UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

Trabajo de Grado Previo a la Obtención del Título de Ingeniera en Sistemas Computacionales

TEMA

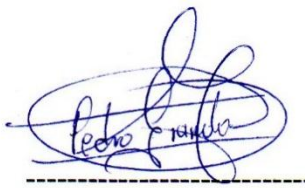
ESTUDIO Y DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES ANDROID

Autor: Mónica Lucía Tapia Marroquín.
Director: Ing. Pedro Granda.

Ibarra – Ecuador
Noviembre 2013

CERTIFICACIÓN

Certifico que la Tesis **"ESTUDIO Y DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES ANDROID"** ha sido realizada en su totalidad por la señorita: Tapia Marroquín Mónica Lucía portadora de la cédula de identidad número: 100268660-6.



Ing. Pedro Granda

DIRECTOR DE TESIS

CERTIFICACIÓN

Ciudad, 25 de octubre de 2013

Señores
UNIVERSIDAD TÉCNICA DEL NORTE
Presente

De mis consideraciones.-

Siendo auspiciantes del proyecto de tesis de la Egresada TAPIA MARROQUÍN MÓNICA LUCÍA con CI: 100268660-6 quien desarrolló su trabajo con el tema "ESTUDIO Y DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES ANDROID", me es grato informar que se han superado con satisfacción las pruebas técnicas y la revisión de cumplimiento de los requerimientos funcionales, por lo que se recibe el proyecto como culminado y realizado por parte de la egresada TAPIA MARROQUÍN MÓNICA LUCÍA. Una vez que hemos recibido la capacitación y documentación respectiva, nos comprometemos a continuar utilizando el mencionado aplicativo en beneficio de nuestra empresa/institución.

La egresada TAPIA MARROQUÍN MÓNICA LUCÍA puede hacer uso de este documento para los fines pertinentes en la Universidad Técnica del Norte.

Atentamente,

Sr. Gualberto Quiñonez

Presidente.

COOPERATIVA DE TAXIS "28 DE ABRIL" TELF. 2958317 IBARRA-ECUADOR

COOPERATIVA "28 DE ABRIL"

UNIVERSIDAD TÉCNICA DEL NORTE

CESION DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Mónica Lucía Tapia Marroquín, con cédula de identidad Nro. 1002686606, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de Propiedad Intelectual del Ecuador, artículos 4, 5 y 6, en calidad de autor del trabajo de grado denominado: **"ESTUDIO Y DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES ANDROID"** , que ha sido desarrollado para optar por el título de: Ingeniera en Sistemas Computacionales en la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En mi condición de autora me reservo los derechos morales de la obra antes citada. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

.....

Nombre: Mónica Lucía Tapia Marroquín

Cédula: 1002686606

Ibarra, a los 25 días del mes de Octubre del 2013

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual ponemos a disposición la siguiente información.

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD:	1002686606
APELLIDOS Y NOMBRES:	TAPIA MARROQUÍN MÓNICA LUCÍA
DIRECCIÓN:	GARCÍA MORENO 10-09 Y PEDRO RODRÍGUEZ
EMAIL:	mony_t0204@hotmail.com
TELÉFONO FIJO:	062642460
TELÉFONO MOVIL:	0995559427

DATOS DE LA OBRA	
TITULO:	ESTUDIO Y DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES ANDROID.
AUTOR:	TAPIA MARROQUÍN MÓNICA LUCÍA
FECHA:	2013-11-25
PROGRAMA:	PREGRADO
TITULO POR EL QUE OPTA:	INGENIERIA EN SISTEMAS COMPUTACIONALES
DIRECTOR:	ING. PEDRO GRANDA

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Mónica Lucía Tapia Marroquín, con cedula de identidad Nro. 1002686606, en calidad de autora y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la

Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Articulo 144.

3. CONSTACIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 25 días del mes de octubre de 2013

EL AUTOR:

.....

Mónica Lucía Tapia Marroquín

CI: 1002686606

DEDICATORIA

Quiero dedicar este trabajo en primer lugar a Dios, por ser quien guía mi vida, por darme la fortaleza y así lograr alcanzar esta meta.

A mis padres Justo Tapia y Josefa Marroquín, quienes han sido el pilar fundamental de mi vida, los admiro y los amo con todo mi corazón, gracias por su apoyo, por su amor, por su esfuerzo y por sus ánimos.

A mi hijo Bryan Alexander, a quien quiero mucho y es la mejor bendición que Dios me ha dado.

A una persona especial quien me ha brindado siempre su apoyo incondicional y me ha motivado en todo momento para culminar mis estudios y cumplir esta meta.

Mónica Tapia

AGRADECIMIENTO

A Dios por iluminar mi camino, darme fuerzas para vencer los obstáculos y culminar mi carrera.

A mis padres dignos de ejemplo, trabajo y constancia quienes me han brindado su amor incondicional, han estado en todos los momentos de mi vida apoyándome y alentándome a seguir adelante.

A mi hijo a quien siempre me ha dado ánimo para salir adelante.

A una persona especial cuyo respaldo y ánimos han sido decisivos en momentos de angustia y desesperación.

A mi director de tesis Ingeniero Pedro Granda, quien durante todo este tiempo me colaboró en el desarrollo de este trabajo.

A mis profesores que han sido participes en mi formación académica dentro de esta prestigiosa Institución.

Agradecer a todas aquellas personas que en mayor o menor medida han ayudado a que este trabajo se desarrolle.

Mónica Tapia

Índice de Contenido

Índice de Figuras

Índice de Tablas

RESUMEN

El presente documento es para los desarrolladores de software con conocimientos previos en lenguaje JAVA interesados en crear aplicaciones móviles con Android, brinda una visión general de una aplicación Android, aquí se describen algunos de los controles disponibles en el toolbox del IDE utilizado, se detalla los conceptos básicos para el aprendizaje del sistema operativo Android. Y finalmente se realiza un pequeño aplicativo para demostrar sus funcionalidades básicas.

El trabajo se ha estructurado de la siguiente manera: una investigación, una introducción a su interfaz y una implementación, el documento presenta seis capítulos.

En el capítulo uno, se hace una comparativa entre las tres tecnologías principales que lideran el mercado móvil de los teléfonos inteligentes.

En el capítulo dos se detallan conceptos técnicos de la plataforma Android, los cuales son necesarios conocer antes de una aplicación.

En el capítulo tres, se detallan los conceptos principales de Android, se describen las herramientas a utilizar para el desarrollo de un proyecto y se hace una breve introducción a la interfaz de programación.

En el capítulo cuatro se desarrolla los módulos descritos en el proyecto Taxímetro.

En el capítulo cinco se desarrolla el proyecto a través de la metodología de software Scrum.

Por último en el capítulo seis se detallan las conclusiones y recomendaciones a las que se ha llegado al realizar la investigación y el proyecto.

SUMMARY

This document is for software developers with some knowledge on JAVA interested in creating Android mobile application, provides an overview of an Android application, here are some of the controls available in the toolbox of the IDE used, detailing the learning basics of the Android OS. And finally made a small application to demonstrate its basic functionalities.

The paper is structured as follows: an investigation, an introduction to its interface and an implementation, the paper presents six chapters.

In chapter one, we make a comparison between the three main technologies that lead the mobile market smartphones.

Chapter two detailed technical concepts of the Android platform, which are necessary to know before an application.

In chapter three, details the main concepts of Android, describes the tools used for the development of the project and a brief introduction to the programming interface.

In chapter four develops the modules described in the project taximeter. In chapter five the project is developed through the Scrum software methodology.

Finally in Chapter Six details the conclusions and recommendations that have been reached in conducting research and project.

CAPÍTULO I

SISTEMAS OPERATIVOS PARA SMARTPHONE

CONTENIDO:

1. Sistemas Operativos para dispositivos móviles.
2. Android
3. IOS
4. Windows Phone

CAPÍTULO 1: SISTEMAS OPERATIVOS PARA DISPOSITIVOS MÓVILES

1.1 SISTEMAS OPERATIVOS PARA DISPOSITIVOS MÓVILES

Un sistema operativo es un programa o conjunto de programas que en un sistema informático gestiona los recursos de hardware y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los demás. Entre algunas de sus características principales tenemos:

- ✓ Gobernar y proveer de un ambiente conveniente de trabajo
- ✓ Hacer uso eficiente del hardware
- ✓ Proveer de una adecuada distribución y asignación de los recursos
- ✓ Administrar y controlar la ejecución de programas

Un sistema operativo móvil es un sistema operativo que controla un dispositivo móvil al igual que las computadoras utilizan Windows, Linux o Mac OS, los sistemas operativos móviles son más sencillos y se orientan más a la conectividad inalámbrica, los formatos multimedia y las diferentes maneras de introducir información en ellos.

El mundo de los móviles va hacia la personalización radical**:** sus funciones se han elevado considerablemente, sirven muchas cosas y existe un sin número de programas que realizan infinidad de tareas como: jugar, hacer tareas de oficina, aprovechar al

máximo la geolocalización, entre otros. También está la Nube, que hace que muchos servicios estén permanentemente online.

Es fundamental que un sistema operativo disponga de una gran variedad de aplicaciones de buena calidad. Aunque no todas sean útiles, lo que cuenta es que cada uno tenga la posibilidad de personalizar su móvil hasta el mínimo detalle.

De un tiempo a acá, el boom de la telefonía móvil ha explotado, impulsado por la aparición de sistemas operativos cada vez más complejos hasta el punto de asemejarse mucho a los NetBooks actuales.

1.2 ANDROID

Hace algunos años, Google decidió que debía expandir su negocio hacia los móviles, así que su mejor estrategia fue crear un sistema operativo móvil propio, gratis y con varios de los más grandes fabricantes de celulares como respaldo, de esta manera nace Android. [**WEB1**]

Android es un sistema operativo móvil basado en el kernel de Linux, con una interfaz de programación Java, diseñado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas, Google TV y otros. Desarrollado por la Open Handset Alliance la cual es liderada por Google. [**WEB2**]

Android permite programar aplicaciones en una variación de Java llamada Dalvik. El sistema operativo proporciona todas las interfaces necesarias para desarrollar

aplicaciones que accedan a las funciones del teléfono (GPS, llamadas, sms, agenda, entre otras.) de una forma muy fácil en un lenguaje de programación muy popular como es Java.

Figura 1.1: SO Android en Teléfonos Móviles
Fuente: [WEB 3]

1.2.1 Breve Historia

En Julio de 2005, Google adquirió Android Inc., en ese entonces la compañía se dedicaba a la creación de software para teléfonos móviles. Una vez dentro de Google, el equipo desarrolló una plataforma basada en el núcleo Linux para dispositivos móviles, que fue promocionado a fabricantes de dispositivos y operadoras con la promesa de proveer un sistema flexible y actualizable, Google adaptó su buscador y sus aplicaciones para el uso en móviles.

Es el principal producto de la Open Handset Alliance, una alianza comercial de un conglomerado de compañías entre fabricantes y desarrolladores de hardware, software y operadores de servicio, dedicadas al desarrollo de estándares abiertos para dispositivos móviles, algunos de sus miembros son Google, HTC, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia y Wind River Systems. Google liberó la mayoría del código de Android bajo licencia Apache, una licencia libre y de código abierto.

1.2.2 Versiones

Cada actualización del sistema operativo Android es desarrollada bajo un nombre en código de un elemento relacionado con postres, los nombres en código están en orden alfabético:

Versión	Descripción
1.0	Liberado el 23 de septiembre de 2008
1.1	Liberado el 9 de febrero de 2009
1.5 (Cupcake) 	Liberado el 15 de septiembre de 2009. Basado en el kernel de Linux 2.6.27 Con una interfaz sencilla y atractiva, GPS, capacidad de grabar y reproducir vídeos, entre otras.
1.6 (Donut) 	Liberado el 15 de septiembre de 2009. Basado en el kernel de Linux 2.6.29 Esta actualización se incluyó novedades como la 'Quick Search Box', control de batería, conexión a VPN, entre otras.
2.0 / 2.1 **(Eclair)** 	Liberado el 26 de octubre de 2009. Basado en el kernel de Linux 2.6.29 En actualización se incluyó un rediseño de la interfaz de usuario, soporte para HTML5, Bluetooth 2.1, soporte para Facebook, entre otras. El SDK 2.0.1 fue liberado el 3 de diciembre de 2009. El SDK 2.1 fue liberado el 12 de enero de 2010.
2.2 (Froyo) 	Liberado el 20 de mayo de 2010. Basado en el kernel de Linux 2.6.32 En esta actualización se incluyó: una optimización general del sistema Android, que mejoraba su rendimiento y memoria, soporte para Adobe Flash entre otras.

2.3 (Gingerbread) 	Liberado el 6 de diciembre de 2010. Basado en el kernel de Linux 2.6.35.7 En esta actualización se incluyó: nuevos efectos, soporte para NFC, mejora en la entrada de datos, audio y gráficos para juegos, etc.
3.0 / 3.1 / 3.2 (Honeycomb) 	Liberado el 22 de febrero del 2011. En esta actualización se incluyó: soporte para tablets, escritorio 3D con widgets rediseñados, Google Talk, entre otras.
4.0 (Ice Cream Sandwich) 	En esta actualización se incluyó: versión que unifica el uso en cualquier dispositivo, tanto en teléfonos, tablets, televisores, netbooks, etc.
4.1 **(Jelly Bean)** 	En esta actualización se incluyó: mejora de la fluidez y de la estabilidad gracias al proyecto "Project Butter", ajuste automático de widgets cuando se añaden al escritorio, cambiando su tamaño y lugar para permitir que los nuevos elementos se puedan colocar nuevas lenguas no occidentales, fin al soporte de Flash Player para Android a partir de esta versión.
4.2 Jelly Bean_mr1	Liberado el 9 de octubre del 2012. En esta actualización se incluye: soporte de rotación de la pantalla principal, arreglo de fallos y mejoras en rendimiento, notificaciones expansión/contracción con un dedo.

Tabla 1.1: Versiones SDK Android

Fuente: [WEB 2]

1.2.3 Características [WEB 4]

- ✓ Framework de aplicaciones, que permite el remplazo y la reutilización de los componentes.

- ✓ Sistema de notificaciones, esta característica es algo en lo que Android sobresale del resto de sistemas operativos móviles.
- ✓ Navegador web integrado, basado en el motor Webkit.
- ✓ Sqlite, para almacenamiento de datos.
- ✓ Lenguaje de programación Java
- ✓ Soporta diversos formatos multimedia (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- ✓ Soporta HTML, HTML5, Adobe Flash Player, entre otros.
- ✓ Incluye un emulador de dispositivos, herramientas para depuración de memoria y plugin para Eclipse.
- ✓ Máquina virtual Dalvik, la cual está optimizada para dispositivos móviles, muy similar a Java.
- ✓ Telefonía GSM.
- ✓ Bluetooth, 3g y Wifi.
- ✓ GPS, Cámara, acelerómetro y brújula.
- ✓ Tienda de aplicaciones gratuitas o pagadas llamada Google Play.
- ✓ Búsqueda por voz versión de Siri

1.3 IOS

iOS, anteriormente denominado iPhone OS es un sistema operativo móvil de Apple. Originalmente desarrollado para el iPhone, para luego ser usado en dispositivos como el iPod Touch, iPad y el Apple TV. Apple, Inc. no permite la instalación de iOS en hardware de terceros. Tenía el 26% de cuota de mercado de sistemas operativos

móviles vendidos en el último cuatrimestre de 2010, detrás de Google Android y Nokia Symbian. En mayo de 2010 en los Estados Unidos, tenía el 59% de consumo de datos móviles (incluyendo el iPod Touch y el iPad). [**WEB 5**]

La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles. Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan al sacudir el dispositivo (por ejemplo, para el comando deshacer) o rotarlo en tres dimensiones (un resultado común es cambiar de modo vertical a horizontal). [**WEB 5**]

iOS se deriva de Mac OS X, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo Unix. iOS cuenta con cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de "Servicios Principales", la capa de "Medios" y la capa de "Cocoa Touch". [**WEB 5**]

1.3.1 Historia

Apple reveló la existencia de iPhone OS en la Macworld Conference & Expo en enero de 2007, aunque el sistema no tuvo un nombre oficial hasta que salió la primera versión beta del iPhone SDK un año más tarde, en marzo de 2008. Antes de esto se consideraba simplemente que el iPhone ejecutaba OS X. A partir de entonces se llamaría iPhone OS. El lanzamiento del iPhone OS tuvo lugar el 29 de junio de 2007. [**WEB 5**]

El interés en el SDK aumentaría en meses siguientes debido al explosivo crecimiento de la plataforma iPhone, que se vio incrementado en septiembre de

2007 del iPod Touch, un dispositivo con las capacidades multimedia del iPhone pero sin la capacidad de hacer llamadas telefónicas. [**WEB 5**]

El 27 de enero de 2010 Steve Jobs, CEO de Apple, anunció el iPad, un dispositivo muy similar al iPod Touch pero con un enfoque más orientado hacia la industria de contenidos. Este dispositivo, apoyado en una pantalla táctil de mayor dimensión, compartiría sistema operativo con sus dos exitosos hermanos, y vendría acompañado de una aplicación oficial para la compra y lectura de libros electrónicos, iBooks. [**WEB 5**]

En de abril de 2010 se estima 185.000 las aplicaciones disponibles para iPhone OS a través de la App Store. El 7 de junio de 2010, durante la presentación del iPhone 4, Steve Jobs anunció que iPhone OS pasaría a ser llamado oficialmente como iOS. El 12 de septiembre de 2012 se presenta iOS 6 durante la presentación del iPhone 5. Tim Cook hace presentación del iPhone 5, iTunes 11 e iOS 6. [**WEB 5**]

1.3.2 Versiones

Versión	Descripción
iOS 1 	Fue lanzado el 29 de junio de 2007, es la primera versión y se distribuye por medio de la página de *itunes*, ofrece actualizaciones y corrección de errores.
iOS 2 	Fue lanzado el 11 de junio de 2008, esta fue incluida de fábrica en el iphone 3G, se incluyó una aplicación que permitía adquirir programas de compañías ajenas a Apple.
iOS 3 	Fue lanzado el 17 de junio de 2009, la función principal que se incluyó en esta actualización fue la de "copiar y pegar". Cabe mencionar que únicamente los equipos que poseían la versión 2, se podían actualizar a este paquete.

iOS 4 	Fue lanzado el 21 de junio de 2010, este software se desempeña de mejor manera en la plataforma del iphone 4, porque en el iphone 3GS, las funciones no se ejecutan a su máxima capacidad.
iOS 5 	Fue lanzada el 6 de junio de 2011, es la segunda versión iOS que no permite que los dispositivos antiguos se actualicen a esta versión, fue lanzado para iPhone 3GS, iPhone 4, iPhone 4S, iPod touch 3G, iPod touch 4G, iPad e iPad.
iOS 6 	El 12 de septiembre de 2012 se presentó iOS 6 durante la presentación del iPhone 5, iTunes 11 e iOS 6.

Tabla 1.2: Versiones iOS

Fuente: [**WEB 6**]

1.3.3 Características

- ✓ Interfaz gráfica está diseñada para el touch screen, con capacidad para gestos, la interfaz se encuentra constituida fundamentalmente de sliders, interruptores y botones, con una respuesta inmediata.
- ✓ Notificaciones, cuando hay una notificación, no se interrumpe.
- ✓ Soporta acelerómetros.
- ✓ Emplea unos 500 MB de almacenamiento, depende del modelo.
- ✓ Diversas aplicaciones para gestionar correo, fotos, cámara, mensajes, clima, notas, contactos, reloj, entre otros.
- ✓ Soporta multitarea, aunque con algunas limitaciones.
- ✓ No dispone de soporte para: Adobe Flash y Java, por este motivo los sitios web con dichas tecnologías no pueden ser visualizados en este sistema operativo.

1.4 WINDOWS PHONE

Windows Phone anteriormente conocido como Windows Mobile, es un sistema operativo móvil desarrollado por Microsoft, diseñado para su uso en teléfonos inteligentes. Windows Phone hace parte de los sistemas operativos con interfaz natural de usuario, se basa en el núcleo del sistema operativo Windows CE y posee un conjunto de aplicaciones básicas que utilizan las APIs de Microsoft Windows.

Microsoft tomó la decisión de no hacer compatible Windows Phone con Windows Mobile, por este motivo las aplicaciones existentes no funcionan en Windows Phone, por lo que es necesario desarrollar nuevas aplicaciones, Windows Phone ofrece una nueva interfaz de usuario e integra varios servicios en el sistema operativo.

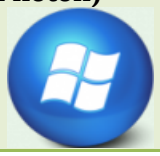
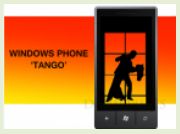
1.4.1 Breve Historia

Windows Phone es el sucesor de la versión del sistema operativo móvil Windows Mobile desarrollado por Microsoft quien comenzó a trabajar desde el 2008 en lo que debía ser un replanteamiento de su sistema operativo, ya que en el año anterior a este había cambiado la industria de la telefonía celular cuando Nokia introdujo su modelo N95 y nuevos competidores como Apple y Android se interesaban en competir con sus respectivos sistemas operativos móviles.

Después de un intenso trabajo, Microsoft presentó a Windows Phone 7, que se lanzó el 1 de septiembre de 2010, y la versión del SDK estuvo disponible a mediados de septiembre de 2010, fue lanzado en Europa y Asia en octubre del

mismo año y en EEUU en noviembre. Inicialmente, Windows Phone 7 estaba destinado para lanzarse durante el 2009, pero luego de varios aplazamientos Microsoft tuvo que desarrollar Windows Mobile 6.5 como una versión de pasajera.

1.4.2 Versiones

Versión	Descripción
Windows Phone 7 (Photon) 	Incluye funciones de integración con los servicios Xbox Live y Zune. La interfaz, conocida como "Metro", ha sido revisada en su totalidad y comparte características visuales similares a la interfaz del dispositivo Zune HD.
Windows Phone 7.5 ('Mango') 	Es una actualización de software para Windows Phone. Este cambio se anunció el 24 de mayo de 2011, y lanzado el 27 de septiembre de 2011. Steve Ballmer mencionó que tendría más de 500 características. Luego Andy Lees anunció que Windows Phone "Mango" incluirá IE9 Mobile y Joe Belfiore dio a conocer el progreso más reciente en la incorporación de Internet Explorer 9 en el Windows Phone, incluyendo soporte para CSS3 Media Queries, y soporte para usar GPS cuando se trabaje con las aplicaciones de ubicación geográfica, entre otros.
Windows Phone 7.5 ('Refresh') 	Es una actualización de software para Windows Phone también conocida como Tango, fue uno de los requisitos de Nokia en su acuerdo con Microsoft, está enfocada a una minimización de los requisitos del sistema operativo para adaptarlo a terminales de menor coste. Anunciada en el MWC 2012 de Barcelona, trae nuevas funciones pero también limitaciones para los terminales de gama baja.

Windows Phone 7.8 	Es una actualización anunciada a principios de 2013, ofrece algunas mejoras como la nueva interfaz de usuario de WP8 y fondos personalizados para la pantalla de bloqueo y se mejoró la gestión de la batería.
Windows Phone 8 	Windows Phone 7 no puede actualizarse a Windows Phone 8.

Tabla 1.3: Versiones Windows Phone
Fuente: [**WEB 7**]

1.4.3 Características

- ✓ Explorador de Windows.
- ✓ Transferencia de archivos: Se mostrará todo en una sola ventana con información detallada y con la posibilidad de pausar la transferencia.
- ✓ Procesadores ARM: Gran capacidad de procesamiento y consumen muy poca energía.
- ✓ Aplicaciones basadas en HTML 5.
- ✓ USB 3.0
- ✓ Windows App Store.
- ✓ Interfaz gráfica. Lo que más se destaca es su interfaz gráfica llamada Metro. La pantalla de inicio, llamada "Start Screen", se compone de "Live Tiles", mosaicos dinámicos que son enlaces a aplicaciones, características, funciones y objetos individuales como contactos, páginas web o archivos multimedia, que muestran información útil y personalizada para el usuario.

- ✓ El sistema es más gráfico. Presenta mayor información gracias a Metro. Las aplicaciones y contenidos se presentan en recuadros que ofrecen un vistazo a la información sobre la aplicación. La interacción con estos cuadros se realiza con un solo toque, al estilo de los dispositivos táctiles.

- ✓ Multitarea y redes sociales.

- ✓ Almacenamiento en la nube.

- ✓ Seguridad y privacidad. Además de sincronizar esos contenidos, con SkyDrive los usuarios tienen acceso a cualquier documento o archivo que se almacene en su cuenta de Microsoft.

- ✓ Nuevo administrador de tareas.

1.5 COMPARATIVA

A continuación se presenta una comparación entre los tres grandes sistemas operativos móviles como son Android, iOS y Windows Phone en la que se analizan sus principales características:

1.5.1 Facilidad de Uso

Android se encuentra a un paso de acercarse a la usabilidad que ofrecen iOS y Windows Phone en sus dispositivo móviles. Tanto la interfaz del iPhone como de los Windows Phone son intuitivas por lo que el usuario es capaz de aprender a usar el dispositivo sin haber tenido conocimiento previo más rápido que un dispositivo Android.

1.5.2 Aplicaciones de terceros

Este es un tema de suma importancia en los smartphone. Aunque AppStore de Apple dispone de muchas más que Google Play, este último cuenta con más aplicaciones gratuitas que AppStore. Por ejemplo: Angry Birds y WhatsApp son gratuitas en Android, pero en AppStore son de paga. En el caso de Marketplace de Windows Phone, dispone de pocas aplicaciones en comparación a iOS y Android.

1.5.3 Duración de la Batería

Windows Phone tiene una excelente autonomía, en el caso de Mango dispone de un nuevo sistema de ahorro de energía, que permite que este se ponga en marcha automáticamente cuando la batería se esté agotando. En cuanto a Apple, tras varios años con iOS (desde 2007), se ha dado cuenta de lo importante que es para sus usuarios la duración de la batería, y se lo ha tomado en serio. En el caso de Android, por utilizar multitarea real hace que la batería se agote muy rápido.

1.5.4 Software del Teclado

iOS y Windows Phone se encuentran casi empatados en este punto, Android queda muy atrás, sin embargo, en Android podemos instalar cualquier teclado personalizado. Así que, en definitiva, pero al hablar de las funcionalidades que vienes de fábrica iOS y Windows Phone dominan en esta categoría

1.5.5 Reconocimiento de Voz

Casi todos los campos de texto que aparecen en el sistema Android pueden ser completado usando la voz. En Windows Phone e iOS sólo se utilizan comandos para funciones esenciales, como llamar y algo más.

1.5.6 Multitarea

Los tres sistemas tienen soporte multitarea, aunque la multitarea de iOS y Windows Phone es limitada.

1.5.7 Personalizable

Android tiene completa libertad, se diría que es casi total. En Windows Phone se puede personalizar el fondo en la pantalla de bloqueo, y cambiar el color de fondo entre claro y oscuro, y el color de énfasis entre 12 colores. En iOS se puede cambiar el fondo tanto en la pantalla de bloqueo como en la que lista las aplicaciones

1.5.8 Widgets

iOS no permite demasiada gala en su escritorio. Android si lo permite al igual que Windows Phone, a través de su mosaico llamado Live Tiles.

1.5.9 Notificaciones

En iOS y Android funcionan bien. Windows Phone no posee esta funcionalidad.

1.5.10 Soporte para Flash

Android si lo posee, mientras que iOS y Windows Phone no lo incorporan.

1.5.11 Core

El Core de Android es Linux, mientras que el de iOS es Darwin. Windows Phone trabaja sobre Windows NT.

1.5.12 Ventajas Android

Su principal ventaja para los desarrolladores de aplicaciones es que es de Open Source, se puede personalizar el teléfono al máximo y modificar funciones del teléfono sencillamente instalando una aplicación, otro punto a favor de Android es la confianza que está recibiendo de los fabricantes. Gracias a ello, la oferta de teléfonos con Android es amplia y la oferta es variada tanto en marcas como en precios.

1.5.13 Desventajas Android

Uno de los aspectos negativos de Android es su fragmentación, aunque va mejorando, actualizar el sistema operativo a nuevas versiones no es sencillo como

con un iPhone, también comparando con iPhone, la cantidad de juegos disponible para Android es menor, pero se lo está resolviendo.

1.5.14 Ventajas iOS

Buen diseño, funcionalidad, facilidad de uso y una variedad de aplicaciones y juegos enorme. Su perfecta integración con servicios en la nube y equipos de sobremesa, especialmente Mac, es otro de sus puntos fuertes, el correo, las redes sociales, se podrá estar siempre conectado.

1.5.15 Desventajas iOS

El sistema de Apple es cerrado, tiene un control más rígido de las aplicaciones publicadas, los costos por disfrutar de un iPhone son altos esto se debe a que existe un fabricante y un modelo.

1.5.16 Ventajas Windows Phone

Contiene un diseño moderno, práctico, atractivo y con características innovadoras, este sistema operativo móvil cuenta con una gran inversión y se ha diseñado para competir con los más grandes, el resultado de esto es un sistema moderno y calificado.

1.5.17 Desventajas Windows Phone

Por ser muy joven la variedad de móviles con Windows Phone no es muy amplia como la que ofrece el sistema operativo Android, la cantidad de software disponible actualmente es baja, pero se encuentra en crecimiento.

1.5.18 Cuadro Comparativo General

A continuación se muestra una tabla comparativa de los sistemas operativos móviles expuestos anteriormente.

Sistema Operativo	**Android**	**iOS**	**Windows Phone**
Interfaz			
Kernel	Linux	OS X	Windows NT
Tipo de SO	Abierto	Cerrado	Cerrado
Lenguaje de Programación Nativo	Java	Objective C	C#
Seguridad	Muy Buena	Susceptible a Malware	Muy Buena
Costos de desarrollo	25 usd una sola vez	99 usd anuales + Mac	99 usd anual
Adaptabilidad	Excelente	Excelente	Excelente
Edad de la Plataforma	Madura	Madura	Joven
Multitarea	Si	Si	Si
Standares Soportados	GSM, CDMA	GSM, CDMA	GSM, CDMA
hardware soportado	amplia gama de dispositivos	iPhone, iPad, iPod touch	Limitada gama de dispositivos
Actualización	Si	Si	Si
Cortar /Copiar/Pegar	Si	Si	Si
Programa de productividad	Google Docs	iWork	Office Mobile

Tienda de Libros	Google Books	iBooks	N/A
Tienda de software	Google Play	App store	Marketplace
Sincronización Wi-Fi	No por defecto	Si	Si
Red Social Integrada	Facebook, Twitter	Facebook, Twitter	Facebook, Twitter, Windows Live
Apps	500000+	650000+	100000+
Soporte para Tablet	Si	Si	No
Navegador	Basado en Chrome	Safari	Internet Explorer
Mapas	Google Maps	Mapas Apple	Bing Maps
Motor de búsqueda predeterminado	Google	Google	Bing
Expansión de almacenamiento	Micro SD	No	No
Soporte en la Nube	Google Sync, Google Drive	iCloud	SkyDrive
Asistente de Voz	S-Voice (Galaxy S III)	Siri	Tellme
Pantalla de Inicio	Iconos y widgets	Iconos	Baldosas(Tiles)
Soporte Flash	Si	No	No
Interfaz de Usuario	Más Técnico	Fácil	Fácil
Personalización	Profunda	Limitada	Ninguna
Notificaciones	Pull-down	Pull-down	Toast y Banners

Vida de la Batería	Poca duración	Media duración	Mayor duración

Tabla 1.4: Cuadro comparativo Android, iOS y Windows Phone

Fuente: [**WEB 8**] [**WEB 9**]

1.6 CONCLUSIONES

En base a la comparativa que se ha realizado se puede decir que el sistema operativo Android es una excelente plataforma para desarrollar aplicaciones, por ser de software libre y estar apadrinado por Google.

Para un desarrollador Android las ventajas son obvias: no hay formularios tediosos para darse de alta, tampoco diferentes certificados que se debe obtener para probar y distribuir las aplicaciones como en el caso de Apple, en Android sólo hay que pagar una sola vez para poder subir aplicaciones a Google Play, hay libertad total para publicar, no hay que esperar a que la aplicación sean revisada sino que al ser subida, inmediatamente estarán disponible.

Pero los inconvenientes que se encuentran en Android a la hora de programar son algunos, el principal es que no hay soporte personalizado de Google, por lo tanto hay que preocuparse de que la aplicación funcione en las distintas versiones de Android, otro sería que al no pasar las aplicaciones por ninguna revisión, puede haber publicaciones de aplicaciones que no valen la pena y tampoco son revisadas para encontrar algún tipo de error. En Google Play se venden menos aplicaciones que en iTunes, ya que hay más descargas de aplicaciones gratuitas que de pago.

CAPÍTULO II

INTRODUCCIÓN

CONTENIDO:

- Introducción a la plataforma Android
- Máquina Virtual Dalvik
- Android y Java ME
- GPS
- Android

CAPÍTULO 2: INTRODUCCIÓN

2.1 INTRODUCCIÓN A LA PLATAFORMA ANDROID

Android es un sistema operativo de código abierto basado en Linux, para ser utilizado en dispositivos con procesadores ARM. Android cuenta con varias librerías y Apis desarrolladas por Google, las aplicaciones corren en una máquina virtual denominada Dalvik.

Dalvik es una parte indispensable del Android, una aplicación antes de ser ejecutada es convertida en un ejecutable de la máquina de Dalvik (formato .dex), que está asignado para optimizar su uso en equipos que tienen muchas restricciones en uso como el procesador y memoria en los teléfonos inteligentes.

El sistema operativo Android tiene un motor de base de datos relacional denominada SQLlite, que permite a cualquier aplicación crear tablas, relaciones, realizar querys y obtener datos en forma eficiente.

El SDK de Android soporta la mayoría de plataformas Java, Standard Edition (Java SE) a excepción de la Abstract Window Toolkit (AWT) y Swing. En lugar del AWT y Swing, el SDK de Android tiene su propio conjunto de interfaz de usuario moderno. Al programar en Android se programa en Java, por lo cual se puede esperar tener una

Java Virtual Machine (JVM), que es responsable de la interpretación, el tiempo de ejecución. Una JVM proporciona la optimización necesaria para ayudar a llevar a Java a niveles de rendimiento comparables a lenguajes compilados como C y C + +.

2.2 MÁQUINA VIRTUAL DALVIK

Dalkiv es el nombre de la máquina virtual que utiliza Android (DalvikVM) que ha sido diseñada para optimizar la memoria y los recursos de hardware en el entorno de teléfonos móviles, Dalvik está basada en registro a diferencia de la máquina virtual de Java que está basada en uso de pilas, Dalvik es un intérprete que sólo ejecuta los archivos ejecutables con formato .dex (Dalvik Executable), que es un formato optimizado para el almacenamiento eficiente y ejecución mapeable en memoria.

Permite que el código sea compilado a un bytecode independiente de la máquina en la que se va a ejecutar, y la máquina virtual interpreta este bytecode al ejecutar el programa. Una de las razones por las que se optó por utilizar la máquina virtual de Java es la necesidad de optimizar los recursos al máximo y enfocar el funcionamiento de los programas hacia un entorno dónde los recursos de memoria, procesador y almacenamiento son escasos.

