ESCUELA SUPERIOR POLITECNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y Computación

Estudio comparativo de dos plataformas de programación de dispositivos móviles

TESINA DE SEMINARIO

Previa a la obtención del título:

INGENIERÍA EN TELEMÁTICA

Presentado por:

Gregorio Pazmiño Vélez

Magdeline Rosero Pérez

Guayaquil - Ecuador
2013

AGRADECIMIENTO

A nuestro director de seminario de graduación Ing. Marcos Millán y a la Ing. Patricia Chávez por ayudarnos en el proceso de revisión del informe de seminario de graduación.

Al Sr. Bolívar Rosero y al Sr. Eduardo Bueno por habernos facilitado las herramientas necesarias para realizar las pruebas y poder realizar las respectivas comparaciones entre las plataformas Android y JME.

DEDICATORIA

A Dios.
A mi padre.
A mi hermano.
A mi madre, quien estuvo allí para apoyarme.
A mi sobrino, quien quiero con mucho amor.

Gregorio Pazmiño Vélez

A Dios.
A mis padres.
A mis hermanos, en especial a mi hermano que está en el cielo porque sé que estará feliz por mi meta alcanzada.

Magdeline Rosero Pérez

TRIBUNAL DE SUSTENTACIÓN

_____

Msc. Marcos Millán T.
Profesor del Seminario de Graduación

_____

Msc. Patricia Chávez B.
Profesora Delegada por la Unidad Académica

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta tesina, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Graduación de la ESPOL)

_____

Magdeline Rosero Pérez

_____

Gregorio Pazmiño Vélez

RESUMEN

Este proyecto tiene como objetivo principal la comparación, análisis e interpretación de los resultados entre dos plataformas de programación de dispositivos móviles usando análisis estadísticos para comprender de mejor manera la diferencias entre estas dos plataformas, ayudando a un programador de aplicaciones de dispositivos móviles a determinar que plataforma elegir para el desarrollo de su aplicación, teniendo en cuenta el auge, alta disponibilidad de herramientas, el mejor desempeño de la plataforma, entre otras variables.

Las plataformas que fueron comparadas son Android y JME, dos de las más usadas actualmente. Se realizaron las pruebas a cada plataforma comparándolas en base a ciertas variables seleccionadas que nos indican cuál de las dos plataformas de programación de dispositivos móviles proporciona mayor ventaja al programador en el momento de desarrollar su aplicación.

Con los resultados obtenidos se pudo conocer con mayor profundidad las características de cada una de estas plataformas, las ventajas y desventajas en diferentes aspectos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

Abreviatura	Significado
A-GPS	GPS Asistido (Assistance-GPS)
AOSP	Proyecto de Código Abierto Android (Android Open Source Project)
API	Interfaz de Programación de Aplicaciones (Application Programming Interface)
ARM	Máquina Avanzada para Ordenadores con Conjunto Reducido de Instrucciones (Advance RISC Machine)
AWT	Kit de Herramientas de Ventana Abstracta (Abstrac Windowing Toolkit)
CDC	Configuración del Dispositivo Conectado (Connected Device Configuration)
CDMA	Multiplexación por División de Código (Code Division Multiple Access)
CPU	Unidad Central de Procedimiento (Central Processing Unit)
CLDC	Configuración Limitada de Dispositivos Conectados (Connected Limited Device Configuration)
DPI	Puntos por Pulgada (Dots Per Inch)
EDGE	Tasas de Datos Mejoradas Para La Evolución de GSM (Enhanced Data Rates for GSM Evolution)
FM	Frecuencia Modulada (Frequency Modulation)
GATT	Perfil Genérico de Atributo (Generic Attribute Profile)

Gmail	Correo Google (Google Mail)
GPL	Licencia Pública General (General Public License)
GPRS	Servicio General de Radio por Paquetes (General Packet Radio Service)
GPS	Sistema de Posicionamiento Global (Global Positioning System)
GPU	Unidad de Procesamiento Gráfico (Graphics Processing Unit)
GSM	Sistema Global para las Comunicaciones Móviles (Global System for Mobile Communication)
HID	Dispositivo de Interfaz Humano (Human Interface Device)
HTC	High Tech Computer
IDC	Corporación de Datos Internacional (International Data Corporation)
IDL	Interfaz de Lenguaje de Descripción (Interface Description Language)
IM	Mensajería Instantanea (Instant Messaging)
JAR	Archivo Java (Java Archive)
JDK	Kit de Desarrollo Java (Java Development Kit)
JME	Edición Micro Java (Java Micro Edition)
JRE	Ambiente de Ejecución Java (Java Runtime Environment)
Java SE	Plataforma Java Edición Estándar (Java Platform Standard Edition)

J2EE	Plataforma Java 2 Edición Empresarial (Java 2 Platform Enterprise Edition)
J2ME	Plataforma Java 2 Edición Micro (Java 2 Platform Micro Edition)
J2SE	Plataforma Java 2 Edición Estándar (Java 2 Platform Standard Edition)
JVM	Máquina Virtual de Java (Java Virtual Machine)
KVM	Máquina Virtual Kilobytes (Kilobytes Virtual Machine)
LTE	Evolución a Largo Plazo (Long Term Evolution)
MIDP	Perfil de Dispositivo de Información Móvil (Mobile Information Device Profile)
MMS	Sistema de Mensajería Multimedia (Multimedia Messaging System)
NDEF	Formato de Intercambio de Datos NFC (NFC Data Exchange Format)
NDK	Kit de Desarrollo Nativo (Native Development Kit)
NFC	Comunicación de Campo Cercano (Near Field Communication)
OHA	Open Handest Alliance
OpenGL ES	Biblioteca Gráfica Abierta para Sistemas Embebidos (Open Graphics Library for Embedded Systems)
OS	Sistema Operativo (Operating System)
PDA	Asistente Personal Digital (Personal Digital Assistant)
PIM	Gestor de Información Personal (Personal Information Manager)

PKI	Infraestructura de Clave Pública (Public Key Infrastructure)
P2P	Par a Par (Peer-to-Peer)
QVGA	Quarter Video Graphics Array
RAM	Memoria de Acceso Aleatorio (Random Access Memory)
RDS	Sistema de Radiodifusión de Datos (Radio Data System)
ROM	Memoria de sólo Lectura (Read Only Memory)
RTP	Protocolo de Transporte en Tiempo Real (Real-Time Transport Protocol)
SDK	Kit deDEsarrollo de Software (Software Development Kit)
SGL	Biblioteca de Escenario Gráfico (Scene Graph Library)
SIP	Protocolo de Iniciación de Sesión (Session Initiation Protocol)
SMS	Servicio de Mensajes Cortos (Short Message Service)
SPP	Perfil de Puerto Serie (Serial Port Profile)
SSL	Capa de Conexión Segura (Secure Sockets Layer)
SVG	Gráficos Vectoriales Redimensionables Scalable (Vector Graphics)
VGA	Adaptador Gráfico de Video (Video Graphics Array)
VM	Máquina Virtual (Virtual Machine)
WIMAX	Interoperatibilidad Mundial para Acceso por Microondas (Worldwide Interoperability for Microwave Access)
XML	Lenguaje de Marcas Extensible (Extensible Markup Language)

INTRODUCCIÓN

En la actualidad existen gran cantidad de plataformas de desarrollo para aplicaciones de dispositivos móviles tales como BlackBerry OS, Android, Windows Phone, entre otros. ¿Será acaso que elegir entre una plataforma u otra, puede causar alguna pérdida? Pues cada persona puede programar en la plataforma que se sienta más confortable, pero el monto de las ganancias puede variar, ya que los desarrolladores esperan tener ingresos monetarios por el desarrollo de la aplicación (algunos son gratuitos, pero obviamente en la mayoría habita el pensamiento de obtener beneficio por su sacrificio); el porcentaje de ganancias puede variar dependiendo de la plataforma que se elija. Si no hay ganancias entonces, o se perdió tiempo programando la aplicación en una plataforma que no es tan usado o se hizo una aplicación cuyo funcionamiento no le gusta a nadie. Nos centraremos en la primera opción.

En este proyecto nos enfocamos en comparar las plataformas Android y JME las mismas que han sido elegidas debido a que su licencia es gratuita y a la gran aceptación a nivel global, con el principal objetivo de ayudar al programador a elegir correctamente en que plataforma desarrollar su aplicación; con cuál le tomará menor tiempo, con cuál puede obtener mayor ganancias o con cuál plataforma le será más fácil desarrollar la aplicación. En este documento se conoce la problemática que ha provocado la existencia de muchas plataformas y los objetivos a corto y largo plazo que presenta el desarrollo de este proyecto.

Antes de introducirnos por completo a la comparación entre las dos plataformas se realizó una breve descripción (historia, características, arquitectura) sobre ellas, con el fin de conocer el funcionamiento de Android y JME. Para la comparación, se escogió ocho variables que fueron analizadas y probadas una por una en cada plataforma.

CAPÍTULO 1

DESCRIPCIÓN GENERAL DEL PROYECTO

En este capítulo se detalla la problemática que surge al momento de la elección de la plataforma de programación de dispositivos móviles, los objetivos alcanzados durante el desarrollo del informe de seminario de graduación y las variables que fueron seleccionadas para la comparación de las plataformas Android y JME.

1.1. ANTECEDENTES

En la actualidad existen varias aplicaciones creadas para diferentes marcas de teléfonos móviles como son Nokia, Blackberry, Samsung, Iphone, Sony Ericsson entre otros. El incremento de fabricación de teléfonos móviles por diferentes empresas ha originado el desarrollo de múltiples plataformas para el

funcionamiento y la programación de aplicaciones en los respectivos dispositivos móviles. A partir de ahí un programador tiene varias opciones para seleccionar en que plataforma desarrollar su aplicación, pero el hecho de no haber realizado una buena elección para la programación de una aplicación implica pérdida de tiempo, crear aplicaciones que al final por los recursos utilizados tengan un valor alto en el mercado provocando que la misma no tenga acogida por parte de los usuarios finales, entonces el desarrollador debe implementar algún método de investigación para saber en qué plataforma realizar su aplicación, para de esta manera tener gran acogida, ganancias y menor tiempo de desarrollo.

1.2. Objetivos

Implementar, analizar y comparar dos plataformas de programación de dispositivos móviles con el fin de conocer que plataforma es la mejor opción para comenzar a desarrollar aplicaciones para dispositivos móviles.

Objetivos Específicos

- Comprender la arquitectura y funcionamiento de Android y JME.
- Evaluar el desempeño de las plataformas desde el punto de vista del programador.

- Analizar que plataforma de programación de dispositivos móviles me permite agregar mayor cantidad de funcionalidades a una aplicación.
- Recopilar información necesaria que será de ayuda al programador para el proceso de selección entre Android y JME.

1.3. Descripción del proyecto

Dentro de los resultados obtenidos por la IDC, el reporte nos indica que se han vendido 154 millones de dispositivos por todo el mundo, donde la marca que se encuentra liderando en el mercado a nivel mundial en el año 2013 es Samsung seguida de su competencia Apple y Nokia. En base a esta estadística escogimos las dos plataformas de programación de dispositivos móviles como son Android y JME ya que son las que más se destacan dentro del mercado internacional y además el tipo de licencia con la que cuentan dichas plataformas son gratuitas.

Las plataformas fueron sometidas a varias pruebas, las suficientes para tener una idea clara sobre el desempeño de cada plataforma. El método de comparación usado fue un método estadístico, obteniendo datos de media aritmética, varianza y desviación estándar; datos necesarios para el análisis comparativo. Después de haber hecho las respectivas comparaciones entre las

plataformas Android y JME se concluye que plataforma es mejor para cada variable analizada, ayudando a los programadores de dispositivos móviles elegir en que plataforma crear su aplicación.

1.4. Variables a analizar

Las variables a analizar son escogidas cuidadosamente para la comparativa entre las dos plataformas:

1) **APIs:** Se comparan las APIs de Android y JME, con esta variable daremos a conocer que plataforma posee mayor disponibilidad para el desarrollo de aplicaciones, tales como son sensores, acceso a la red, geolocalización, etc.

2) **Licencia:** Identificaremos que tipo de licencia presenta cada plataforma, es decir si son de código abierto, si son gratuitas o tienen algún costo.

3) **Lenguaje de programación:** Se observará que lenguaje de programación se usa para el desarrollo de aplicaciones móviles en cada plataforma.

4) **Líneas de código:** Visualizar y comparar la cantidad de líneas de código que se utiliza en una aplicación determinada tanto en Android como en JME.

