la aplicación, aquí se tienen que declarar todas las actividades, el nombre es AndroidManifest.xls.

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

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="ERMICH"
      android:theme="@style/AppTheme" >

      <activity
         android:name=".Splash"
         android:label="@string/app_name">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

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

```
<activity
   android:name=".Extras">
</activity>
<activity
   android:name=".Almacen">
</activity>
<activity
   android:name=".MenuActivity">
</activity>

<activity
   android:name=".WelcomeActivity">
</activity>
<activity
   android:name=".MainActivity">
</activity>
<activity
   android:name=".NotePad">
</activity>
<activity
   android:name=".NoteEdit">
</activity>

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

```
        </activity>
        <activity
            android:name=".NoteView">
        </activity>
    </application>

</manifest>
```