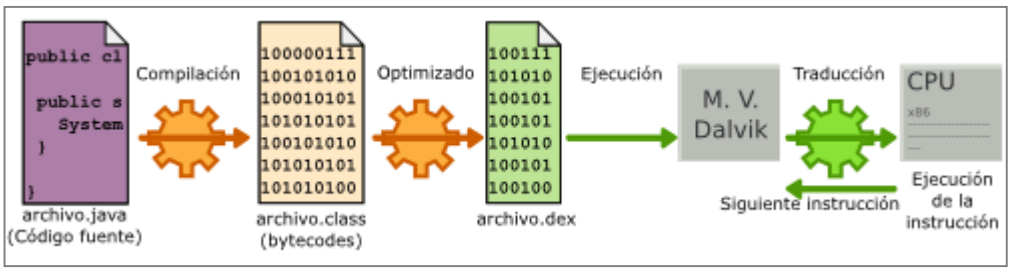


Figura 2.1: Funcionamiento de la Máquina Virtual Dalvik

Fuente: [**WEB 10**]

La herramienta "dx" incluida en el SDK de Android permite transformar las clases compiladas (.class) por un compilador de lenguaje Java en formato .dex.

La máquina virtual Dalvik ha sido optimizada para que múltiples instancias de esta máquina puedan funcionar al mismo tiempo con un impacto muy bajo en el rendimiento de la memoria del dispositivo. Su objetivo es proteger a las aplicaciones, de forma que el cierre o fallo inesperado de alguna de ellas no afecte de ninguna forma a las demás.

2.2.1 Dalvik y Java Virtual Machine

La máquina virtual de Java, que se encuentra en casi todas las computadoras, se basa en el uso de las pilas. Dalvik utiliza los registros, ya que los teléfonos móviles están optimizados para la ejecución basada en estos.

Aunque utiliza el lenguaje Java para programar aplicaciones Android, el bytecode de Java no es ejecutable en un sistema Android, así como también las librerías Java que utiliza Android son levemente distintas a las utilizadas en Java Standard Edition (Java SE) o en Java Mobile Edition (Java ME), guardando también características en común.

2.2.2 Android y JavaME

Android utiliza un lenguaje muy conocido como es Java, lo cual ayuda a que cualquier programador q tenga un mínimo de experiencia pueda comenzar a programar sus aplicaciones sin mayor complicación, incluye las APIs más importantes de este lenguaje como java.util, java.io o java.net.

La plataforma Android viene con todo lo que se necesita en un solo paquete: el sistema operativo, controladores de dispositivo, bibliotecas centrales, la JNI, el VM Dalvik optimizada, y el entorno de desarrollo de Java. Los programadores pueden tener la seguridad de que cuando se desarrollan nuevas aplicaciones, todas las bibliotecas principales estarán disponibles en el dispositivo.

Figura 2.2: Disponibilidad de Java para configuraciones informáticas diversas
Fuente: Propia

La figura muestra la disponibilidad de Java para configuraciones informáticas diversas. Java Platform, Standard Edition (Java SE) es adecuado para configuraciones de escritorio y estaciones de trabajo. Java Platform, Enter-premio Edition (Java EE) está diseñado para configuraciones de servidor.

Java Platform, Micro Edition (Java ME) es una edición de Java que está recortada para dispositivos. Además, dos conjuntos de configuración están disponibles para Java ME. La primera con-figuración se llama Connected Device Configuration (CDC).

Java ME para CDC implica una comparación abajo versión de Java SE con menos paquetes, menor número de clases dentro de los paquetes, e incluso menos campos y métodos en las clases. Para los aparatos y dispositivos que se encuentran más constreñido, Java define una configuración llamada Connected Limited Device Configuration (CLDC).

2.3 GPS

Es un sistema global de navegación por satélite que permite determinar en cualquier parte del mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros (aunque en los Smartphone rara vez es así de preciso). El sistema fue desarrollado, instalado y actualmente operado por el Departamento de Defensa de los Estados Unidos.

2.3.1 Como Funciona

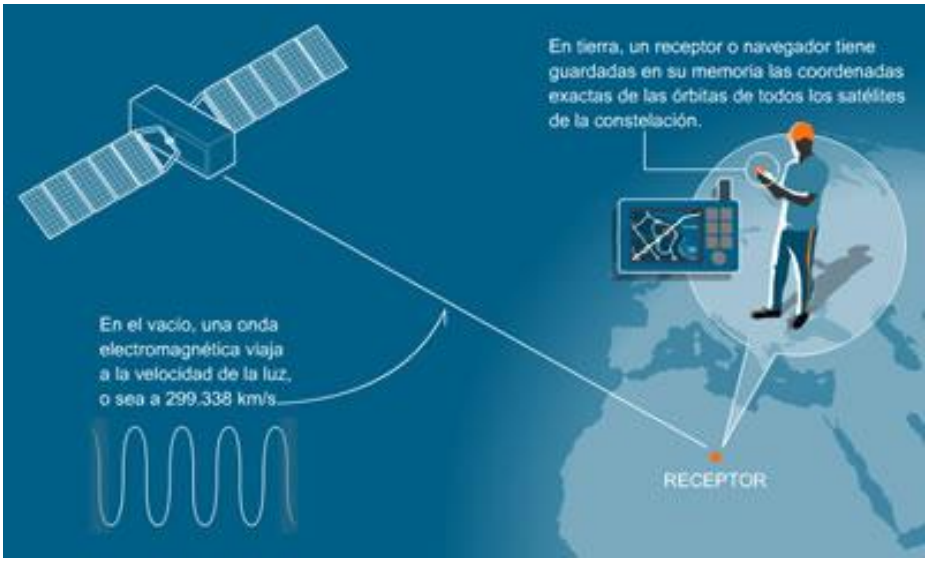


Figura 2.3: Funcionamiento GPS
Fuente: [**WEB 11**]

El funcionamiento es sencillo:

- El dispositivo manda una señal a los satélites, y al menos espera la respuesta de tres de ellos.

- Cada satélite en cobertura le envía una señal con el identificador y la hora de recepción del mensaje.
- El dispositivo calcula la diferencia entre la hora actual y la de recepción de cada satélite, y "triangula" la posición en referencia a los resultados).

En exteriores este método es muy fiable, en interiores pierde todo su potencial, ya que depende de satélites, y por consiguiente de cobertura externa.

2.3.2 Integración con telefonía móvil

Actualmente dentro del mercado de la telefonía móvil la tendencia es la de integrar, por parte de los fabricantes, la tecnología GPS dentro de sus dispositivos. El uso y masificación del GPS está exclusivamente extendido en los teléfonos móviles Smartphone, lo que ha hecho surgir todo un ecosistema de software para este tipo de dispositivos, así como nuevos modelos de negocios que van desde el uso del terminal móvil para la navegación tradicional punto-a-punto hasta la prestación de los llamados Servicios Basados en la Localización.

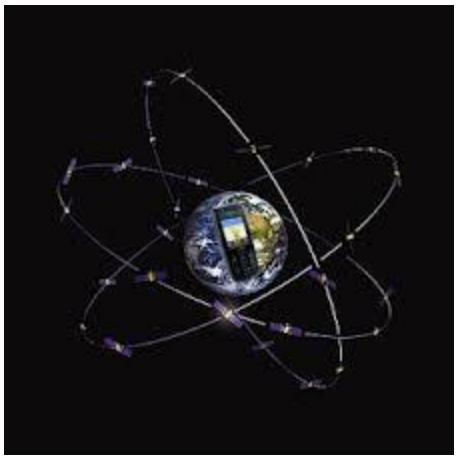


Figura 2.4: GPS en Telefonía Móvil

2.3.3 GPS actuales

Existen varios modelos, algunos ya antiguos, y otros todavía operativos. Pero enfocados a dispositivos móviles, se puede mencionar tres: [**WEB 12**]

- GPS: La base en la que se sustentan casi todos los dispositivos móviles. Como principal ventaja está el hecho de ser el estándar en geolocalización y que no depende de la cobertura 3g. Como desventaja, la poca fiabilidad, la dependencia de buena cobertura exterior y los tiempos de espera entre envío y recepción de información.
- A-GPS: El A-GPS fue creado precisamente por el tiempo de espera inicial del GPS, a-GPS hace uso de Internet para dar una posición más o menos acertada y ayudar al GPS a triangular más rápido. Depende de datos móviles, y no mejora al GPS simplemente le da una pequeña ayuda inicial. Su funcionamiento es sencillo y rápido. Se conecta a internet para descargar un paquete de datos con la información de la antena a la que está conectado.
- GLONASS: Los satélites GPS son dominio de EEUU, los de GLONASS son rusos. Este sistema lleva más de 30 años, pero por ser soviéticos, no es muy conocido. Por este motivo Rusia amenazó con aumentar los impuestos de dispositivos móviles que no tuvieran soporte a GLONASS para que los fabricantes comenzarán a tomarlo en serio. GLONASS entra en funcionamiento cuando el GPS falla. No hace falta estar perdido en el desierto para que no haya cobertura GPS, sino que simplemente una ciudad con edificios altos tiene a su vez puntos ciegos, y la ayuda de GLONASS es mejor. Tiene más margen de error que el GPS, pero la unión de estos dos hace más efectivo el posicionamiento.

2.3.4 Arquitectura GPS

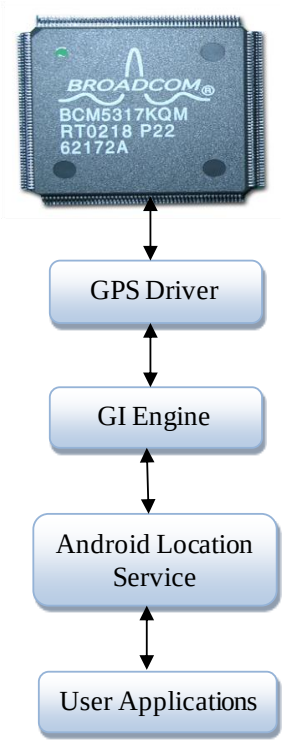


Figura 2.5: Arquitectura GPS
Fuente: [**WEB 13**]

2.3.4.1 El Chip

Se trata de un receptor de radiofrecuencia que se comunica directamente con los satélites GPS.

2.3.4.2 El Driver

Sirve de controlador para el chip, el GPS Driver usa API's de nivel Bajo para comunicarse con el Chip GPS. Consiste sencillamente en una serie de archivos que se encuentran en /System/lib/hw o /Vendor/Lib/hw y usualmente del tipo gps.*.os dependiendo de la versión de Android (ejemplos: gps.default.so o

gps.aries.so). Cabe mencionar que son los propios fabricantes quienes liberan este producto.

2.3.4.3 GL Engine

Este es el encargado de suministrar la localización de los nodos de radiofrecuencia cercanas, guardando los datos obtenidos en el GPS Lock.

Consiste de un path (/system/bin) acompañado de nombres como glgps o gpsd. Funciona utilizando unos parámetros de configuración que consisten en archivos .xml o .conf (glconfig.xml, gps.xml, gpsconfig.xlm, gps.conf). Estos nombres dependen de la versión de Android y se encuentran generalmente en (/system/etc, /system/etc/gps o /vendor/etc).

Dependiendo de la configuración y de la plataforma, GL Engine toma la información de Localización de las torres móviles y lee el NVRAM (es el sitio donde se almacena data del GPS lock), esta información se encuentra generalmente en /data/gps en archivos acabados en .sto y .dat (ejemplo: gldata.sto, lto.dat, xtra.bin, epo.dat).

Al hacer uso de toda esta información, el GL Engine asiste al GPS Driver: por lo tanto es capaz de detectar múltiples satélites GPS para los que está programado el GPS Driver, pero para fijar la posición necesita información adicional que puede descargar de los satélites GPS (de manera muy lenta) o bien puede acceder a Internet para usar los servidores SUPL/NTP (mayor velocidad), inmediatamente guarda todos estos datos en el NVRAM para próximos usos.

2.3.4.4 Android Framework o Servicio de Localización

Consiste en una serie de clases como puede ser el Location Manager, que proporcionan servicios para que una aplicación use el GL Engine. Es una especie de puente entre la aplicación y el GL Engine.

Pertenecen al Framework de Android, y por lo tanto de java. Se trata de un grupo de funciones contenidas en el paquete *android.location* del framework que controlan diferentes aspectos del GPS con el fin de comunicarse entre Apps y GL Engine.

Algunos ejemplos son:

- addGpsStatusListener (GpsStatus.Listener listener): Cambia el estado del GPS a *escuchando*.
- getProviders (Criteria criteria, boolean enabledOnly): Retorna una lista de torres cercanas que satisfacen X criterios pasados a la función.

2.3.4.5 Aplicaciones de Usuario

La última capa son las aplicaciones que se tiene instaladas en el terminal, se trata de cualquier aplicación que use el GPS. Google Maps, Navigation, etc.

2.4 ANDROID

Android es el nombre esencial para un sistema operativo enfocado al uso del mismo en dispositivos móviles, tomando en cuenta que al inicio se lo creo para ser usado solamente en teléfonos celulares. En la actualidad se puede encontrar a Android en todo tipo de dispositivos como tablets, Smartphone, netbooks, entre otros.

Android es una pila de software para dispositivos móviles que incluye un sistema operativo basado en Linux, middleware y una serie de aplicaciones esenciales, con una interfaz de programación Java.El SDK de Android proporciona todas las

herramientas y APIs necesarias para desarrollar aplicaciones, incluye un compilador, un depurador y un emulador de dispositivo, así como su propia máquina virtual para ejecutar programas Android.

2.4.1 Arquitectura

La arquitectura del sistema operativo Android está formada por capas de software donde cada una puede utilizar los servicios de la capa inferior. Comenzando por la capa inferior se encuentra el conjunto de drivers basados en Linux, esta parte no es pública. Un nivel más arriba se encuentra un conjunto de librerías que no son accesibles directamente sino a través del nivel superior llamado Framework de aplicaciones y junto a la capa de aplicaciones son totalmente públicas, por lo tanto los usuarios pueden acceder libremente.

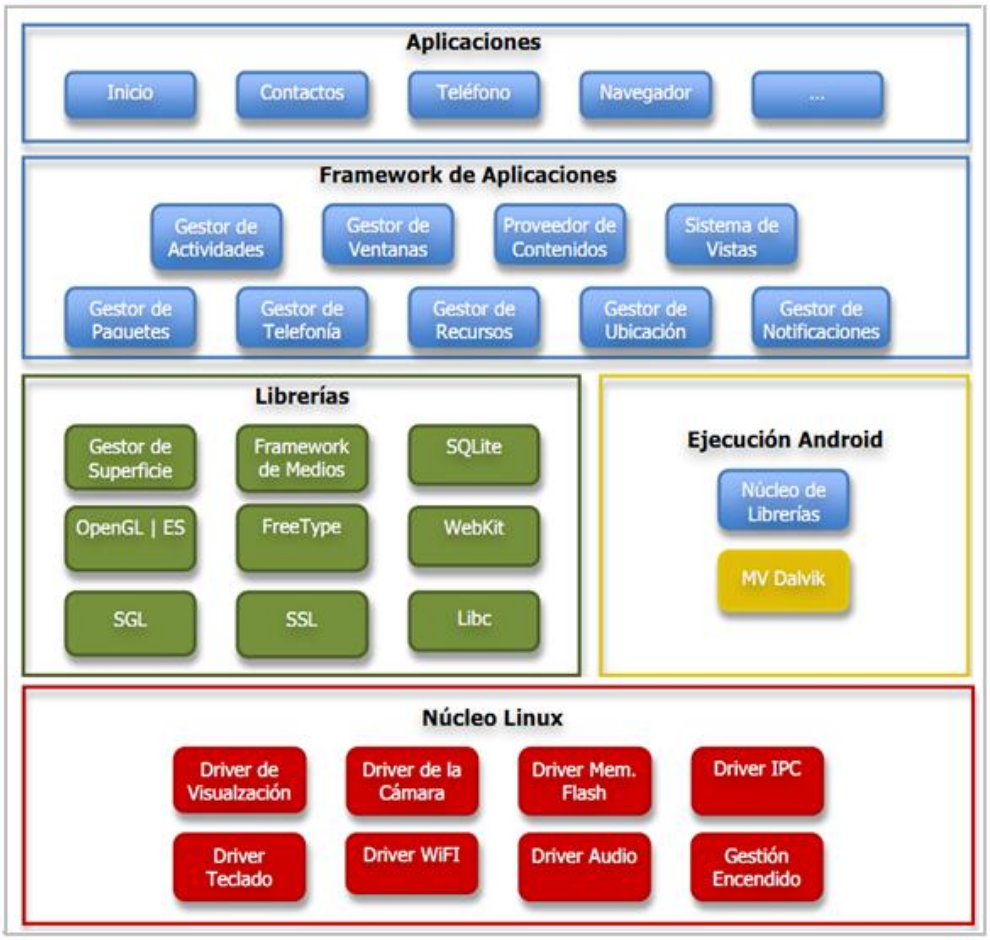


Figura 2.6: Arquitectura del Sistema Operativo Android
Fuente: [**WEB 14**]

2.4.1.1 Aplicaciones

En la capa de aplicaciones es el lugar donde se incluyen todas las aplicaciones del dispositivo. Las aplicaciones básicas incluyen un cliente de email, programa de SMS, calendario, mapas, navegador, contactos, entre otros, las aplicaciones generalmente se encuentran escritas en lenguaje Java.

2.4.1.2 Framework de Aplicaciones

Esta capa se encuentra formada por las clases y servicios que utilizan las aplicaciones para realizar trabajo. La mayor parte de los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik.

2.4.1.3 Librerías

Android incluye en su base de datos un conjunto de bibliotecas o librerías que están escritas en C o C++ , que son expuestas a todos los desarrolladores a través del framework de aplicaciones Android System C library, librerías de medios, librerías de gráficos, 3D, SQlite, entre otras.

2.4.1.4 Runtime

Android incorpora un set de librerías que aportan la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalkiv ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalkiv ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros, y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida "dx".

2.4.1.5 Núcleo Linux

Android depende de Linux versión 2.6 para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de

controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software

El núcleo del sistema operativo Android está basado en el kernel de Linux versión 2.6, similar al que puede incluir cualquier distribución de Linux, como Ubuntu, solo que adaptado a las características del hardware en el que se ejecutará Android, es decir, para dispositivos móviles.

El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores. De esta forma también nos evitamos el hecho de quebrarnos la cabeza para conocer las características precisas de cada teléfono. Si necesitamos hacer uso de la cámara, el sistema operativo se encarga de utilizar la que incluya el teléfono, sea cual sea. Para cada elemento de hardware del teléfono existe un controlador (o *driver*) dentro del kernel que permite utilizarlo desde el software.

El kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación (*networking*), etc.

2.4.2 Anatomía de una Aplicación

Una aplicación Android se compone de muchos partes o componentes, a continuación se describen las más importantes:

2.4.2.1 Activity

Una actividad representa a la capa de presentación o controlador de una aplicación, una aplicación puede tener varias actividades independientes que trabajan juntas durante el tiempo de ejecución de una aplicación, una de las

actividades se marca como la inicial al momento de arrancar una aplicación.Toda actividad se inicia como respuesta a un *Intent*.

2.4.2.2 Intents

Las Intenciones son los mensajes del sistema que se encuentran corriendo en el interior del dispositivo. Se encargan de notificar a las aplicaciones de varios eventos: cambios de estado en el hardware (ej. cuando se inserta una SD al teléfono), notificaciones de datos entrantes (ej. cuando llega un SMS) y eventos en las aplicaciones (ej. cuando una actividad es lanzada desde el menú principal).

2.4.2.3 Content providers

Proveedores de Contenido, este elemento ofrece un conjunto de datos almacenados en el dispositivo para que se pueda acceder y compartir entre varias aplicaciones. Se puede guardar datos en archivos del sistema, en una base de datos SQLite, en la web o en otro lugar de almacenamiento persistente a la que la aplicación pueda tener acceso. A través del proveedor de contenido, otras aplicaciones pueden consultar o incluso modificar los datos (si el proveedor de contenidos lo permite).

2.4.2.4 Services

Las actividades, las intenciones y los proveedores de contenido son de corta duración y pueden ser detenidos en cualquier momento.

Por el contrario, los servicios están diseñados para seguir corriendo, y si es necesario, de manera independiente de cualquier actividad. Por ejemplo el reproductor de música, que es un servicio que puede mantenerse corriendo mientras se envía un SMS o alguna otra función en nuestro teléfono.

2.4.2.5 BroadcastReceiver

No realiza ninguna acción por sí mismo, recibe y reacciona ante anuncios de tipo broadcast, existen muchos originados (por el sistema Batería baja), las aplicaciones pueden lanzar un broadcast, no tienen UI pero pueden iniciar una actividad para atender al anuncio.

2.4.2.6 Views

Las vistas son los widgets de interfaz de usuario, por ejemplo, botones, campos de texto entre otros. La clase base para todos los dictámenes es *android.view.View*. La disposición de las vistas es gestionada por las subclases de *android.view.ViewGroups*. Visto a menudo tienen atributos que pueden ser utilizados para cambiar su apariencia y comportamiento.

2.4.2.7 Widgets

Son componentes interactivos que se utilizan en la pantalla de inicio de Android. Presentan típicamente algún tipo de dato y permite al usuario realizar acciones a través de ellos. Por ejemplo, un widget puede mostrar un breve resumen de los correos electrónicos nuevos y si el usuario selecciona una dirección de correo electrónico podría iniciar la aplicación de correo electrónico seleccionada.

2.4.3 Ciclo de vida de una Aplicación

Las aplicaciones están formadas por actividades, en cierto momento una actividad pasa al primer plano y se coloca por encima de otra formado una pila de actividades, el botón back cierra la actividad y recupera de la pila la anterior. Las aplicaciones en Android no tienen control de su ciclo de vida, deben estar preparadas para su terminación en cualquier momento, cada aplicación se

ejecuta en su propio proceso. El runtime de Android gestiona el proceso de cada aplicación y por extensión de cada Actividad que contenga. Las aplicaciones Android están constituidas a partir de la combinación de los siguientes bloques:

- **Activity.-** Una actividad es cada una pantalla dentro de una aplicación, es la responsable de presentar los elementos visuales y reaccionar a las acciones del usuario, toda actividad se inicia como respuesta a un Intent.

- **Intent.-** Android usa una clase especial llamada "Intent" para moverse de una pantalla a otra. Una intención describe lo que una aplicación desea hacer. Cuando se lanzan Intent el sistema busca que actividades son capaces de dar respuesta a ese Intent y elige la más adecuada.

- **Service.-** Un servicio es una tarea que se ejecuta durante periodos prolongados y no interacciona con el usuario.

- **ContentProvider.-** Un proveedor de contenido es una clase que implementa un conjunto estándar de métodos para que otras aplicaciones almacenen o recuperen

Una aplicación no requiere utilizar todos estos componentes. Los componentes que se vaya a utilizar en la aplicación se definen en el archivo llamado "AndroidManifest.xml".

2.4.4 Anatomía de una Actividad

Una actividad es cada pantalla que se quiere mostrar, aunque a veces puede no tener UI. Las actividades son el elemento más usado en una aplicación para Android. Cuando una actividad se inicia se pone en pausa la anterior y se envía a la pila, el usuario puede navegar como en la web por las actividades, para pasar de una activity a otra se realiza mediante intenciones (Intent) realizando startactivity (URI). La URI permite que cada actividad puede ser reutilizada por otras aplicaciones y reemplazadas fácilmente.

2.4.4.1 Ciclo de vida de una Actividad

Las actividades son manejadas mediante una pila de actividades, cuando una nueva actividad es iniciada la anterior es pausada y almacenada en una pila de actividades, cuando la nueva actividad termine, la siguiente actividad en la pila es reactivada. El sistema puede resolver finalizar una actividad pausada o detenida por motivos de memoria, cada actividad es responsable de salvar su estado de forma que sea posible restaurarla tras haber sido pausada o detenida. Una actividad puede estar en cuatro estados:

2.4.4.1.1 Activa

Cuando la actividad ha sido iniciada por el usuario, está actualmente ejecutándose y se encuentra en primer plano y por tanto en la parte superior de la pila.

2.4.4.1.2 Pausada

Cuando la actividad ha sido iniciada por el usuario, está actualmente ejecutándose y está visible, pero una notificación o alguna otra cosa esta

sobrepuesta en alguna parte de la pantalla. Durante este estado, el usuario puede ver la actividad, pero no es posible interactuar con ella.

2.4.4.1.3 Detenida

Cuando la actividad ha sido iniciada por el usuario, sigue ejecutándose pero se encuentra oculta por otras actividades que se han lanzado. Cuando una actividad se encuentra en este estado, la actividad no es capaz de mostrar información significativa para el usuario de manera directa, pero puede hacerlo mediante el uso de notificaciones.

2.4.4.1.4 Finalizada

Una vez que la actividad ha completado su ejecución libera todos los recursos que estaba utilizando y finaliza el ciclo. Si es llamada de nuevo iniciará el ciclo completamente a diferencia de una actividad detenida, la cual aún almacena su estado de ejecución a fin de recuperarlo cuando sea llamada de nuevo.

Figura 2.7: Ciclo de vida de una Actividad
Fuente: [**WEB 15**]

Una actividad puede iniciar otra dentro de la misma aplicación. De la misma forma puede iniciar otra de una aplicación externa.

2.4.4.2 Métodos del ciclo de vida

Algunos de los métodos del ciclo de vida son los siguientes:

Estado	**Descripción**	**Siguiente**
onCreate()	(Bundle *savedInstanceState*): Creación de la Actividad, puede recibir información de estado de instancia, por si se reanuda desde una instancia pausada anterior.	onStart()
onRestart()	Se ejecuta después de que la aplicación se haya detenido, justo después de que sea iniciada de nuevo.	onStart()
onStart()	Inicio de la actividad recién creada.	onResume() o onStop()
onResume()	Se ejecuta cuando la Activity interactúa con el usuario, en éste plano la Activity está en la cima de la pila.	onPause()
onPause()	Se ejecuta cuando el sistema está a punto de continuar una Activity anterior.	onResume() o onStop()
onStop()	Se ejecuta cuando la Activity deja de ser visible al usuario, porque otra Activity ha continuado y pasa a un lugar más prioritario de la pila. Fin de la actividad	onRestart() o onDestroy()
onDestroy()	Es la llamada final a la actividad, después de ésta, es totalmente destruida.	Ninguno

Tabla 2.1: Métodos del Ciclo de Vida
Fuente: [**WEB 16**]

2.4.5 Librerías

Android ofrece una gama de APIs para el desarrollo de aplicaciones. A continuación se describe una lista de las APIs principales que ofrece Android:

Librería	**Descripción**
android.util	Paquete básico de servicios públicos, contiene las clases de bajo nivel, como contenedores especializados, formateadores de cadenas, y de análisis XML de servicios públicos.
android.os	Paquete de sistema operativo, permite el acceso a los servicios básicos como el paso de mensajes, comunicación entre procesos y funciones de reloj.
android.graphics	Es el suministro de las clases de bajo nivel como lienzos de apoyo, colores y las primitivas de dibujo.
android.text	Las herramientas de procesamiento de texto para mostrarlo y analizarlo.
android.database	Proporciona las clases de bajo nivel necesario para la manipulación de cursores cuando se trabaja con bases de datos.
android.content	El contenido de la API se utiliza para admirar el acceso a los datos y a la publicación, proporcionando los servicios para hacer frente a los recursos, los proveedores de contenido y los paquetes.
android.view	Las vistas son un núcleo de la interfaz de usuario. Todos los elementos de la interfaz se construyen

	utilizando una serie de vistas que proporcionan los componentes de interacción con el usuario.
android.widget	Construido sobre el paquete de Vista, están las clases widget, elementos de la interfaz de usuario para su uso en las aplicaciones. Se incluyen listas, botones y diseños.
com.google.android.maps	API que proporciona acceso a los controles de mapas que se puede utilizar en una aplicación. Incluye el control MapView, superposición y la clase MapController utilizados para anotar y controlar los mapas.
android.app	Paquete que proporciona el acceso al modelo de solicitud, incluye la actividad de servicios y las API que forman la base de todas las aplicaciones.
android.provider	Para facilitar el acceso a los desarrolladores a determinados proveedores de contenidos estándar, el paquete proveedor ofrece clases para todas sus distribciones.
android.telephony	Las API´s de telefonía dan la posibilidad de interactuar directamente con el dispositivo de Teléfono, permitiendo realizar, recibir y controlar las llamadas de teléfono, estado y mensajes SMS.
android.webkit	Ofrece funciones para trabajar con contenido basado en web, incluyendo un control WebView para incrustar los navegadores en sus actividades y un administrador de cookies.
android.location	Da a la aplicación acceso a la ubicación física del dispositivo actual. Los servicios de localización

	ofrecen acceso genérico a información de localización utilizando cualquier posición de hardware o tecnología disponible en el dispositivo.
android.media	Las API de los medios de comunicación proporcionan soporte para reproducción y grabación de audio.
android.opengl	Android ofrece un potente motor 3D que utiliza la API de OpenGL ES que se puede utilizar para crear interfaces de usuario en 3D.
android.hardware	El hardware de la API expone un sensor incluyendo la cámara, acelerómetro, sensores y una brújula.

Tabla 2.2: Librerías Android
Fuente: [**WEB 17**]

CAPÍTULO III

ENTORNO DE DESARROLLO

CONTENIDO:

- Estructura de un Proyecto Android
- Entorno de Trabajo
- Interfaz de Usuario
- Widgets.

CAPÍTULO 3: ENTORNO DE DESARROLLO

3.1 ESTRUCTURA DE UN PROYECTO ANDROID

Es muy importante saber que archivos componen un proyecto Android antes de empezar a realizar aplicaciones para esta plataforma. Cuando se crea un nuevo proyecto Android, se genera automáticamente la estructura de carpetas necesaria para la aplicación, en la imagen siguiente se encuentran los elementos creados inicialmente para un nuevo proyecto Android:

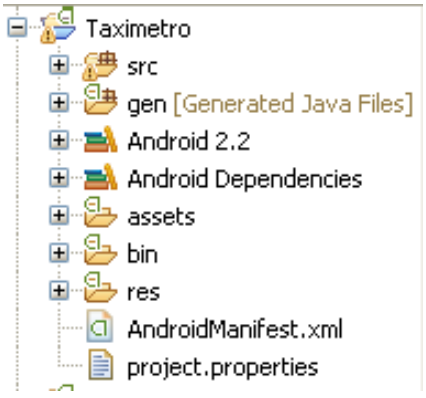


Figura 3.1: Estructura de un Proyecto
Fuente: Propia

A continuación se explica los elementos principales de los ficheros que contendrán el código, recursos o instrucciones de generación de una aplicación:

3.1.1 La carpeta /src

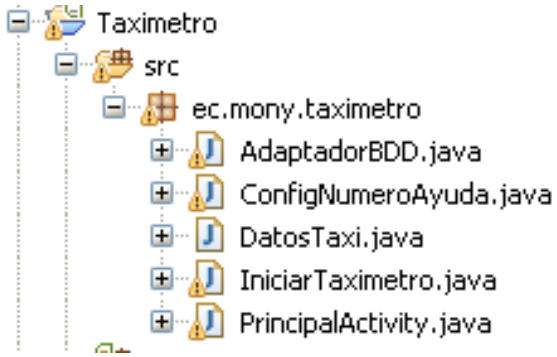


Figura 3.2: Carpeta /src
Fuente: Propia

Esta carpeta contiene todo el código fuente, pueden estar agrupadas en paquetes (carpetas) para ayudar a categorizar la lógica, este tipo de estructura es igual a la de Java, es donde se alojan las subclases de activity creadas.

3.1.2 La carpeta /assets

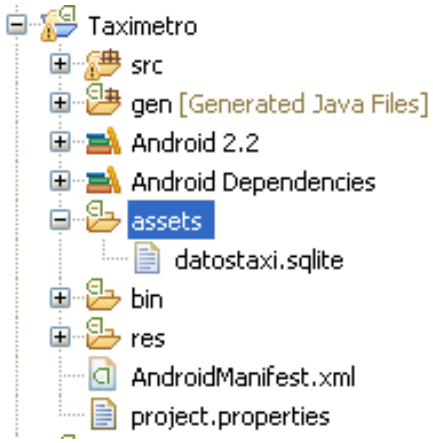


Figura 3.3: Carpeta /assets
Fuente: Propia

Aquí se puede almacenar carpetas y archivos que servirán como dato en bruto, para ser llamados como archivos planos.

3.1.3 La carpeta /res

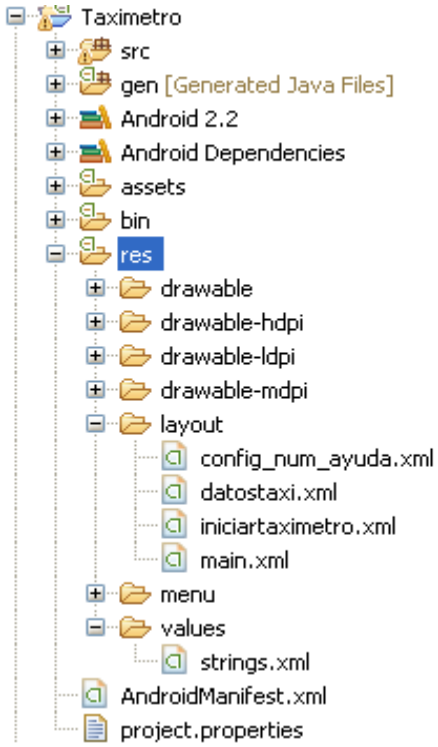


Figura 3.4: Carpeta /res
Fuente: Propia

Dentro de esta carpeta se almacenan recursos de la aplicación como imágenes, valores, plantillas, cadenas de texto, etc:

Carpeta	**Definición**
/res/drawable/	Contiene las imágenes que se van a utilizar en la aplicación, como fondo de un widget o como icono. Para utilizar diferentes recursos dependiendo de la resolución del dispositivo se divide en varias subcarpetas: ✓ /drawable-ldpi

	✓ /drawable-mdpi ✓ /drawable-hdpi
/res/ values/	contiene el archivo strings.xml en el que se declara las variables que se utiliza en la aplicación, como por ejemplo: <string name=”actualizar”>Actualizar</string>
/res/ layout/	Contiene los archivos XML que definen el diseño de las diferentes interfaces gráficas de la aplicación, por defecto en el proyecto creado vendrá el mail.xml. Para definir distintos layouts dependiendo de la orientación del dispositivo se puede dividir en dos subcarpetas: ✓ /layout ✓ /layout-land
/res/ menu/	Especificaciones XML que definen menús.
/res/anim/	Contiene la definición de las animaciones utilizadas por la aplicación.
/res/xml/	Contiene los ficheros XML utilizados por la aplicación.
/res/raw/	Contiene recursos adicionales, comunmente en formato diferente a XML, que no se incluyan en el resto de carpetas de recursos.

Tabla 3.1: Recursos de la carpeta /res
Fuente: [**WEB 18**]

3.1.4 La carpeta gen/

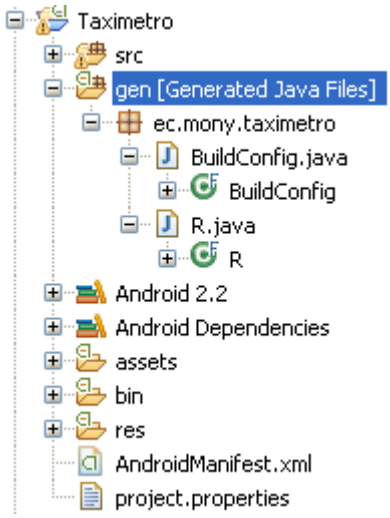


Figura 3.5: Carpeta /gen
Fuente: Propia

Contiene ficheros Java autogenerados, destacando *R.java*, el cual contiene constantes numéricas, corresponde cada una con uno de los recursos de la carpeta res/ y de la carpeta *assets/;build.xml:* contiene una sentencia de comandos para compilar la aplicación e instalarla en un dispositivo. El archivo que se requiere para la ejecución de la aplicación en el dispositivo móvil es el de extensión *.apk*. Este es un fichero comprimido que contiene el .dex, el manifiesto, y los recursos y se encuentra dentro de la carpeta bin/.