5) **Tamaño de aplicación:** Con esta variable tenemos conocimiento que aplicación, la misma para Android y JME, ocupa mayor espacio en memoria.

6) **Emulación:** En esta variable tomamos el tiempo que tarda el emulador en ejecutar una aplicación, ya que el cálculo de velocidad y tamaño que ocupa en memoria son limitaciones que presenta el emulador.

7) **Conectividad:** Se toma en cuenta que tipos de conectividad se encuentra disponible en ambas plataformas, nos permitirá conocer las limitantes que tienen Android y JME.

8) **Múltiples pantallas:** Conociendo que existen diferentes tamaños de pantallas en los dispositivos móviles, se analiza si se debe realizar ajustes en el código para adaptar la visualización del programa en el dispositivo móvil.

CAPÍTULO 2

PLATAFORMAS DE PROGRAMACIÓN DE DISPOSITIVOS MÓVILES: ANDROID Y JME

En este capítulo se ilustra una breve descripción sobre cómo fueron evolucionando las plataformas Android y JME. Además se muestra sus arquitecturas, los tipos de máquinas virtuales que utilizan cada una de ellas y las configuraciones que se emplean de acuerdo a la capacidad de memoria y capacidad de procesamiento de una categoría específica de dispositivos.

2.1. ¿Qué es Android?

Android es un sistema operativo que fue pensado para teléfonos móviles y está basado en el kernel de Linux. Además, se utiliza una máquina virtual personalizada que fue diseñada para optimizar los

recursos de memoria y el hardware en un entorno móvil. Android puede ser extendido liberalmente para incorporar nuevas tecnologías de vanguardia que van surgiendo. La plataforma continuará evolucionando a medida que la comunidad de desarrolladores continúen trabajando para construir aplicaciones móviles innovadoras. [1]

2.2. Breve Historia de Android

En el 2005 Google compra una pequeña empresa llamada Android Inc. En el año 2007 se lanzó la Open Handset Allience, que es una empresa conformada por varios fabricantes de teléfonos celulares, en esa fecha se lanzó la primera versión de Android junto con su SDK, con la finalidad de que los programadores empiecen a crear sus propias aplicaciones. Desde el año 2007 Google ha lanzado varias versiones de Android, llegando a ser aceptada por los usuarios poco a poco. Android ya comenzaba a ganar terreno, hasta que en diciembre del 2010 Google lanza la versión 2.3 denominado “Gingerbread” y se situó como el sistema operativo de móviles más vendido en el mundo. En febrero del 2011 Google lanza la versión 3.0 “HoneyComb”, que está diseñado para tablets en lugar de teléfonos móviles, lo que significa que Android ha trascendido de los teléfonos celulares para trascender a dispositivos

más grandes. Android tiende a continuar evolucionando tal y como lo ha venido haciendo hasta ahora. [1][2] [3]

En la tabla I se observa todas las versiones de Android junto con el porcentaje de dispositivos que se encuentra presente cada versión. Dirigirse al gráfico 2.1 para observar la representación de la tabla I.

Tabla I.- Distribuciones de Android

Versión de Android	API	% de dispositivos
Froyo (2.2)	8	1,6
Gingerbread (2.3-2.3.7)	10	24,1
HoneyComb (3.2)	13	0,1
Ice Cream Sandwich (4.0.3-4.0.4)	15	18,6
Jelly Bean (4.1)	16	37,4
Jelly Bean (4.2)	17	12,9
JellyBean (4.3)	18	4,2
Kitkat	19	1,1

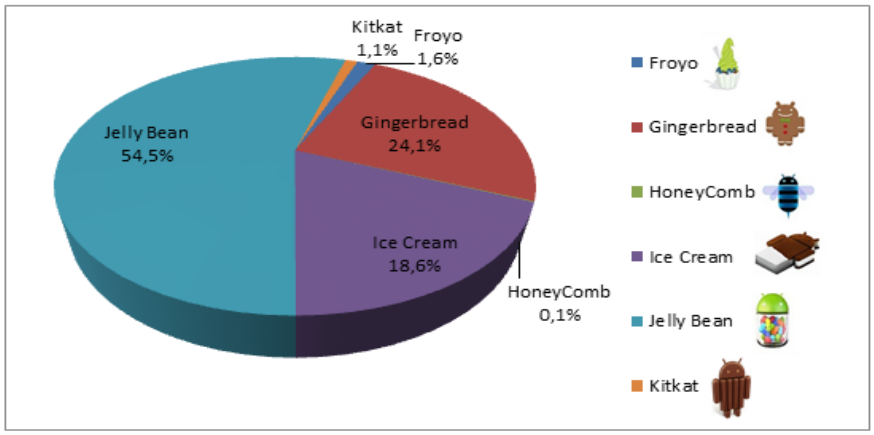


Figura 2.1.- Distribuciones de las versiones Android. [4]

2.3. Aplicaciones nativas de Android

Los dispositivos con Android vienen con una suite de aplicaciones preinstaladas que forman parte del Proyecto Android Código Abierto(en inglés AOSP), incluyendo un cliente de correo electrónico, una aplicación de mensajería de texto, un navegador web basado en WebKit, reproductor de música, cámara, grabación de video, entre otros. En muchos casos los dispositivos Android también tienen instalados ciertas aplicaciones propietarias de Google, tales como es el caso del Google Play Store, un correo electrónico Gmail, mensajería instantánea Google Talk y reproducción de videos YouTube.

Los datos guardados y usados por muchas de estas aplicaciones nativas, como la lista de contactos, son también usados por las aplicaciones de terceros. Similarmente las aplicaciones pueden responder eventos tales como las llamadas entrantes.

Ese necesario notar que la compatibilidad de los dispositivos, la plataforma y el SDK pueden tener ciertas variaciones en lo que respecta a la interfaz gráfica, pero las aplicaciones funcionarán de la misma manera en todos los dispositivos compatibles de Android. [5]

2.4. Características del SDK de Android

Android no diferencia entre las aplicaciones básicas del teléfono y las aplicaciones de terceros. Todos ellos pueden ser construidos para tener igualdad de acceso a las capacidades de un teléfono que proporciona a los usuarios una amplia gama de aplicaciones y servicios. Con los dispositivos basados en la plataforma Android, los usuarios son capaces de adaptar totalmente el teléfono a sus intereses, pueden intercambiar pantalla de inicio del teléfono, el estilo del marcador, o cualquiera de las aplicaciones, incluso pueden instruir a sus teléfonos para utilizar su aplicación de visualización de fotos favorita para manejar la visualización de todas las fotos.

Entre las características más destacadas de Android tenemos la compatibilidad de redes GSM, EDGE, 3G, 4G, LTE para hacer o recibir llamadas, enviar o recibir datos a través de redes móviles. API para los servicios basados en locación tales como GPS y detección de ubicación basados en red; Acesso Wi-Fi y conexión de redes par a par; Control completo de multimedia, incluyendo reproducción y grabación con la cámara y micrófono; APIs para el uso de sensores; Librería para el uso de Bluetooth y hardware NFC. [5]

2.5. Pila de software de Android

La pila de software de Android, es un kernel de Linux y una inmensa colección de librerías de C/C++ expuesta a través una aplicación que provee algún servicio. Está compuesta por los elementos mostrados en la figura 2.2.

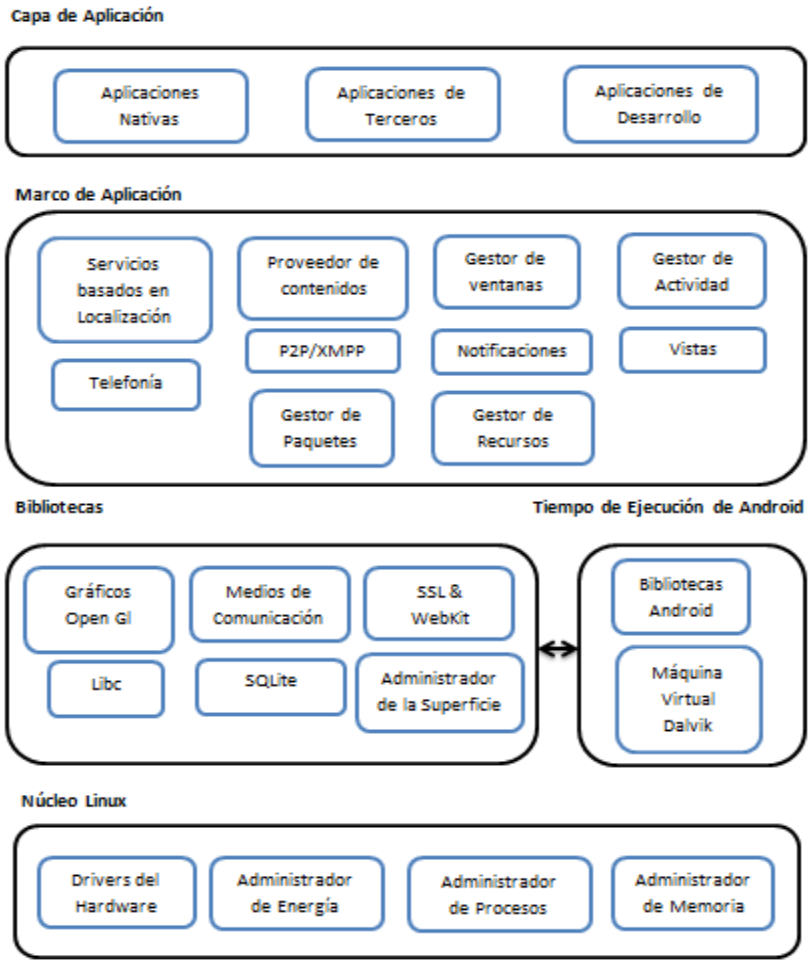


Figura 2.2.- Pila de Software de Android [6]

2.5.1. Núcleo de Linux

Servicios del núcleo (incluyendo drivers de hardware, administración de memoria y procesos, seguridad, redes, administración de poder) son manipulados por el kernel Linux versión 2.6. El kernel también provee una capa de abstracción entre el hardware y la renombrada pila.

2.5.2. Librerías

Se ejecutan sobre el kernel. Android incluye varias librerías C/C++, tales como libc y SSL, que incluye una libraría de medios de comunicación para la reproducción de audio y video y librerías graficas que incluyen OpenGL para graficos en 2D y 3D.

2.5.3. Tiempo de ejecución de Android

El tiempo de ejecución es lo que hace un teléfono Android como un teléfono con Linux. Incluyendo las librerías del núcleo y la máquina virtual Dalvik.

2.5.4. Aplicaciones de framework

Las aplicaciones de framework proveen las clases usadas para crear aplicaciones en Android. También brinda una

extracción genérica para el acceso del hardware, la gestión de la interfaz gráfica y recursos de la aplicación.

2.5.5. Capa de aplicación

Todas las aplicaciones, tanto nativas como de terceras partes, son construidas sobre la capa de aplicación. [7]

2.6. La Máquina Virtual Dalvik

La máquina virtual Dalvik es uno de los elementos principales de Android, basado en el kernel de Linux. Así mismo usando la tradicional Java VM, como Java ME, Android usa su propia máquina virtual diseñada para asegurar que múltiples instancias se ejecuten eficientemente en el mismo dispositivo móvil.

El Dalvik VM manipula las funcionalidades de bajo nivel, incluyendo seguridad, hilos y administración de procesos y memoria. También es posible escribir aplicaciones en C/C++. [7]

Si la velocidad y la eficiencia en tu aplicación escrito en C/C++ es requerida, Android provee de un conjunto de herramientas nativas de desarrollo. El NDK está diseñado para permitir la creación de

librerías C++ usando las librerías libc y libm, junto con el acceso nativo a OpenGL.

2.7. Arquitectura de una Aplicación Android

La arquitectura de Android fomenta la reutilización de componentes, habilitando la posibilidad de publicar y compartir Activities, Services y datos con otras aplicaciones.