3.1.5 La carpeta bin/

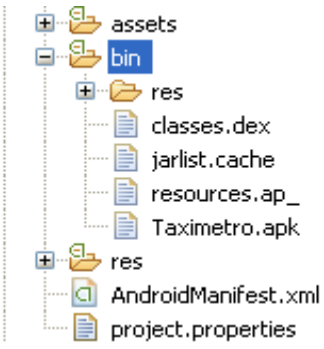


Figura 3.6: Carpeta /bin
Fuente: Propia

Guarda la aplicación ya compilada, el archivo classes.dex es el ejecutable, también contiene las clases compiladas y al archivo *.ap_ que mantiene los recursos de la aplicación.

3.1.6 La carpeta libs/

Contiene los JARs (Java Archive es un tipo de archivo que permite ejecutar aplicaciones escritas en Java) que la aplicación requiera para su ejecución, el fundamental es android.jar, el .jar usado para crear aplicaciones que hagan uso de Google Maps es maps.jar

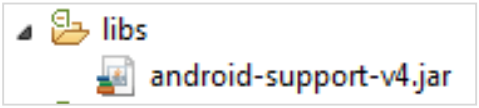


Figura 3.7: Carpeta /libs
Fuente: Propia

3.1.7 El archivo AndroidManifest.xml

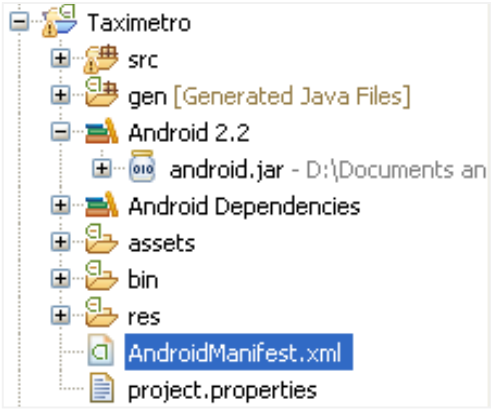


Figura 3.8: Archivo AndroidManifest.xml
Fuente: Propia

Este archivo es el *config* de una aplicación, en este se declara el nombre de la aplicación, la versión, el icono, las actividades, permisos, etc.

Se puede decir que el archivo AndroidManifest.xml, es uno de los archivos más importantes de una aplicación y que todas las aplicaciones o librerías deben contener en la raíz del proyecto. En este archivo, se definen las características del proyecto como el nombre de la aplicación, la versión, el icono, las actividades, paquete o los permisos que va a requerir la aplicación. También es donde se describe los componentes de la aplicación.

Se lo puede visualizar y editar a partir de un editor visual o un editor de archivos XML. La estructura de un "Android Manifest" es la que se muestra a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="ec.mony.taximetro"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <application android:icon="@drawable/ic_launcher" android:label="@string/app_name">
        <activity android:name="PrincipalActivity"
                  android:screenOrientation="portrait"
                  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=".ConfigNumeroAyuda" android:screenOrientation="portrait"/>
        <activity android:name=".DatosTaxi" android:screenOrientation="portrait"/>
        <activity android:name=".IniciarTaximetro" android:screenOrientation="portrait"/>
        <activity android:name=".Menu"/>
    </application>
</manifest>
```

Figura 3.9: Esquema del Archivo AndroidManifest.xml
Fuente: Propia

A continuación se describe sus partes:

<manifest>	• ***package = "ec.mony.taximetro"*** **Es el paquete del programa con el cual se referencia mi aplicación en GooglePlay y el teléfono.** • ***android:versionCode = "1"*****: Hace referencia al número de versión de desarrollo de nuestro programa, cada versión final que se desea publicar debe tener un versionCode distinto.** • ***android:versionName = "1"*****: Es el número de versión del programa.**
<uses-sdk>	En este tag se determina las distintas versiones Android que va a utilizar en la aplicación, tanto sobre qué versiones va a correr como qué versión fue utilizada para realizar las pruebas. Mediante el atributo *android:minSdkVersion*, se establece a partir de qué versión de Android la aplicación podrá correr.
<application>	Este elemento define las entrañas de la aplicación (nombre, actividad principal, icono, etc.) que el archivo AndroidManifest describe.
<uses-permissions>	Mediante este elemento se especifica los permisos que va a necesitar la aplicación para poder ejecutarse, son las condiciones que deberá aceptar el usuario antes de instalar en su dispositivo. Por ejemplo, si se desea utilizar funcionalidades con GPS o envió de SMS del

	teléfono, hay que indicar que la aplicación requiere estos permisos.

Tabla 3.2: Partes del Archivo AndroidManifest.xml
Fuente: [**WEB 19**]

3.2 ENTORNO DE TRABAJO

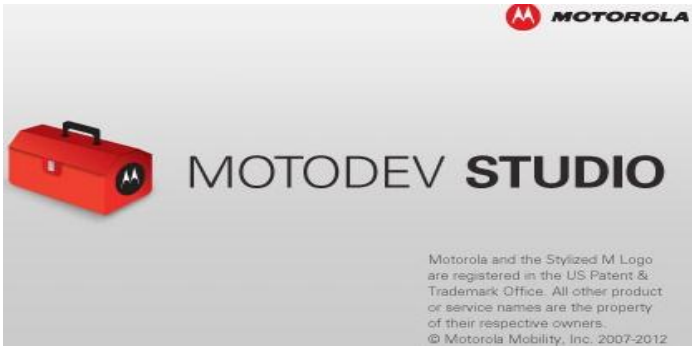
Antes de empezar a desarrollar en android primero se debe seleccionar el entorno de trabajo IDE donde se van a crear los proyectos.

Paso 1: Descargar e instalar JDK (Java Development Kit).- Para poder empezar a desarrollar aplicaciones en Android, primero se debe tener correctamente instalado y configurado el JDK.

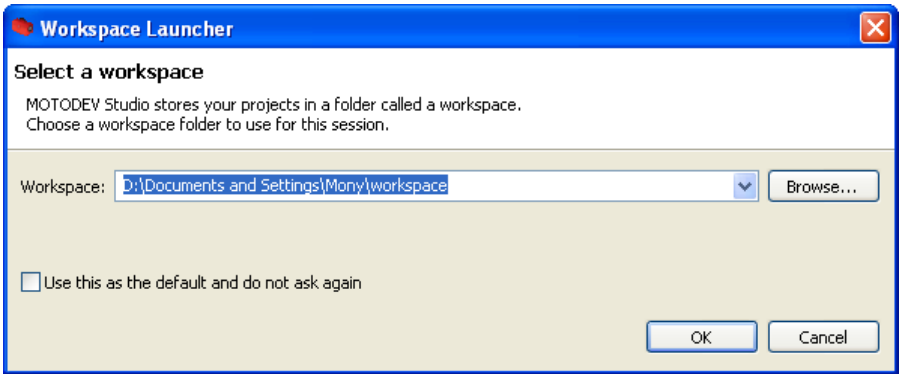
Paso 2: Descargar MOTODEV Studio for Android, que es un entorno de desarrollo gratuito creado por Motorola para el desarrollo de aplicaciones Android. Se basa en Eclipse y añade una serie de características adicionales, incluye el ADT necesario para el desarrollo, incluye muchas mejoras, asistentes más cómodos de ayuda para el desarrollo de android como por ejemplo un visor de Base de Datos , noticias , incluso el emulador se encuentra dentro del mismo IDE, además se puede ver cómo reacciona la aplicación ante una llamada entrante o un mensaje entrante, incluye snipes que son porciones de código ya armadas para trabajar, se puede configurar un equipo remoto para poder conectarme a un dispositivo android real remotamente, se puede firmar desde dentro de la aplicación y publicarla directamente en google play.

Se lo puede descargar desde la página de Motorola http://developer.motorola.com. La descarga está disponible para Mac, Linux y Windows en las versiones de 32 y 64 bit, la ventaja de esto es que desde un instalador se puede tener todo lo que hace falta para iniciarse en android, pero primero tiene que estar registrado como usuario en la página de Motorola para descargarse, se tiene que llenar un formulario de manera sencilla, una vez descargado seguimos los siguientes pasos:

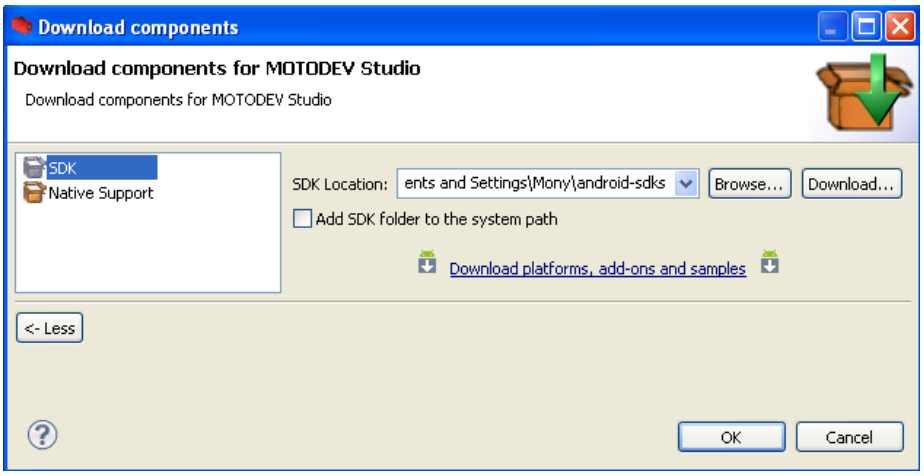
1. Instalar Motodev Studio for Android, la instalación es el clásico siguiente, siguiente y siguiente.



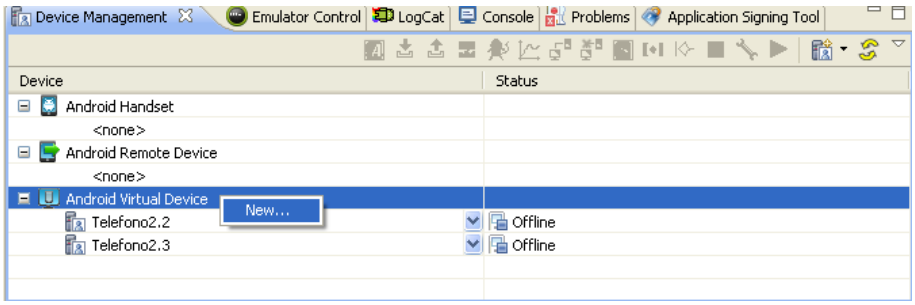
2. Una vez instalado lo ejecutamos, seleccionamos el entorno de trabajo y OK



3. Aparecerá una ventana de configuración, en esta ventana se puede descargar el sdk y las APIs de google si no se las ha descargado antes, pero si ya está descargado se puede seleccionar la ruta donde se encuentra y automáticamente reconoce todas las versiones que se encuentran disponibles

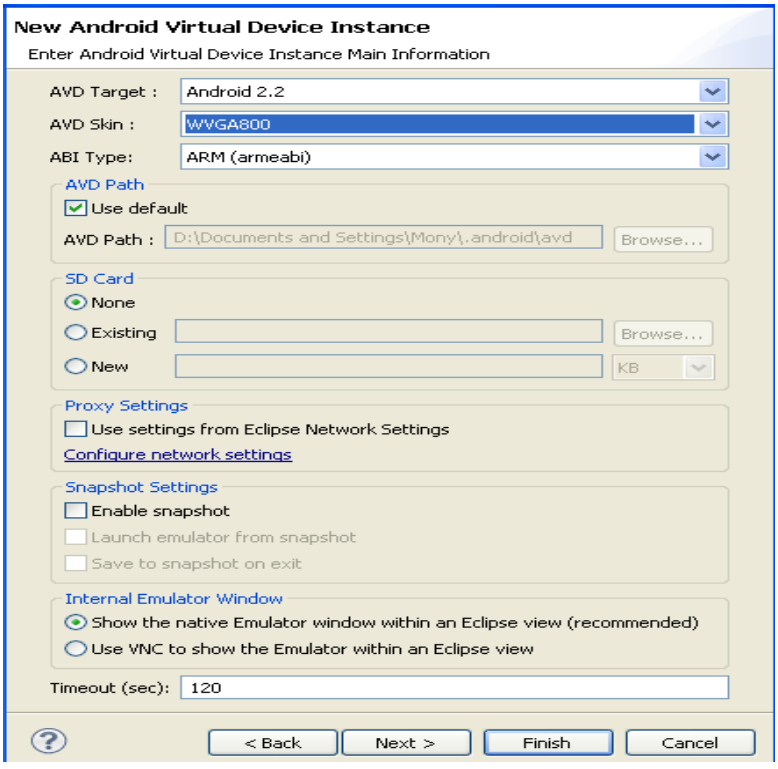


4. Terminando de instalar los paquetes que se necesita, se procede a crear el dispositivo virtual, es decir un emulador de un celular android que servirá para ver y ejecutar las aplicativos y simular como se verían en un equipo real, dar clic *Android Virtual Device-New* en el recuadro de la parte inferior del programa, como se indica en la siguiente pantalla:



En la siguiente pantalla, ingresamos el nombre del dispositivo que puede ser cualquiera:

Aparecerá la siguiente pantalla, en ella se ingresa el nombre que llevará el emulador, en el combo se mostraran todas las versiones que hayamos instalado previamente, seleccionamos la versión que indicada en mi caso la versión 2.2 del SDK.

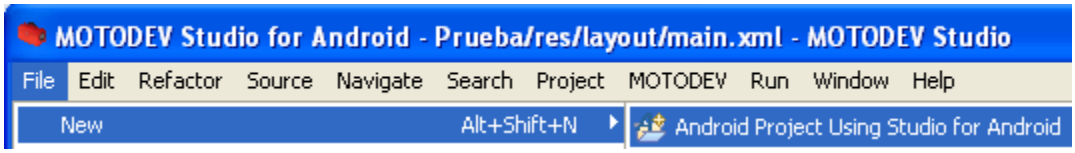


El emulador permite desarrollar y probar aplicaciones Android sin necesidad de utilizar un dispositivo físico. El dispositivo móvil virtual que se ejecuta en el computador, tiene una interfaz como la siguiente:

Figura 3.10: Emulador virtual AVD
Fuente: Propia

3.3 CREAR UN PROYECTO

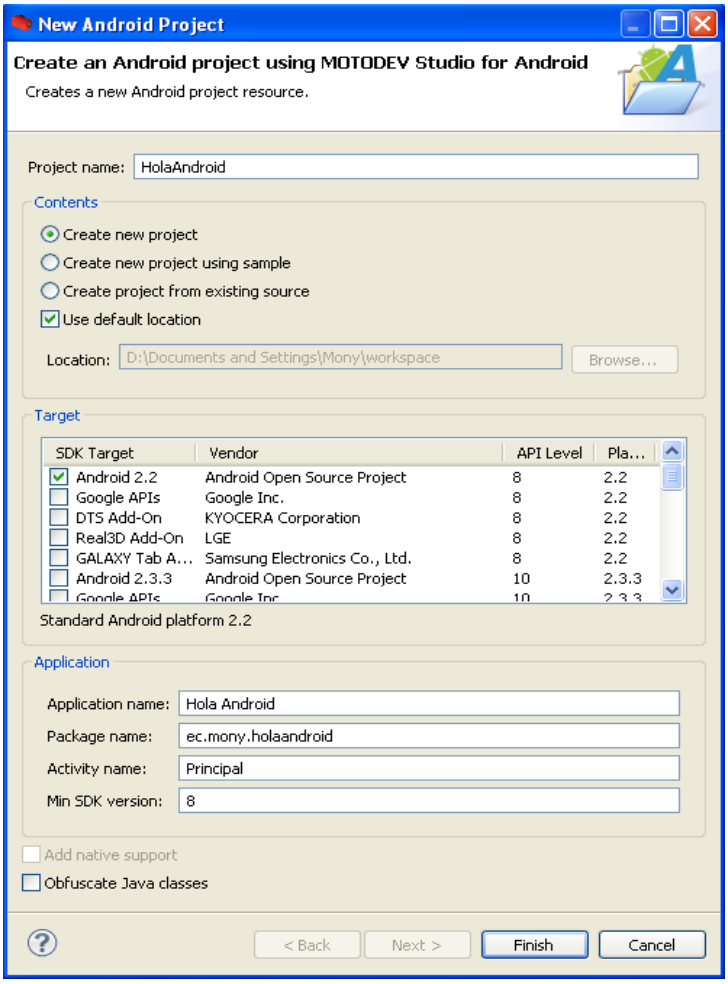
Para confeccionar un programa en MotodevStudio, es necesario crear un proyecto, para hacerlo elegimos la ruta *"File – New – Android Project Using Studio for Android"*.



Aparecerá una ventana donde se escribirá la siguiente información:

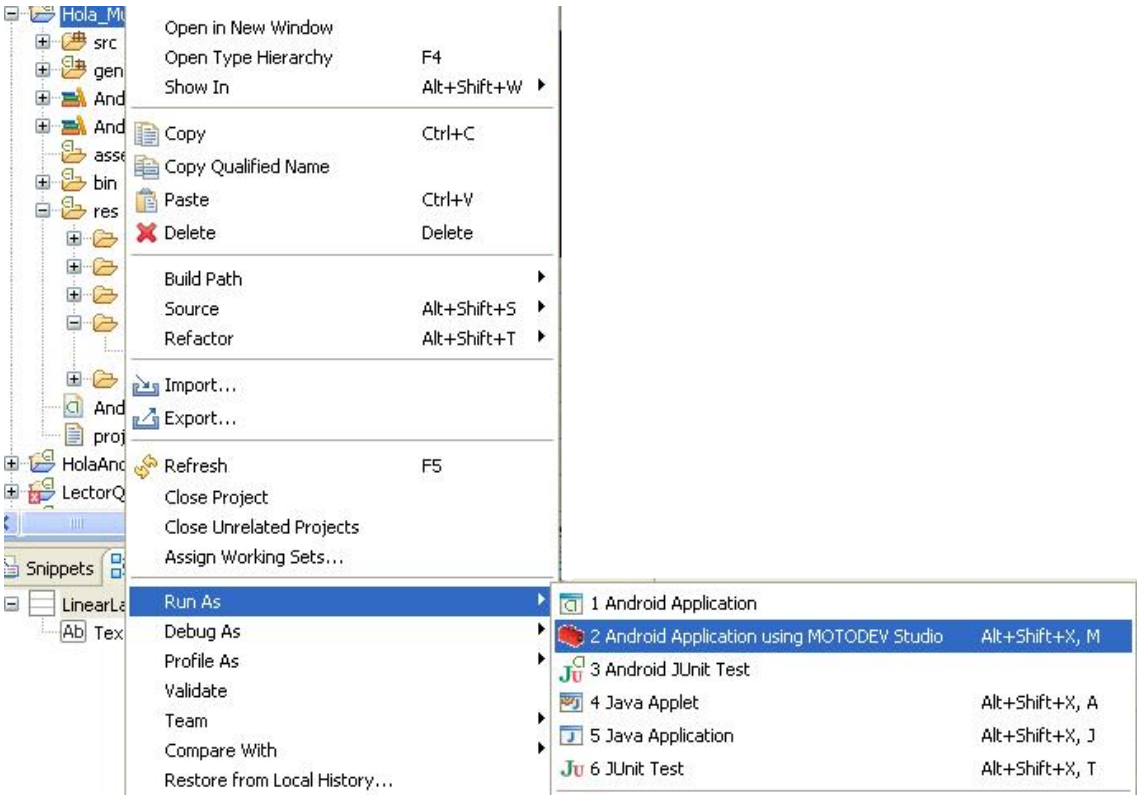
- **Project Name:** Hola Andriod

- **Target:** Android 2.2
- **Application name:** Hola, Android
- **Package name:** ec.mony.holaandroid
- **Activity name:** Principal
- **Min SDK Version**: 8

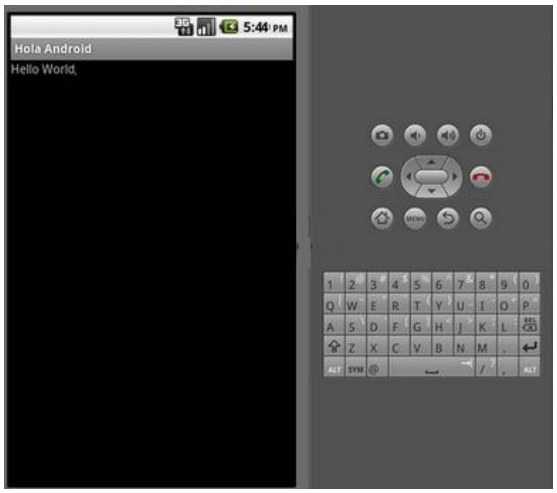


Por último presionamos **"Finish"** para crear el proyecto. Esto creará una aplicación ejemplo donde mostrará una ventana con un mensaje. Más adelante veremos la construcción y modificación de esta estructura.

Para hacerlo funcionar. Damos clic derecho en el proyecto, escogemos **Run As -> Android Aplication using Motodev Studio.**



Se abrirá la ventana del Emulador y luego de cargar del Sistema Operativo, se verá como aparece la aplicación con una ventana de bienvenida al mundo de Android.



3.4 INTERFAZ DE USUARIO

3.4.1 Layouts

Los *layouts* son elementos no visuales encaminados a controlar la distribución, posición y dimensiones de los controles que se insertan en su interior. Estos componentes extienden a la clase base ViewGroup, como muchos otros componentes contenedores, a continuación se detallan los diferentes tipos de layout:

3.4.1.1 FrameLayout

Un FrameLayout coloca todos sus controles hijos alineados con su esquina superior izquierda, de forma que cada control quedará oculto por el control siguiente (a menos que éste último esté transparencia). Se lo utiliza para mostrar un único control en su interior, a modo de contenedor (*placeholder*) sencillo para un sólo elemento sustituible, por ejemplo una imagen. [**WEB 20**]

Los componentes incluidos en un FrameLayaut podrán establecer sus propiedades *android:layout_width y android:layout_height*, que podrán tomar los valores "**fill_parent**" (para que el control hijo tome la dimensión de su layout contenedor) o "**wrap_content**" (para que el control hijo tome la dimensión de su contenido). [**WEB 20**]

Ejemplo:

```
<FrameLayout xmlns:android=
      "http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">
      <EditText android:id="@+id/txtTitulo"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"/>
</FrameLayout>
```

Figura 3.11: FrameLayout

Fuente: Propia

3.4.1.2 LinearLayout

Es el siguiente layout en cuanto a nivel de complejidad, este apila uno tras otro todos sus elementos hijos de forma horizontal o vertical según se establezca su propiedad android:orientation. [**WEB 20**]

Al igual que en un FrameLayout, los elementos contenidos en un LinearLayout pueden establecer sus propiedades **android: layout_width** y **android:**

layout_height para determinar sus dimensiones dentro del layout. Pero en el caso de un LinearLayout, tendremos otro parámetro, la propiedad **android: layout_weight.** [**WEB 20**]

Ejemplo:

```
<LinearLayout xmlns:android=
      "http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="vertical">
      <EditText android:id="@+id/txtTitulo"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"/>
      <EditText android:id="@+id/btnAceptar"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"/>
</LinearLayout>
```

Esta propiedad va a permitir dar a los elementos contenidos en el layout unas dimensiones proporcionales entre ellas. Esto es más difícil de explicar que de comprender con un ejemplo. Si incluimos en un LinearLayout vertical dos cuadros de texto (EditText) y a uno de ellos le establecemos un layout_weight=”1″ y al otro un layout_weight=”2″ conseguiremos como efecto que toda la superficie del layout quede ocupada por los dos cuadros de texto y que además el segundo sea el doble (relación entre sus propiedades weight) de alto que el primero. [**WEB 20**]

Ejemplo:

```
<LinearLayout xmlns:android=
      "http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="vertical">
      <EditText android:id="@+id/txtTitulo1"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
      <EditText android:id="@+id/txtTitulo2"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="2"/>
</LinearLayout>
```

Figura 3.12: LinearLayout
Fuente: Propia

3.4.1.3 TableLayout

Permite distribuir sus elementos hijos de forma tabular, definiendo las filas y columnas necesarias, y la posición de cada componente dentro de la tabla. La estructura de la tabla se define de manera similar a HTML, o sea, indicando las filas que compondrán la tabla (objetos **TableRow**), y dentro de cada fila las columnas necesarias, con la salvedad de que no existe ningún objeto

especial para definir una columna, sino que directamente insertaremos los controles necesarios dentro del TableRow y cada componente insertado (que puede ser un control sencillo o incluso otro **ViewGroup**) corresponderá a una columna de la tabla. De esta forma, el número final de filas de la tabla se corresponderá con el número de elementos TableRow insertados, y el número total de columnas quedará determinado por el número de componentes de la fila que más componentes contenga. [**WEB 20**]

Por norma general, el ancho de cada columna se corresponderá con el ancho del mayor componente de dicha columna, pero existen una serie de propiedades que nos ayudarán a modificar este comportamiento:

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

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

- **android: collapseColumns.** Indicará las columnas de la tabla que se quieren ocultar completamente.

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

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

Ejemplo:

```
<TableLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:stretchColumns="1">
      <TableRow>

            <Button
                android:id="@+id/btnCelda1.1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Celda 1.1" />
            <Button
                android:id="@+id/btnCelda1.2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Celda 1.2" />
      </TableRow>
      <TableRow>
            <Button
                android:id="@+id/btnCelda2.1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Celda 2.1" />
            <Button
                android:id="@+id/btnCelda2.2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Celda 2.2" />
      </TableRow>
      <TableRow>
          <Button
                android:id="@+id/btnCelda3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Celda 3"
                  android:layout_span="2"/>
      </TableRow>
</TableLayout>
```

Figura 3.13: TableLayout
Fuente: Propia

3.4.1.4 RelativeLayout

Este layout permite especificar la posición de cada elemento de forma relativa a su elemento padre o a cualquier otro elemento incluido en el propio layout. De esta forma, al incluir un nuevo elemento *X* podremos indicar por ejemplo que debe colocarse debajo del elemento *Y* y alineado a la derecha del layout padre. [**WEB 20**]

Ejemplo:

```
<RelativeLayoutxmlns:android=
      "http://schemas.android.com/apk/res/android
android:layout_width="fill_parent"
android:layout_height="fill_parent">
      <EditText android:id="@+id/txtTitulo"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"/>
      <Button android:id="@+id/btnAceptar"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/txtNombre"
      android:layout_alignParentRight="true"/>
</RelativeLayout>
```

Figura 3.14: RelativeLayout
Fuente: Propia

En el ejemplo, el botón btnAceptar se colocará debajo del cuadro de texto txtTitulo (android:layout_below="@id/TxtNombre") y alineado a la derecha del layout padre (android:layout_alignParentRight="true"), además de dejar un margen a su izquierda de 10 pixeles (android:layout_marginLeft="10px").

En un RelativeLayout tenemos varias propiedades para colocar cada control donde deseemos. Las principales son:

Tipo	**Propiedades**
Posición relativa a otro control	android:layout_above. android:layout_below. android:layout_toLeftOf. android:layout_toRightOf. android:layout_alignLeft. android:layout_alignRight. android:layout_alignTop. android:layout_alignBottom. android:layout_alignBaseline
Posición relativa al layout padre	android:layout_alignParentLeft. android:layout_alignParentRight.

	android:layout_alignParentTop. android:layout_alignParentBottom. android:layout_centerHorizontal. android:layout_centerVertical. android:layout_centerInParent
Opciones de margen (también disponibles para el resto de layouts)	android:layout_margin. android:layout_marginBottom. android:layout_marginTop. android:layout_marginLeft. android:layout_marginRight
Opciones de espaciado o padding (también disponibles para el resto de layouts)	android:padding. android:paddingBottom. android:paddingTop. android:paddingLeft. android:paddingRight

Tabla 3.3: Propiedades RelativeLayout
Fuente: Propia

3.4.2 Botones

Aquí se va a centrar en los diferentes tipos de botones y cómo podemos personalizarlos. El SDK de Android nos proporciona tres tipos de botones: el clásico (Button), el de tipo on/off (ToggleButton), y el que puede contener una imagen (ImageButton). En la imagen siguiente vemos el aspecto por defecto de estos tres controles. [**WEB 21**]

Figura 3.15: Interfaz visual Botones
Fuente: Propia

3.4.2.1 Control Button

Un control de tipo Button es el botón más básico que se puede utilizar. En el ejemplo siguiente definimos un botón con el texto "Presioname" asignando su propiedad android:text. Además de esta propiedad se podría utilizar muchas otras como el color de fondo (android:background), estilo de fuente (android:typeface), color de fuente (android:textcolor), tamaño de fuente (android:textSize), etc. [**WEB 21**]

```
<Buttonandroid:id="@+id/btnBoton1"
      android:text=""@string/presioname"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>
```

3.4.2.2 Control ToggleButton

Un control de tipo ToggleButton es un tipo de botón que puede permanecer en dos estados, encendido/apagado. En este caso, en vez de definir un sólo texto para el control se define dos, dependiendo de su estado. Así, se puede asignar las propiedades android:textOn y android:textoOff para definir ambos textos. Veamos un ejemplo a continuación. [**WEB 21**]

```
<ToggleButton
      android:id="@+id/btnBoton2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textOff="OFF"
      android:textOn="ON"/>
```

3.4.2.3 Eventos de un botón

El más común de todos los eventos es el evento onClick, para definir la lógica de este evento tendremos que implementarla definiendo un nuevo objeto View.OnClickListener() y asociándolo al botón mediante el método setOnClickListener(). La forma más habitual de hacer esto es la siguiente: [**WEB 21**]

```
final Button btnBoton1 =
(Button)findViewById(R.id.btnBoton1);
btnBoton1.setOnClickListener(new View.OnClickListener() {
@Override
publicvoid onClick(View arg0)
{
lblMensaje.setText("Se ha presionado el boton");
   }
});
```

En un botón de tipo ToggleButton es de utilidad conocer en qué estado ha quedado el botón tras ser pulsado, para lo que podemos utilizar su método isChecked(). En el siguiente ejemplo se comprueba el estado del botón tras ser pulsado y se realizan acciones distintas según el resultado. [**WEB 21**]

```
final ToggleButton btnBoton2 =
(ToggleButton)findViewById(R.id.btnBoton2);
btnBoton2.setOnClickListener(new View.OnClickListener() {
@Override
publicvoid onClick(View arg0)
     {
if(btnBoton2.isChecked())
            lblMensaje.setText("Botón 2: ON");
else
             lblMensaje.setText("Botón 2: OFF");
}
     });
```

3.4.2.4 Personalizar el aspecto un botón

Para cambiar la forma de un botón se podría asignar una imagen a la propiedad android:background, pero esta solución no nos serviría de mucho porque

siempre se mostraría la misma imagen incluso con el botón pulsado, dando poca sensación de elemento "clickable". [**WEB 21**]

La solución perfecta pasaría por tanto por definir diferentes imágenes de fondo dependiendo del estado del botón. Pues bien, Android nos da total libertad para hacer esto mediante el uso de *selectores*. Un *selector* se define mediante un fichero XML localizado en la carpeta /res/drawable, y en él se pueden establecer los diferentes valores de una propiedad determinada de un control dependiendo de su estado. [**WEB 21**]

Por ejemplo, si se quiere dar un aspecto plano a un botón ToggleButton, se puede diseñar las imágenes necesarias para los estados "pulsado" (en el ejemplo *toggle_on.png*) y "no pulsado" (en el ejemplo *toggle_off.png*) y crear un selector como el siguiente:

```
<selector
      xmlns:android=
      "http://schemas.android.com/apk/res/android">
      <itemandroid:state_checked="false"
android:drawable="@drawable/toggle_off"/>
      <itemandroid:state_checked="true"
      android:drawable="@drawable/toggle_on"/>
</selector>
```

Este selector lo guardamos por ejemplo en un fichero llamado toggle_style.xml y lo colocamos como un recurso más en nuestra carpeta de recursos /res/drawable. Ahora hacemos referencia a este nuevo recurso que hemos creado en la propiedad android:background del botón:

```
<ToggleButtonandroid:id="@+id/btnBoton4"
      android:textOn="ON"
      android:textOff="OFF"
      android:padding="10dip"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
android:background="@drawable/toggle_style"/>
```

En la siguiente imagen se ve el aspecto por defecto de un ToggleButton y como ha quedado el ToggleButton personalizado.

3.4.3 Imágenes, etiquetas y cuadros de texto

Aquí vamos a ver tres componentes básicos imprescindibles en las aplicaciones: las imágenes (**ImageView**), las etiquetas (**TextView**) y por último los cuadros de texto (**EditText**). [**WEB 22**]

3.4.3.1 ControlImageView

El control ImageView permite mostrar imágenes en la aplicación. La propiedad más relevante es **android:src**, que permite indicar la imagen a mostrar. Nuevamente, lo normal será indicar como origen de la imagen el identificador de un recurso de nuestra carpeta /res/drawable, por ejemplo android:src=”@drawable/imagen”. Además de esta propiedad, existen algunas otras útiles en algunas ocasiones como las destinadas a establecer el tamaño máximo que puede ocupar la imagen, android:maxWidth y android:maxHeight. [**WEB 22**]

```
<ImageViewandroid:id="@+id/imgFoto"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/icon"/>
```

En la lógica de la aplicación, podríamos establecer la imagen mediante el método setImageResorce(…), pasándole el ID del recurso a utilizar como contenido de la imagen. [**WEB 22**]

```
ImageView img= (ImageView)findViewById(R.id.ImgFoto);
img.setImageResource(R.drawable.icon);
```

3.4.3.2 Control TextView

El control TextView es otro de los clásicos en la programación de GUIs, las etiquetas de texto, y se utiliza para mostrar un determinado texto al usuario. Al igual que en el caso de los botones, el texto del control se establece mediante la propiedad android:text. A parte de esta propiedad, la naturaleza del control hace que las más interesantes sean las que establecen el formato del texto mostrado, que al igual que en el caso de los botones son las siguientes: android:background (color de fondo), android:textColor (color del texto), android:textSize (tamaño de la fuente) y android:typeface (estilo del texto: negrita, cursiva, …). [**WEB 22**]

```
<TextView android:id="@+id/txtEtiqueta"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Texto..."
      android:background="#A4C639"
      android:textAppearance=
      "?android:attr/textAppearanceMedium"
      android:typeface="serif"/>
```

También se puede manipular estas propiedades desde código. Como ejemplo, en el siguiente fragmento recuperamos el texto de una etiqueta con getText(), y posteriormente le concatenamos unos números, actualizamos su contenido mediante setText() y le cambiamos su color de fondo con setBackgroundColor().[**WEB 22**]

```
final TextView txtEtiqueta = (TextView)findViewById(R.id.
txtEtiqueta);
String texto = lblEtiqueta.getText().toString();
texto += "El contenido va aqui";
txtEtiqueta.setText(texto);
```

3.4.3.3 EditText

El control EditText es el componente de edición de texto que proporciona la plataforma Android. Permite la introducción y edición de texto por parte del usuario, por lo que en tiempo de diseño la propiedad más interesante a establecer, además de su posición/tamaño y formato, es el texto a mostrar, atributo android:text. [**WEB 22**]

```
<EditTextandroid:id="@+id/etxtTexto"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
android:layout_below="@id/txtEtiqueta"/>
```

De igual forma, desde nuestro código podremos recuperar y establecer este texto

```
final EditText etxtTexto =
            (EditText)findViewById(R.id.etxtTexto);
 String texto = etxtTexto.getText().toString();
 etxtTexto.setText("Escribe algo!");
```

3.4.4 Checkbox y RadioButton

Aquí se va a ver cómo utilizar otros dos tipos de controles básicos en muchas aplicaciones, los *checkboxes* y los *radio buttons*. [**WEB 23**]

3.4.4.1 Control CheckBox

Un control *checkbox* se acostumbra utilizar para marcar o desmarcar opciones en una aplicación, y en Android está representado por la clase del mismo nombre, CheckBox. La forma de definirlo en la interfaz y los métodos disponibles para manipularlos desde el código son análogos a los ya comentados para el control ToggleButton. [**WEB 23**]

De esta forma, para definir un control de este tipo en el *layout* se puede utilizar el código siguiente, que define un *checkbox* con el texto "Pulsame":

```
<CheckBoxandroid:id="@+id/chPulsame"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/pulsame"/>
```

En cuanto a la personalización del control extiende indirectamente del control TextView, por lo que todas las opciones de formato son válidas también para este control.

En el código de la aplicación se usan los métodos isChecked() para conocer el estado del control, y setChecked(*estado*) para establecer un estado concreto para el control.

```
if (checkBox.isChecked()) {
      checkBox.setChecked(false);
}
```

En lo que tiene que ver a los eventos que puede lanzar este control, el más interesante es el que informa de que ha cambiado el estado del control, que recibe el nombre de onCheckedChanged. Para implementar las acciones de este evento se puede utilizar la siguiente lógica:

```
final CheckBox cb =(CheckBox)findViewById(R.id.chPulsame);
cb.setOnCheckedChangeListener(new OnCheckedChangeListener()
{

      @Override
      publicvoid onCheckedChanged(CompoundButton
      buttonView, boolean isChecked) {
            if (isChecked) {
                cb.setText("Checkbox Pulsado");
            }
else {
                 cb.setText("Checkbox No Pulsado");
            }
      }
});
```

3.4.4.2 Control RadioButton

Al igual que los controles *checkbox*, un *radiobutton* puede estar marcado o desmarcado, pero en este caso suelen utilizarse dentro de un grupo de opciones donde una, y sólo una, de ellas debe estar marcada obligatoriamente, es decir, que si se marca una de ellas se desmarcará automáticamente la que estuviera activa anteriormente. En Android, un grupo de botones *radiobutton* se define mediante un elemento RadioGroup, que a su vez contendrá todos los elementos RadioButton necesarios. A continuación un ejemplo de cómo definir un grupo de dos controles *radiobutton* en nuestra interfaz: [**WEB 23**]

```
<RadioGroupandroid:id="@+id/gruporb"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">
<RadioButtonandroid:id="@+id/rbUno"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/uno"/>
<RadioButtonandroid:id="@+id/rbDos"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
android:text="@string/dos"/>
```

En primer lugar se ve cómo se define el grupo de controles indicando su orientación vertical u horizontal. Luego se añaden todos los objetos RadioButton necesarios indicando su ID mediante la propiedad android:id y su texto mediante android:text. [**WEB 23**]

Una vez definida la interfaz se procede a manipular el control desde el código java haciendo uso de los diferentes métodos del control RadioGroup, los más importantes: check(id) para marcar una opción determinada mediante su ID, clearCheck() para desmarcar todas las opciones, y getCheckedRadioButtonId() que como su nombre indica devolverá el ID de la opción marcada (o el valor -1 si no hay ninguna marcada). Ejemplo:

```
final RadioGroup rb = (RadioGroup)findViewById(R.id.gruporb);
rb.clearCheck();
rb.check(R.id.rbUno);
int rbSeleccionado = rb.getCheckedRadioButtonId();
```

En cuanto a los eventos lanzados, al igual que en el caso de los checkboxes, el más importante será el que informa de los cambios en el elemento seleccionado, llamado también en este caso onCheckedChange. Vemos cómo tratar este evento del objeto RadioGroup:

```
final TextView txtMensaje=(TextView)findViewById(R.id.txtMen);
final RadioGroup rb = (RadioGroup)findViewById(R.id.gruporb);
rb.setOnCheckedChangeListener(
new RadioGroup.OnCheckedChangeListener() {
publicvoid onCheckedChanged(RadioGroup group, int checkedId) {
      txtMensaje.setText("ID opcion seleccionada: " + checkedId);
     }
});
```

Lo que se ha hecho anteriormente nos quedará de la siguiente manera:

3.4.5 Lista Desplegables

Al igual que en otros *frameworks* Android dispone de diversos controles que nos permiten seleccionar una opción dentro de una lista de posibilidades. Así, podremos utilizar listas desplegables (Spinner), listas fijas (ListView), tablas (GridView) y otros controles específicos de la plataforma como por ejemplo las galerías de imágenes (Gallery), a continuación se va a describir un elemento importante y común a todos ellos, los *adaptadores*. [**WEB 23**]

3.4.5.1 Adaptadores en Android (adapters)

Un adaptador representa una interfaz común al modelo de datos que existe por detrás de todos los controles de selección que se ha comentado anteriormente. O

sea, todos los controles de selección accederán a los datos que contienen a través de un adaptador. [**WEB 24**]

Además de proveer de datos a los controles visuales, el adaptador también será responsable de generar a partir de estos datos las vistas específicas que se mostrarán dentro del control de selección. Por ejemplo, si cada elemento de una lista estuviera formado a su vez por una imagen y varias etiquetas, el responsable de generar y establecer el contenido de todos estos "*sub-elementos*" a partir de los datos será el propio adaptador. [**WEB 24**]

Android proporciona varios tipos de adaptadores sencillos, los más comunes son los siguientes:

- ArrayAdapter. Se utiliza para preveer datos a un control de selección a partir de un array de objetos de cualquier tipo.
- SimpleAdapter. Se utiliza para mapear datos sobre los diferentes controles definidos en un fichero XML de layout.
- SimpleCursorAdapter. Se utiliza para mapear las columnas de un cursor sobre los diferentes elementos visuales contenidos en el control de selección.

A continuación se va a describir la forma de utilizar un ArrayAdapter con los diferentes controles de selección disponibles. Se va a crear un adaptador de tipo ArrayAdapter para trabajar con un array genérico de java: [**WEB 24**]

```
final String[] datos =
new String[]{"Opcion1"," Opcion2"," Opcion3","
Opcion4"," Opcion5"};
//Creamos el adaptador
ArrayAdapter<String> adaptador =
new ArrayAdapter<String>(this,
      android.R.layout.simple_spinner_item, datos);
```

El adaptador se crea con tres parámetros: [**WEB 24**]

1. El *contexto*, que será una referencia a la actividad donde se crea el adaptador.
2. El ID del *layout* sobre el que se mostrarán los datos del control. En este caso le pasamos el ID de un layout predefinido en Android (android.R.layout.simple_spinner_item), formado únicamente por un control TextView.
3. El *array* que contiene los datos a mostrar.

Con esto ya se ha creado el adaptador para los datos que se va a mostrar y faltaría asignar este adaptador a nuestro control de selección para que muestre los datos en la aplicación.

3.4.5.2 Control Spinner

Las listas desplegables en Android se llaman Spinner. Funcionan de forma similar al de cualquier control de este tipo, el usuario selecciona la lista, se muestra una especie de lista emergente al usuario con todas las opciones disponibles y al seleccionarse una de ellas ésta queda fijada en el control. Para añadir una lista de este tipo a la aplicación se puede utilizar el código siguiente: [**WEB 22**]

```
<Spinner
android:id="@+id/spinner1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"/>
```

Para enlazar el adaptador a este control se utiliza el siguiente código java:

```
final Spinner combo =
(Spinner)findViewById(R.id.spinner1);
adaptador.setDropDownViewResource(
      android.R.layout.simple_spinner_dropdown_item);
combo.setAdapter(adaptador);
```

Empezamos obteniendo una referencia al control a través de su ID. Y en la última línea se asigna el adaptador al control mediante el método setAdapter(). Para personalizar el aspecto de cada elemento en dicha lista emergente tenemos el método setDropDownViewResource(*ID_layout*), al que podemos pasar otro ID de layout distinto al primero sobre el que se mostrarán los elementos de la lista emergente. En este caso se ha utilizado otro layout predefinido an Android para las listas desplegables (android.R.layout.simple_spinner_dropdown_item). [**WEB 24**]

Con estas líneas de código se conseguirá mostrar un control como el que se ve a continuación:

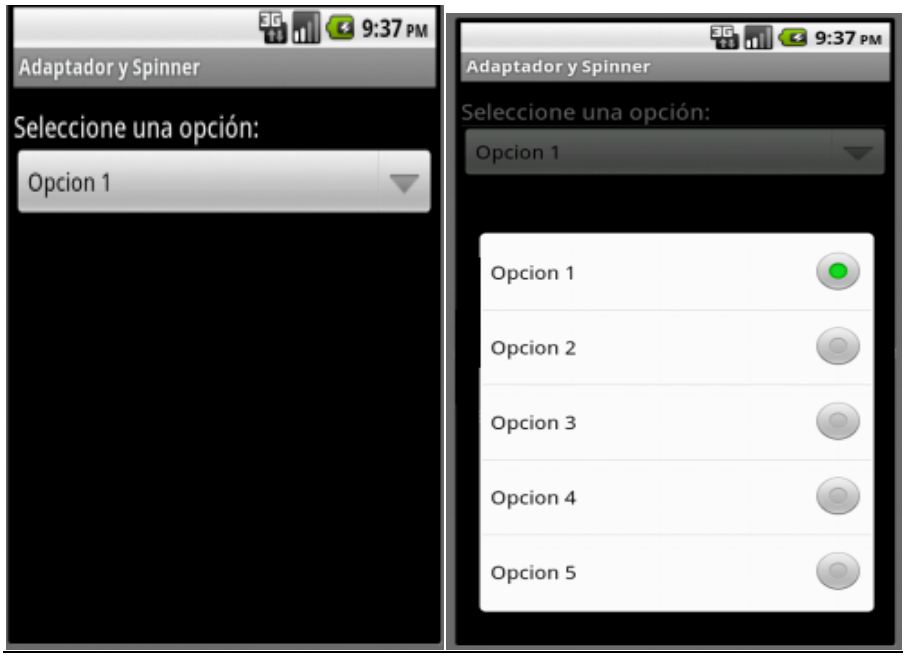


Figura 3.16: Interfaz visual Control Spinner

Fuente: Propia

En cuanto a lo que tiene que ver a los eventos lanzados por el control Spinner, se va a utilizar el generado al seleccionarse una opción de la lista desplegable, onItemSelected, asignándole su controlador mediante el método setOnItemSelectedListener():[**WEB 24**]

```
combo.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
//onItemSelected sera llamado cada vez que se seleccione
//una opción en la lista desplegable
publicvoid onItemSelected(AdapterView<?> parent,
android.view.View v, int posicion, long id) {
      txtMensaje1.setText("Ha escogido: " + datos[posicion]);
      }
      //onNothingSelected sera llamado cuando
//no haya ninguna opción seleccionada
//esto puede ocurrir, si el adaptador no tiene datos
publicvoid onNothingSelected(AdapterView<?> parent) {
txtMensaje1.setText("");
  }
});
```

3.4.6 Listas

Un control ListView muestra al usuario una lista de opciones seleccionables directamente sobre el propio control, sin listas emergentes como en el caso del control Spinner. En caso de existir más opciones de las que se pueden mostrar sobre el control se podrá hacer scroll sobre la lista para acceder al resto de elementos. [**WEB 25**]

Primero empezamos añadiendo un control ListView a nuestra interfaz de usuario:

```
<ListViewandroid:id="@+id/listaOpciones"
          android:layout_width="wrap_content"
     android:layout_height="wrap_content"/>
```

Para enlazar los datos con el control se puede utilizar el mismo código que ya se vio para el control Spinner. Primero se definirá un array con los datos de prueba, y luego se creará el adaptador de tipo ArrayAdapter y se lo asignará al control mediante el método setAdapter():[**WEB 25**]

```
final String[] lista =new
String[]{"Opcion1","Opcion2","Opcion3","Opcion4","Opcion5"};
            ArrayAdapter<String> adaptador1 = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,
                                lista);
ListView listaOpciones =
(ListView)findViewById(R.id.listaOpciones);
listaOpciones.setAdapter(adaptador1);
```

En este caso, para mostrar los datos de cada elemento se ha utilizado otro layout genérico de Android para los controles de tipo ListView (android.R.layout.simple_list_item_1), formado únicamente por un TextView con unas dimensiones determinadas. La lista creada quedará de la siguiente manera: [**WEB 25**]

Figura 3.17: Interfaz visual Listas

Fuente: Propia

Para realizar cualquier acción al pulsar sobre un elemento de la lista creada se implementará el evento onItemClick:

```
listaOpciones.setOnItemClickListener(new
OnItemClickListener() {
@Override
      publicvoid onItemClick(AdapterView<?> a, View v, int
      position, long id) {
//Aqui ira el codigo necesario
       }
 });
```

3.4.7 Grids

El control GridView presenta al usuario un conjunto de opciones seleccionables divididas en filas y columnas, sus propiedades más importantes son:

Propiedad	**Descripción**
android:numColumns	Indica el número de columnas de la tabla o "auto_fit" para que sea calculado por el propio sistema operativo a partir de las siguientes propiedades
android:columnWidth	Indica el ancho de las columnas de la tabla
android:horizontalSpacing	Indica el espacio vertical entre celdas.
android:stretchMode	Indica qué hacer con el espacio horizontal sobrante. Si se establece al valor "columnWidth" este espacio será absorbido a partes iguales por las columnas de la tabla. Si se establece "spacingWidth" será absorbido a partes iguales por los espacios entre celdas

Tabla 3.4: Propiedades de un GridView
Fuente: [**WEB 26**]

A continuación se va a definir un GridView:

```
<GridView
android:id="@+id/gridOpciones"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:numColumns="3">
</GridView>
```

Una vez definida la interfaz de usuario, se procede a crear un array genérico que contenga los datos de prueba, se declara un adaptador de tipo ArrayAdapter pasándole en este caso un layout genérico (simple_list_item_1, compuesto por un simple TextView) y se asocia el adaptador al control GridView mediante el método setAdapter():

```
final String[] opcion = new String[15];
for(int i=0; i<=15; i++)
    opcion[i] = "Opcion " + i;
 ArrayAdapter<String> adaptador = new ArrayAdapter<String>
       (this, android.R.layout.simple_list_item_1, opcion);
final GridView gridOpciones =
      (GridView)findViewById(R.id.gridOpciones);
gridOpciones.setAdapter(adaptador);
```

Los datos del array se añadirán al control GridView ordenados por filas, y si no caben todos en la pantalla se podrá hacer scroll sobre la tabla. A continuación se ve cómo queda la aplicación de prueba:

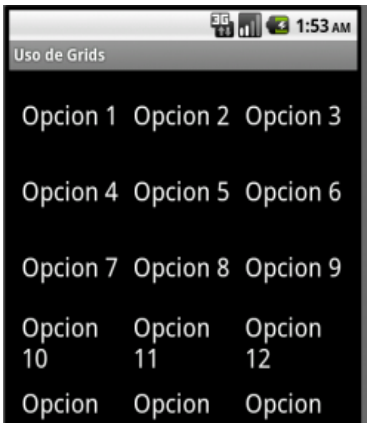


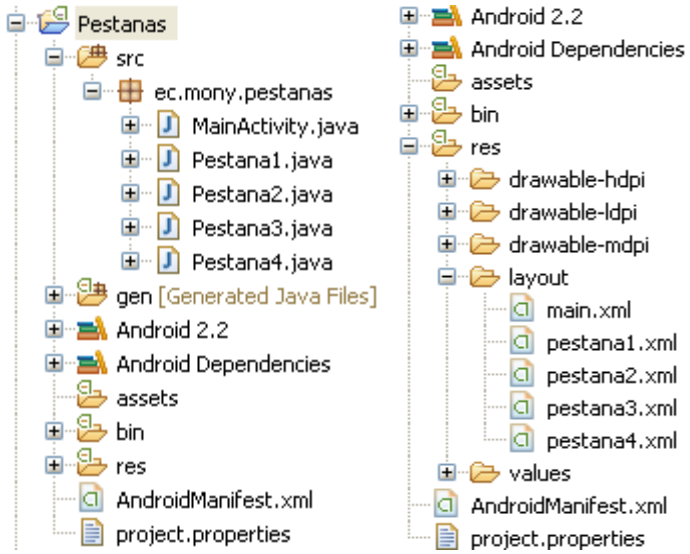
Figura 3.18: Interfaz visual Grids
Fuente: Propia

3.4.8 Pestañas

El elemento principal de un conjunto de pestañas es el control *TabHost*, el cual será el contenedor principal de un conjunto de pestañas. La sección de pestañas se representará mediante un elemento *TabWidget*, y como contenedor para el contenido de las pestañas se añade un *FrameLayout*. [**WEB 27**]

A continuación se desarrollará un pequeño ejemplo en el que se va a utilizar pestañas:

- ✓ Crear un nuevo proyecto llamado Pestanas.
- ✓ Luego crear cuatro actividades que se utilizará en la aplicación nombrándolas Pestana1.class, Pestana2.class, Pestana3.class, Pestana4.class; de igual manera crear el respectivo layout para cada actividad: pestana1.xml, pestana2.xml, pestana3.xml, pestana4.xml, el proyecto al momento tendrá la siguiente forma:



- ✓ Ahora se enlazará cada layout a su respectiva actividad, de la siguiente manera:

```
Public class Pestana1 extends Activity {
      @Override
      publicvoid onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
            setContentView(R.layout.pestana1);
      }
}
```

✓ Ahora se debe vincular cada una de las actividades al Manifiesto para que puedan ser llamadas y trabajar con ellas, así que procedemos a incluirlas en el archivo AndroidManifest.xml de la siguiente manera:

```
<activityandroid:name=".Pestana1"/>
<activityandroid:name=".Pestana2"/>
<activityandroid:name=".Pestana3"/>
<activityandroid:name=".Pestana4"/>
```

✓ Ahora se procede a modificar el layout principal *main.xml* para que contenga los recursos necesarios para utilizar las pestañas en el proyecto, primero se define un*TabHost* que contendrá los elementos principales de la actividad, luego se crea un LinearLayout vertical para contener ordenadamente los 2 elementos principales que son el TabWidget y el FrameLayout de la siguiente manera:

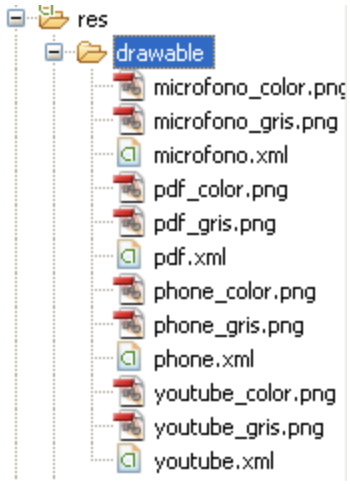
```
<TabHost
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@android:id/tabhost"
android:layout_width="match_parent"
android:layout_height="match_parent">
<LinearLayout
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<TabWidget
android:id="@android:id/tabs"
android:layout_width="match_parent"
android:layout_height="wrap_content"/>
<FrameLayout
android:id="@android:id/tabcontent"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
      </LinearLayout>
</TabHost>
```

- ✓ Se necesita un icono para cada pestaña y dos imágenes para cada icono que representen si la pestaña está seleccionada o no, puede ser una a color y la otra en gris por ejemplo:

- ✓ Primero se copia las dos imágenes en la carpeta res/drawable/ del proyecto (se crea la carpeta si no existe), llamando a la imagen blanca ic_tab_artists_white.png y a la gris ic_tab_artists_grey.png. Ahora tenemos que crear un *Drawable* (que representa algo que puede ser dibujado) en XML que especifique que imagen usar para cada estado. Creamos un nuevo fichero en res/drawable/phone.xml e insertamos el código:

```
<?xmlversion="1.0"encoding="utf-8"?>
<selectorxmlns:android=
      "http://schemas.android.com/apk/res/android">
<itemandroid:drawable="@drawable/phone_color"
            android:state_selected="true"/>
      <itemandroid:drawable="@drawable/phone_gris"/>
</selector>
```

Cuando el estado de la imagen cambie, la imagen se cambiará automáticamente por la otra definida en estos XML, el archivo queda de la siguiente manera:



- ✓ Lo siguiente que se debe hacer es codificar la actividad principal MainActivity.class, esta clase extiende de TabActivity, se crea un TabHost, así como un Tab.Spec llamado spec que contendrá las especificaciones de la pestaña a mostrar, también se crea el intent que se usará para llamar cada una de las 4 actividades ya creadas anteriormente, y se llama a los recursos, los cuales son los iconos de las pestañas y sus respectivos .xml:

- ✓ El resultado del proyecto se lo muestra en la siguiente imagen:

Figura 3.19: Interfaz visual Pestañas
Fuente: Propia

3.4.9 Widgets

Un widget se diferencia de una aplicación en varias cosas en que no debe extender de la clase Activity si no de AppWidgetProvider, tiene ciertas limitaciones respecto de los elementos que puede usar en este, debe tener un tamaño determinado en la pantalla, etc. [**WEB 28**]

Pasos a seguir para crear un widget:

1. Definir la Interfaz gráfica (*layout*) mediante un archivo XML en /res/layout
2. Configuración XML del widget (AppWidgetProviderInfo) en /res/xml/
3. Implementación de la funcionalidad del widget (AppWidgetProvider), especialmente su evento de actualización.
4. Declaración del widget dentro del *Android Manifest* de la aplicación

A continuación se va a crear un widget estático básico, que servirá para entender la estructura interna de este tipo de componente.

El widget a crearse consistente en un simple marco rectangular verde con un mensaje de texto predeterminado ("*Mi Primer Ejemplo, Trabajando con Widget*").

Luego de crear el proyecto, se define la interfaz del widget que estará compuesta por dos *frames* (FrameLayout), uno verde exterior y uno azul interior más pequeño para simular el marco, y una etiqueta de texto (TextView) que contendrá el mensaje a mostrar, el layout xml quedará de la siguiente manera:

```
<FrameLayout
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#19DE5A"
android:padding="10dip">
<FrameLayout
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#0B87BC"
android:padding="5dip">
<TextViewandroid:id="@+id/txtMensaje"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:gravity="center"
android:textSize="14dip"
android:textColor="#FFFFFF"
android:text="@string/mi_primer_widget"/>
</FrameLayout>
</FrameLayout>
```

Debido a que el layout de los widgets de Android está basado en un tipo especial de componentes llamados *RemoteViews*, no es posible utilizar en su interfaz todos los contenedores y controles existentes en android, sino los más básicos que se indican a continuación:

- Contenedores: FrameLayout, LinearLayout y RelativeLayout
- Controles: Button, ImageButton, ImageView, TextView, ProgressBar, Chronometer y AnalogClock.

A continuación se va a crear un nuevo fichero XML donde se va a definir un conjunto de propiedades del widget, como por ejemplo su tamaño en pantalla o su frecuencia de actualización. Este XML se deberá crear en la carpeta *\res\xml*, el cual se llamará m*iwidget_provider.xml:*

```
<?xmlversion="1.0"encoding="utf-8"?>
<appwidget-provider
xmlns:android="http://schemas.android.com/apk/res/android"
android:initialLayout="@layout/main"
android:minWidth="146dip"
android:minHeight="72dip"
android:label="Mi Primer Widget"
android:updatePeriodMillis="3600000"
/>
```

Para este widget se define las siguientes propiedades:

- initialLayout: referencia al layout XML que se ha creado en el paso anterior.
- minWidth: ancho mínimo del widget en pantalla, en dp (*density-independent pixels*).
- minHeight: alto mínimo del widget en pantalla, en dp (*density-independent pixels*).
- label: nombre del widget que se mostrará en el menú de selección de Android.
- updatePeriodMillis: frecuencia de actualización del widget, en milisegundos.

3.4.9.1 Tamaño del Widget

Debe definirse siempre minWidth y minHeight que definen el tamaño mínimo que ocupa en la pantalla por defecto.

La pantalla home de android esta diagramada en forma de grilla, la cantidad de filas y columnas depende del tamaño del display que posea el dispositivo. Puede calcularse el tamaño mediante esta fórmula: [**WEB 28**]

$$minWidth = (num\_celdas * 74) - 2$$

$$minHeight = (num\_celdas * 74) - 2$$

En nuestro caso ocupara dos celdas de alto y una de ancho.

Ahora se debe crear una nueva clase llamada Miwidget, la cual contendrá lo siguiente:

```
package ec.mony.miprimerwidget;

import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.Context;

publicclass MiWidget extends AppWidgetProvider {
      @Override
      publicvoid onUpdate(Context context,
      AppWidgetManager appWidgetManager,
      int[] appWidgetIds) {
      //Aqui ira el codigo para
      //actualizar el widget
          }
}
```

El último paso es declarar el widget dentro del manifest de la aplicación, dentro del elemento <application>:

```
<receiverandroid:name=".MiWidget"android:label="Mi Primer Widget">
      <intent-filter>
            <actionandroid:name=
            "android.appwidget.action.APPWIDGET_UPDATE"/>
      </intent-filter>
      <meta-data
            android:name="android.appwidget.provider"
            android:resource="@xml/miwidget_provider"/>
</receiver>
```

El widget se declarará como un elemento <receiver>, a continuación una breveexplicación de cada uno de sus elementos:

- Atributo name: Referencia a la clase java del widget, en este caso la clase llamada MiWidget.

- Elemento <intent-filter>, donde se indicará los "eventos" a los que responderá el widget, como el evento principal de un widget está definido en el método onUpdate(), se añadirá el evento APPWIDGET_UPDATE, para detectar la acción de actualización.

- Elemento <meta-data>, donde se hace referencia con su atributo *resource* al XML de configuración*widget_provider.xml* que se creó anteriormente.

Por último ejecutamos el proyecto para probar el widget creado. Una vez que el emulador haya ejecutado la aplicación, se tiene que dirigir a la pantalla principal del teléfono, para realizar una pulsación larga sobre el escritorio o tecla Menú, seleccionamos la opción *Widgets*, y escogemos el nombre del widget creado, el resultado será lo siguiente:

Figura 3.20: Interfaz visual Widgets
Fuente: Propia

3.4.10 Menús

Es uno de los componentes importantes que debe incluir tu aplicación. Los teléfonos Android tienen una tecla dedicada a lanzar los menús dentro de una aplicación.

En Android existen tres tipos de menús que se puede utilizar dentro de las aplicaciones: [**WEB 29**]

- **Menús Principales**. Son los más comunes, aparecen en la zona inferior de la pantalla al pulsar el botón 'menu' del teléfono.
- **Submenús**. Son menús secundarios que se pueden mostrar al pulsar sobre una opción de un menú principal.
- **Menús Contextuales**. Útiles en muchas ocasiones, aparecen al realizar una pulsación larga sobre algún elemento de la pantalla.

3.4.10.1 Creando un menú

Primero se debe crear un archivo XML con la definición del menú para después referenciarlos desde el código Java. Manejar los menús de esta forma es una buena práctica porque nos permite separar el contenido del menú del código de nuestra aplicación.

Para crear un recurso de menú, se necesita de un archivo XML que se guardará en el directorio *res* > *menu* con la siguiente estructura: [**WEB 29**]

- ✓ **<menu>** Este elemento debe ser el nodo raíz del archivo. Crea un objeto *Menu* dentro del código. Puede contener uno o más elementos <*item*> y <*group*>.

- ✓ **<item>** Representa un ítem dentro del menú y en la parte del código crea un objeto de tipo *MenuItem*. Este elemento puede contener un elemento <*menu*>anidados que permitirá crear submenús.

- ✓ **<group>** Es un elemento opcional e invisible que se puede anidar dentro de un <*item*>, permite categorizar opciones que compartan propiedades tales como el estado activo/inactivo o visibilidad.

A continuación se va a crear un menú a partir de su definición en XML. Los ficheros XML de menú se deben colocar en la carpeta "res\menu" del proyecto y tendrán una estructura semejante al ejemplo siguiente:

```
<?xmlversion="1.0"encoding="utf-8"?>
<menuxmlns:android="http://schemas.android.com/apk/res/android"
>
      <itemandroid:id="@+id/menu1"android:title="Opcion1"
      android:icon="@drawable/ico"></item>
      <itemandroid:id="@+id/menu2"android:title="Opcion2"
      android:icon="@drawable/ico1"></item>
      <itemandroid:id="@+id/menu3"android:title="Opcion3"
      android:icon="@drawable/ico2"></item>
```

Como se puede observar el elemento raíz es <*menu*>que contendrá una serie de elementos <item> que se corresponderán con las distintas opciones a mostrar en el menú. Para cada uno de ellos hemos definido un *id*, un *icono* que es un recurso de imagen almacenado en el directorio *res* > *drawable* y un *título*, que es el texto que desplegará cada.

Una vez definido el menú en el fichero XML, se procede a implementar el evento onCreateOptionsMenu() de la actividad que se va a mostrar.

```
@Override
publicboolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu1, menu);
returntrue;
}
```

En este evento se debe"*inflar*" el menú. primerose hace una referencia al *inflater* mediante el método getMenuInflater() y luego se genera la estructura del menú llamando a su método infate() pasándole como parámetro el ID del menu definido en XML, que en nuestro caso será R.menu.menu, finalmente se devuelve valor *true* para confirmar que debe mostrarse el menú. El resultado de correr la aplicación será la siguiente:

Figura 3.21: Interfaz visual Menús
Fuente: Propia

Una vez construido el menú, la implementación de cada una de las opciones se incluirá en el evento *onOptionsItemSelected()* de la actividad que mostrará el menú. Este evento recibe como parámetro el item de menú que ha sido pulsado por el usuario, cuyo *ID* se puede recuperar con el método *getItemId()*, según el *ID* se podrá saber qué opción se ha pulsada y ejecutar unas acciones u otras. En este caso muestra un mensaje de texto diferente según la opción que se ha pulsa, se deberá cambiar el contenido de cada *case* según la funcionalidad que se desee dar a la aplicación:

```
@Override
publicboolean onOptionsItemSelected(MenuItem item) {
      switch (item.getItemId()) {
case R.id.menu1:
                  Toast.makeText(getApplicationContext(), "Ha
                  pulsado Opcion 1",
                Toast.LENGTH_SHORT).show();
returntrue;
case R.id.menu2:
                  Toast.makeText(getApplicationContext(), "Ha
                  pulsado Opcion 2",
                Toast.LENGTH_SHORT).show();
returntrue;
case R.id.menu3:
                  Toast.makeText(getApplicationContext(), "Ha
                  pulsado Opcion 3",
                Toast.LENGTH_SHORT).show();
returntrue;
default:
returnsuper.onOptionsItemSelected(item);
}
}
```

Esta es la vista final de la pantalla cuando se pulsa el botón MENÚ:

Por último recordar que se necesita importar los siguientes paquetes para que eso funcione la aplicación:

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.Toast;
```

3.4.11 Gestión de Preferencias

En android se puede almacenar información de varias maneras, para grandes cantidades de datos se dispone de bases de datos mediante sqlite y para guardar pequeñas configuraciones o datos únicos, la plataforma pone a disposición la clase SharedPreferences.

SharedPreferences almacena información accesible desde cualquier actividad de la aplicación, y permanece almacenada a pesar de que se salga de la misma.Se guardan en un archivo XML en forma de **clave-valor**, y se guardan en el sistema de archivos de la siguiente forma:

/data/data/com.namespace.de.nuestra.aplicacion/shared_prefs/nombre_de_nuestras_prefs.xml

Para usar la clase SharedPreferences en Android se debe importar:

import android.content.SharedPreferences;

Toda la gestión se centraliza en la clase *SharedPrefences*, que representará a una colección de preferencias, una aplicación Android puede gestionar varias colecciones de preferencias, que se diferenciarán mediante un identificador único.

Para obtener una referencia a una colección determinada se utiliza el método getSharedPrefences() al que se pasa el identificador de la colección y un *modo de acceso*, el modo de acceso indicará qué aplicaciones tendrán acceso a la colección de

preferencias y que operaciones tendrán permitido realizar sobre ellas, hay tres posibilidades de acceso:

Modo de Acceso	**Descripción**
MODE_PRIVATE	Sólo la aplicación podrá leer y modificar datos de estas preferencias.
MODE_WORLD_READABLE	Sólo la aplicación podrá escribir datos de configuración, el resto de aplicaciones del dispositivo Android podrá leer los datos de configuración pero no modificarlos.
MODE_WORLD_WRITEABLE	Todas las aplicaciones podrán leer y modificar estas preferencias.

Tabla 3.5: Modos de acceso método getSharedPrefences()
Fuente: [**WEB 30**]

Para obtener una referencia a una colección de preferencias llamada por ejemplo "Datos" y como modo de acceso exclusivo para la aplicación se hará lo siguiente:

```
SharedPreferences prefe=
getSharedPreferences("Datos",Context.MODE_PRIVATE);
```

Ya obtenida una referencia a la colección de preferencias, se puede obtener, insertar o modificar preferencias utilizando los métodos get o put correspondientes al tipo de dato de cada preferencia, por ejemplo, para obtener el valor de una preferencia llamada "email" de tipo String se escribe lo siguiente:

```
String mail = prefe.getString("email", "mony@hotmail.com");
```

Al método getString() se le pasa el nombre de la preferencia a recuperar y un segundo parámetro con un valor por defecto, el cual será devuelto por el método getString() si la preferencia solicitada no existe en la colección.

Para actualizar o insertar nuevas preferencias el proceso será igual, con la única diferencia de que la actualización o inserción no se la hará directamente sobre el objeto *SharedPreferences*, sino sobre su objeto de edición *SharedPreferences.Editor*, a este objeto se accede mediante el método edit() de la clase *SharedPreferences*. Ya obtenida la referencia al editor, se utiliza los métodos *put* correspondientes al tipo de datos de cada preferencia para actualizar/insertar su valor, por ejemplo putString(clave, valor), para actualizar una preferencia de tipo String, una vez actualizado/insertado los datos necesarios se debe llamar al método commit() para confirmar los cambios:

```
SharedPreferences prefe =
      getSharedPreferences("Datos",Context.MODE_PRIVATE);
SharedPreferences.Editor editor = prefe.edit();
editor.putString("correo", "monica@hotmail.com");
editor.putString("nombre", "ejemplo");
editor.commit();
```

Estos ficheros XML se almacenan en una:

/data/data/**ec.mony.preferencias**/shared_prefs/**Datos**.xml

Si se abre este fichero con cualquier editor de texto mostrará un contenido como el siguiente:

```
<?xmlversion='1.0'encoding='utf-8'standalone='yes'?>
<map>
    <stringname="nombre">ejemplo</string>
    <stringname="correo">monica@hotmail.com</string>
</map>
```

3.4.12 Base de Datos SQLite

SQLite es un motor open source de bases de datos ligero, ideal para entornos móviles, ofrece características necesarias como poca memoria, no necesita servidor, precisa poca configuración y es transaccional.

Android incorpora todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, entre ellas una completa API para llevar a cabo de manera sencilla todas las tareas necesarias. En Android, la forma típica para crear, actualizar, y conectar con una base de datos SQLite es a través de una clase auxiliar llamada *SQLiteOpenHelper*, o sea de una clase propia que derive de ella y que se debe personalizar para adaptarse a las necesidades concretas de la aplicación.

La clase *SQLiteOpenHelper* tiene un constructor, que normalmente no se necesita sobrescribir, y dos métodos abstractos, *onCreate()* y *onUpgrade()*, que se debe personalizar con el código necesario para crear la base de datos y para actualizar su estructura respectivamente.

Como ejemplo, se va a crear una base de datos llamada BDLibros, con una tabla llamada Libros que contendrá sólo cuatro campos: código, titulo, autor y año. La aplicación debe permitir:

- ✓ Ingreso de Libros.
- ✓ Consulta de Libros por código
- ✓ Eliminación de Libros
- ✓ Actualización de datos de los Libros.