Todas las aplicaciones hechas en Android poseen la siguiente arquitectura:

1. **Administrador de Actividades.-** Controla el ciclo de vida de los activities, incluyendo la administración de la pila de la actividad.
2. **Views.**- Usado para construir las interfaces de usuarios para las actividades.
3. **Proveedor de Contenido.**- Permite a la aplicación compartir datos.
4. **Administrador de Notificaciones.-** Provee un consistente y no intrusivo mecanismo para avisos al usuario.
5. **Administrador de Recurso.**- Habilita los recursos no codificados, tales como strings y gráficas, para ser externalizados.
6. **Intents.-** Provee un mecanismo para la transferencia de datos entre las aplicaciones. [7]

2.8. Breve historia de las versiones de Java

Java se creó como una herramienta para ser usada en un proyecto de set-top-box en una pequeña operación denominada The Green Project en Sun Microsystems en el año 1991. El número de integrantes que conformaban el proyecto estaba formado por tres personas cuyo dirigente era James Gosling. Los objetivos de Glosing eran implementar una máquina virtual y un lenguaje con una estructura de sintaxis similar a C++. Entre junio y julio de 1994 el equipo reorientó la plataforma hacia la web. Naugthon uno de los integrantes del proyecto creó un prototipo de navegador, WebRunner que más tarde sería conocido como HotJava. En ese mismo año se hizo una demostración de HotJava y la plataforma Java a los ejecutivos de Sun Microsystems y se tuvo que esperar hasta el 23 de mayo de 1995, durante las conferencias de SunWorld para que saliera a la luz pública Java y HotJava.

El 9 de enero de 1996, Sun fundó el grupo empresarial JavaSoft para que se encargue del desarrollo tecnológico. Dos semanas más tarde fue publicada la primera versión de Java que fue JDK 1.0. Desde entonces Java ha experimentado numerosos cambios, así como un enorme incremento en el número de clases y paquetes que componen la biblioteca estándar. El 19 de febrero de 1997 salió al mercado la versión JDK 1.1 la cual incluía una reestructuración intensiva del modelo de eventos AWT. El 8 de diciembre de 1998

aparece una nueva versión de Java denominada J2SE 1.2, desde esta y las siguientes versiones fueron denominadas Java 2 y el nombre de J2SE, reemplazo a JDK para distinguir la plataforma a base de J2EE y J2ME, los cambios que se integraron en esta nueva versión fueron la API gráfica (Swing) que fue integrada en las clases básicas, JavaPlug-in, Java IDL y colecciones. El 11 de diciembre del 2006 Sun Microsystems decide cambiar nuevamente el nombre de J2SE por Java SE siendo la versión Java SE 6 la versión publicada ese año. En la figura 2.3 se visualiza como han ido cambiando las versiones de Java al pasar de los años. [8]

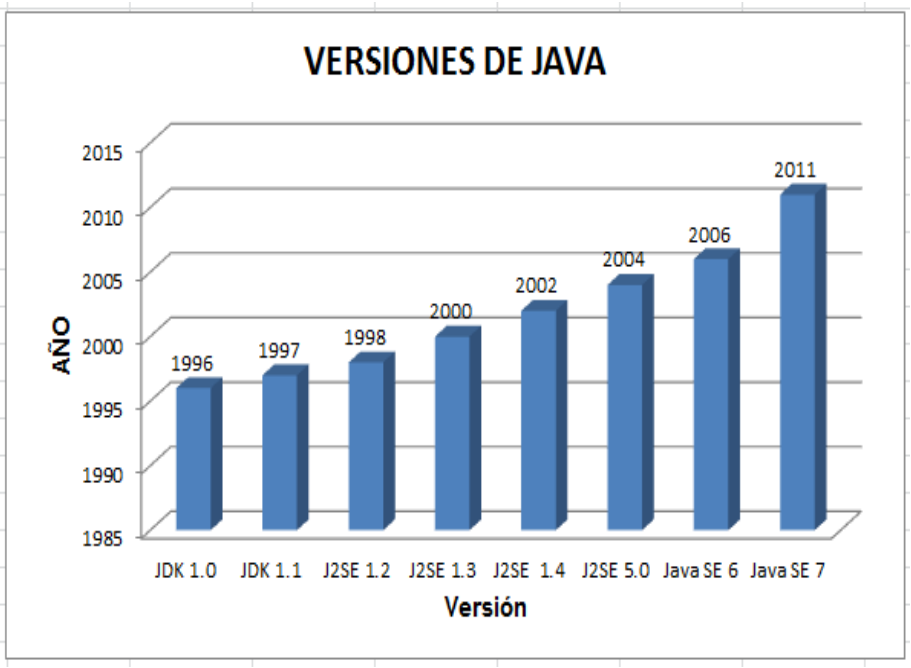


Figura 2.3.- Versiones de java

2.9. Introducción a la Plataforma Java Micro Edition

JME (anteriormente “J2ME”) es un entorno de desarrollo robusto y flexible enfocado a la aplicación de la tecnología Java en dispositivos con capacidades limitadas, tales como teléfonos móviles, PDAs, electrodomésticos inteligentes, TV, impresoras, etc.

Java Micro Edition es una plataforma de aplicaciones móviles más antigua que está siendo considerablemente utilizada. Esta edición tiene unos componentes básicos que la diferencian de las otras versiones de Java, uno de estos componentes es el uso de una máquina virtual denominada KVM, la misma que requiere poca memoria para funcionar. [9]

Las tecnologías JME contienen un JRE muy optimizado, especialmente desarrollado para el mercado de gran consumo, cubren una amplia gama de dispositivos donde se permite ejecutar programas de seguridad, conectividad, buscapersonas y utilidades en tarjetas inteligentes. Existen varias plataformas relacionas con java como se muestra en la figura 2.4. Cada plataforma está enfocada al desarrollo de una aplicación específica como se detalla a continuación.

- **JEE:** Se utiliza básicamente para aplicaciones basadas en la web o aplicaciones basadas en la red.

- **JSE:** Está orientada al desarrollo de aplicaciones independientes de la plataforma.
- **JME:** Se encuentra orientada a dispositivos con capacidades restringidas es decir con limitaciones de proceso y capacidad gráfica.[10]

Figura 2.4.- Ediciones de Java

La figura 2.5 nos indica que JEE es un súper conjunto de JSE pues contiene todas las funcionalidades de éste y más características, en cambio JME es un subconjunto de JSE con una única excepción que es el paquete javax.microedition el cual no se encuentra dentro de los paquetes de JSE.

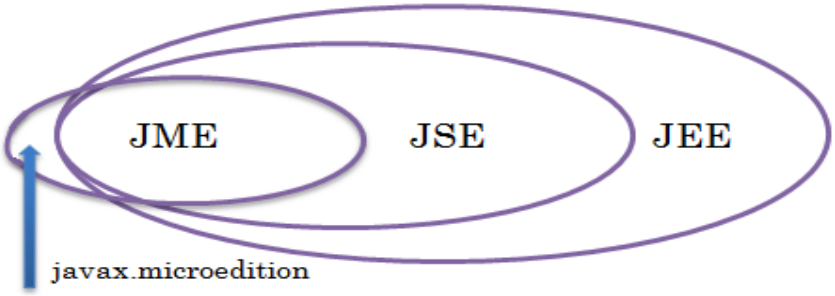


Figura 2.5.- Relación entre las APIS de la plataforma Java [10]

La tecnología Java ofrece, asimismo, métodos de creación de servicios Web, transferencia de información XML, numerosos protocolos de red, conjunto de herramientas y la aplicación Java Web Start. JME se encuentra divido entre configuraciones, perfiles y APIs operacionales, la cual provee información específica acerca de las APIs y de los dispositivos de las diferentes familias.

Las APIs opcionales definen específicamente una funcionalidad adicional que puede ser incluida en una configuración particular (o perfil). Actualmente hay una gran cantidad de configuraciones y perfiles, las más relevantes de JME se encuentran ilustradas en la figura 2.6.

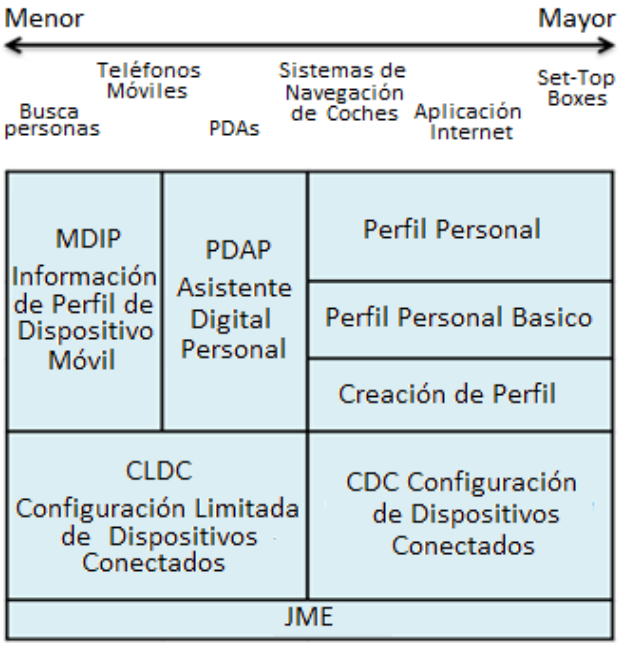


Figura 2.6.- Perfiles y configuraciones comunes de JME. [11]

2.10. Características de Java ME

La plataforma de programación de dispositivos móviles JME define configuraciones y perfiles, como elementos básicos para el desarrollo de aplicaciones que se adaptan a las características de un amplio rango de dispositivos. Cada configuración se utiliza de acuerdo a la memoria, la capacidad de procesamiento y de entrada/salida de una categoría específica de dispositivos.

2.10.1. Configuraciones

Una configuración define las características del lenguaje java y un conjunto mínimo de APIs eficiente para desarrollar las aplicaciones destinadas a un amplio rango de dispositivos. La decisión acerca de qué configuración aplicar sobre un dispositivo se basa principalmente en la disponibilidad y capacidades de memoria, pantalla, conexión de red y procesador de dicho dispositivo. Las características típicas de aquellos dispositivos que se ajustan a cada una de las actuales configuraciones son:

- **CDC**: La configuración de dispositivos conectados está orientado a dispositivos de información compartidos, fijos y de conexión permanente como son los decodificadores de televisión digital, televisores con internet, etc. La configuración CDC utiliza una máquina virtual llamada CVM, además presenta requisitos mínimos de hardware

para su correcto funcionamiento como son: 512 KB de memoria para ejecutar Java, 256 KB de memoria para la ejecución de programas, conexión de red y un procesador de 32 bits.

- **CLDC:** La configuración limitada de dispositivos conectados está orientado a dispositivos móviles personales y proporciona un nivel mínimo de funcionalidades para desarrollar aplicaciones para un determinado conjunto de dispositivos como por ejemplo dispositivos móviles, PDA, agendas electrónicas, entre otros. Los requisitos mínimos de hardware para esta configuración son: 160 KB de memoria disponible para Java donde 128 KB son destinados para la máquina virtual Java que para esta configuración recibe el nombre de KVM y para las librerías del API de CLDC, las 32KB restantes se lo utiliza para el sistema de ejecución. Los otros requerimientos son un procesador de 16 bits, un consumo bajo de batería y conexión de red. [12]

La figura 2.7 muestra un mapa conceptual indicando los dos tipos de configuraciones que presenta JME para una mayor comprensión sobre este tema.