Primero se creará una clase derivada de SQLiteOpenHelper que se llamará*AdminBDDOpenHelper,* donde se sobrescribe los métodos *onCreate()* y *onUpgrade()* para que se adapten a la estructura de datos indicada anteriormente:

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

publicclass AdminBDDOpenHelper extends SQLiteOpenHelper {
      //Sentencia SQL para crear la tabla de Libros
String sqlLibro =
      "CREATE TABLE Libros (codigo int primary key, titulo text,
      autor text, ano int)";
      public AdminBDDOpenHelper(Context context, String nombre,
      CursorFactory factory, int version) {
            super(context, nombre, factory, version);
      }

      @Override
      publicvoid onCreate(SQLiteDatabase db) {
            //Se ejecuta la sentencia SQL para crear la tabla
db.execSQL(sqlLibro);
      }

      @Override
      publicvoid onUpgrade(SQLiteDatabase db, int versionAnte,
      int versionNue) {
      //Se elimina la versión anterior de la tabla
db.execSQL("DROP TABLE IF EXISTS Libros");
//Se crea la nueva versión de la tabla
db.execSQL(sqlLibro);
      }
}
```

Primero se define una variable llamado sqlLibro donde se almacena la sentencia SQL para crear la tabla llamada Libros con los campos mencionados.

El método onCreate() será ejecutado automáticamente por la clase AdminBDDDBHelper cuando sea necesaria la creación de la base de datos. Las tareas que se hacen en este método serán la creación de todas las tablas necesarias y la inserción de los datos iniciales si son necesarios. Para la creación de la tabla se usa la sentencia SQL ya definida y la se la ejecuta contra la base de datos utilizando el método más común de los disponibles en la API de SQLite

proporcionada por Android, llamado execSQL(). Este método se limita a ejecutar directamente el código SQL que se le pasa como parámetro.

El método onUpgrade() se lanzará automáticamente cuando se llama a una actualización de la estructura de la base de datos o una conversión de los datos.

Una vez definida la clase *helper*, se procede a crear un objeto de la clase *AminBDDOpenHelper*al que se lo pasará el contexto de la aplicación (en el ejemplo una referencia a la actividad principal), el nombre de la base de datos, un objeto CursorFactory que no será necesario (en ese caso se pasa el valor null), y por último la versión de la base de datos que se necesitará.

Una vez que se tiene una referencia al objeto UsuariosSQLiteHelper, se llama a su método *getReadableDatabase()* o *getWritableDatabase()* para obtener una referencia a la base de datos, dependiendo de si se desea consultar los datos o también realizar modificaciones, respectivamente.

La clase principal que implementará las inserciones, eliminaciones, actualizaciones y consultas quedará de la siguiente manera:

```
import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

publicclass MainActivity extends Activity {
      private EditText etxtcod,etxtTitulo,etxtAutor,etxtAnio;

@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

etxtcod=(EditText)findViewById(R.id.etxtCodigo);
etxtTitulo=(EditText)findViewById(R.id.etxtTitulo);
etxtAutor=(EditText)findViewById(R.id.etxtAutor);
etxtAnio=(EditText)findViewById(R.id.etxtAno);
}

//Aqui va le codigo que se encuentra debajo
```

```
//METODO PARA INGRESAR NUEVOS LIBROS
publicvoid IngresarLibro(View v) {
AdminBDDOpenHelper admin=new AdminBDDOpenHelper(this,
"administracion", null, 1);
        SQLiteDatabase bd=admin.getWritableDatabase();
        String codigo=etxtcod.getText().toString();
        String titulo=etxtTitulo.getText().toString();
        String autor=etxtAutor.getText().toString();
        String anio=etxtAnio.getText().toString();
ContentValues registro=newContentValues();
registro.put("codigo", codigo );
registro.put("titulo",titulo );
registro.put("autor",autor );
registro.put("ano",anio );
bd.insert("libros", null, registro);
bd.close();
etxtcod.setText("");
etxtTitulo.setText("");
etxtAutor.setText("");
etxtAnio.setText("");
        Toast.makeText(this, "Nuevo Libro Ingresado",
            Toast.LENGTH_SHORT).show();
    }
```

```
//METODO PARA ELIMINAR LOS LIBROS CREADOS
publicvoid eliminarLibro(View v) {
        AdminBDDOpenHelper admin=newAdminBDDOpenHelper(this,
            "administracion", null, 1);
        SQLiteDatabase bd=admin.getWritableDatabase();
        String codigo=etxtcod.getText().toString();
int cant=bd.delete("libros", "codigo="+codigo+"",null);
bd.close();
etxtcod.setText("");
etxtTitulo.setText("");
etxtAutor.setText("");
etxtAnio.setText("");
if (cant==1)
Toast.makeText(this, "Se eliminó el Libro dicho codigo",
            Toast.LENGTH_SHORT).show();
else
             Toast.makeText(this, "No existe el Libro con
            dicho codigo",Toast.LENGTH_SHORT).show();
}
```

```
//METODO PARA CONSULTAR UN LIBRO POR CODIGO
publicvoid consultarLibro(View v) {
        AdminBDDOpenHelper admin=new AdminBDDOpenHelper(this,
            "administracion", null, 1);
        SQLiteDatabase bd=admin.getWritableDatabase();
        String codigo=etxtcod.getText().toString();
Cursor fila=bd.rawQuery
            ("select titulo,autor,ano  from libros where
            codigo="+codigo+"",null);
if (fila.moveToFirst())
        {
etxtTitulo.setText(fila.getString(0));
etxtAutor.setText(fila.getString(1));
etxtAnio.setText(fila.getString(2));
        }
else
             Toast.makeText(this, "No existe el Libro con dicho
            codigo", Toast.LENGTH_SHORT).show();
        bd.close();
}
```

```
//METODO PARA ACTUALIZAR UN LIBRO
publicvoid actualizarLibro(View v) {
AdminBDDOpenHelper admin=new AdminBDDOpenHelper(this,
            "administracion", null, 1);
        SQLiteDatabase bd=admin.getWritableDatabase();
        String codigo=etxtcod.getText().toString();
        String titulo=etxtTitulo.getText().toString();
        String autor=etxtAutor.getText().toString();
        String anio=etxtAnio.getText().toString();
ContentValues registro=newContentValues();
registro.put("titulo",titulo);
registro.put("autor",autor);
registro.put("ano",anio);
            int cant = bd.update("libros", registro,
            "codigo="+codigo, null);
bd.close();
if (cant==1)
            Toast.makeText(this, "Se actualizaron los datos",
            Toast.LENGTH_SHORT).show();
else
            Toast.makeText(this, "No existe el Libro con dicho
            codigo", Toast.LENGTH_SHORT).show();
    }
//Y se cierra la clase principal
}
```

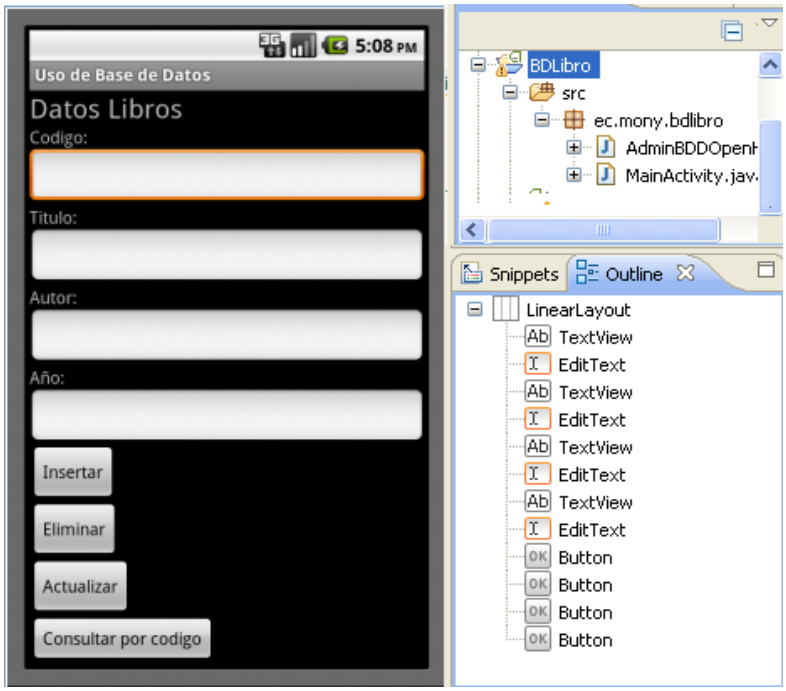


Figura 3.22: Interfaz visual ejemplo BDD
Fuente: Propia

CAPÍTULO IV

FUNCIONAMIENTO DEL APLICATIVO

CONTENIDO:

- Funcionamiento del Aplicativo
- Definición de Módulos

CAPÍTULO 4: FUNCIONAMIENTO DEL APLICATIVO

4.1 FUNCIONAMIENTO DEL APLICATIVO

4.1.1 Objetivos

En el aplicativo Taxímetro se definieron los siguientes objetivos:

- ✓ Permitir que el usuario encienda el servicio de GPS.
- ✓ Modificar los valores para el cálculo del precio a pagar tanto diurno como nocturno.
- ✓ Permitir que el usuario ingrese un número celular de la persona q desee que le llegue el SMS de auxilio en caso de un accidente.
- ✓ Acceder a la información del Taxi siempre y cuando el usuario ingrese la placa.
- ✓ Dar a conocer el tiempo transcurrido, la distancia y el valor a pagar por la carrera.

4.1.2 Alcance

El sistema funcionara sobre dispositivos móviles que cuenten mínimo con la versión 2.2 (Froyo) de Android.

La aplicación Taxímetro tiene la finalidad de ser un apoyo tanto para el dueño del taxi como para el usuario final, ya que mucha veces al tomar un taxi es imposible saber cuánto costará la carrera hasta que se le pregunta al taxista, y al realizar el pago queda la duda si se pagó el importe adecuado por dicho servicio, porque existen diversas formas de alterar los taxímetros, es por esto que esta aplicación servirá para hacer nuestro propio conteo de tarifa Taxímetro, aunque el precio se lo tomará como aproximación.

4.2 DEFINICIÓN DE MÓDULOS

El aplicativo Taxímetro contiene los siguientes módulos:

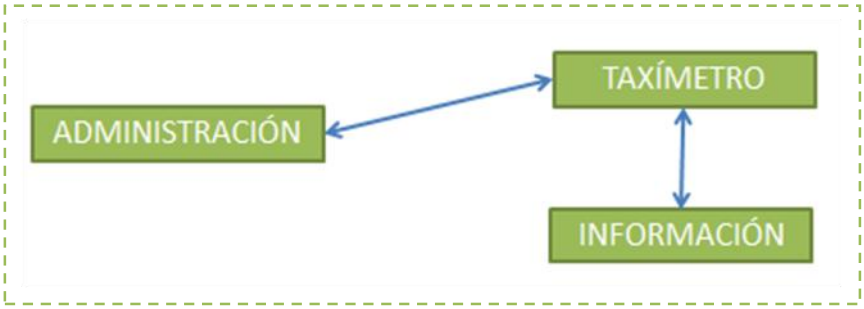


Figura 4.1: Módulos del Sistema
Fuente: Propia

4.2.1 Módulo de Administración

4.2.1.1 Actualización de Tarifa Diurna

Para la gestión de la tarifa diurna se definió el siguiente caso de uso y su respectiva documentación:

- **Diagrama Caso de Uso**

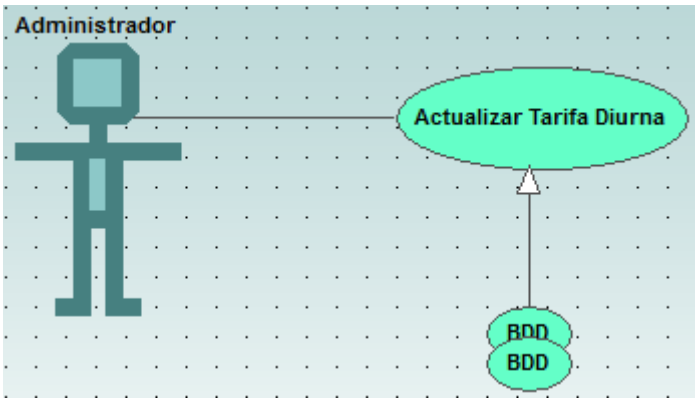


Figura 4.2: Proceso de Actualización de Tarifa Diurna
Fuente: Propia

- **Descripción Caso de Uso**

Caso de Uso:	Actualizar Tarifa Diurna
Autor:	Mónica Tapia
Fecha:	01/09/2012
Descripción: Permite actualizar las Tarifas.	
Actores: Administrador, usuario del sistema.	
Precondiciones: El administrador es cualquier persona que tenga acceso a la aplicación.	
Flujo Normal: 1. El actor pulsa sobre la opción Actualizar Tarifa Diurna 2. El sistema extrae los valores de la Tarifa Diurna de la base de datos. 3. El actor modifica los valores de los diferentes parámetros necesarios para el cálculo del Precio a pagar por la carrera y guarda. 4. El sistema será actualizado.	
Flujo Alternativo: Ninguno.	

Tabla 4.1: Proceso de Actualización de Tarifa Diurna
Fuente: Propia

4.2.1.2 Actualización de Tarifa Nocturna

Para la gestión de la tarifa nocturna se definió el siguiente caso de uso y su respectiva documentación:

- **Diagrama Caso de Uso**

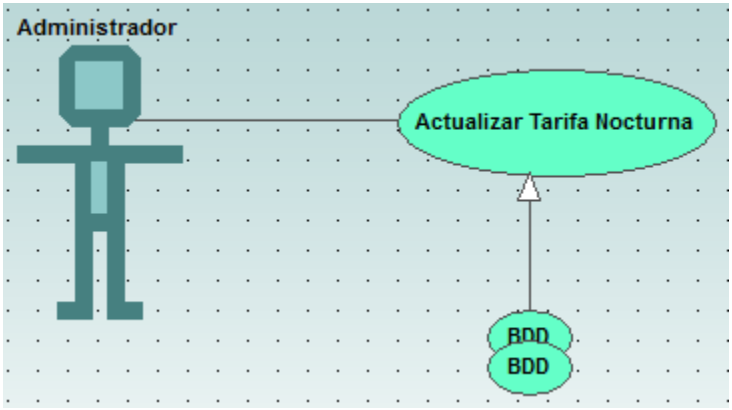


Figura 4.3: Proceso de Actualización de Tarifa Nocturna

Fuente: Propia

- **Descripción Caso de Uso**

Caso de Uso:	Actualizar Tarifa Nocturna
Autor:	Mónica Tapia
Fecha:	01/09/2012
Descripción: Permite actualizar las Tarifas.	
Actores: Administrador, usuario del sistema.	
Precondiciones: El administrador es cualquier persona que tenga acceso a la aplicación.	
Flujo Normal: 1. El actor pulsa sobre la opción Actualizar Tarifa Diurna 2. El sistema extrae los valores de la Tarifa Nocturna de la base de datos.	

3. El actor modifica los valores de los diferentes parámetros necesarios para el cálculo del Precio a pagar por la carrera y guarda. 4. El sistema será actualizado.
Flujo Alternativo: Ninguno.

Tabla 4.2: Proceso de Actualización de Tarifa Nocturna
Fuente: Propia

4.2.2 Módulo Taxímetro

4.2.2.1 Cálculo de Tarifa

Para la gestión del cálculo de tarifa se definió el siguiente caso de uso y su respectiva documentación:

- **Diagrama Caso de Uso**

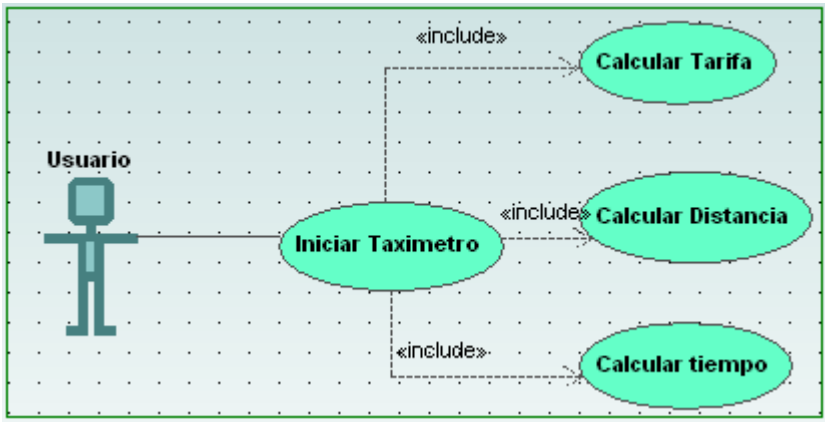


Figura 4.4: Proceso Iniciar Taxímetro
Fuente: Propia

- **Descripción Caso de Uso**

Nombre:	Iniciar Taxímetro
Autor:	Mónica Tapia
Fecha:	03/09/2012
Descripción: Permite iniciar y terminar el Taxímetro, escoger el horario y enviar un mensaje de auxilio.	
Actores: Usuario del sistema.	
Precondiciones: El usuario debe haber iniciado el servicio de GPS con anticipación.	
Flujo Normal: 1. El actor pulsa sobre el botón Taxímetro 2. El sistema abre la ventana Taxímetro 3. El usuario escoge el horario (Diurno y Nocturno). 4. El sistema extrae de la base de datos las tarifas con las que va a trabajar para el cálculo del Precio de la carrera. 5. El actor pulsa sobre el botón para iniciar taxímetro. 6. El sistema va mostrando el valor a pagar, la distancia que recorre y el tiempo transcurrido durante la carrera. 7. El actor pulsa sobre el botón terminar Taxímetro. 8. El sistema muestra un cuadro de dialogo con el valor a pagar y la tarifa con la que se ha calculado dicho valor. 9. El actor pulsa el botón OK.	
Flujo Alternativo:	

El sistema no podrá obtener distancia, si el dispositivo no cuenta con plan de datos, o no ha iniciado el servicio GPS con anticipación. El sistema no enviará un SMS de auxilio si no se guardó el número celular del destinatario.

Tabla 4.3: Proceso Iniciar Taxímetro
Fuente: Propia

4.2.2.2 Enviar Mensaje de Auxilio

Para la gestión del envió del mensaje de auxilio se definió el siguiente caso de uso y su respectiva documentación:

- **Diagrama Caso de Uso**

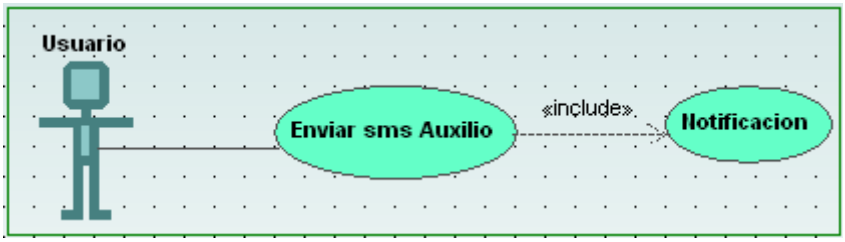


Figura 4.5: Proceso Enviar Mensaje
Fuente: Propia

- **Descripción Caso de Uso**

Nombre:	Enviar mensaje de auxilio
Autor:	Mónica Tapia
Fecha:	05/09/2012

Descripción: Permite enviar un mensaje de auxilio al número celular que se ingresó en el caso de uso Ingresar número Destinatario.
Actores: Usuario del sistema.
Precondiciones: El usuario debe disponer de saldo para el envío de mensajes. El usuario debe haber ingresado el número celular en el caso de uso Ingresar Destinatario. El usuario debe haber consultado la placa del taxi para que se puedan enviar dichos datos en el mensaje de auxilio
Flujo Normal: 1. El actor pulsa sobre el botón SMS Auxilio, en caso de algún accidente. 2. El sistema envía un mensaje al número que se debió haber guardado anteriormente. 3. El actor recibe una notificación de que el mensaje ha sido enviado.
Flujo Alternativo: El sistema no enviará un SMS de auxilio si no se guardó el número celular del destinatario.
Poscondiciones: El mensaje ha sido envido con éxito.

Tabla 4.4: Proceso Enviar Mensaje
Fuente: Propia

4.2.3 Módulo Información

4.2.3.1 Ingresar número destinatario

Para la gestión de ingreso del número destinatario se definió el siguiente caso de uso y su respectiva documentación:

- **Diagrama Caso de Uso**

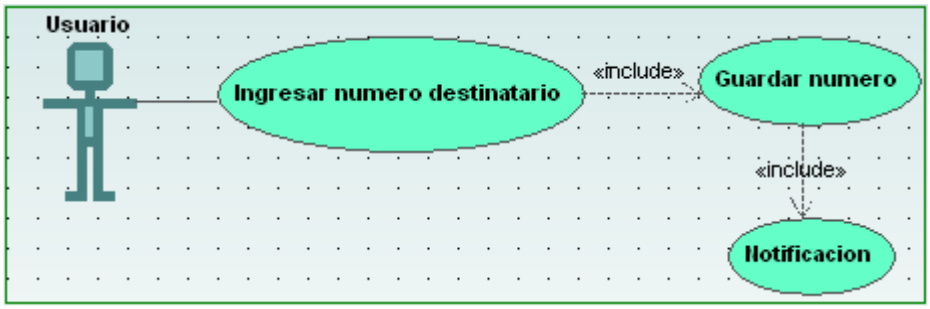


Figura 4.6: Proceso Ingresar número destinatario
Fuente: Propia

- **Descripción caso de Uso**

Nombre:	Ingresar número Destinatario
Autor:	Mónica Tapia
Fecha:	08/09/2012
Descripción: Permite introducir un número celular para él envió de un mensaje de auxilio en caso de un accidente.	
Actores: Usuario del sistema.	
Precondiciones: Ninguna.	

Flujo Normal: 1. El actor pulsa sobre el botón para configuración. 2. El sistema muestra una caja de texto para introducir el número celular del destinatario. 3. El actor introduce el número celular. 4. El sistema comprueba la validez del número y lo almacena.
Flujo Alternativo: El sistema comprueba la validez del número, si el número es incorrecto, se avisa al actor de ello permitiéndole que lo corrija
Poscondiciones: El número ha sido almacenado en el sistema.

Tabla 4.5: Proceso Ingresar número destinatario
Fuente: Propia

4.2.3.2 Consultar datos Taxi

Para la gestión de consulta de los datos del taxi se definió el siguiente caso de uso y su respectiva documentación:

- **Diagrama Caso de Uso**

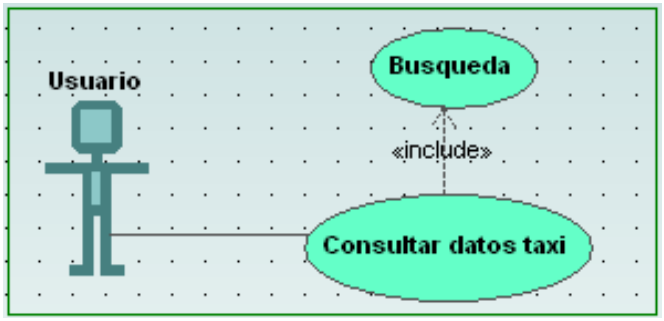


Figura 4.7: Proceso Consultar datos Taxi
Fuente: Propia

- **Descripción caso de Uso**

Caso de Uso:	Consultar datos Taxi
Autor:	Mónica Tapia
Fecha:	08/09/2012
Descripción: Permite conocer si el Taxi que tomamos es legal o no.	
Actores: Usuario del sistema.	
Precondiciones: Se necesita el número de placa del Taxi.	
Flujo Normal: 1. El actor pulsa sobre el botón datos taxi. 2. El sistema muestra una caja de texto para introducir la placa del taxi. 3. El actor introduce la placa y pulsa sobre el botón buscar. 4. El sistema hace una consulta y devuelve los datos del Taxi.	
Flujo Alternativo: 1. El sistema comprueba la validez de la placa, si no se encuentra la placa, se avisa al actor de ello.	
Poscondiciones: Los datos de las carreras serán mostrados con éxito	

Tabla 4.6: Proceso Consultar datos Taxi
Fuente: Propia

4.2.3.3 Historial de Carreras

Para la gestión de consulta del historial de carreras se definió el siguiente caso de uso y su respectiva documentación:

- **Diagrama Caso de Uso**

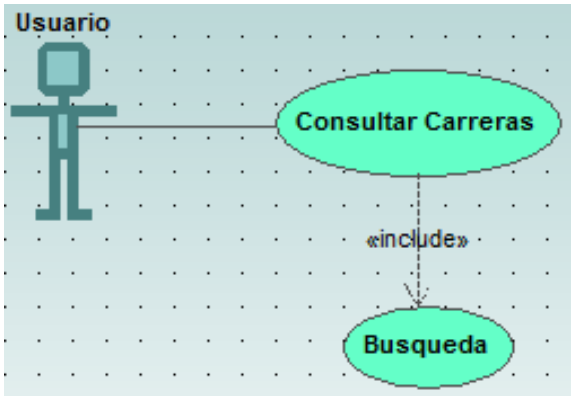


Figura 4.8: Proceso Consultar Historial de carreras
Fuente: Propia

- **Descripción caso de Uso**

Caso de Uso:	Consultar Historial de Carreras
Autor:	Mónica Tapia
Fecha:	09/09/2012
Descripción: Muestra la información de todas las carreras realizadas.	
Actores: Usuario del sistema.	

Precondiciones: Ninguna.
Flujo Normal: 1. El actor pulsa sobre el menú Historial. 2. El sistema muestra realiza una búsqueda a la base de datos y obtiene todas las carreras realizadas hasta la fecha, comenzando desde la carrera más reciente.
Flujo Alternativo: Si no ha realizado ninguna carrera, el sistema no mostrara nada.
Poscondiciones: Los datos de todas las carreras realizadas hasta la fecha serán mostrados con éxito.

Tabla 4.7: Proceso Consultar Historial de Carreras
Fuente: Propia

CAPÍTULO V

DESARROLLO DEL APLICATIVO

CONTENIDO:

- Metodología de Desarrollo
- Desarrollo del Aplicativos
- Introducción
- Gestión del Proyecto

CAPÍTULO 5: DESARROLLO DEL APLICATIVO

5.1 METODOLOGÍA DE DESARROLLO

La metodología de desarrollo a utilizar en este proyecto es SCRUM, que es una metodología de desarrollo ágil y flexible para el desarrollo de software, se basa en un proceso de trabajo constante, iterativo e incremental.

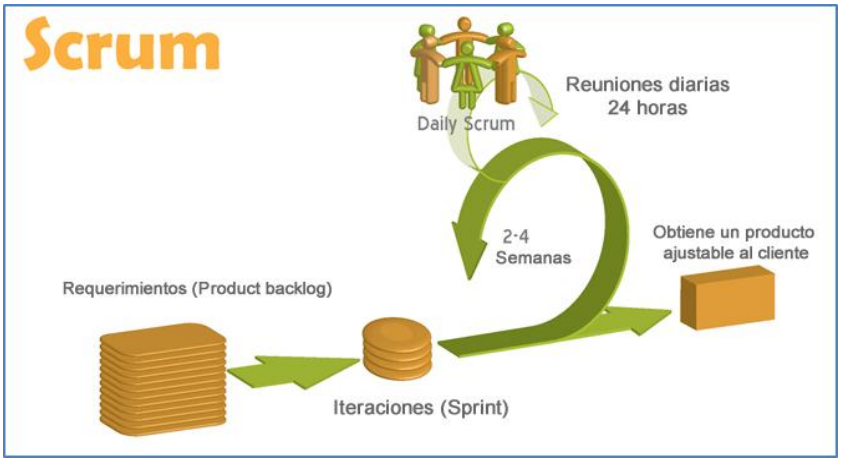


Figura 5.1: Metodología Scrum
Fuente: [**WEB 31**]

5.1.1 Roles en la metodología SCRUM

Existen dos aspectos fundamentales a diferenciar, los actores y las acciones:

- Los actores son los que ejecutarán las acciones, estos son:

ProductOwner, es la persona responsable de transmitir al equipo de desarrollo la visión del producto que se desea crear, conoce los requerimientos y marca las prioridades del proyecto o producto.

Scrum Master es la persona que asegura el seguimiento de la metodología guiando las reuniones y ayudando al equipo ante cualquier problema que pueda

aparecer, garantiza que el equipo de trabajo no tenga impedimentos u obstáculos para abordar sus tareas dentro del proyecto.

ScrumTeam son las personas responsables de desarrollar y entregar el producto**.**

Usuarios o Cliente, son los beneficiarios finales de la aplicación a desarrollar.

Las acciones tienen relación directa con los actores las acciones de Scrum forman parte de un ciclo iterativo repetitivo y tienen como objetivo minimizar el esfuerzo y maximizar el rendimiento en el desarrollo.

- Las acciones fundamentales son:

Definición del proyecto (*ProductBacklog*): Consiste en un documento que recoge el conjunto de requerimientos que se asocian al proyecto. El ProductOwneres el responsable de realizarlo y establecer las prioridades de cada requerimiento. Es un documento de alto nivel, que contiene descripciones genéricas (no detalladas), y que está sujeto a modificaciones a lo largo del desarrollo.

Definición del Sprint (*Sprint Backlog*): Un sprint debe entenderse como un subconjunto de requerimientos, que provienen del ProductBacklog, para ser ejecutadas durante un periodo entre 1 y 4 semanas de trabajo. El sprint backlogses el documento que describa las tareas que son necesarias para realizar el subconjunto de requerimientos.

5.1.2 Ventajas

- ✓ Definición de iteraciones con fechas cortas de trabajo. (dos a cuatro semana)
- ✓ Trabaja principalmente en los requerimientos de más prioridad.
- ✓ Presenta resultados al cliente en poco tiempo.
- ✓ Existen reuniones para solucionar los problemas de retraso.

- ✓ Permite ajustar el producto de software de acuerdo a las necesidades del cliente debido a que se tiene iteraciones cortas.
- ✓ Ayuda a las empresas a ahorrar tiempo y dinero.
- ✓ Es aplicable a cualquier tecnología y lenguaje de programación.

5.1.3 Desventajas

- ✓ El mantenimiento del producto puede complicarse debido a no tener mucha documentación.
- ✓ Permite fallar si es necesario.
- ✓ No es conveniente usarlo cuando el equipo de trabajo es muy grande.
- ✓ El equipo de trabajo siempre no debe estar distribuido geográficamente.
- ✓ Equipo de trabajo con poca experiencia.

5.2 MVC EN ANDROID

Android utiliza el patrón de diseño Modelo Vista Controlador (MVC), que consiste en separar los datos de una aplicación, la interfaz de usuario y la lógica de negocios en tres componentes distintos que se relacionan para tener como resultado una aplicación.

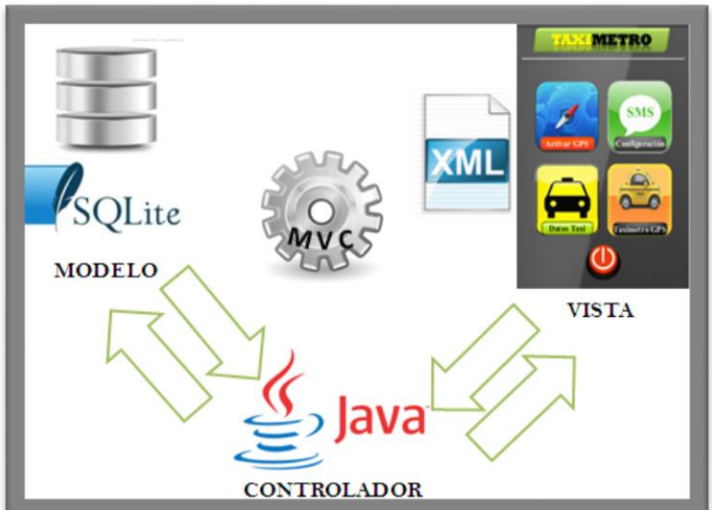


Figura 5.2: Diagrama MVC del Aplicativo
Fuente: [**WEB 32**]

5.2.1 Modelo

Como modelo se refiere a las representaciones que se construye basada en la información con la que operará una aplicación. El modelo que se elige dependiendo de las necesidades de información de la aplicación.

5.2.2 Vista

La vista es la interfaz con la que va a interactuar el usuario. En Android, las interfaces se construyen en XML.

5.2.3 Controlador

Son todas las clases que ayudan a darle vida a la interfaz que se ha construido previamente y que permite desplegar y consumir información para el usuario. Estos controladores se programan en lenguaje Java y son el core de la aplicación.

5.3 DESARROLLO DEL APLICATIVO

5.3.1 Participantes o equipo de trabajo

El conjunto de participantes para el desarrollo del sistema Taxímetro se describe a continuación:

Nro.	Rol	Nombre
1	Cliente (Owner)	Sr. Gualberto Quiñonez
2	Facilitador (Scrum master)	Mónica Tapia
3	Equipo SCRUM (Scrum team)	Mónica Tapia

Tabla 5.1: Participantes del Sistema Taxímetro
Fuente: Propia

5.3.2 Lista de Requerimientos (PRODUCT BACKLOG)

A continuación se describen los siguientes requerimientos:

Id	Requerimiento
R1	El sistema funcione utilizando el servicio de GPS
R2	Búsqueda de los datos del taxi en base a la placa
R3	En caso de un accidente se pueda enviar un mensaje de auxilio
R3	De la opción de escoger el tipo de horario del servicio, sea este diurno o nocturno
R4	Muestre la distancia recorrida
R5	Muestre el tiempo transcurrido
R6	Muestre el valor de la carrera
R7	Muestre un Historial de todas las carreras realizadas
R8	Actualice los valores de los diferentes parámetros para el cálculo del precio de la carrera
R9	Menús de información

Tabla 5.2: Requerimientos del Sistema
Fuente: Propia

5.3.3 Priorizar los requerimientos (RELEASE BACKLOG)

Luego de analizar la lista de requerimientos con el cliente (Owner) se estableció el siguiente cuadro de prioridades:

Id	Requerimiento	Prioridad
R2	Búsqueda de los datos del taxi en base a la placa	1
R1	El sistema funcione utilizando el servicio de GPS	2
R4	Muestre la distancia recorrida	3

R5	Muestre el tiempo transcurrido	4
R6	Muestre el valor de la carrera	5
R3	En caso de un accidente se pueda enviar un mensaje de auxilio	6
R9	Menús de información	7
R8	Actualice los valores de los diferentes parámetros para el cálculo del precio de la carrera	8
R7	Muestre un Historial de todas las carreras realizadas	9

Tabla 5.3: Priorizar los Requerimientos
Fuente: Propia

5.3.4 Planificación del proyecto (SPRINT)

El proyecto tiene un estimado de 93 días para su culminación y la jornada de trabajo será de 5 horas diarias. Siguiendo con la metodología, tenemos que cada uno de los requerimientos agruparlos en iteraciones.

Id	Desde	Hasta	Días	Nombre	Meta	Requerimiento
I1	2012-10-15	2012-11-09	20	Análisis y construcción del Proyecto	Diseñar el proyecto de acceso a datos, diseñar la interfaz de usuario.	Estudio de la tecnología Android
I2	2012-11-12	2012-11-30	15	Búsqueda de datos Taxi por placa.	Creación del filtro de búsqueda de acuerdo a los	R2, R1

					requerimientos	
I3	2012-12-03	2013-01-11	30	Costo Carrera Taxi.	Calcular el costo de la carrera en base a la distancia y tiempo transcurrido	R4,R5,R6
I4	2013-01-14	2013-01-31	14	Número para el Mensaje de auxilio, notificaciones, menús de información	Almacenar los datos del taxi para ser enviados en el mensaje, notificaciones, menús	R3,R9
I5	2013-02-01	2013-02-20	14	Actualizar Tarifas, Historial Carreras	Modificar las tarifas en caso de actualización de la Ley de Tránsito y Transporte.	R7, R8

Tabla 5.4: Agrupación de requerimientos en Iteraciones
Fuente: Propia

5.3.5 Iteración 1. Análisis y construcción del Proyecto

5.3.5.1 Modelo de Datos

En esta iteración se inició primeramente con el diseño del modelo de datos, obteniendo finalmente el siguiente la siguiente base de datos:

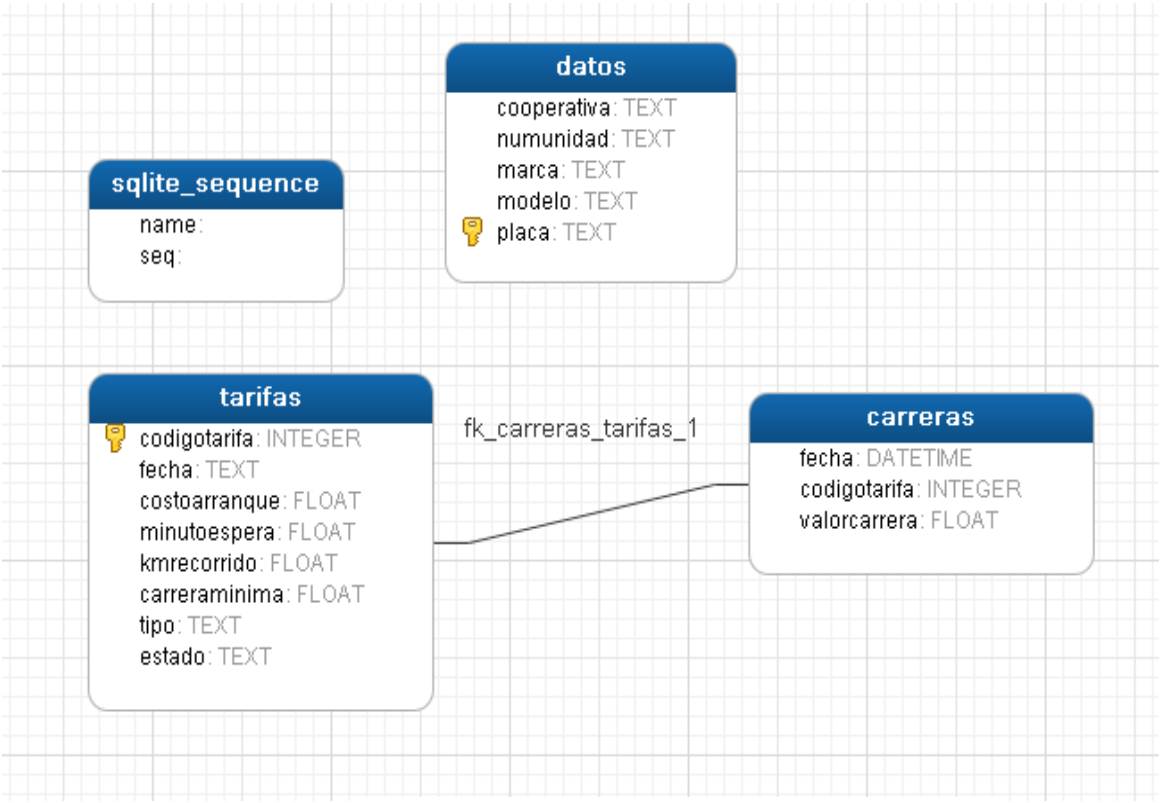


Figura 5.3: Base de Datos Taxímetro
Fuente: Propia

A continuación se crea la clase de acceso a los datos, la misma q será añadida en la carpeta **assets** del proyecto Taxímetro, la clase de acceso a la base de datos `AdaptadorBDD`, la cual extiende de la de la Api android.database.sqlite que es una clase de ayuda para gestionar la creación de bases de datos.

También se diseñó el diagrama de arquitectura que se muestra a continuación:

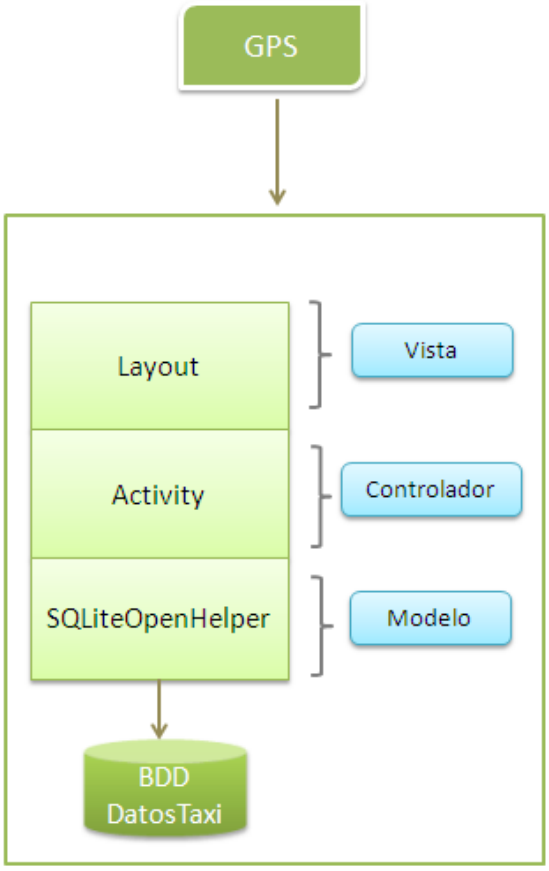


Figura 5.4: Diagrama de Arquitectura
Fuente: Propia

5.3.5.2 Interfaz de Usuario

Aquí se define la estructura visual para la interfaz de usuario, como es el caso de la interfaz de usuario para una actividad de aplicación. Se puede declarar el diseño mediante un objeto de tipo Layout, este es un contenedor de una o más vistas y controla su comportamiento y posición.

La interfaz de usuario para la aplicación Taxímetro es la siguiente:

Figura 5.5: Interfaz Menú Principal Aplicación Taxímetro
Fuente: Propia

Figura 5.6: Interfaz de Ingreso del Numero Celular de la Aplicación
Fuente: Propia

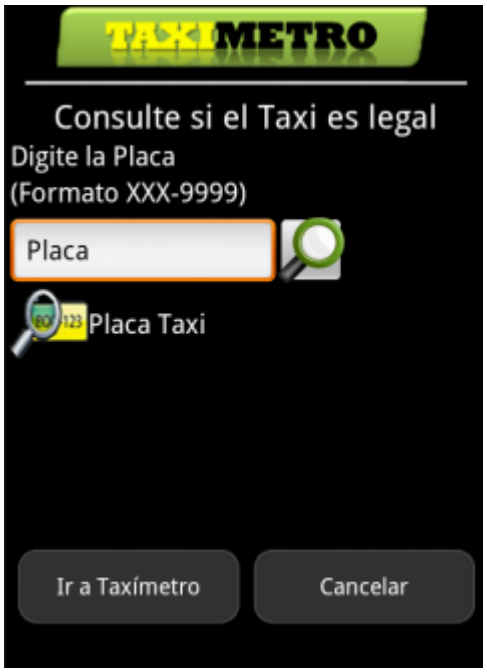


Figura 5.7: Interfaz de consulta de datos de la Aplicación Taxímetro
Fuente: Propia

Figura 5.8: Interfaz de inicio del Taxímetro de la Aplicación
Fuente: Propia

Figura 5.9: Interfaz Administración Tarifa Diurna de la Aplicación
Fuente: Propia

Figura 5.10: Interfaz Administración Tarifa Nocturna de la Aplicación
Fuente: Propia

Figura 5.11: Interfaz Historial de Carreras de la Aplicación Taxímetro
Fuente: Propia

5.3.6 Iteración 2. Búsqueda de datos Taxi por placa.

Búsqueda de datos Taxi, acceso al servicio GPS

La clase de búsqueda de los datos del Taxi en base a la placa se llama `DatosTaxi.java`, misma que está implementada dentro del paquete del proyecto Taxímetro.

Ver caso de uso *Figura 4.6: Proceso Consultar datos taxi*

5.3.7 Iteración 3. Costo Carrera Taxi.

La clase para el cálculo de las Carreras se llama `IniciarTaximetro.java`, misma que está implementada dentro del paquete del proyecto Taxímetro.

Ver caso de uso *Figura 4.3: Proceso Iniciar Taxímetro*

5.3.8 Iteración 4. Mensaje de auxilio, notificaciones, menús de información

La clase para el ingreso del número celular, para posteriormente ser guardado para él envió del mensaje de auxilio en caso de un accidente se llama `ConfigNumAyuda.java`, misma que está implementada dentro del paquete del proyecto Taxímetro.

Ver caso de uso *Figura 4.4: Proceso Iniciar Enviar Mensaje*

5.3.9 Iteración 5. Actualizar Tarifas, Historial Carreras

Las clases para la actualización de tarifas se llaman `AdministracionDia.java` y `AdministracionNoche.java`, mismas que están implementadas dentro del paquete del proyecto Taxímetro.
Ver caso de uso *Figura 4.3: Proceso Actualización de Tarifa Diurna*
Ver caso de uso *Figura 4.3: Proceso Actualización de Tarifas Nocturna*

5.4 INTRODUCCIÓN