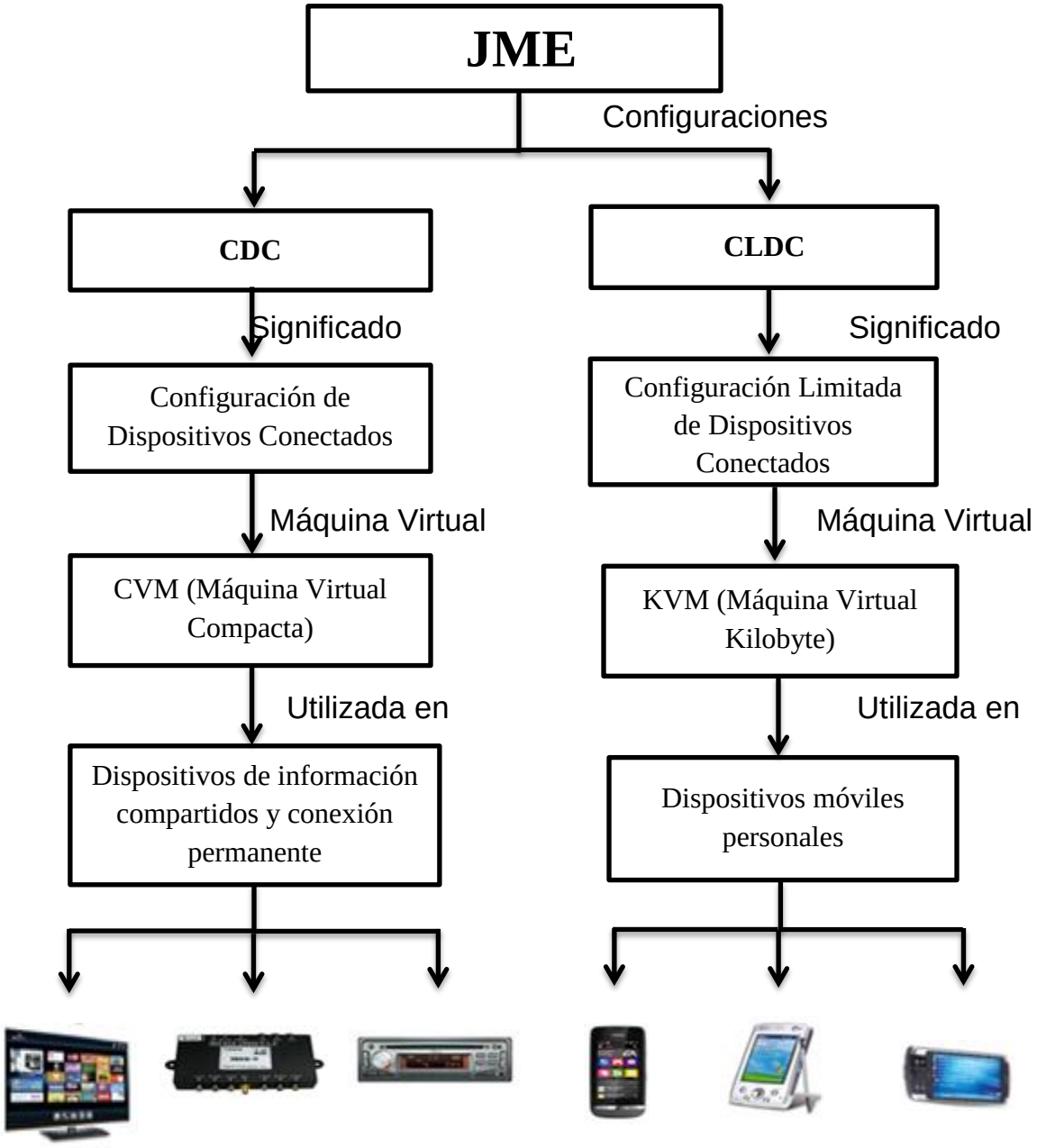


Figura 2.7.- Mapa Conceptual de las configuraciones de JME

2.10.2. Perfiles

Un perfil es una extensión de una configuración que proporciona a un programador las librerías necesarias para desarrollar una aplicación para un tipo de dispositivo en particular. Por ejemplo, MIDP define APIs para componentes de interfaz de usuario, manejo de entrada de datos y eventos, almacenamiento persistente, comunicaciones y temporizadores, todo ello teniendo en cuenta las limitaciones de pantalla y memoria de los dispositivos móviles.

Los perfiles identifican un grupo de dispositivos por la funcionalidad que proporcionan y el tipo de aplicaciones que se ejecutarán en ellos. Las APIs se las establece mediante un perfil las mismas que definen las características de un dispositivo. Al momento de realizar una aplicación se debe tener en cuenta que un perfil se construye sobre una configuración determinada. [12] El perfil MIDP establece las capacidades del dispositivo, por lo tanto, especifica las APIs relacionadas con:

1) La aplicación (control de la aplicación MIDP).
2) Interfaz de usuario. Almacenamiento permanente.
3) Trabajo en red.
4) Temporizado

2.11. MIDlet

Es un programa desarrollado en la plataforma para dispositivos móviles JME, utiliza la máquina virtual KVM. Las aplicaciones JME desarrolladas bajo la especificación MIDP, se denominan MIDLets.

Los MIDlets son empaquetados en ficheros ".jar" el cual está compuesto de: Clases de MIDlet, clases de soporte, recursos (imágenes, sonidos…etc), manifiesto (fichero ". mf") y un descriptor (fichero ".jad"). [12]

2.12. Anatomía de un MIDP

Las APIs disponibles para una aplicación MIDlet vienen en paquetes, tanto en CLDC y MIDP, como se muestra en la Tabla II, los paquetes marcados con un + son nuevos en CLDC 1.1 y MIDP 2.0.

Tabla II.- Paquetes MIDP

CLDC 1.1	MIDP 2.0
java.lang	javax.microedition.lcdui
+java.lang.ref	+javax.microedition.lcdui.game
java.io	+javax.microedition.media
java.util	+javax.microedition.media.control
javax.microedition.io	javax.microedition.midlet
	+javax.microedition.pki
	javax.microedition.rms

CLDC define un núcleo de APIs, tomados mayoritariamente desde J2SE. Este incluye las clases fundamentales de java en java.lang y colecciones simples desde java.util. También especifica una API para acceso a redes dentro del paquete javax.microedition.io.

Opcionalmente, los programadores de dispositivos móviles pueden modificar las APIs de java según las características específicas del teléfono. Los dispositivos MIDP, serán habilitados para ejecutar diferentes aplicaciones. La figura 2.8 muestra un mapa de estas posibilidades.

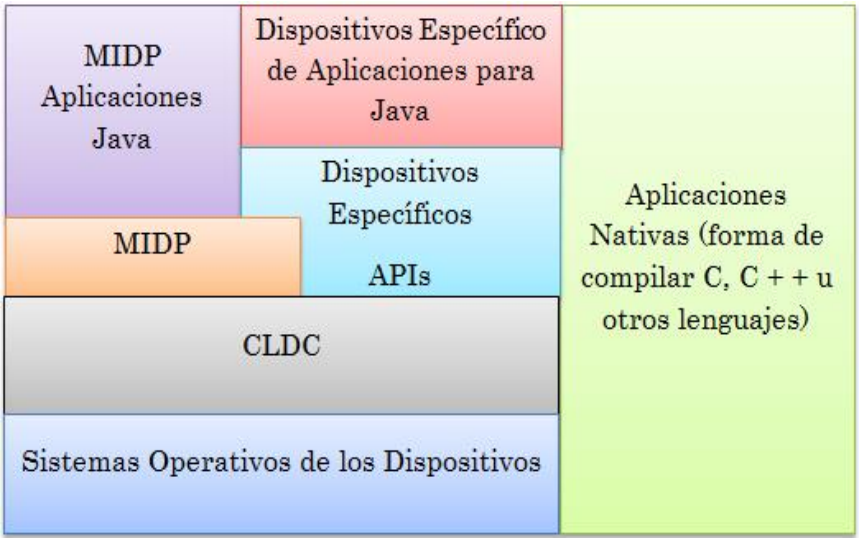


Figura 2.8.- Componentes de software MIDP [13]

Cada dispositivo implementa algún tipo de OS como ejemplo el dispositivo móvil Nokia Asha 311 tiene el sistema operativo Serie

40. Las aplicaciones nativas corren directamente sobre esta capa y representa un mundo antes de MIDP en diferentes tipos de dispositivos, cada uno con su propio sistema operativo y aplicaciones nativas. Las aplicaciones de MIDP usan solamente CLDC y APIs MIDP.

2.13. Ventajas de MIDP

Las principales ventajas de MIDP son las siguientes:

1) **Portabilidad:** La ventaja de usar java sobre otras herramientas para desarrollo de aplicaciones de dispositivos pequeños es ser portable. Usted podría escribir aplicaciones para dispositivos con lenguaje C/C++, pero el resultado podría ser útil para una simple plataforma en específico. Las aplicaciones escritas usando las APIs MIDP serán ejecutables en cualquier dispositivo MIDP.

2) **Seguridad:** Esta es la segunda ventaja de MIDP sobre otras plataformas para desarrollo de dispositivos móviles. Java es muy bien conocido por su habilidad de descargar aplicaciones de manera segura. La JVM usado en la CLDC solo implementa un bytecode parcial, lo cual significa que parte de la importante tarea de la verificación de los bytecode deben ser realizados fuera del dispositivo MIDP. La CLDC no permite cargadores de clases para aplicaciones definidas. Significa que la mayoría de

entrega de aplicaciones dinámicas es independiente de los dispositivos específicos. Las aplicaciones MIDP ofrecen una importante promesa de seguridad, esto significa que existen errores de restricción es decir que en una aplicación MIDP nunca estará habilitado para escribir en la memoria del dispositivo que no pertenece a la JVM. Una aplicación MIDP no va a estropear otra aplicación en el mismo dispositivo.

En MIDP 2.0, la suite de MIDlets puede ser firmada criptográficamente y luego verificada en el dispositivo, lo cual da a los usuarios mayor seguridad al ejecutar una aplicación descargada. Una arquitectura de nuevos permisos también permite al usuario denegar el acceso de código no confiable. Por ejemplo si usted instala un programa MIDLET sospechoso en su teléfono móvil, éste estará habilitado para ejecutarse si el usuario explícitamente se lo permite. [13]

CAPÍTULO 3
COMPARACIÓN DE LAS PLATAFORMAS ANDROID Y JME

Se comparan las plataformas escogidas para determinar en cada variable, cuál de los dos, Android o JME, presenta ventaja con respecto al otro. Las variables analizadas son: APIs, licencia, lenguaje de programación, líneas de código, tamaño de la aplicación, emulación, conectividad y múltiples pantallas.

3.1. APIs

El significado de las siglas es Interfaz de Programación de Aplicaciones, la cual es un conjunto de librerías que le brinda al

programador desarrollar sus aplicaciones con mayor facilidad. En este variable se analizó cuál de las dos plataformas, ya sea Android o JME le dan mayor facilidad al programador al momento de agregar recursos a su aplicación y de esta manera tenga más acogida por los usuarios. Existen APIs que, tanto Android como JME comparten, estas son las APIs que heredan directamente de la superclase *Object:* java.text, java.util, java.security, java.math, java.lang, java.net, java.io. [14]

Después de comparar las APIs entre ambas plataformas (para ver las APIs y su funcionalidad dirigirse al ANEXO E), se observó que Android presenta mayor cantidad de APIs, aparte de eso, no hay una API de JME que cumpla una funcionalidad que Android no pueda cumplir, para un mejor entendimiento de lo antes mencionado ver Figura 3.1.

Figura 3.1.- Diagrama de Venn de APIs: Android y JME

La gráfica de pastel (ver Figura 3.2) me ayuda a concluir que existe mayor cantidad de APIs en la plataforma de programación de dispositivos móviles Android, lo que significa que dicha plataforma

le facilita al programador agregar más recursos a la aplicación y de esta manera conseguir un mayor aprovechamiento de las funcionalidades que traen consigo los dispositivos móviles de hoy en día, provocando así una mayor acogida por parte de los usuarios finales y permitiéndole al programador obtener mayor cantidad de ganancias por la creación de dichas aplicaciones.

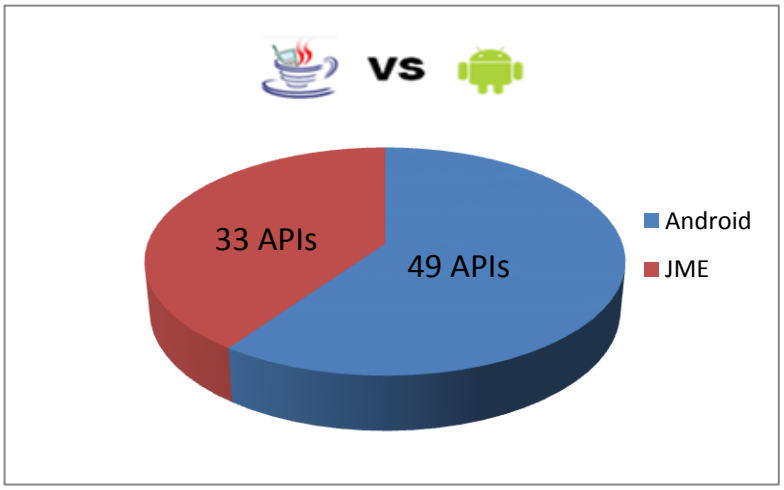


Figura 3.2.- Cantidad de APIs de Android y JME.

El uso de las APIs avanzadas de Android como son: android.location, android.media, android.opengl, android.hardware, android.bluetooth, android.wifi, android.net.wifi.p2p entre otras ofrecen al programador una funcionalidad más eficaz para aprovechar los recursos del dispositivo móvil al momento de realizar una aplicación permitiendo así que los usuarios que la utilicen se sientan satisfechos.

El uso de la API de localización de JME no resulta tan eficiente al utilizarla en una aplicación debido a que utiliza el sistema de posicionamiento global para conocer la ubicación de un objeto consumiendo más batería que si utilizaremos las torres de telefonía móvil o las señales Wi-Fi como lo hace el API de localización de Android.

Se observó que una de las interfaces de programación de aplicaciones con la que no cuenta JME pero que nos proporciona Android es android.net.wifi.p2p que utiliza el Wi-Fi Direct el cual nos permite conectar varios dispositivos Wi-Fi entre sí sin necesidad de un punto de acceso intermedio.