El presente proyecto tiene como finalidad desarrollar una aplicación para dispositivos móviles Android, en el que se implemente sus funcionalidades básicas.

El proyecto Taxímetro para dispositivos móviles Android es un buen sistema para demostrar funcionalidades de los dispositivos móviles como el servicio de GPS y él envió de mensajes, entre otras.

5.5 GESTIÓN DEL PROYECTO

5.5.1 Resumen

Un taxímetro normal es un aparato de medida mecánico o electrónico instalado en taxis, que mide el valor a cobrar en relación a la distancia recorrida y al tiempo transcurrido.

Debido al auge que van adquiriendo los dispositivos móviles, se toma la decisión de realizar un taxímetro para celulares Android.
La aplicación se basa en el GPS del propio celular por lo que los cálculos pueden ser aproximados.

Utiliza base de datos y el servicio de envío de mensajes de un dispositivo a otro.

Para realizar la aplicación que cumpla con estos requisitos, se necesita de las siguientes herramientas de programación con sus respectivas versiones:

Herramientas	Versiones
Jdk	1.6+
Android SDK	2.2
MotodevStudio	4.0
Base de Datos	SQLite

Tabla 5.5: Herramientas a utilizar en el sistema
Fuente: Propia

Estas herramientas deben ser instaladas y configuradas previamente para poder iniciar con la aplicación.

Además se necesita de un celular android, ya que en el emulador la aplicación no funciona completamente, se realiza hasta la negociación, pero el momento de utilizar el servicio GPS no lo hace, por limitaciones del pc.

Es importante encender el GPS con anticipación para que este ubique la posición rápidamente al momento de iniciar el taxímetro y no ocurra inconvenientes para el usuario.

5.5.2 Vista General

5.5.2.1 Propósito, los Alcances, y Objetivos

El proyecto contempla el desarrollo e implementación del sistema Taxímetro, de manera que cumpla los requerimientos especificados anteriormente, los mismos que se detallan en los casos de uso descritos anteriormente.

Muchas veces al tomar un taxi es imposible saber cuánto costará la carrera hasta que se le pregunta al taxista, y al realizar el pago queda la duda si se pagó el importe adecuado por dicho servicio, ya que existen diversas formas de alterar los taxímetros, es por esto que esta aplicación servirá para hacer nuestro propio conteo de tarifa Taxímetro, aunque el precio se lo tomará como aproximación.

Para ello se considera necesario el desarrollo de un Taxímetro para usuarios finales Android, así como una base de datos que almacene los datos principales de las Cooperativas de Taxis de la ciudad de Ibarra.

El proyecto funcionará sobre teléfonos inteligentes que cuenten mínimo con la versión 2.2 (Froyo) de Android.

5.5.2.2 Suposiciones y Restricciones

Las suposiciones y restricciones respecto al sistema, y que se derivan directamente de las entrevistas con el cliente son las siguientes:

- ✓ Seguridad de la información: el usuario cuenta con las tarifas descritas en la Dirección de Tránsito y Transporte y en caso de que estas cambien, el usuario podrá modificarlas.
- ✓ Gestión de flujos de trabajo, confiabilidad de los datos.

El sistema debe diseñarse como una aplicación para dispositivos móviles Android para ser utilizado por el usuario, sea este el propietario del taxi o el cliente del taxi.

5.5.3 Organización

5.5.3.1 Participantes en el Proyecto

Los participantes del proyecto se describen en la Tabla 4.1 del capítulo anterior.

5.5.3.2 Interfaz del proyecto

El equipo de desarrollo interactuará activamente con los participantes para especificación y validación de los procesos generados.

5.5.3.3 Roles y Responsabilidades

Los roles y responsabilidades ya se encuentran definidas en el capítulo anterior.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

CONTENIDO:

- Conclusiones
- Recomendaciones
- Glosario
- Referencias

CAPÍTULO 6: CONCLUSIONES Y RECOMENACIONES

6.1 CONCLUSIONES

- La utilización de plataformas abiertas nos permite crear software de buena calidad sin preocuparnos de licencias, sus actualizaciones son constantes ya que se trata de software libre, además al reducir el presupuesto para un proyecto hace de Android una plataforma muy atractiva para el desarrollo de aplicaciones móviles.
- Se logró desarrollar la aplicación "Taxímetro" para teléfonos inteligentes con sistema Operativo Android que simula un taxímetro físico pero haciendo uso de la tecnología GPS para calcular un aproximado de la tarifa.
- Debido a un limitante que tiene el IDE de desarrollo Motodev Studio for Android como es las resoluciones de pantalla, solo abarca tres tipos de resoluciones por lo que la aplicación no se adapta a Tables.
- Se ha logrado hacer buen uso de las herramientas SQLite contenidas en el framework de Android, logrando implementar con éxito la aplicación propuesta como proyecto de esta tesis, demostrando así la facilidad de desarrollar aplicaciones para la plataforma Android.
- El uso de SCRUM como metodología de desarrollo, nos permite obtener resultados en poco tiempo dando mayor prioridad a los requerimientos, esto representa una ventaja ya que se prueba la aplicación y sus nuevas funcionalidades mucho más rápido que al desarrollarlo con otras metodologías.
- El presente trabajo es un aporte para desarrollo social y tecnológico de la región, ya que mediante este documento podemos aprender una arquitectura que en el medio tiene muy buena demanda y que actualmente en nuestra región no se la está explotando.

6.2 RECOMENDACIONES

- Para empezar a programar en Android es aconsejable tener experiencia básica en el lenguaje de programación Java y conocimientos en Programación Orientada a Objetos.

- Si vamos a realizar un estudio de nuevas tecnologías se recomienda seguir la documentación provista por el desarrollador de esta tecnología, también es conveniente leer foros especializados ya que en los mismos se pueden encontrar respuestas a las inquietudes y problemas que puedan surgir en el proceso de aprendizaje.

- Se recomienda desarrollar una aplicación utilizando una versión baja del API, esto permitirá que la aplicación sea compatible con una mayor cantidad de dispositivos.

- En cuanto al diseño se recomienda que los botones de acción no sean demasiado pequeños debido a que la mayoría de teléfonos inteligentes son táctiles y se utiliza el dedo para su manipulación.

- Según las comparativas obtenidas, el sistema operativo Android es una excelente plataforma para desarrollar aplicaciones, por ser de software libre y estar respaldado por Google, que actualmente es uno de los gigantes de la informática.

- Para que el aplicativo Taxímetro trabaje de una manera precisa y rápida, se recomienda que el dispositivo cuente con plan de datos.

GLOSARIO

Java: Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje deriva de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel.

XML: XML eXtensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C), permite definir la gramática de lenguajes específicos para estructurar documentos grandes.

API: Interfaz de Programación de Aplicaciones (Application Programming Interface) es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Open Handset Alliance: Es una alianza comercial de 84 compañías liderada por Google, que se dedica a desarrollar estándares abiertos para dispositivos móviles. Entre sus miembros tenemos: HTC, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia entre otros.

SDK: Siglas en inglés de software development kit *(kit de desarrollo de software)* es un conjunto de herramientas de desarrollo de software que permite al programador crear aplicaciones para un sistema concreto, proporciona las bibliotecas API y herramientas de desarrollo necesarias para crear, probar y depurar aplicaciones.

IDE: Entorno de desarrollo, es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Open Source: Código abierto, es el término con el que se conoce al software distribuido y desarrollado libremente.

Smartphone: Es un teléfono inteligente generalmente táctil, que puede comunicarse a través de Wi-Fi, bluetooth, conexión a internet, envío de mensajería, e-mails. Es un teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades semejantes a una mini computadora y conectividad. Una característica clave de un Smartphone es que las aplicaciones adicionales pueden ser instaladas en el dispositivo.

Java Development Kit (**JDK**): Es un software que provee herramientas de desarrollo para la creación de programas en Java.

JAR: Un archivo jar siglas en inglés, **J**ava **Ar**chive, es un tipo de archivo que permite ejecutar aplicaciones escritas en el lenguaje Java. Los archivos JAR están comprimidos con el formato ZIP y cambiada su extensión a .jar.

BIBLIOGRAFÍA

- **[WEB 1]:** Android (2012). Que es Android?. Recuperado el 10 de Abril 2012, de http://www.celularesandroid.com/que-es-android/
- **[WEB 2]:** Wikipedia (2012). Android. Recuperado el 10 de Abril 2012, de http://es.wikipedia.org/wiki/Android
- **[WEB 3]:** TinEye (2012). Dispositivo Android. Recuperado el 20 de Abril 2012, de http://www.tineye.com/search/14da41d3e23d464e85d5214f336a9efd0e578987.
- **[WEB 4]:** Angel Vilchez (2009). Que es Android, Características y Aplicaciones. Recuperado el 20 de Abril 2012, de http://www.configurarequipos.com/doc1107.html
- **[WEB 5]:** Wikipedia (2012). IOS. Recuperado el 21 de Abril 2012, de http://es.wikipedia.org/wiki/IOS_%28sistema_operativo%29
- **[WEB 6]:** Wikipedia (2012). Anexo: Historial de versiones de iOS. Recuperado el 21 de Abril 2012, de http://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_iOS
- **[WEB 7]:** Wikipedia (2012). Windows Phone. Recuperado el 23 de Abril 2012, de http://es.wikipedia.org/wiki/Windows_Phone
- **[WEB 8]:** La Motora (2013). iOS 5 vs. Android 4.0 Ice Cream Sandwich vs. Windows Phone 7.5 Mango. Recuperado el 2 de Abril 2013, de http://www.lamotora.com/index.php/blog/128-comparacion-movil
- **[WEB 9]:** Taringa (2011). iOS vs Android vs Windows Phone 7, ¿Cual smartphone comprar. Recuperado el 2 de Abril 2012, de http://www.taringa.net/posts/celulares/8066308/IOS-vs-Android-vs-Windows-Phone-7-Cual-smartphone-comprar.html
- **[WEB 10]:** Condesa (2011). La máquina virtual Dalvik. Recuperado el 25 de Abril 2012, de http://androideity.com/2011/07/07/la-maquina-virtual-dalvik/
- **[WEB 11]:** Victor's Hut(2007). Cómo funciona el GPS. Recuperado el 2 de Mayo 2012, de http://victorhut.wordpress.com/2007/03/02/como-funciona-el-gps/

- **[WEB 12]:** PabloYglesias (2011). Adentrándose en el mundo GPS. Recuperado el 2 de Mayo 2012, de http://planetared.com/2013/01/adentrandose-en-el-mundo-del-gps-nociones-basicas-i/
- **[WEB 13]:** PabloYglesias (2011). Adentrándose en el mundo GPS: Arquitectura. Recuperado el 2 de Mayo 2012, de http://www.pabloyglesias.com/adentrandose-en-el-mundo-del-gps-arquitectura-ii/
- **[WEB 14]:** Nirman (2011). Sistema Operativo Android. Recuperado el 3 de Marzo2012, de http://www.aplicaciones-android.org/sistema-operativo-android/
- **[WEB 15]:** Reimon & Richard (2013). Activity – entender y usar una Actividad. Recuperado el 3 de Marzo 2013, de http://jarroba.com/activity-entender-y-usar-una-actividad/
- **[WEB 16]:** Condesa (2011). Ciclo de vida de una actividad. Recuperado el 3 de Marzo 2012, de http://androideity.com/2011/07/06/ciclo-de-vida-de-una-actividad/
- **[WEB 17]:** Software de Comunicaciones (2011). Librerías Básicas. Recuperado el 15 de Mayo 2012, https://sites.google.com/site/swcuc3m/home/android/api
- **[WEB 18]:** Sgoliver (2010). Estructura de un proyecto Android. Recuperado el 15 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1278
- **[WEB 19]:** GDroid (2011). Conociendo el archivo AndroidManifest.xml. Recuperado el 28 de Mayo 2012, de http://gdroid.com.mx/blog/2011/07/01/conociendo-el-archivo-androidmanifest-xml/
- **[WEB 20]:** Sgoliver (2010). Interfaz de usuario en Android: Layouts. Recuperado el 28 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1341
- **[WEB 21]:** Sgoliver (2010). Interfaz de usuario en Android: Controles básicos (I). Recuperado el 29 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1363
- **[WEB 22]:** Sgoliver (2010). Interfaz de usuario en Android: Controles básicos (II). Recuperado el 29 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1373
- **[WEB 23]:** Sgoliver (2010). Interfaz de usuario en Android: Controles básicos (III). Recuperado el 29 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1387

- **[WEB 24]:** Sgoliver (2010). Interfaz de usuario en Android: Controles de selección (I). Recuperado el 30 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1404
- **[WEB 25]:** Sgoliver (2010). Interfaz de usuario en Android: Controles de selección (II). Recuperado el 30 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1414
- **[WEB 26]:** Sgoliver (2010). Interfaz de usuario en Android: Controles de selección (IV). Recuperado el 29 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1439
- **[WEB 27]:** Sgoliver (2010). Interfaz de usuario en Android: Pestañas (Tabs). Recuperado el 29 de Mayo 2012, de http://www.sgoliver.net/blog/?p=2112
- **[WEB 28]:** Sgoliver (2010). Interfaz de usuario en Android: Widgets (I). Recuperado el 29 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1684
- **[WEB 29]:** Sgoliver (2010). Menús en Android. Recuperado el 29 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1756
- **[WEB 30]:** Sgoliver (2010). Preferencias en Android. Recuperado el 29 de Mayo 2012, de http://www.sgoliver.net/blog/?p=1731
- **[WEB 31]:** Diego Salma (2009). Scrum. Recuperado el 2 de Junio 2012, de http://www.diegosalama.com/2009/08/21/scrum-funciona/
- **[WEB 32]:** Condesa (2012). La importancia del MVC en Android. Recuperado el 25 de Mayo de 2012, de http://androideity.com/2012/05/10/la-importancia-del-mvc-en-android/

GPS

- PabloYglesias. (2013). Adentrándose en el mundo del GPS: Arquitectura (I). Recuperado en enero de 2013, de http://www.pabloyglesias.com/blog/adentrandose-en-el-mundo-del-gps-nociones-basicas-i/

- Adrián Latorre. (2012). Cómo está estructurado el GPS en Android. Recuperado en enero de 2013, de http://www.elandroidelibre.com/2012/12/cmo-est-estructurado-el-gps-en-android.html

SQLite

- Wikipedia. (2012). SQLite. Recuperado en junio de 2012, de http://es.wikipedia.org/wiki/SQLite

- Lars Vogel. (2012). Android SQLite Database and ContentProvider - Tutorial. Recuperado en junio de 2012, de http://www.vogella.com/articles/AndroidSQLite/article.html

  http://www.sqlite.org/

Android

- Wikipedia. (2012). Android. Recuperado en mayo de 2012, de http://es.wikipedia.org/wiki/Android

- AngelVilchez (2009). Que es Android: Características y Aplicaciones. Recuperado en mayo de 2012, de http://www.configurarequipos.com/doc1107.html.

- OSL. (2011). Documentación del curso de Desarrollo Android. Recuperado en marzo de 2012, dehttp://www.softwarelibre.ulpgc.es/cursos/android

- Txema Rodríguez (2012). Android en 2012: desarrollando aplicaciones. Recuperado el mayo de 2012, de http://www.xatakandroid.com/aplicaciones-android/android-en-2012-desarrollando-aplicaciones

- Guru_Andrea. (2012). Versiones de Android.Recuperado en mayo de 2012, dehttp://comunidad.movistar.es/t5/Blog-Android/TutorialQu%C3%A9-tiene-cada-versi%C3%B3n-de-Android/ba-p/435041

- Alkar. (2010). AndroidFroyo 2.2. Recuperado en mayo de 2012, dehttp://http://www.genbeta.com/sistemas-operativos/android-22-froyo-un-repaso-a-las-principales-novedades

- Pablo Diaz. (2011). Android Gingerbread 2.3. Recuperado en mayo de 2012, de http://www.tecnofans.es/android/articulo/android-gingerbread-2-3-mejoras-frente-a-froyo-2-2/25539/

- Santiago Hollmann (2011). Android Manifest. Recuperado en junio de 2012, de http://piedralibre.wordpress.com/2010/11/25/android-manifest-una-ntroduccion/

- Android (2012). Foros. Recuperado en octubre de 2012, de http://www.android-spa.com

- Ignacio González Sainz, Javier Perez Pacheco, Adrián Ruiz Contreras, Jorge Silva C (2012). WikiDroid. Recuperado en octubre de 2010, de http://www.wikidroid.es

- Condesa (2012). Programación Android**.** Recuperado en octubre de 2012, de http://androideity.com/

- sgoliver.net blog (2012). Curso Programación Android**.** Recuperado en octubre de 2012, de http://www.sgoliver.net/blog/

- Universidad Politécnica de Valencia (2011). Programación para dispositivos móviles**.** Recuperado en octubre de 2012, de http://www.androidcurso.com/

- Universidad Carlos III de Madrid (2009). Software de Comunicaciones. Recuperado en octubre de 2012, de https://sites.google.com/site/swcuc3m/home/android

- Javier (2012). Windows Phone 8 vs iOS 6.0 vs Android 4.0: Comparación. Recuperado en octubre de 2012, de http://applelizados.com/windows-phone-8-ios-6-0-android-4-0-comparacion/
- Taringa (2012). Tutoriales Android. Recuperado en octubre de 2012, de http://www.taringa.net/posts/videos/14032597/Tutoriales-Android.html.
- Dani Alonso (2011). Comparativa de iOS, Android y Windows Phone**.** Recuperado en mayo de 2012, de http://technicidio.wordpress.com/2011/07/04/comparativa-ios-iphone-apple-android-google-windows-phone-microsoft/ Dani Alonso
- Dieyoarx (2011). iOS vs Android vs Windows Phone 7, ¿Cual smartphone comprar. Recuperado en octubre de 2012, de http://www.taringa.net/posts/celulares/8066308/IOS-vs-Android-vs-Windows-Phone-7-Cual-smartphone-comprar.html

SCRUM

- Alejandro Mazaeda. (2007). SCRUM: Desarrollo Ágil de Software. Recuperado en junio de 2012, de http://www.taringa.net/posts/info/867068/SCRUM_-Desarrollo-_gil-de-software.html
- Xavier Albaladejo. (2010). ¿Proyectos complejos? ¿Necesitas resultados? Conoce Scrum. Recuperado en junio de 2012, de http://www.proyectosagiles.org/beneficios-de-scrum
- Alejandro Mazaeda. (2007). SCRUM: Desarrollo Ágil de Software. Recuperado en junio de 2012, de http://www.proyectosagiles.org/
- Itzcoalt Alvarez M (2010). Desarrollo Ágil con Desarrollo Ágil con SCRUM. Recuperado julio de 2012, de http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:sg07.p02.scrum.pdf

Taxímetro

- Wikipedia. (2012). Taxímetro. Recuperado en mayo de 2012, de http://es.wikipedia.org/wiki/Taxímetro
- Google play. (2012). TAXO - Taxímetro Argentino. Recuperado en mayo de 2012, de https://play.google.com/store/apps/details?id=ar.com.skaplun.Taxo&feature=search_result
- CONSEJO NACIONAL DE TRANSITO Y TRANSPORTE (2003). Resolución Nº 001-DIRñ2003-CNTTT. Recuperado en febrero de 2012, de http://www.ant.gob.ec/tarifas/001DIR.pdf

ANEXOS

CONTENIDO:

ANEXO A

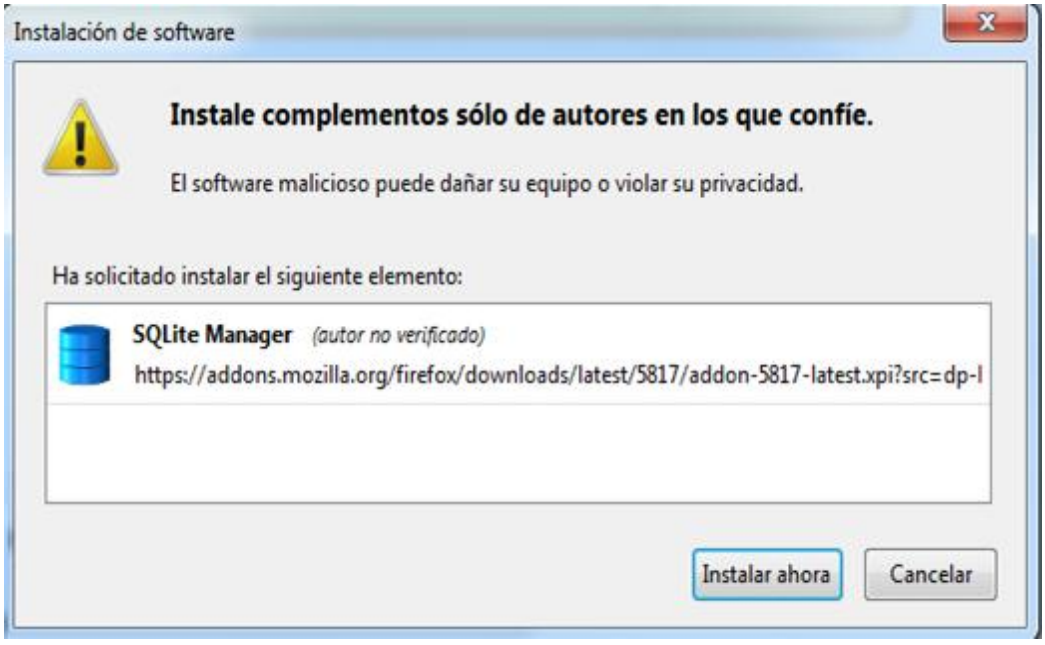
MANUAL DE TÉCNICO

INSTALACIÓN DE LA BASE DE DATOS

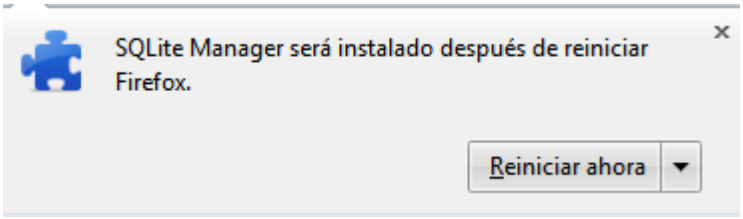
SQLite es un sistema de gestión de base de datos relacional de software libre, a continuación se muestra su instalación:

- SQLite Manager se lo puede instalar como complemento de Mozilla, se lo descarga de la siguiente dirección https://addons.mozilla.org/es/firefox/addon/sqlite-manager/ haciendo clic en el botón "Agregar a Firefox":

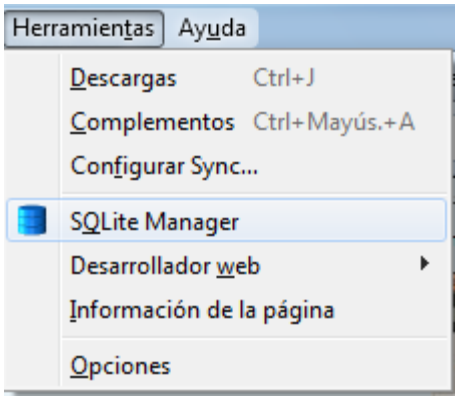
- Seguidamente aparece una ventana de verificación, dar clic en Instalar ahora



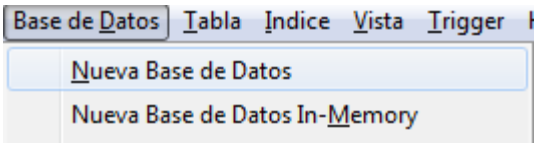
- Y por último dar clic en Reiniciar ahora



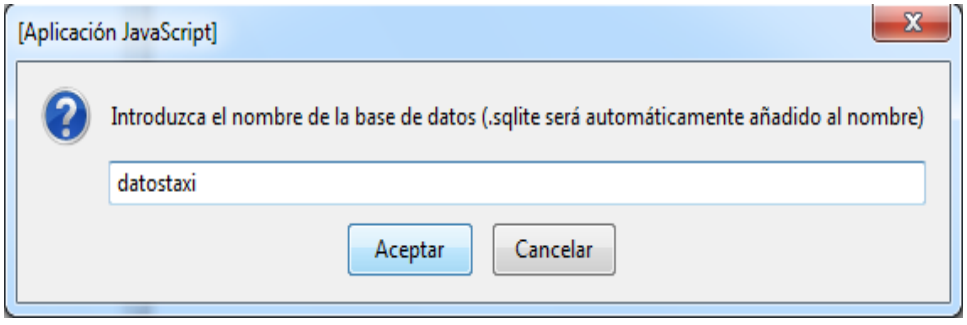
- Una vez instalado, ir a Herramientas y escoger SQLite Manager.



- Para crear una base de datos, dar clic en la pestaña Base de Datos y Nueva Base de Datos:

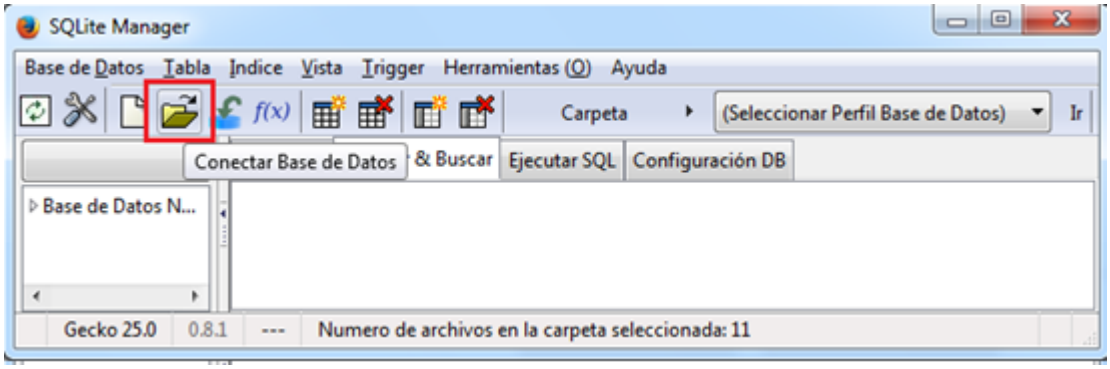


- Escribir el nombre de la base de datos en este caso "datostaxi" y dar clic en Aceptar.

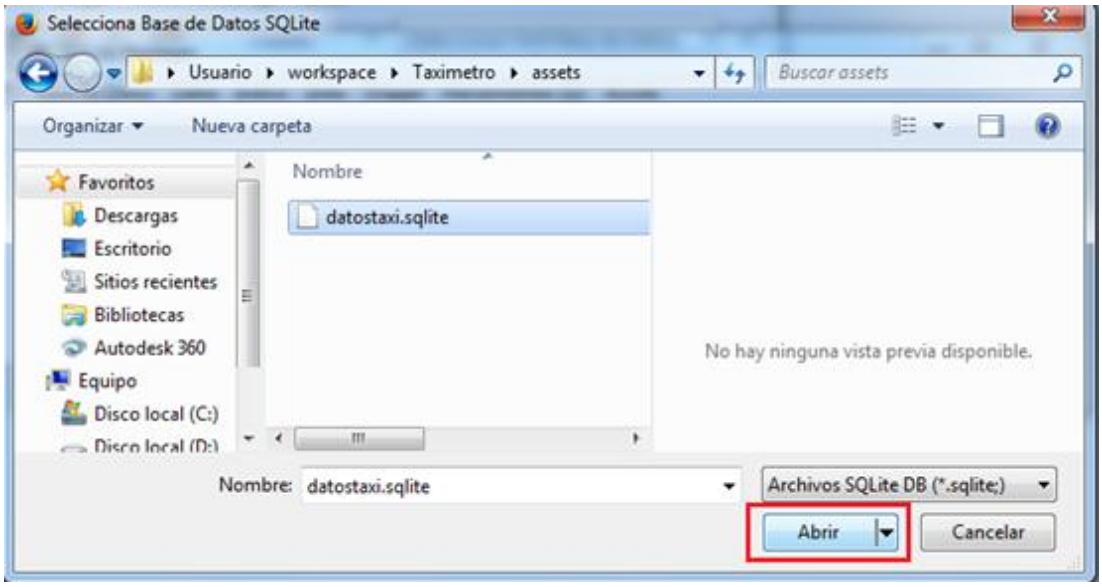


Para crear las tablas de la base de datos podemos hacerlo mediante un script o mediante el entorno gráfico.

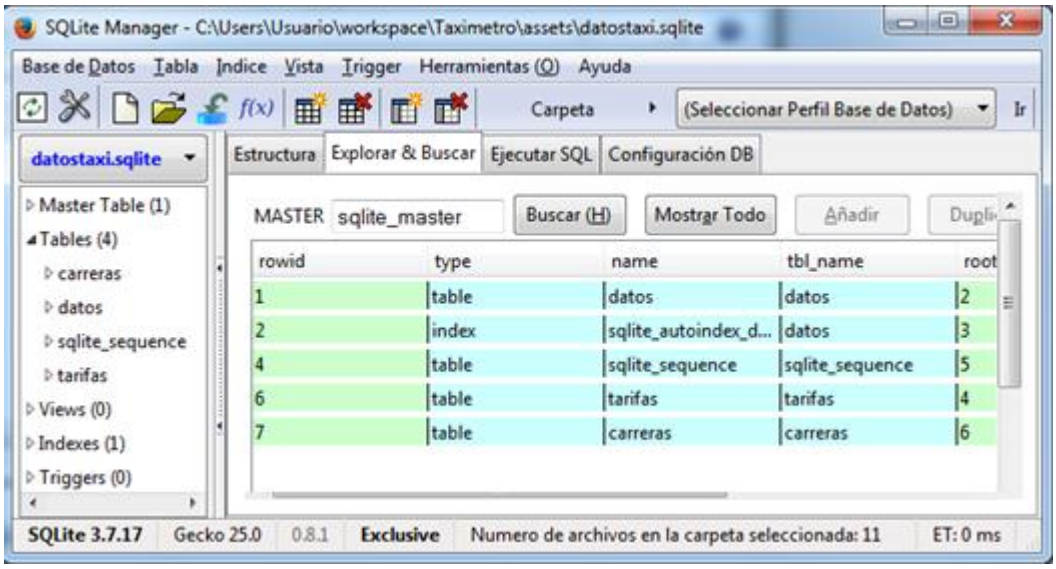
- Para abrir una base de datos existente, dar clic en "Conectar Base de Datos" como se muestra en la figura:



- Escoger la ubicación del archivo de la Base de Datos en este caso dirigirse al proyecto *Taximetro-assets-datostaxi.sqlite* y dar clic en "Abrir":



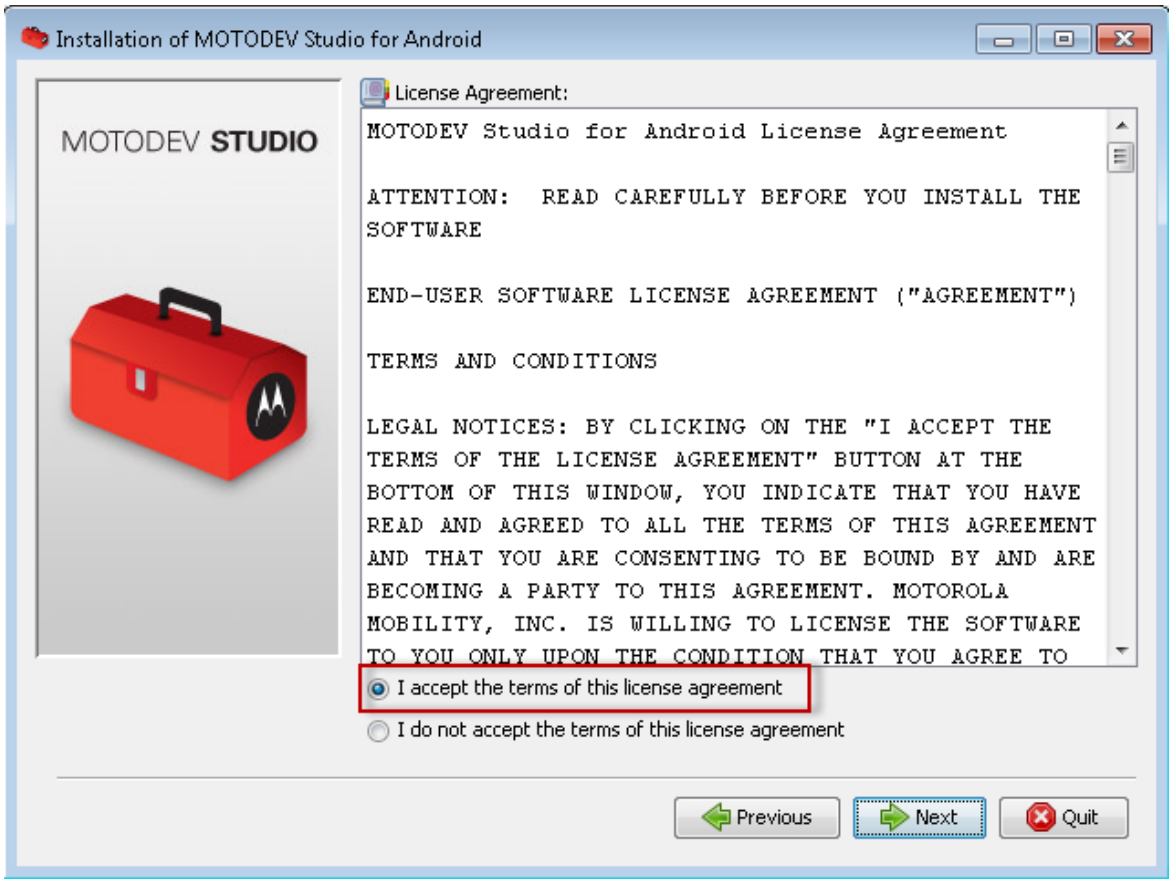
- Y listo ya se podrá visualizar y administrar la base de datos "datostaxi.sqlite".



INSTALACION DEL IDE DE DESARROLLO MOTODEV STUDIO

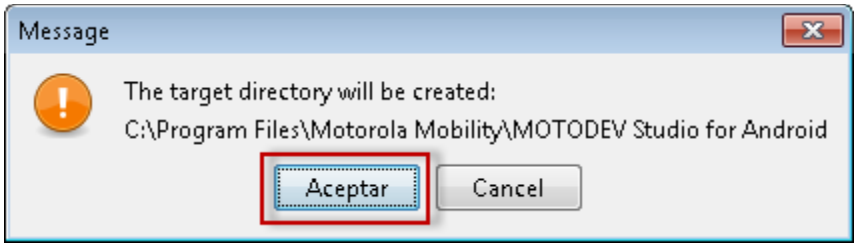
- Instalar y configurar el Kit de Desarrollo de Java JDK
- Instalar el IDE de desarrollo Motodev Studio, una vez descargado abrir el ejecutable:

- El proceso de instalación es sencillo, es el clásico siguiente siguiente. En la ventana que se muestra a continuación se acepta la licencia y dar clic en Next:

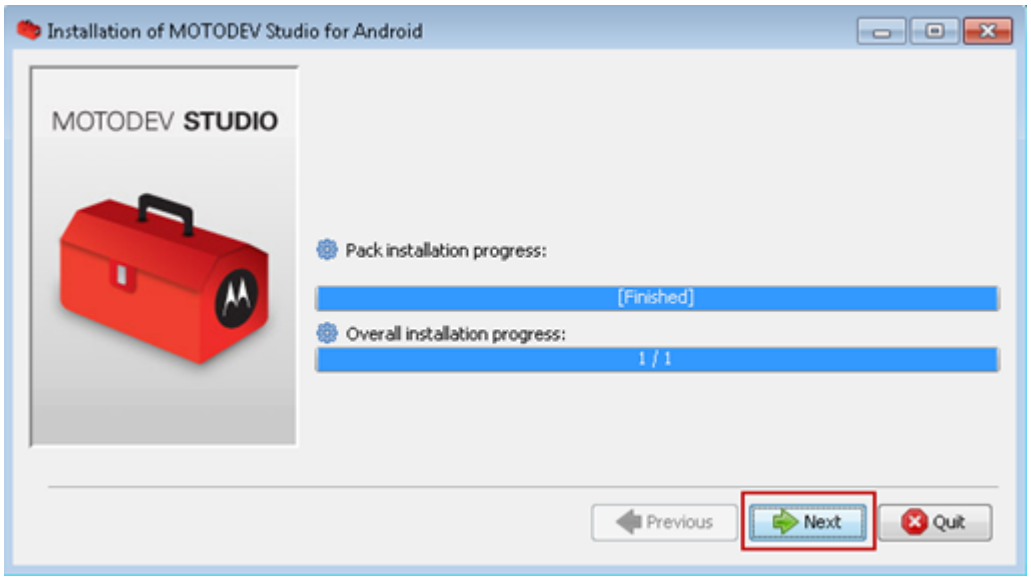


- En la siguiente ventana se especifica la dirección en la que se va a instalar Motodev Studio:

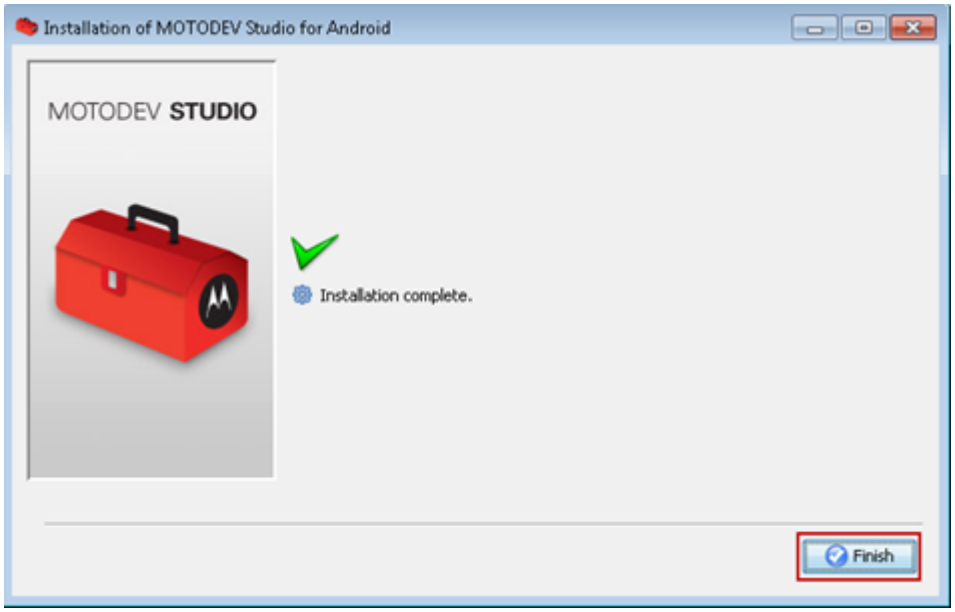
- Si la ruta de instalación no existe, el programa notifica que se creará el directorio:



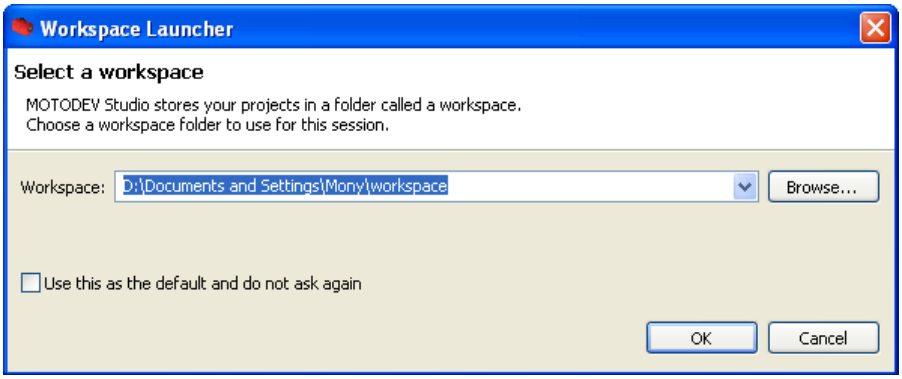
- En las siguientes ventanas dar clic en Next, hasta llegar a la ventana donde comenzará el proceso de instalación, una vez que haya finalizado dar clic en siguiente:



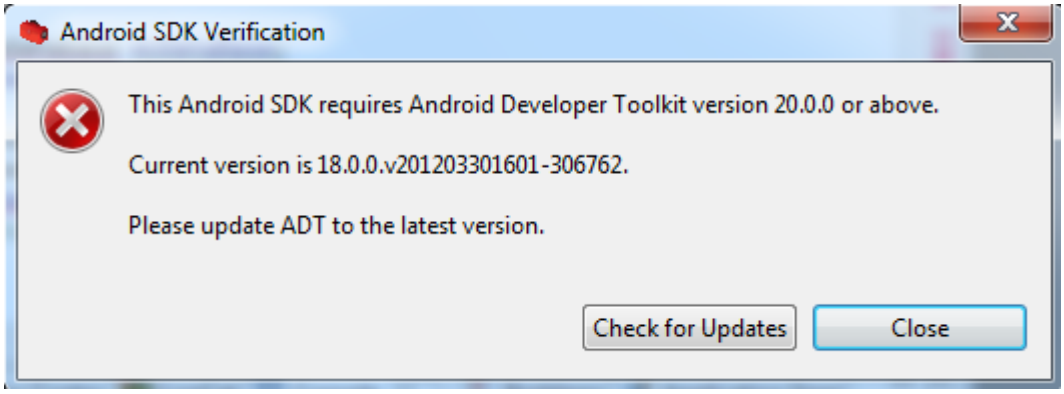
- Una vez que la instalación ha terminado dar clic en Finish:



- Una vez instalado abrir el IDE, aparecerá una ventana en la que muestra la dirección donde se guardarán los proyectos por defecto "workspace", si se desea se puede escoger otra ruta:



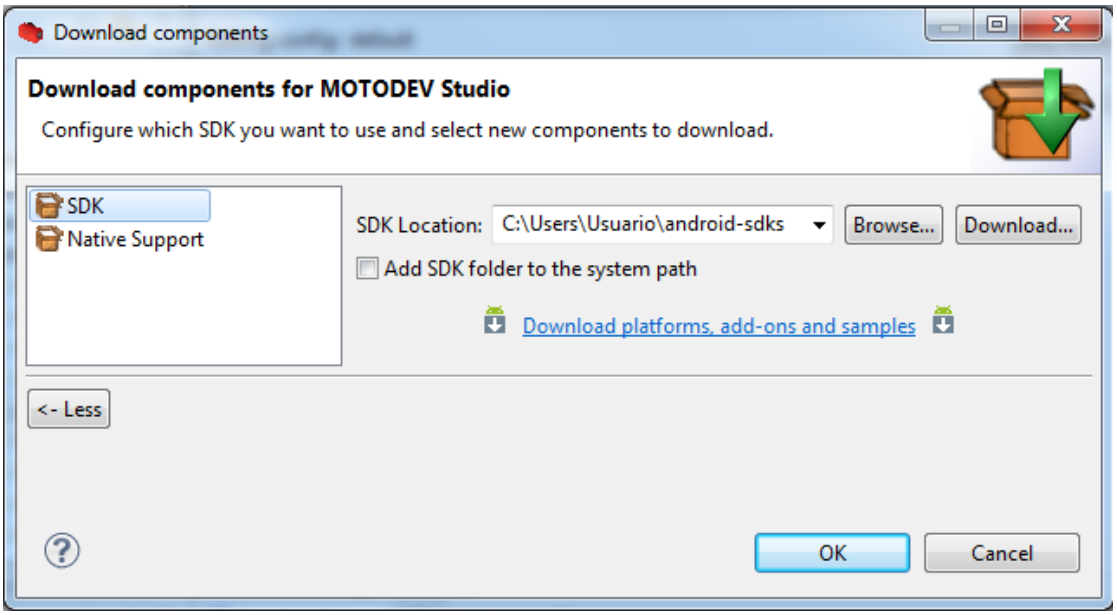
Una vez que se abre el entorno de desarrollo, aparecerá una ventana en la que se escoge la opción "*Close for Updates"*, esto es porque entre los componentes actualizados, no actualiza el ADT (Android Developer Toolkit).



Reiniciar Motodev Studio.

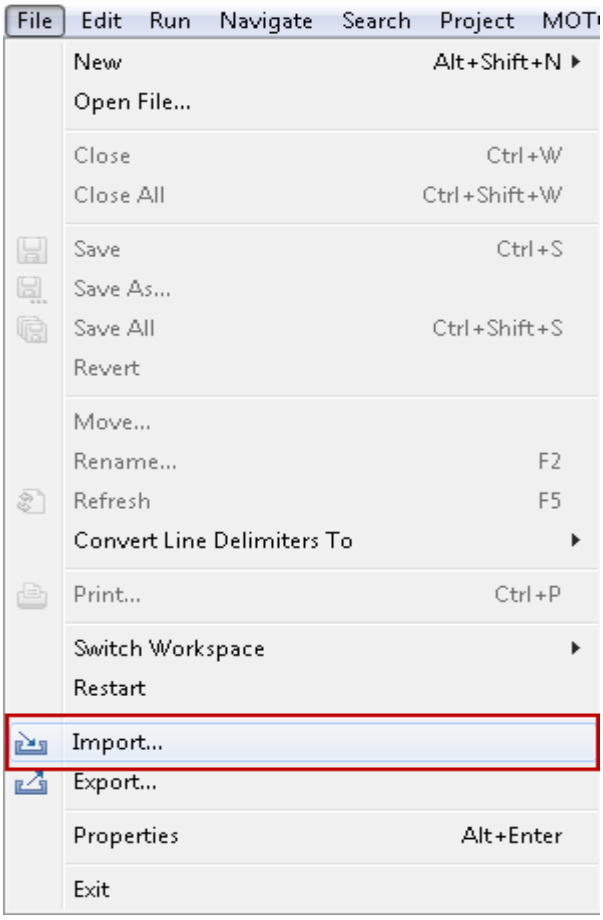
- Al iniciar Motodev Studio nuevamente aparecerá una ventana de configuración, en la cual se puede descargar el sdk y las APIs de google si no se las ha descargado

antes, pero si ya está descargado se puede seleccionar la ruta donde se encuentra el SDK y automáticamente reconoce todas las versiones que se encuentran disponibles:

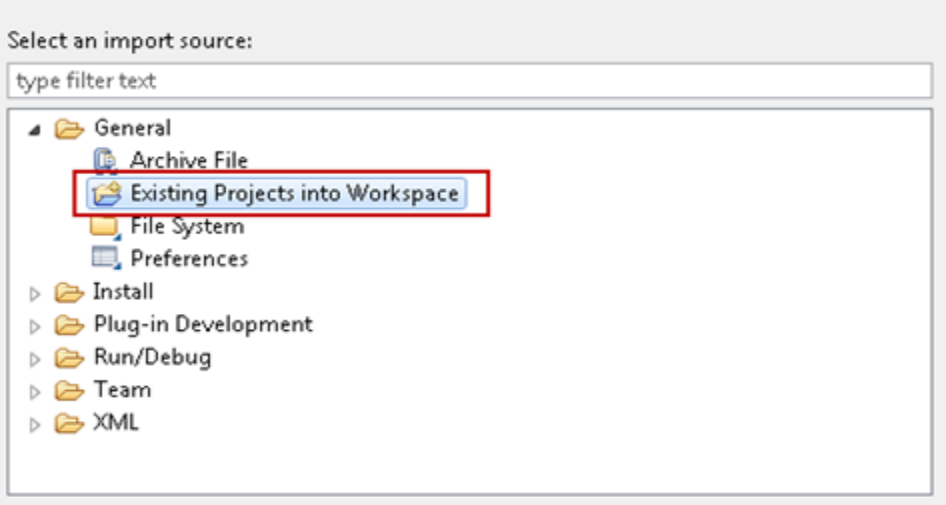


- Una vez que se ha terminado de instalar los paquetes que se necesita, se procede a crear el dispositivo virtual, como se explica en el capítulo 3 enunciado 3.2:

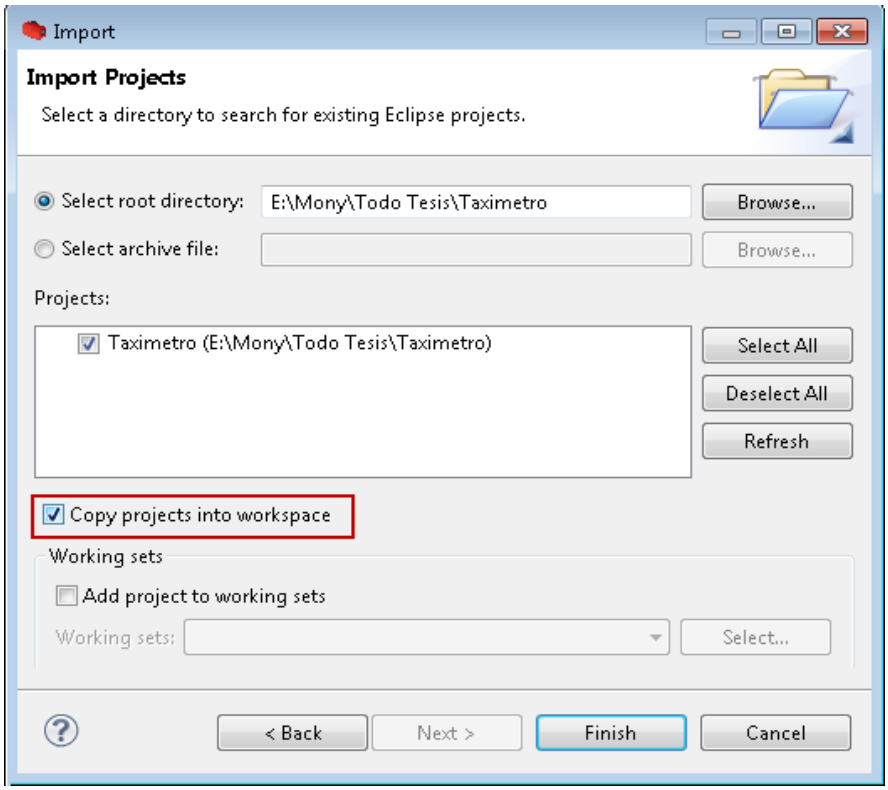
- Una vez creado el AVD, ya se podrá empezar a realizar aplicaciones Android, si ya se tiene un proyecto y se desea abrir la aplicación en IDE, se procede a importarlo: dirigirse a "File" y escoger la opción "Import" como se muestra en la figura:



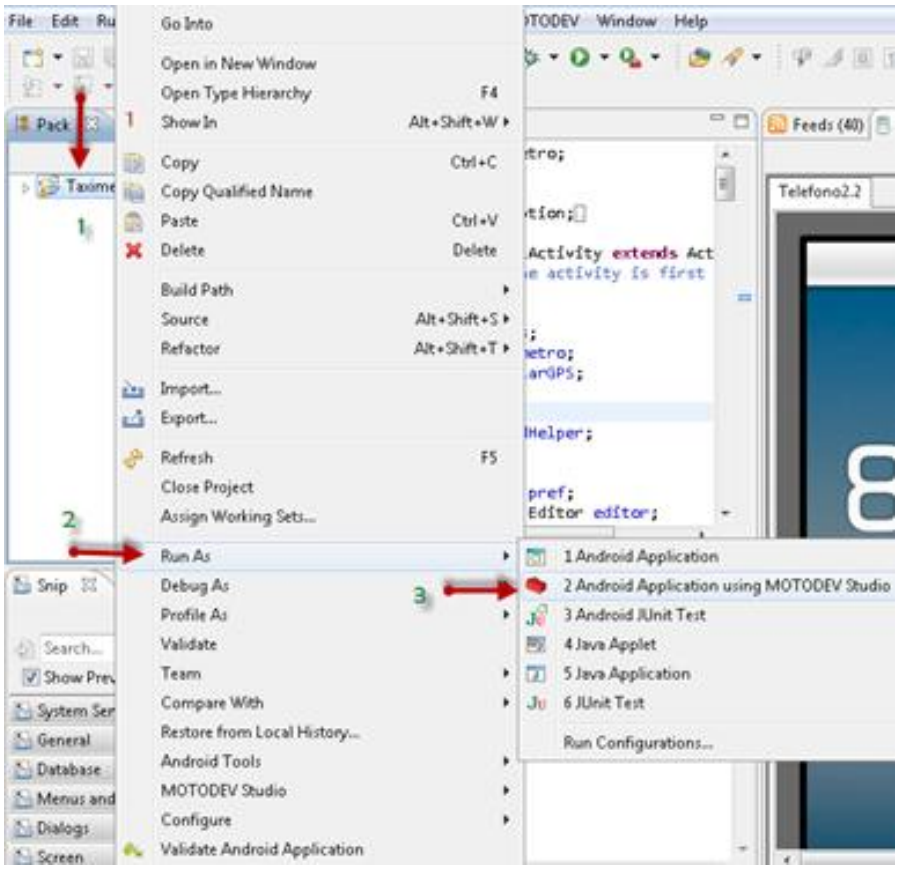
- En la siguiente ventana seleccionar la opción "General" y escoger un proyecto existente en el workspace:



- En la siguiente ventana seleccionar la ruta en la que se encuentra el proyecto, luego dar clic en la opción "Copy projects into *workspace*" para que se copie el proyecto en el *workspace* y finalizar:



Una vez que se haya cargado el proyecto ya se lo podrá correr haciendo clic derecho sobre el *proyecto –Run As - Android Aplication using MOTODEV Studio*, y listo.


Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Assign Working Sets...
Run As
Debug As
Profile As
Validate
Team
Compare With
Restore from Local History...
Android Tools
MOTODEV Studio
Configure
Validate Android Application
1 Android Application
2 Android Application using MOTODEV Studio
3 Android JUnit Test
4 Java Applet
5 Java Application
6 JUnit Test
Run Configurations...
1
2
3


TAXIMETRO
Activar GPS
SMS
Configuración
Datos Taxi
Taxímetro GPS

PUBLICAR LA APLICACIÓN EN GOOGLE PLAY

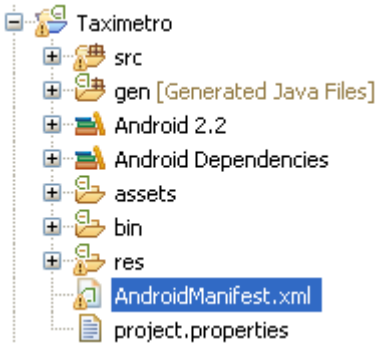
Una vez que ya se tiene lista la aplicación, el paso siguiente es publicarla para que los usuarios empiecen a utilizarla. El primer paso que se debe hacer es firmar la aplicación para luego publicarla en Google Play.

➢ Firmar una Aplicación Android

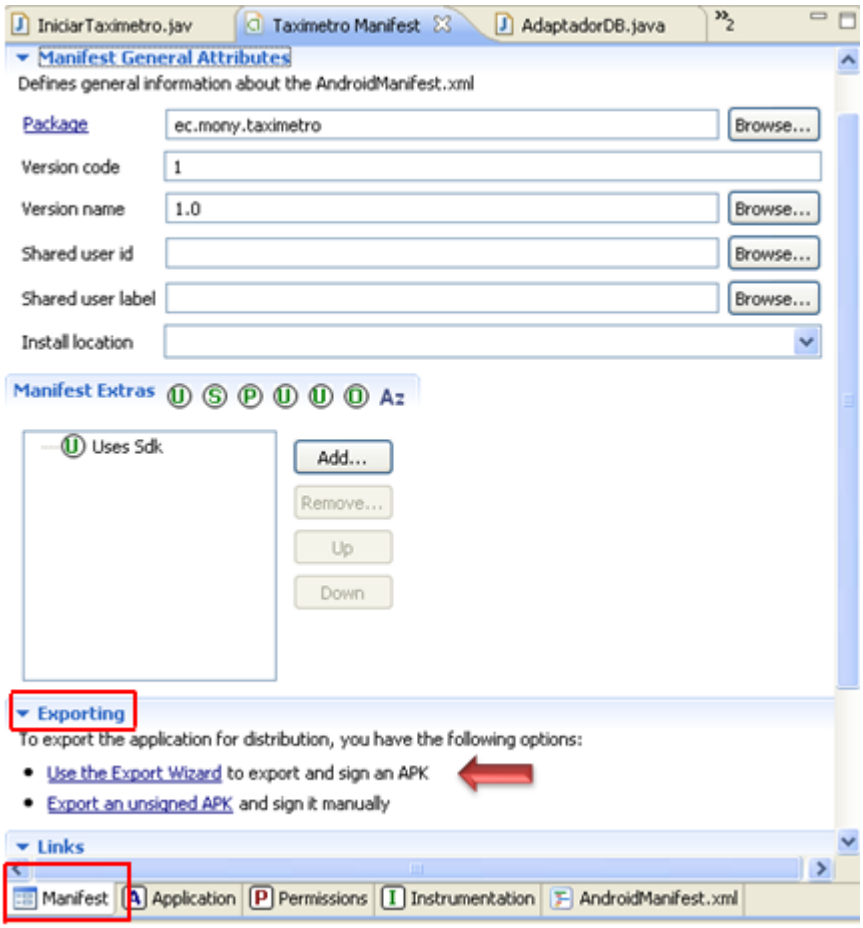
Se firman las aplicaciones como medida de seguridad y como requisito de garantía, para poder distribuir e instalar la aplicación sin problemas, para que de esta manera solamente el autor de dicha aplicación pueda modificarla y actualizarla.

Los pasos para firmar la aplicación son los siguientes:

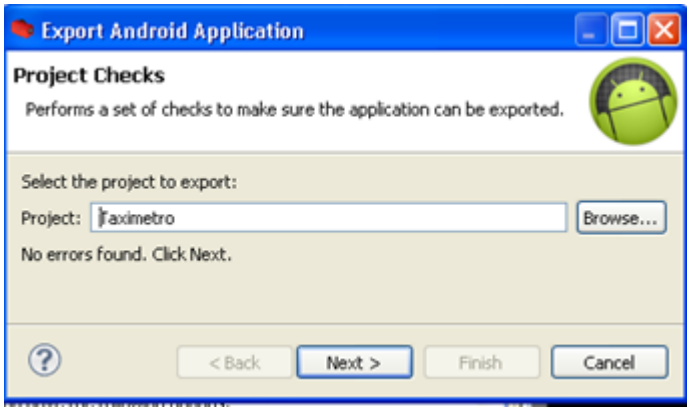
1. Abrir el IDE de desarrollo Motodev Studio, en el árbol de directorios abril el archivo AndroidManifest.xml.



2. Ubicarse en la primera pestaña llamada *Manifest* y en la sección *Exporting* como se muestra en la imagen:



3. Dar clic en la opción *Use the Export Wizard*.

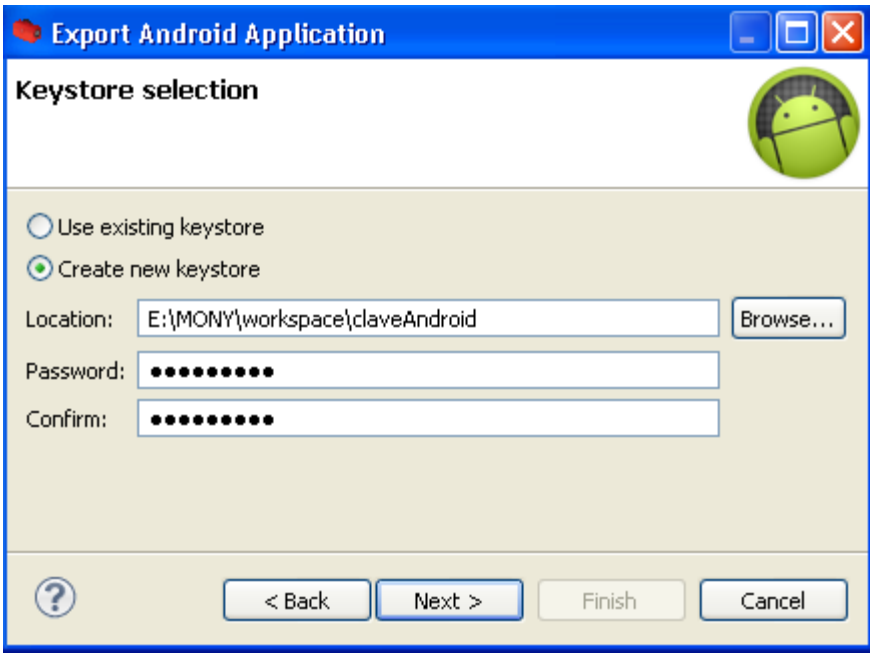


En la primera pantalla se encuentra seleccionada de forma automática la aplicación en la que se está trabajando, hacer clic en el botón Siguiente.

4. Es necesario para firmar una aplicación tener una keystore, este es un almacén de claves en dónde se encuentran todos los certificados validados que se pedirán. Como es la primera vez que se va a firmar la aplicación se debe crear una keystore. Clic en la opción Create new keystore y llenar los campos:

   Location: Es el directorio en dónde deseamos que se guarde la keystore y se le da un nombre cualquiera.

   Password: keystore deberá tener una contraseña de 6 dígitos o más.



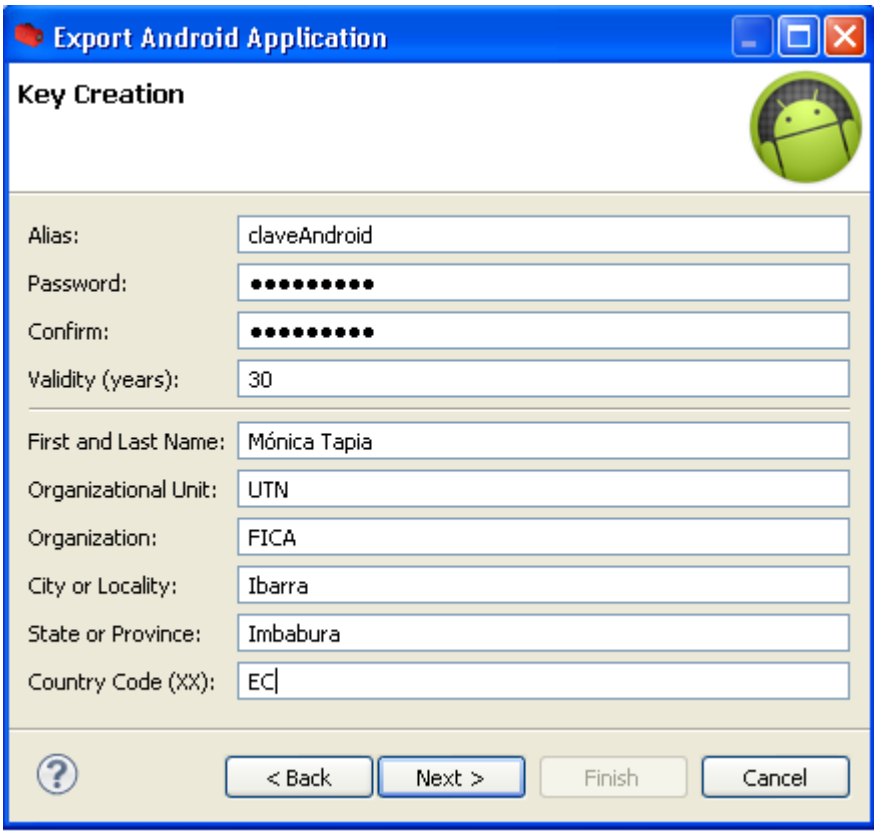
5. A continuación se describen los datos que se debe llenar en la siguiente pantalla:

   Alias: Un alias para la keystore, puede ser el mismo que el del nombre o una abreviación del mismo.

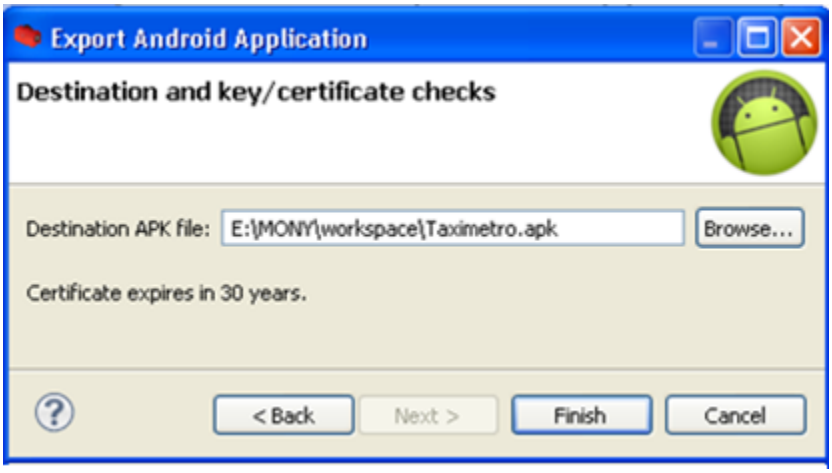
   Password: Nuevamente se asigna una contraseña.

   Validity (years): En esta parte definimos la duración de la validación del keystore en años.

Los siguientes campos corresponden a información personal y de la organización o empresa.



6. En la siguiente pantalla, se escoge el directorio de destino dónde se va a guardar el archivo ".apk" firmado y también se indicará cuándo expirará el certificado. Dar clic en Finalizar y ya está firmada la aplicación.



PUBLICAR LA APLICACIÓN

- **Crear cuenta de Google**

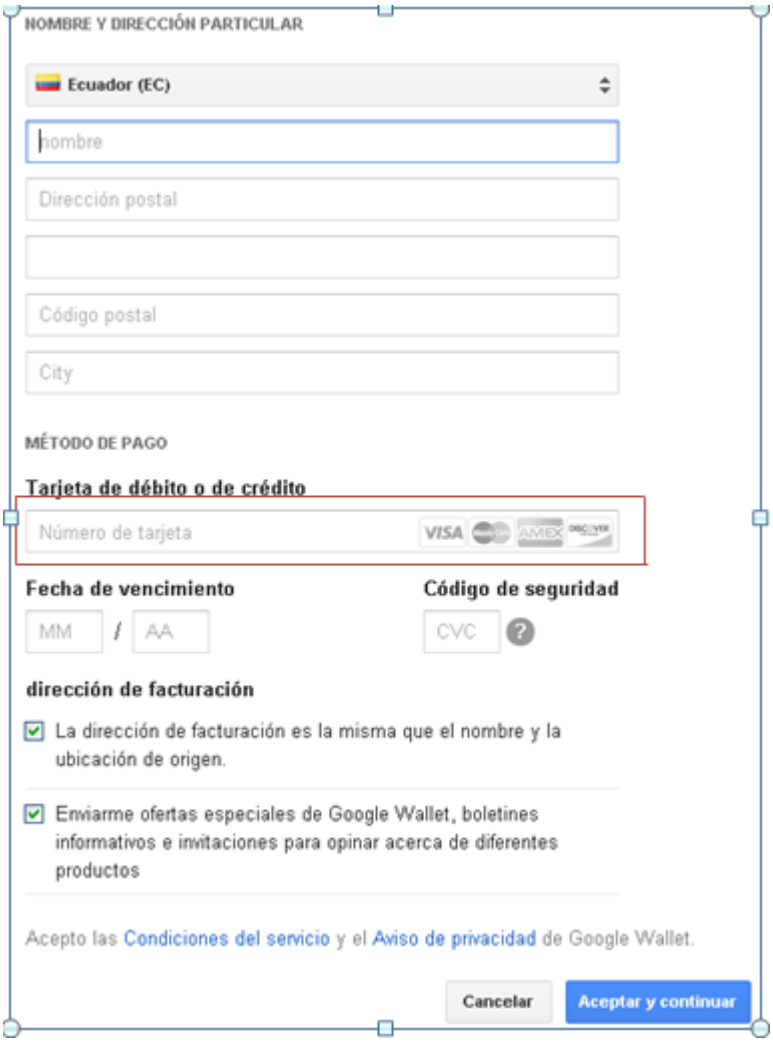
  Puede ser una cuenta de GMail, Youtube, Google Apps, entre otras. Esta cuenta estará asociada a la aplicación y servirá para realizar futuras modificaciones.

- **Acceder a la consola Google Play para desarrolladores**

  La consola de Google Play será el centro donde se controle todas las apps, su estado y estadísticas. Se accede desde **https://play.google.com/apps/publish/v2/** con la cuenta *gmail*. Cuando se accede a la cuenta por primera vez, se debe registrar dicha cuenta como desarrollador.

  El primer paso es aceptar el acuerdo para desarrolladores:

El segundo paso es pagar una cuota de $25 mediante tarjeta de crédito:



- **Subir app**

  Luego de haber realizado los pasos anteriores ya se podrá subir la aplicación, hacer clic en Añadir nueva aplicación y subir el archivo APK firmado (el tamaño máximo de un archivo APK es de 50 MB) y darle un nombre:

Seguidamente escoger la opción Entrada en Play Store:

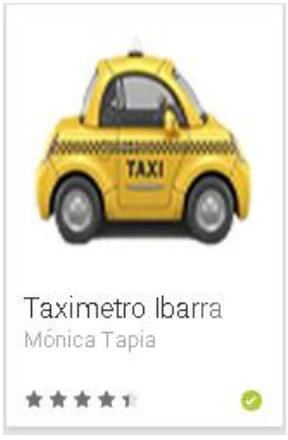
En esta opción se procede a llenar la siguiente información:

Nombre	Descripción
Idioma	El idioma predeterminado es el inglés de EE.UU. Se puede incluir traducciones del nombre y de la descripción de la aplicación para promocionarla para usuarios que hablen otros idiomas. Las traducciones se mostrarán en Google Play
Nombre	Es el nombre de la aplicación, como va a aparecer en Google Play. Se puede añadir un nombre por idioma
Descripción	Es el detalle descriptivo de la aplicación. En este campo, se puede utilizar un máximo de 4.000 caracteres

Cambios recientes	Se puede añadir notas sobre los cambios específicos de la versión más reciente de la aplicación.
Texto promocional	Es el texto situado junto al gráfico promocional en un lugar destacado de Google Play.
Tipo de aplicación	Google Play se divide en dos tipos principales de aplicaciones: Aplicaciones y Juegos.
Categoría	Se debe elegir una categoría para la aplicación.
Elementos Gráficos	Capturas de pantalla (mínimo 2, máximo 8) JPEG o PNG de 24 bits (no alpha) longitud mínima para los laterales: 320 píxeles, longitud máxima para los laterales: 3840 píxeles, y el icono de alta resolución (512x512) PNG de 32 bits (alfa).
Información del Contacto	Sitio Web, correo y teléfono del desarrollador

Una vez llenados todos los datos descritos anteriormente, se procede a la publicación de la aplicación.

Finalmente la aplicación Taxímetro estará disponible en Google Play en la siguiente dirección web: https://play.google.com/store/apps/details?id=ec.mony.taximetro&hl=es



Fuente: Android Developer: Cómo subir aplicaciones http://support.google.com/googleplay/android-developer/answer/113469?hl=es

ANEXO B

MANUAL DE USUARIO

SISTEMA TAXÍMETRO PARA USUARIOS FINALES

INDICACIONES GENERALES

El sistema Taxímetro ayudará a que el servicio de taxis sea eficiente en cuanto al cálculo del valor de la carrera a pagar, este sistema es más justo para el conductor del taxi y para el usuario.

TIPOS DE USUARIOS

El sistema cuenta con un tipo de usuario como es el Usuario final o sea el pasajero.

INGRESO AL SISTEMA

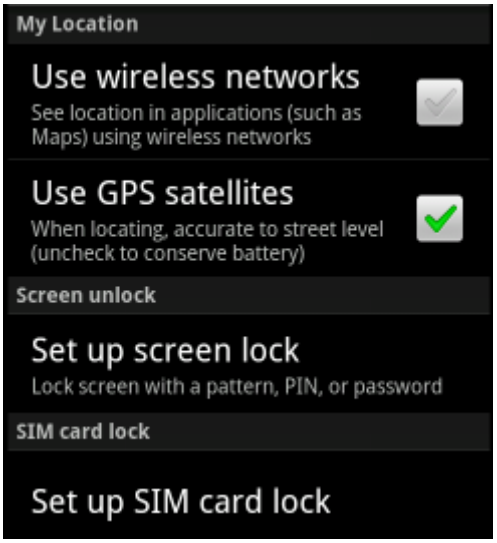
1. Primero se debe descargar e instalar la aplicación "Taxímetro Ibarra" de la página de Google Play.
2. Para ingresar al sistema se debe presionar en el siguiente icono:

3. Se desplegará la siguiente pantalla de Inicio.

DESCRIPCIÓN DE LAS OPCIONES DEL MENÚ

- 

Activar GPS.- Esta opción le permite al usuario activar el servicio de GPS, necesario para iniciar el Taxímetro.



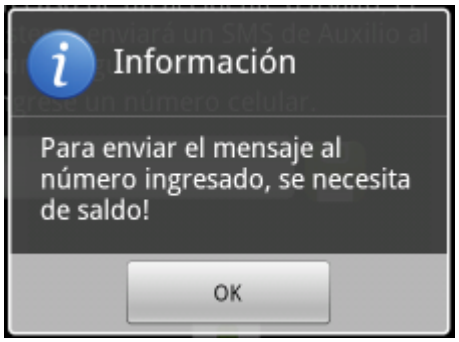
Nota: En caso de que el usuario no activara el servicio GPS el Taxímetro no iniciará.

- 

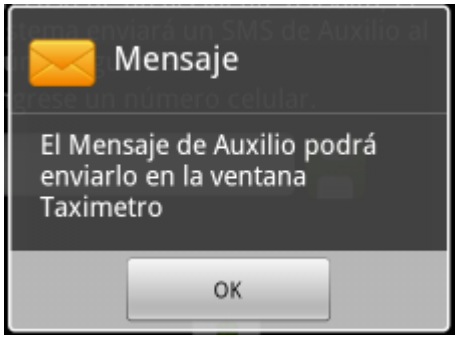
Configuración SMS.- Esta opción le permite al usuario ingresar un número celular para él envió del SMS de auxilio en caso de un accidente.

Sub Menú: Esta ventana cuenta con dos submenús, los cuales se muestran al presionar la tecla menú del dispositivo.

Menú Acerca de: Contiene con una breve información dirigida al usuario.



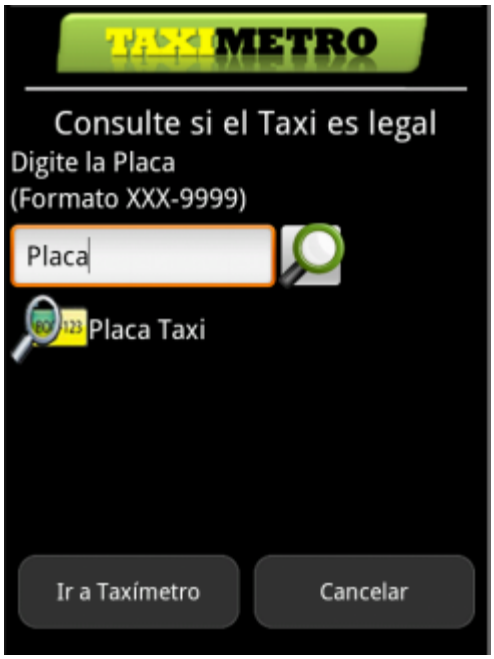
Menú SMS: Contiene una breve información dirigida al usuario.



Nota: En caso de que el usuario no haya ingresado ningún número, el servicio *Enviar SMS Auxilio* no realizará ninguna acción.

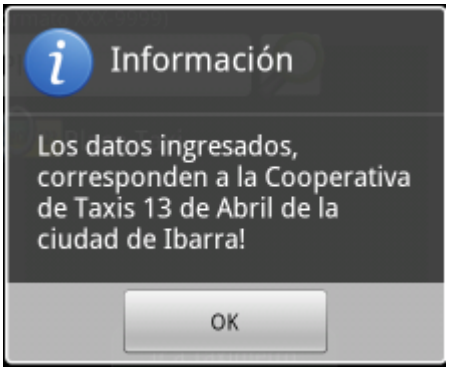
- 

Datos Taxi.- Esta opción le permite al usuario ingresar la placa del Taxi, para obtener sus datos y de esta manera verificar si el Taxi que abordó es legal o no.



Sub Menú: Esta ventana cuenta con dos submenús, los cuales se muestran al presionar la tecla menú del dispositivo.

Menú Información: Contiene con una breve información dirigida al usuario.



Menú Datos SMS: contiene una breve información dirigida al usuario.

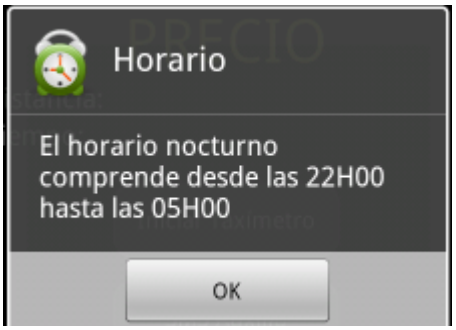
- **Taxímetro GPS.-** Esta opción le permite al usuario iniciar el Taxímetro, escoger el tipo de horario diurno o nocturno y enviar el mensaje de Auxilio en caso de un accidente.

  Nota: Una vez que se ha iniciado el taxímetro, el botón Terminar Taxímetro se bloqueará hasta que el usuario desbloquee el candado para que pueda terminarlo.

Sub Menú: Esta ventana cuenta con submenús, los cuales se muestran al presionar la tecla menú del dispositivo.

Menú Acerca de: contiene una breve información dirigida al usuario.

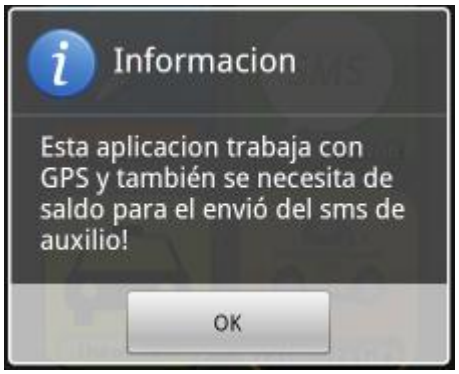
Menú Horario: contiene una breve información dirigida al usuario.



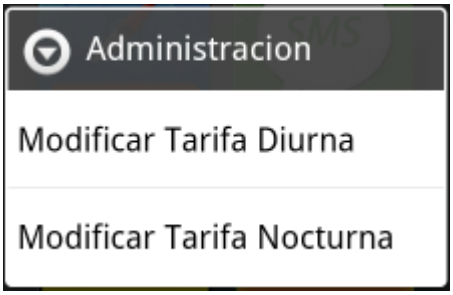
Menú Historial: En esta opción se mostrará todas las carreras q el usuario ha realizado hasta la fecha.

- **Sub Menús en la pantalla Principal:**

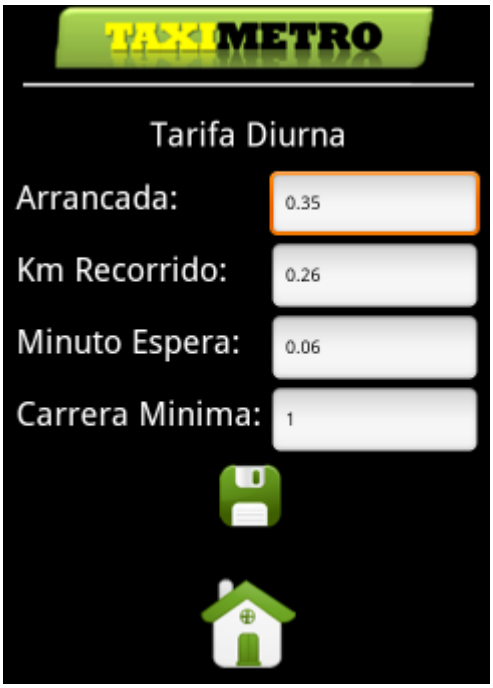
Menú Acerca de: contiene una breve información dirigida al usuario.



Menú Administración: Esta opción muestra un submenú donde se da a escoger la tarifa que el usuario desea modificar:

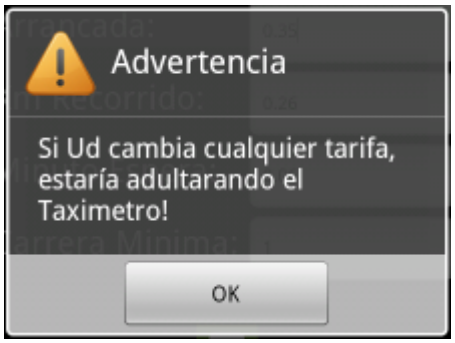


- ✓ **Tarifa Diurna.-** Esta opción permite al usuario modificar las diferentes modalidades de Tarifas para el cálculo del precio de la carrera diurna.



Sub Menú: Esta ventana cuenta con dos submenús, los cuales se muestran al presionar la tecla menú del dispositivo.

Menú Advertencia: Contiene un mensaje de advertencia dirigido al usuario.

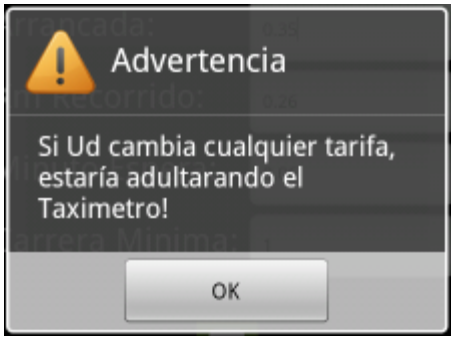


Menú Tarifa Vigente: Contiene los valores de las tarifas vigentes.

- ✓ **Tarifa Nocturna.-** Esta opción permite al usuario modificar las diferentes modalidades de Tarifas para el cálculo del precio de la carrera nocturna.

Sub Menú: Esta ventana cuenta con dos submenús, los cuales se muestran al presionar la tecla menú del dispositivo.

Menú Advertencia: Contiene un mensaje de advertencia dirigido al usuario.



Menú Tarifa Vigente: Contiene los valores de las tarifas vigentes.

ANEXO C

ANTEPROYECTO

Universidad Técnica del Norte

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

ANTEPROYECTO DE TRABAJO DE

GRADO DATOS GENERALES

1. TEMA: Estudio y desarrollo de aplicaciones para dispositivos móviles Android.
2. ÁREA/LÍNEA DE INVESTIGACIÓN: Investigación / Desarrollo de Software
3. ENTIDADQUEAUSPICIA: Cooperativa de Taxis "28 de Abril" Nº 30
4. DIRECTOR: Ing. Pedro Granda
5. AUTOR: Mónica Lucía Tapia Marroquín **DIRECCIÓN:** García Moreno 10-09 **TELÉFONO:** 095559427 **CORREOELECTRÓNICO:** mony_t0204@hotmail.com LUGARDETRABAJO: TELÉFONOTRABAJO: DIRECCIÓNDETRABAJO:
6. DURACIÓN: 9 meses
7. INVESTIGACIÓN: Nueva(X) Continuación ()
8. PRESUPUESTO:$2380

PARA USO DEL CONSEJO ACADÉMICO	
FECHA DE ENTREGA:	FECHA DE REVISIÓN:
APROBADO:SI ( ) NO( )	FECHA DE APROBACIÓN:
OBSERVACIONES:	

Universidad Técnica del Norte

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
ESCUELA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

PLAN DEL PROYECTO DE TITULACIÓN

Propuesto por: Mónica Lucía Tapia Marroquín	**Áreas Técnicas del Tema:** Investigación / Desarrollo de Software
Director Sugerido: Ing. Pedro Granda	**Fecha:** 24 de enero de 2012

1. Tema

Estudio y desarrollo de aplicaciones para dispositivos móviles Android.

2. Problema

2.1 Antecedentes

Los dispositivos móviles han evolucionado mucho en estos últimos años tanto en hardware como en software. El incremento de los servicios que se proveen hoy en día en Internet, el avance tecnológico, la tendencia hacia dispositivos más pequeños y más rápidos, junto con la necesidad del acceso a la información en cualquier lugar y momento, son las razones para el surgimiento de nuevas tecnologías para el acceso a la información y al internet desde cualquier tipo de dispositivos incluyendo a los teléfonos celulares, existen varios sistemas operativos para dispositivos móviles pero ninguno ha presentado tantas ventajas como Android.

2.2 Situación Actual

A medida que los teléfonos móviles crecen en popularidad, los sistemas operativos con los que trabajan alcanzan mayor importancia. Algunos sistemas operativos para dispositivos móviles más utilizados son Android en representación de Linux, iPhone OS

en representación de Mac OS X y Windows Mobile; de estos Android ha pasado a ser el sistema operativo más utilizado.

Android, más que un sistema operativo, representa todo un paquete de software para dispositivos móviles que incluye gran cantidad de drivers, gestor de bases de datos, un completo framework de aplicaciones, y numerosas aplicaciones de usuario. Está basado en el núcleo de Linux y todas sus aplicaciones se escriben en lenguaje Java, disponiendo además de una máquina virtual específica llamada Dalvik.

Google es el principal promotor de Android y pretende aprovechar al máximo la cada vez mayor capacidad de los dispositivos móviles, que llegan a incluir componentes como GPS, pantallas táctiles, conexiones rápidas a Internet, entre otras.

2.3 Prospectiva

Se espera que en los próximos años el acceso a Internet desde dispositivos móviles crezca exponencialmente y llegue a superar el uso de dispositivos convencionales como PCs de escritorio o Laptops.
Por lo cual es imprescindible realizar un estudio de las aplicaciones que se pueden desarrollar en este sistema operativo Android ya que su uso se extiende rápidamente, ya que es el sistema operativo más utilizado en dispositivos inteligentes.

3. Objetivos

3.1 Objetivo General

Estudiar el funcionamiento y las posibilidades que ofrece el sistema operativo móvil Android comenzando por conocer sus características hasta llegar al desarrollo de una aplicación que demuestre sus funcionalidades básicas.

3.2 Objetivos Específicos

- Realizar un estudio sobre la tecnología Android, conocer sus características, obtener ventajas y desventajas al momento de realizar una aplicación.
- Desarrollar una aplicación simulando un Taxímetro para el sistema operativo Android.
- Diseñar una Interfaz amigable para mejorar la usabilidad de la aplicación Taxímetro para aumentar el interés para crear aplicaciones para dispositivos móviles.
- Desarrollar el sistema con una metodología de desarrollo ágil como es Scrum.
- Generar los diagramas correspondientes al diseño del sistema.

4. Alcances

El sistema funcionara sobre dispositivos móviles que cuenten mínimo con la versión 2.2 (Froyo) de Android, se realizará la investigación y mediante el aplicativo de TAXÍMETRO se determinará como realizar una aplicación para Android.

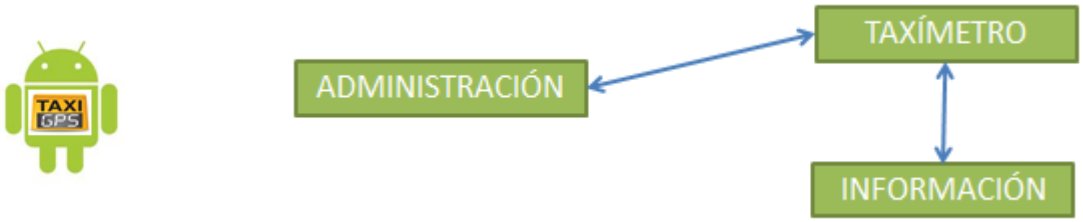
El sistema contara con los siguientes módulos:

- MÓDULO DE ADMINISTRACIÓN
  - ✓ Ingreso y actualización del Valor Mínimo

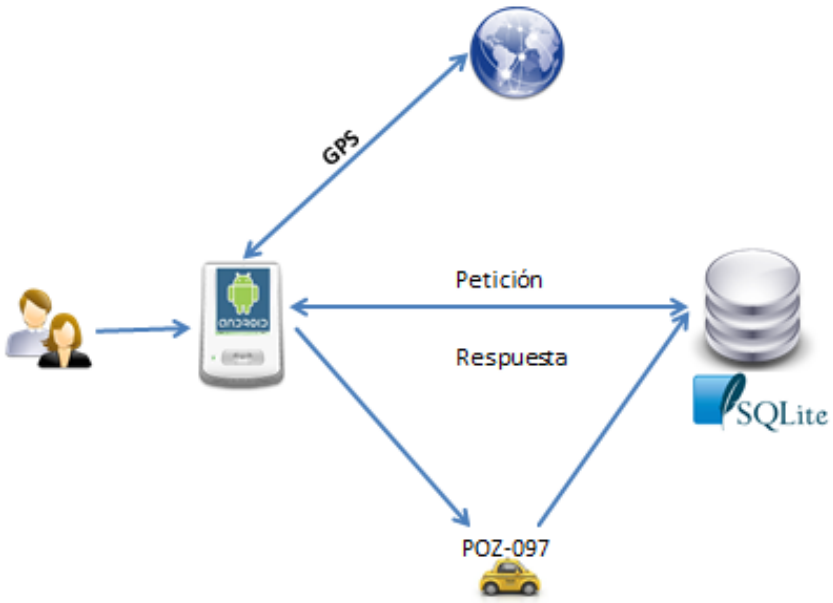
- MÓDULO DE TAXIMETRO
  - ✓ Calculo de tarifa

- MÓDULO DE INFORMACIÓN
  - ✓ Verificar datos de Taxi
  - ✓ Verificar Datos de Taxista

MÓDULOS:



ESQUEMA:



5. Justificación

Se justifica esta investigación debido a la importancia que van ganando los dispositivos móviles en el ámbito personal y profesional, brindándonos servicios que se hacen cada vez más indispensables en nuestras labores cotidianas. En nuestro país no se ha difundido lo suficiente el desarrollo de aplicaciones para dispositivos móviles que no sea Java ME,

además desde su lanzamiento Android ha ganado un espacio muy importante, empresas como HTC, ACER, HUAWEI, LG, MOTOROLA, SAMSUNG,SONYERICSSON, ALCATEL han adoptado a Android como sistema operativo para sus dispositivos.

La aplicación TAXÍMETRO se justifica ya que mucha veces al tomar un taxi es imposible saber cuánto costará la carrera hasta que se le pregunta al taxista, y al realizar el pago queda la duda si se pagó el importe adecuado por dicho servicio, ya que existen diversas formas de alterar los taxímetros, es por esto que esta aplicación servirá para hacer nuestro propio conteo de tarifa Taxímetro, aunque el precio se lo tomará como aproximación.

Las herramientas a utilizar son las siguientes:

Lenguaje de Programación:	Java
IDE de desarrollo:	Motodev Studio for Android
Gestor de Base de Datos:	SQLite
Versión del Sistema Operativo	Froyo+

6. Contexto

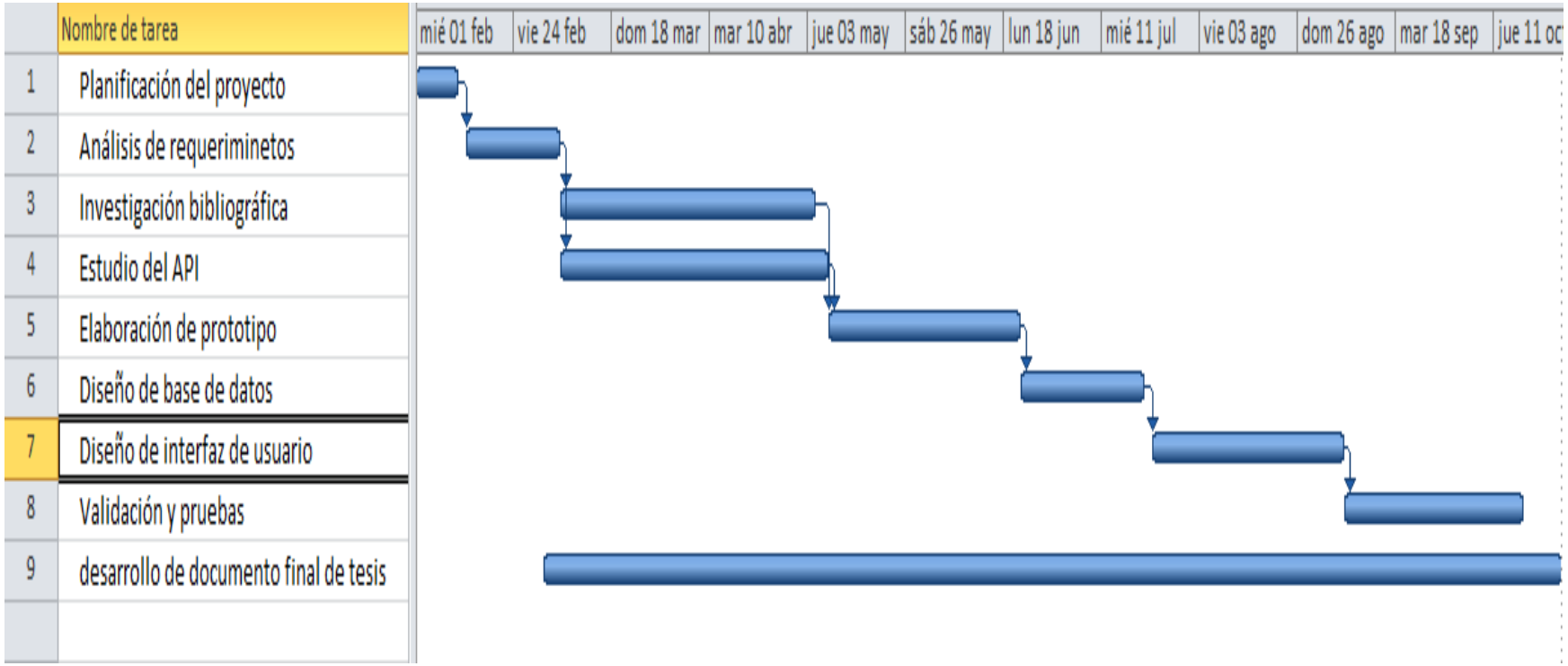
Tema	Realizado por	Tecnología
Análisis y estudio de transacciones seguras con el protocolo wap y tecnologías para aplicaciones web movil", con el aplicativo: Implementación y desarrollo de una aplicación wap-cmmerce Orientada al uso de tecnología celular.	Ramírez Velasteguí Mónica y Gudiño Silva Marcelo	.NET

Se realizará la investigación del desarrollo de aplicaciones con software libre con el cual se realizará la aplicación Taxímetro para el sistema operativo para móviles Android con acceso a una base de datos SQLite.

7. Contenidos

ÍNDICE

8 .Cronograma de Actividades



9. Presupuesto

Costos de Hardware

Descripción	Costo Empresa $	Costo Tesista $	Costo Real $
Computador Portátil	0.00	1000.00	1000.00
Computador Escritorio	0.00	0.00	0.00
Impresora	0.00	100.00	0.00
Celular con Android	0.00	700.00	700.00
Total de Hardware	**0.00**	**1800.00**	**1700.00**

Costos de Software

Descripción	Costo Empresa $	Costo Tesista $	Costo Real $
Internet	0.00	180.00	180.00
Motor de Base de Datos SQLite	0.00	0.00	0.00
IDE de desarrollo	0.00	0.00	0.00
Total de Software	**0.00**	**180.00**	**180.00**

Materiales de oficina

Descripción	Costo Empresa $	Costo Tesista $	Costo Real $
Resmas hojas de papel bond	2.00	20.00	20.00
Copias (documentos y libros)	0.00	80.00	80.00
DVD's, esferos	0.00	10.00	10.00
memoria flash	0.00	0.00	0.00
Total de Materiales de oficina	**2.00**	**110.00**	**110.00**

Varios

Descripción	Costo Empresa $	Costo Tesista $	Costo Real $
Movilización	0.00	60.00	60.00
Capacitación	0.00	80.00	80.00
Imprevistos	0.00	50.00	50.00
Empastado y Anillado	0.00	200.00	200.00
Infraestructura Física	0.00	0.00	0.00
Total de Varios	**0.00**	**390.00**	**390.00**
Total Costos	**2.00**	**2480.00**	**2380.00**

10. Bibliografía

Android

- Wikipedia. (2012). Android. Recuperado el 9 de enero de 2012, de http://es.wikipedia.org/wiki/Android

- AngelVilchez (2009). Que es Android: Características y Aplicaciones. Recuperado el 9 de enero de 2012, dehttp://www.configurarequipos.com/doc1107.html.

- OSL. (2011). Documentación del curso de Desarrollo Android. Recuperado el 10 de enero de 2012, de http://www.softwarelibre.ulpgc.es/cursos/android

- TxemaRodríguez (2012). Android en 2012: desarrollando aplicaciones. Recuperado el 10 de enero de 2012, de http://www.xatakandroid.com/aplicaciones-android/android-en-2012-desarrollando-aplicaciones

- Guru_Andrea. (2012). Versiones de Android.Recuperado el 11 de enero de 2012, de http://comunidad.movistar.es/t5/Blog-Android/TutorialQu%C3%A9-tiene-cada-versi%C3%B3n-de-Android/ba-p/435041

- Alkar. (2010). AndroidFroyo 2.2. Recuperado el 12 de enero de 2012, de http://www.genbeta.com/sistemas-operativos/android-22-froyo-un-repaso-a-las-principales-novedades

- Pablo Díaz. (2011). Android Gingerbread 2.3. Recuperado el 12 de enero de 2012, de http://www.tecnofans.es/android/articulo/android-gingerbread-2-3-mejoras-frente-a-froyo-2-2/25539/

SQLite

- Wikipedia. (2012). SQLite. Recuperado el 12 de enero de 2012, de http://es.wikipedia.org/wiki/SQLite

Taxímetro

- Wikipedia. (2011). Taxímetro. Recuperado el 12 de enero 2012, de http://es.wikipedia.org/wiki/Taxímetro