3.2. Licencia

Es una autorización que el autor les concede a otras personas para el uso y distribución de sus programas. Existen muchos tipos de licencias, de ellas hemos resumido a las cuatro más conocidas, mostrados en la figura 3.3.

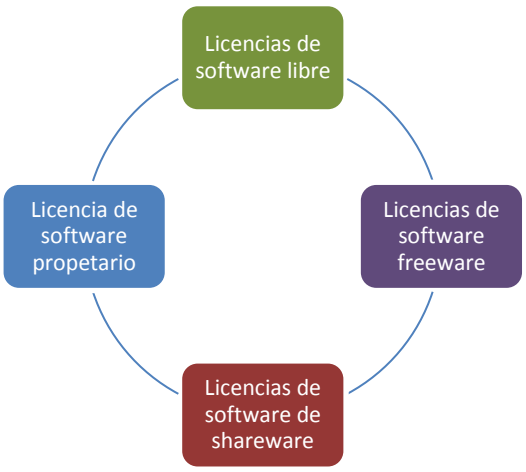


Figura 3.3.- Tipos de licencias de software

Tipos de licencia:

- Licencia de software propietaria: La libre distribución, uso, copia o modificación del programa está prohibido por el autor. Para usar, copiar o distribuir, se debe pagar por el producto.
- Licencia de software Shareware: Uso limitado en tiempo o capacidades para que el usuario lo evalúe, una vez acabado el tiempo de evaluación, el usuario debe pagar.
- Licencia de software Freeware: Uso y copia del producto de manera libre y gratuita.
- Licencia de Software Libre: Estudiar, copiar, usar, modificar y redistribuir el programa libremente y de manera gratuita. Aquí se incluye el Código fuente. [15]

JME es una plataforma libre con licencia GPL es decir que le permite al usuario hacer uso de la plataforma, modificarla y distribuir

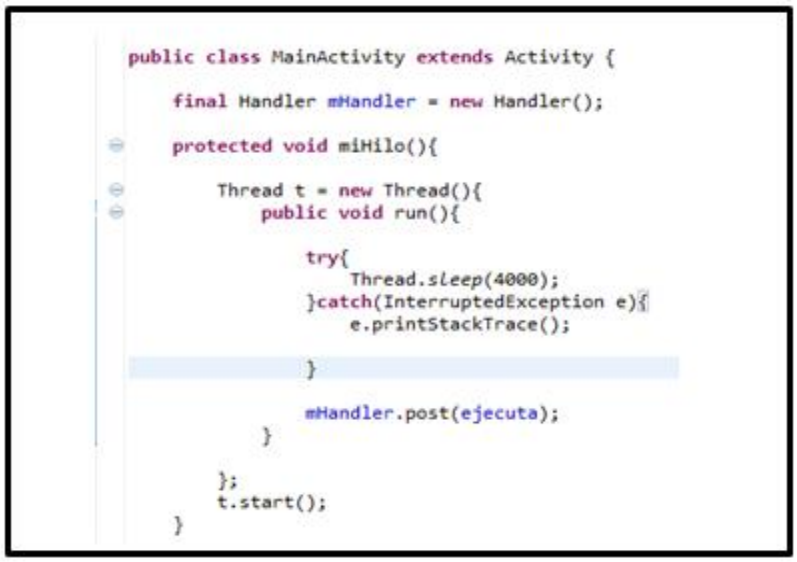
las modificaciones que se le haya hecho a la misma. La licencia GPL establece que todo programa o cualquier material que haya sido desarrollado bajo esta licencia debe estar disponible para ser compartido con el resto de personas.

Android es una plataforma libre bajo dos licencias Apache 2.0 y GNU GPL 2 es decir que la persona está en la libertad de usar, modificar y distribuir el software sin estar obligado a liberar el código fuente que este modificado. Por otro lado la licencia Apache 2.0 también indica que los programas derivados no necesariamente deben ser de este mismo tipo de licencia, ni siquiera exige que el software se tenga que distribuir libremente y con código abierto, pero exige que se realice un informe a los receptores que en la distribución se ha usado código con licencia Apache. [16]

Ambas plataformas poseen en esencia el mismo tipo de licencia GPL, se puede distribuir, modificar y usar libremente. La diferencia radica en que Android usa también la licencia Apache v2.0, esto será beneficioso para los programadores que quieran modificar a su gusto la plataforma Android sin la necesidad de distribuirla bajo la misma licencia, es decir, la modificación puede ser vendida. La licencia solo cubre el código de la plataforma Android, no se verá afectado los programas que se realicen bajo esta plataforma, las aplicaciones pueden ser vendidas.

3.3. Lenguaje de Programación

Las aplicaciones para Android pueden ser desarrolladas sobre varios lenguajes de programación, los cuales son: Java (ver Figura 3.4), VisualBasic (ver Figura 3.5), C# (ver Figura 3.6) y si se desea programar de manera nativa se la puede hacer en C/C++ (ver Figura 3.7). Existe una plataforma para desarrollar aplicaciones para Android sin la necesidad de saber algo de programación y esta se llama AppInventor. En las gráficas se observa la sintaxis de los diferentes lenguajes de programación para Android. [17]



```
public class MainActivity extends Activity {

    final Handler mHandler = new Handler();

    protected void miHilo(){

        Thread t = new Thread(){
            public void run(){

                try{
                    Thread.sleep(4000);
                }catch(InterruptedException e){
                    e.printStackTrace();

                }

                mHandler.post(ejecuta);
            }

        };
        t.start();
    }
```

Figura 3.4. Lenguaje Java para Android

Para cada lenguaje de programación de aplicaciones Android existe un entorno de desarrollo.

Eclipse es un IDE que facilita el desarrollo de aplicaciones Android escritas en lenguaje Java.

Basic4Android es un entorno de programación para desarrollar aplicaciones basados en Android con lenguaje VisualBasic, está orientado a personas que empezaron a programar de manera más grafica que abstracta.

Mono es un entorno pensada para programadores que desarrollan aplicaciones en lenguaje C# y .NET pero esta vez para la plataforma Android. [17]

```
Sub Activity_Resume

End Sub

Sub Activity_Pause (UserClosed As Boolean)

End Sub



Sub Button1_Click
    mio = EditText1.Text
    If ordenador > mio Then Label2.Text = "Mi número es mayor"
    If ordenador < mio Then Label2.Text = "Mi número es menor"
    If ordenador = mio Then Label2.Text = "HAS ACERTADO"
End Sub
```

Figura 3.5.- Lenguaje Visual Basic para Android

```
using Android.Widget;
using Android.OS;

namespace prueba
{
    [Activity (Label = "prueba", MainLauncher = true)]
    public class MainActivity : Activity
    {
        int count = 1;

        protected override void OnCreate (Bundle bundle)
        {
            base.OnCreate (bundle);
            SetContentView (Resource.Layout.Main);

            Button button = FindViewById<Button> (Resource.Id.myButton);
            button.Click += delegate {
                button.Text = string.Format ("{0} clicks!", count++);
            };
        }
    }
}
```

Figura 3.6.- Lenguaje C# para Android

```
#include <stdio.h>
#include <stdlib.h>

#include <egl.h>

NativeWindowType displayWindow;

const EGLint config16bpp[] =
{
EGL_RED_SIZE, 5,
EGL_GREEN_SIZE, 6,
EGL_BLUE_SIZE, 5,
EGL_NONE
};

GLfloat colors[3][4] =
{
    {1.0f, 0.0f, 0.0f, 1.0f},
    {0.0f, 1.0f, 0.0f, 1.0f},
    {0.0f, 0.0f, 1.0f, 1.0f}
};
```

Figura 3.7.- Lenguaje C para Android

AppInventor es una plataforma desarrollada pensando en personas que no tengan un gran nivel de experiencia programando. Se puede construir una aplicación de manera gráfica (con bloques). Ver figura 3.8. [17]

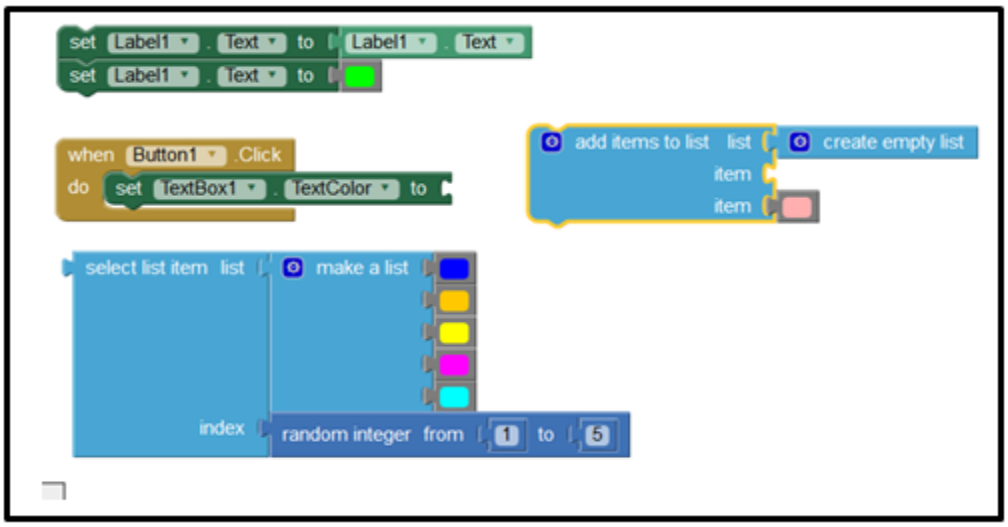


Figura 3.8.- AppInventor para aplicaciones Android.

Por otro lado, JME posee un único lenguaje de programación que es lenguaje Java para dispositivos móviles, no hay elección a más.
En esta variable Android tiene su ventaja, puesto que muchas personas se verán atraídas a desarrollar aplicaciones Android usando cualquiera de los cuatro lenguajes descritos anteriormente o personas que no sepan mucho de programación podrán desarrollar su aplicación usando un entorno de desarrollo grafico llamado AppInventor, algo que no tendrá efecto al tratar de desarrollar

aplicaciones en JME, para esto se debe tener conocimiento del lenguaje orientado a objetos, en este caso Java.

3.4. Líneas de Código

Desarrollamos tres aplicaciones para comparar las líneas de código que se crean al programar cada una de las Apps tanto para Android como para JME. De esta forma sabríamos que plataforma ayudaría al programador a realizar una aplicación en menor tiempo. El código de las tres aplicaciones se las puede apreciar en el ANEXO D.

La primera aplicación que se realizó es un sencillo programa que muestra un texto en pantalla. Los datos obtenidos fueron los siguientes: la plataforma Android sumó un total de 8 líneas de código para la creación de esta aplicación mientras que en la plataforma JME se obtuvo 17, lo cual significa que la plataforma Android decremento en un 53% de líneas de código utilizadas en JME.

La segunda aplicación es una calculadora que realiza las operaciones básicas (suma, resta, multiplicación y división). Las líneas de código obtenidas en Android fueron 58 mientras que en JME fueron 105, es decir que JME excede a Android por un 45%.

La tercera aplicación es un juego, se trata del tres en raya. Los resultados obtenidos fueron los siguientes: la plataforma Android sumó un total de 136 líneas de código para realizar la aplicación mientras que en la plataforma JME se obtuvo 181, lo que significa JME supera en un 25% al número de líneas utilizadas en Android.

En definitiva, dada que en estas tres aplicaciones se vio la clara ventaja de Android (ver la Figura 3.9), se puede concluir que JME tiende a generar más líneas de código para la creación de una aplicación. Para un programador le será más óptimo desarrollar una aplicación en Android, ya que al tener menos líneas de código en su programa, le permitirá reducir el tiempo de creación de la aplicación.

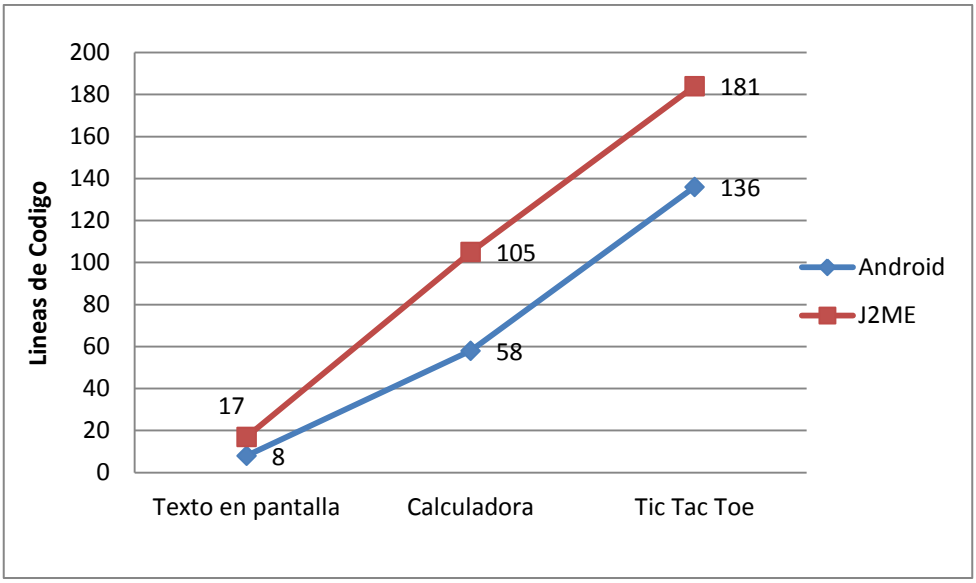


Figura 3.9.- Comparativa Android y J2ME: Líneas de código.

3.5. Tamaño de la Aplicación

Para la comparación de esta variable se usan las tres aplicaciones realizadas por nosotros y otras cuatro aplicaciones descargadas desde el internet hasta el teléfono móvil y comparamos el tamaño que ocupan en memoria. Todos los tamaños de la aplicaciones fueron comparadas en kilobytes (kB).

Los resultados de comparar la primera aplicación fueron los siguientes: en Android nos mostró un tamaño de 20kB y JME nos mostró un tamaño de 4kB, lo cual indica que el tamaño de la aplicación realizada para Android es cinco veces mayor que la desarrollada en JME.

La segunda aplicación nos muestra para Android un peso de 20kB y para JME un peso de 4kB, los mismos resultados que los de la aplicación anterior. En la tercera aplicación encontramos que Android tiene un peso de 28kB, en cambio JME un peso de 8kB. En esta ocasión la aplicación en Android es 3.5 veces mayor que la realizada en JME. En la cuarta aplicación (twitter), se obtuvo que en Android pesa 7065.6 kB y la aplicación en JME pesa 716.8 kB. En la quinta aplicación (whatsapp) obtuvimos, para Android un peso de 16547.8 kB y para JME un peso en memoria de 1433 kB. En los resultados de la sexta (Facebook). Android tiene un peso de 17489.92 kB y JME presenta un peso de 133.12 kB. Por último la

séptima aplicación (Angry Bird), nos indica que en Android pesa 24688,26 kB y en JME tiene un peso 2560 kB.

Las aplicaciones hechas en la plataforma JME pesan menos que las aplicaciones hechas en Android, esto es debido a que Android requiere más recursos para la realización de las aplicaciones. Ver figura 3.10.

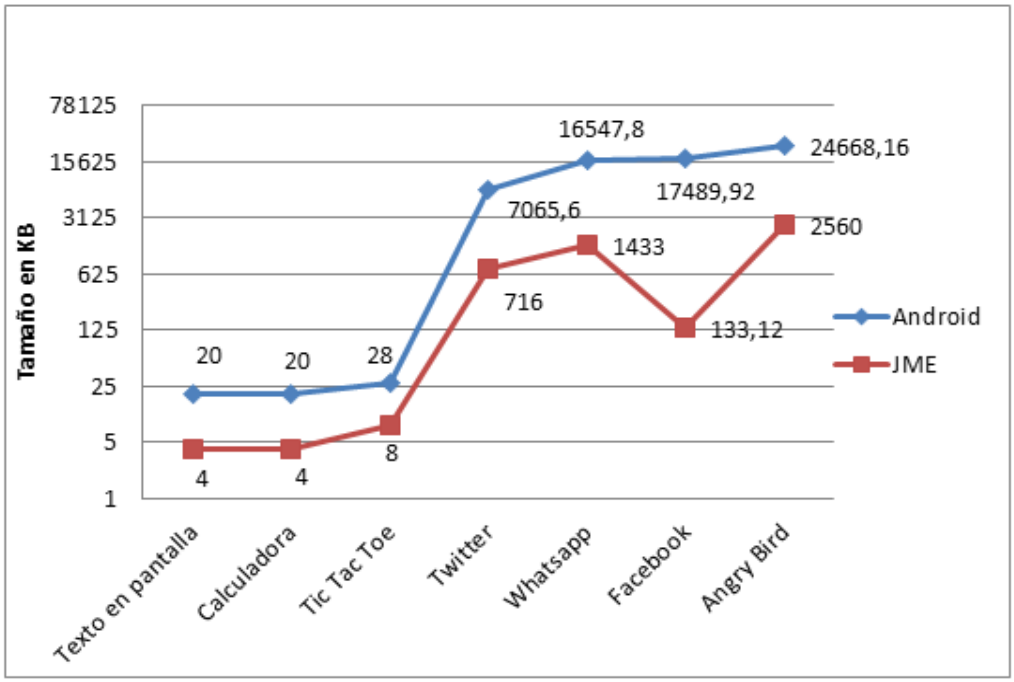


Figura 3.10.- Gráfica Android vs JME: Comparación del tamaño de las aplicaciones

3.6. Emulación

Un emulador me da una idea de lo que el teléfono va a ser en la práctica aunque no tendrá acceso a todas las funcionalidades del sistema operativo. [18] Por ejemplo un emulador no me dará el valor correcto de la velocidad de un teléfono en particular ya que el equipo donde ejecutamos nuestro emulador tiene un conjunto diferente de hardware en comparación con el teléfono real. En esta variable se calculó el tiempo promedio de carga de una aplicación tanto en el emulador de Android como el de JME, y además la tendencia con la que pueden ir variando los datos por encima o por debajo del valor promedio.

La fórmula que utilizamos para calcular el tamaño de la muestra para una población infinita es la siguiente:

$$n = \frac{Z^2 p.q}{e^2}$$ (Ecuación 1)

Dónde:

n = Tamaño de la muestra

$Z = Valor\ correspondiente\ al\ nivel\ de\ confiamza$

$pq = Varianza\ de\ la\ población$

$e = Error\ muestral$

De la Ecuación 1 con Z = 1.96, pq = 0.25 y un error e = 5%, se obtuvo una muestra total de 384 pruebas, tanto para Android como para JME.

Muestras tomadas del emulador de Nokia asha 311

n: Numero de muestras

n=384

$\bar{X}$: Media

$\bar{X} = \sum_{i=1}^{n} \frac{X_i}{n}$ (Ecuación 2)

$\bar{X} = 27.175\ seg$

S^2: Varianza

$S^2 = \sum_{i=1}^{n} \frac{(X_i - \bar{X})^2}{n-1}$ (Ecuación 3)

$S^2 = 2.295$

S : Desviación Estándar

$s = \sqrt{\sum_{i=1}^{n} \frac{(X_i - \bar{X})^2}{n-1}}$ (Ecuación 4)

$s = 1.515$

Muestras tomadas en el emulador de Android

De la ecuación 1 se obtuvo una muestra de:
n=384

De la ecuación 2 se obtuvo una media de:
$\bar{X} = 15.703$

De la ecuación 3 una varianza de:
$S^2 = 0.015$

De la ecuación 4 se obtuvo una desviación estándar de:
$s = 0{,}123$

Luego de realizar las 384 pruebas se obtuvo que el tiempo promedio que tarda el emulador de Android en ejecutar la aplicación es de 15.703 seg el cual resulto menor que el de JME que fue de 27.175 seg eso significa que el emulador de Android es más rápido que el de JME como se observa en la figura 3.11, permitiendo también que la depuración sea rápida y haciendo posible ver los errores de manera ágil. .

Los valores obtenidos en la desviación estándar nos permiten concluir que la tendencia con la que va a ir variando los tiempos en

JME será de 1.5 con respecto a su valor promedio mientras que en Android será de 0.12.

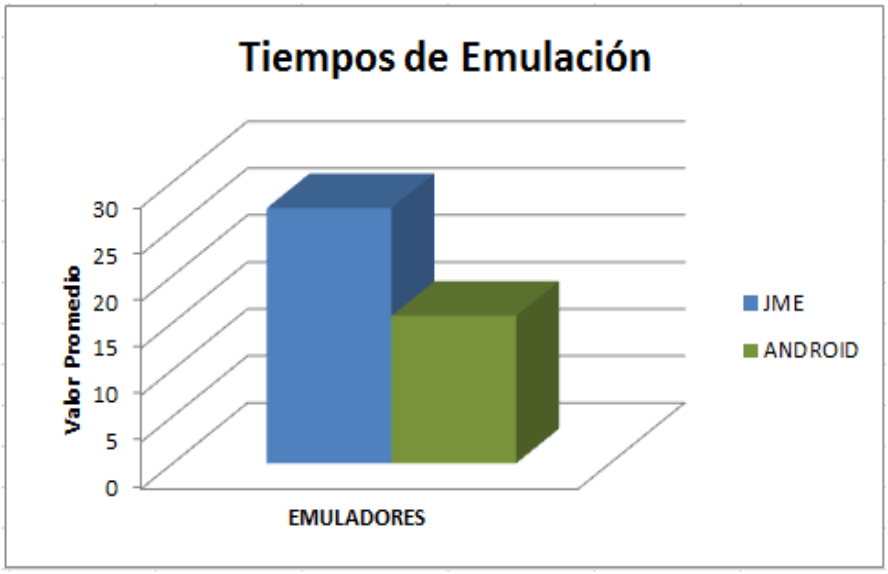


Figura 3.11.- Tiempo promedio: Ejecución en el emulador

Comparando el gráfico de dispersión de JME con el grafico de dispersión con Android, figura 3.12 y figura 3.13 respectivamente, nos ayuda a visualizar de mejor manera que tan alejados se encuentran nuestros datos del valor promedio, permitiéndonos de esta manera concluir con mayor certeza que el emulador de Android es más preciso que JME, puesto que los datos obtenidos al hacer pruebas en el emulador de JME se encuentran muy dispersos, exceptuando por algunos datos aberrantes de Android

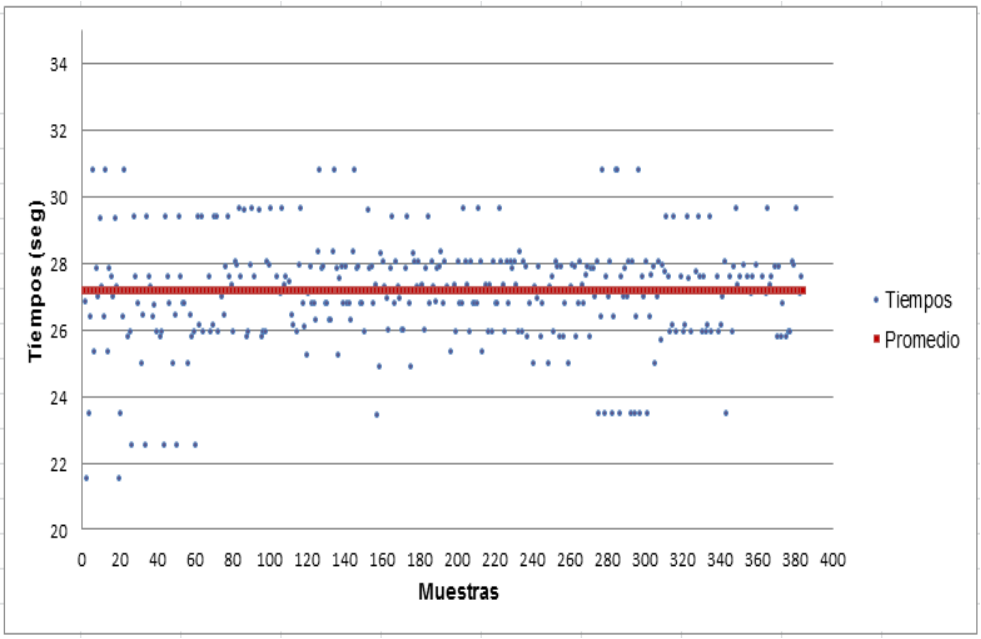


Figura 3.12.- Dispersión de datos en JME

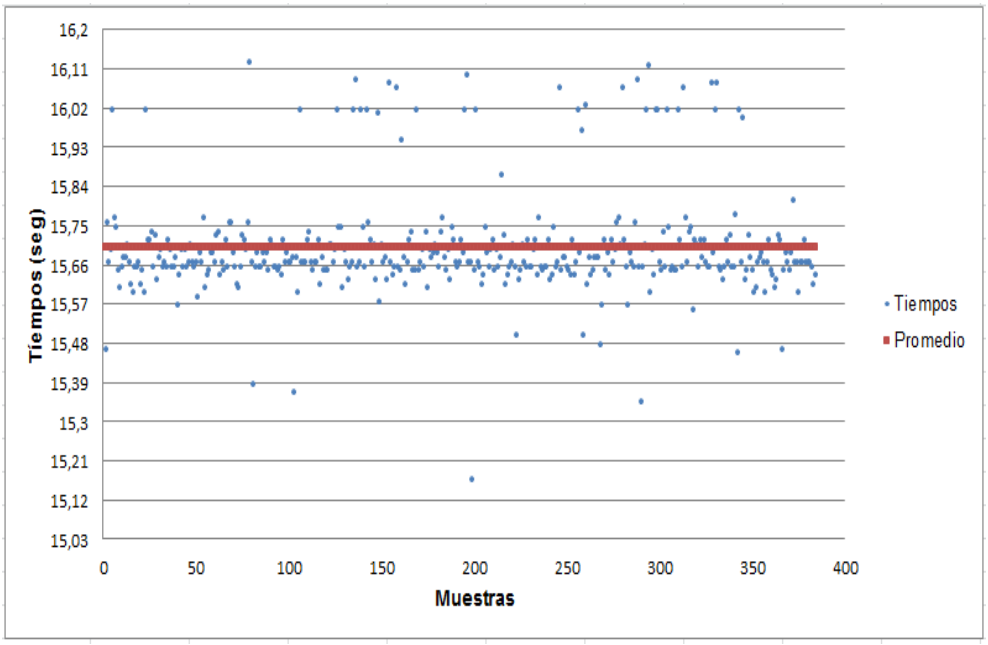


Figura 3.13.- Dispersión de datos en Android

3.7. Conectividad

Esta variable nos permite conocer cuántos tipos de conectividad son soportados por las plataformas Android y JME. La plataforma Android, según hemos hecho las pruebas, también permite una conexión Wi-Fi (para tener acceso a internet) y conexiones par a par mediante Wi-Fi, permite conexiones Wimax, CDMA, LTE (implementada en el Ecuador por CNT), ED-VO, GSM/EDGE, conexiones GPRS. Provee comunicaciones NFC, que permite leer mensajes NDEF en dispositivos NFC etiquetados, la ventaja de NFC con respecto a otras tecnologías de comunicación inalámbrica como lo son infrarrojos y Bluetooth, radica en que todo celular o dispositivo móvil con comunicación NFC puede ser usado como tarjeta de crédito, tarjetas de autenticación y posee cifrado. [19]

Android también permite conexiones con otros dispositivos móviles mediante Bluetooth, en su última versión (kitkat), da soporte a Bluetooth HID sobre GATT, esto permite conectar el teléfono celular o una tablet con periféricos de baja potencia, tales como ratones, joystick y teclados. [20]

Todo este tipo de conectividades que soporta Android y el hecho de que se puedan realizar aplicaciones con cualquiera de ellas, es gracias a la API que proporciona la plataforma.

Por otro lado JME permite conectarme a una red mediante Wi-Fi, para acceder a internet, permite conexiones a otros dispositivos móviles mediante Bluetooth, conexión GPRS, conexión infrarroja y también soporta conectividad NFC.

Dado que Android es una tecnología que avanza rápidamente en lo que se refiere a plataformas de desarrollo de dispositivos móviles y al avance tecnológico hardware de los dispositivos móviles, posee mayor cantidad de conectividad que JME. Esto es un recurso favorable tomando como ejemplo, la conectividad Wimax, tecnología de internet inalámbrico la cual está cogiendo mayor acogida debido a su menor costo y al mayor alcance que la red inalámbrica Wi-Fi hasta ahora no ha sido implementada en la plataforma JME.

3.8. Múltiples pantallas

Android se ejecuta en una variedad de dispositivos que ofrecen diferentes tamaños de pantalla y densidades como se observa en el ANEXO F. Para ello la plataforma Android ofrece la herramienta de apoyo de múltiples pantallas para permitirle al programador crear su aplicación de tal manera que con solo un archivo .apk pueda abarcar todas las configuraciones de pantallas compatibles en lugar de programar una aplicación para cada tipo de pantalla. Las variables que se toman en cuenta al momento de diseñar la interfaz

de usuario para la plataforma Android son las siguientes: Tamaño, densidad, orientación, resolución y píxel independiente de la densidad. Esta última variable me permite mantener el tamaño físico de los elementos de la interfaz de usuario cuando se muestre en las pantallas con diferentes densidades como se muestra en la Figura 3.14 pero al no tener en cuenta esta variable el diseño de la interfaz de usuario se verá afectada como se observa en la Figura 3.15. [21]

Fig. 3.14.- Aplicación con soporte para diferentes densidades

Figura 3.15.- Aplicación sin soporte para diferentes densidades

Los valores de tamaño y de la densidad utilizados en la aplicación, es muy importante ya que la combinación de estas dos variable me permiten con una sola aplicación abarcar un rango de dispositivos que se ajustan de acuerdo a las configuraciones generalizada de pantalla que tiene Android como se presenta en la figura 3.16.

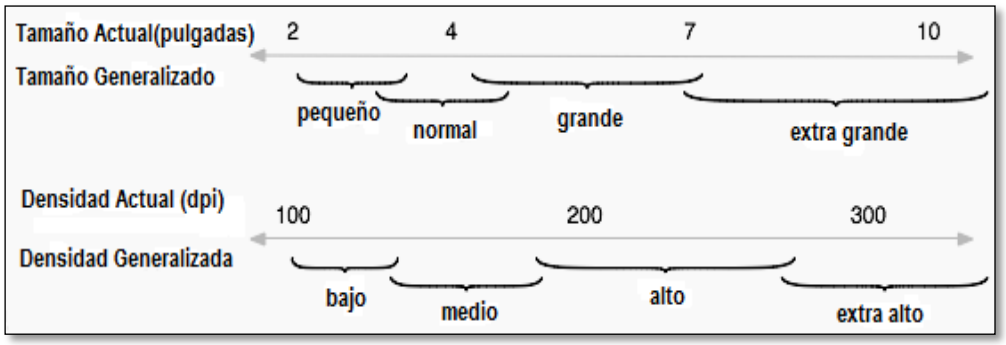


Figura 3.16.- Esquema generalizada de pantallas compatibles

La plataforma JME está enfocada hacia teléfonos móviles pequeños por lo cual esta plataforma no presenta una diversidad de pantallas como Android (ver ANEXO F), los métodos que utiliza para obtener el tamaño de pantalla de un dispositivo son getWidth() y getHeigth() los cuales permiten al programador ajustar los elementos utilizados en la aplicación al tamaño de pantalla de un teléfono en particular, pero si se desea adaptar el tamaño de una imagen para cualquier tipo de pantalla lo que utiliza JME es el método de redimensión de imagen private Image resizeImage(Image src). [22]

La desventaja que presenta esta plataforma es que al redimensionar la imagen al tamaño de la pantalla del dispositivo se pierde la calidad de la imagen lo cual no ocurre en la plataforma Android ya que utiliza la variable de píxel independiente a la densidad la cual permite mantener la definición de la imagen en diferentes pantallas.

Desde el punto de vista del programador es más conveniente que utilice la plataforma Android si desea que el diseño de la interfaz de usuario no se vea afectada al momento de probar su aplicación en una variedad de dispositivos móviles.

CONCLUSIONES

1. De acuerdo con lo explicado en la variable APIs del capítulo 3, debido a la gran cantidad de APIs que provee la plataforma Android se pueden realizar desde aplicaciones sencillas hasta aplicaciones complejas y de gran utilidad como la de realizar una aplicación para conectar dos dispositivos que soportan wi-fi directo y luego comunicarse a través de distancias mucho más larga que bluetooth.

2. Las pruebas hechas en la variable emulador que se encuentra en el capítulo 3, nos indica que Android provee un emulador con un 42.24% mayor en rapidez de la que proporcionan los emuladores (Nokia y Toolkit) para JME.

3. Ambas plataformas me permiten adaptar mi aplicación a diferentes pantallas pero con la diferencia de que la plataforma Android dentro de

sus parámetros utiliza la variable “píxel independiente de la densidad”, la cual permite que al probar la aplicación en varios dispositivos no afecte al diseño de interfaz de usuario, esta funcionalidad no me brinda la plataforma JME.

4. En la actualidad existe gran auge de dispositivos móviles con pantallas grandes (tablets), de las cuales no hay uno que use la plataforma JME para crear aplicaciones, por lo que la plataforma JME está quedando rezagada.

5. En la variable lenguaje de programación del capítulo 3 se observó que con la herramienta App Inventor, crear una aplicación para el sistema operativo Android es muy sencillo, pero obviamente no estará habilitado para todas las funcionalidades que presenta la API de Android. No se podrá crea una aplicación con conectividad WIMAX o NFC.

6. Como se analizó en el capítulo 3 “Comparación de las plataformas Android y JME”, la plataforma Android se destacó en la mayoría de las variables, excepto por la variable tamaño de aplicación que fue la única en que la plataforma JME superó a Android.

RECOMENDACIONES

1. A pesar de que el emulador de Android es rápido comparado con el emulador de JME, es mejor cargar la aplicación a un dispositivo móvil ya que el tiempo en que la aplicación es ejecutada en el dispositivo es menor al tiempo ejecutada en el emulador y el funcionamiento del programa puede ser observado en tiempo real.

2. Si se realiza una aplicación para un dispositivo móvil en específico, entonces se aconseja borrar elementos innecesarios tales como, las carpetas *drawable* no usadas que se encuentran dentro de la carpeta *res,* con la finalidad de aminorar el peso de la aplicación.

3. Debido a la alianza que existe entre las empresas Nokia y Microsoft, los emuladores de Nokia solo pueden ser instalados en los sistemas operativos propietarios de Microsoft, por lo tanto la instalación del emulador Nokia SDK 2.2 de la plataforma JME solo puede efectuarse en los sistemas operativos Windows.

4. Tal como se observó en el capítulo 3, la ventaja de Android sobre JME en muchas variables, y debido a la gran aceptación de Android a nivel global, es recomendable elegir la plataforma Android para desarrollar sus aplicaciones.

ANEXOS

ANEXO A

Descripción de los teléfonos móviles

Los teléfonos que se presentan en la Figura A.1 y Figura A.2 fueron usados para el análisis de las variables tamaño de la aplicación y multitarea.

Características del teléfono Samsung Galaxy Mini

Samsung Galaxy Mini trabaja con un sistema operativo Android versión 2.2 (Froyo) y puede ser actualizado a la versión Android 2.3 (Gingerbread). Posee un CPU con una velocidad de 600 MHz y arquitectura ARM. Un procesador gráfico GPU marca Adreno 200. Contiene sensores de proximidad, acelerómetro y brújula. El tamaño de su pantalla es de 240 x 320 píxeles, 3.14 pulgadas. [23]

Figura A.1: Teléfono Samsung Galaxy Mini

Soporta mensajes SMS, MMS, correo electrónico, IM. Posee radio FM. Tiene GPS con soporte A-GPS. Posee un emulador de java. Batería de Litio de 120 mA, en estado fijo tiene 570 horas de duración, mientras el celular esté siendo usado tiene una duración de 9 horas con 30 minutos. Posee una ranura de tarjeta a la cual se le puede insertar una memoria microSD con capacidad de hasta 32 GB, una memoria interna con 160 MB de capacidad y 384 MB de memoria RAM. Comunicación GPRS y EDGE, Bluetooth, puerto USB, y conexión a redes inalámbricas Wi-Fi. [23]

Características del teléfono Nokia Asha 311

Posee un sistema operativo Serie 40 propietario de Nokia. Cuenta con un CPU de 1 GHz de velocidad. Tiene sensores de proximidad y acelerómetros. Soporta mensajería vía SMS, MMS, correo electrónico, IM. La plataforma de programación que utiliza es JME. Tiene incluido una cámara de 3.15 MP con 2048 x 1356 pixeles de resolución, capacidad para fotos y videos con resolución VGA. [24]

Figura A.2: Teléfono Nokia Asha 311

Batería de Litio de 1110 mA y 3.7 voltios, en estado fijo (sin uso) tiene una duración de 696 horas en tecnología 2G y 768 horas en tecnología 3G. En uso continuo de llamada tiene una duración de 14 horas en tecnologías 2G y de 6 horas en tecnología 3G. Escuchando música tiene una duración de 40 horas en ambas tecnologías. El tipo de comunicación es GPRS y EDGE.

Tiene acceso a la conectividad Wi-Fi, conexión Bluetooth y un puerto USB denominado microUSB de segunda generación. Posee una ranura de tarjeta en la cual puede insertarse una memoria microSD de hasta 32 GB de capacidad, una memoria interna de 140 MB de capacidad, memoria RAM de 128 MB y una memoria ROM de 256 MB. [24]

ANEXO B

Instalación del IDE con Oracle Java ME SDK en Netbeans

Previo a la instalación del IDE debemos tener instalado el entorno de desarrollo Netbeans, los pasos para la instalación se encuentra en el link que está en la referencia [25].

Una vez instalado Netbeans en nuestra computadora lo primero que debemos hacer es descargar el JAVA ME SDK de la siguiente página *http://www.oracle.com/technetwork/java/javame/javamobile/download/sdk/index.html*, aceptamos los acuerdos de la licencia y lo descargamos.

Para instalar el Oracle Java ME SDK

1. Descargue el oracle-jmesdk-3-4-rr-win32-bin.exe archivo instalador y guárdelo en cualquier ubicación del ordenador
2. Haga doble clic en el archivo de instalación para ejecutar el instalador y siga las instrucciones.
3. Cuando la instalación se haya completado, el (TM) ME Platform SDK 3.4 icono Administrador de dispositivos Java () aparece en la bandeja del sistema de Windows.

Ahora procederemos activar la función de JME en Netbeans:

1. Seleccione Tools > Plugins > Avaliable Plugins, hacemos clic en la casilla Java ME y le damos Activate. (Ver Figura B.1)

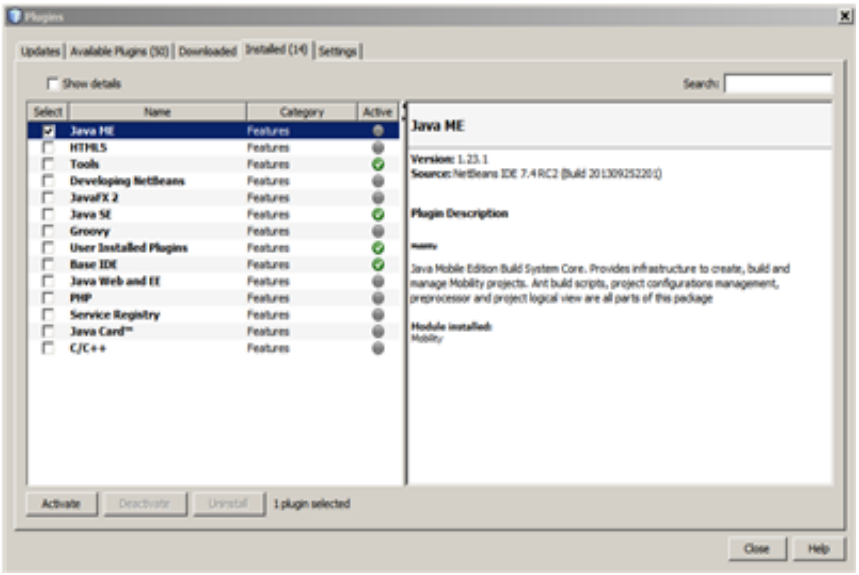


Figura B.1: Selección de Plugins

2. En el panel de bienvenida del cuadro de diálogo Instalador, haga clic en Activate. (Ver Figura B.2)

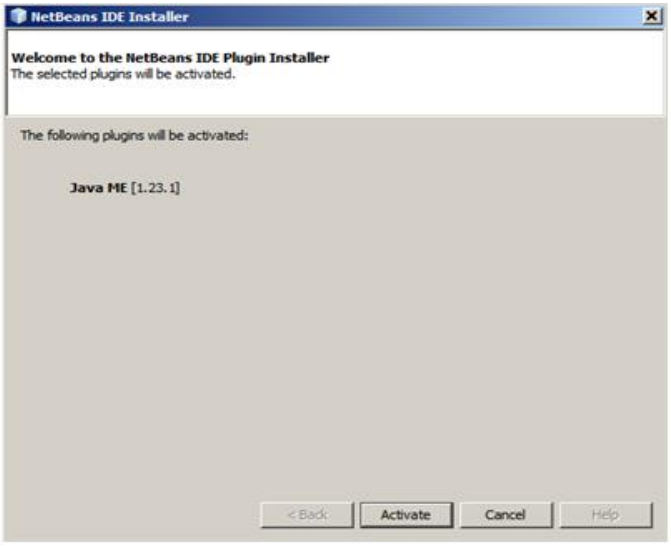


Figura B.2: Activación de Plugins

3. Cuando la activación se ha realizado correctamente, haga clic en Finalizar.

Luego proceder a instalar la plataforma Java ME Oracle:

1. Seleccione Tools > Java Platforms, luego se presentara la ventana de la Figura B.3

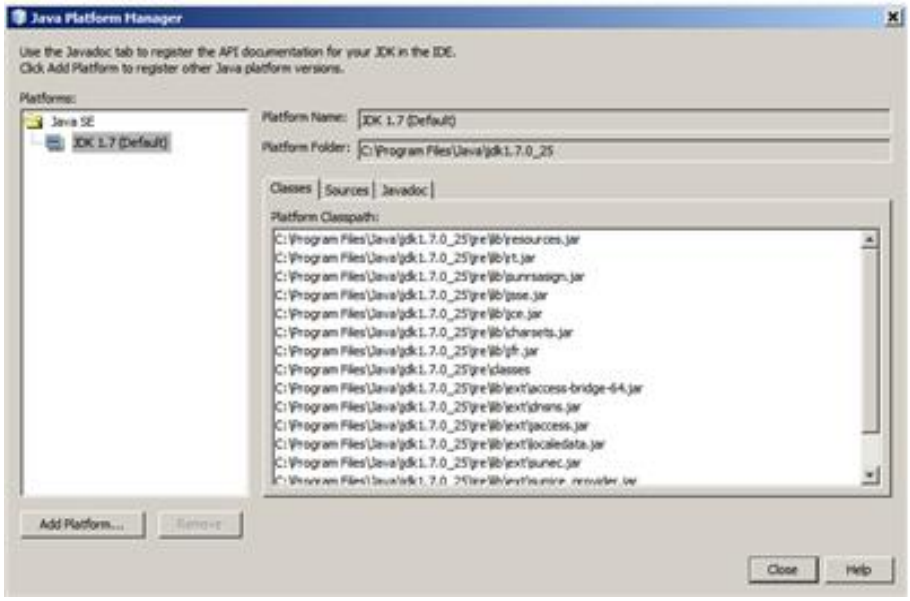


Figura B.3: Añadir Plataforma Java ME

2. Hacer clic en Add Platform. En el cuadro Seleccione Tipo de plataforma del asistente Add Plataforma Java, seleccione Java ME CLDC Plataforma Emulador y haga clic en Siguiente. (Ver Figura B.4)
3. Seleccione el directorio donde se encuentra la plataforma Java ME SDK que al inicio habíamos descargado.

4. Tan pronto como el IDE detecta la plataforma, haga clic en Finish en el panel plataformas, detectado del asistente el complemento Java.

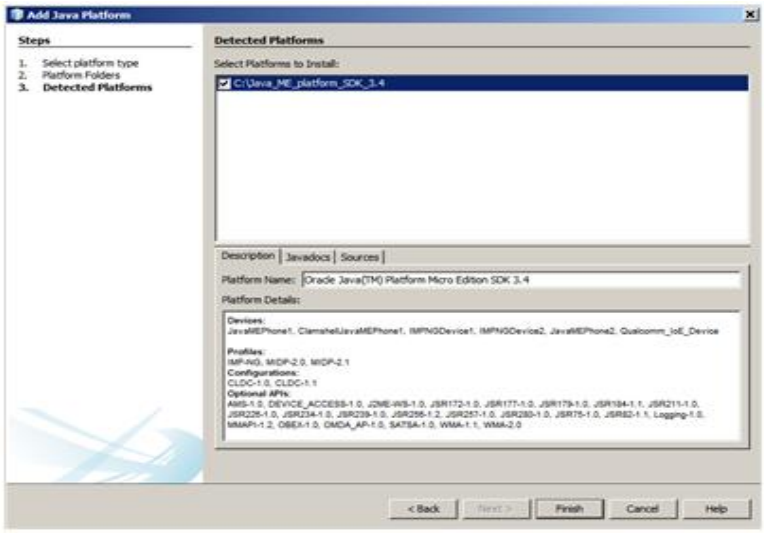


Figura B.4: Selección de la plataforma JME

5. La plataforma Oracle Java ME ya se encuentra registrada en Netbeans (Ver Figura B.5)

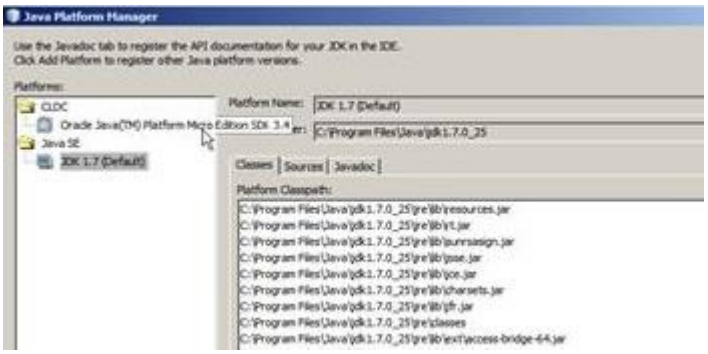


Figura B.5: Plataforma JME instalada

Si deseamos desarrollar interfaces gráficas de usuario rápidamente en la plataforma JME en Netbeans deberíamos instalar Visual Mobile Designer el cual nos permite crear nuestra aplicación con solo arrastrar los elementos que deseamos colocar en la misma y luego procediendo a generar eventos a cada elemento agregado.

Instalación del Visual Mobile Designer

1. Seleccione Tools > Plugins > Avaliable Plugins > dar clic en la casilla Visual Mobile Designer
2. Seleccione Install que se encuentra en la parte inferior izquierda de la ventana.
3. Si se instaló correctamente usted podrá observar la opción Visual MIDlet al momento de crear un nuevo archivo (Ver Figura B.6). [26]

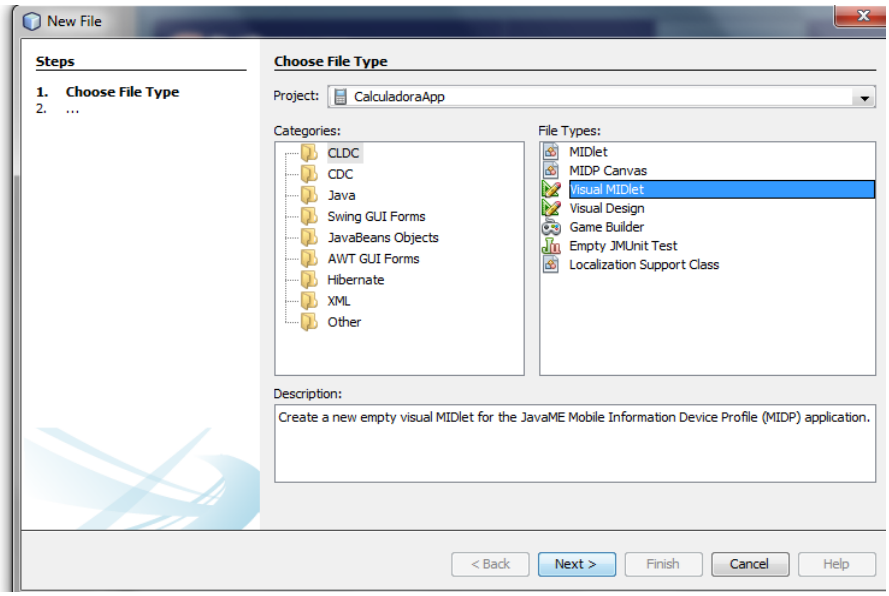


Figura B.6: Creación de un Visual Midlet

ANEXO C

Instalación del SDK de Android

Para la instalación del SDK de Android previamente se debería tener instalado Eclipse en nuestra computadora pero para facilitar la instalación podríamos descargar el paquete ADT para iniciar rápidamente el desarrollo de aplicaciones en Android. Este paquete incluye los componentes de Android SDK y una versión del IDE de Eclipse con una función del Android Developer Tools para agilizar el desarrollo de aplicaciones. El link de descarga del paquete es http://developer.android.com/sdk/index.html (Ver Figura C.1)

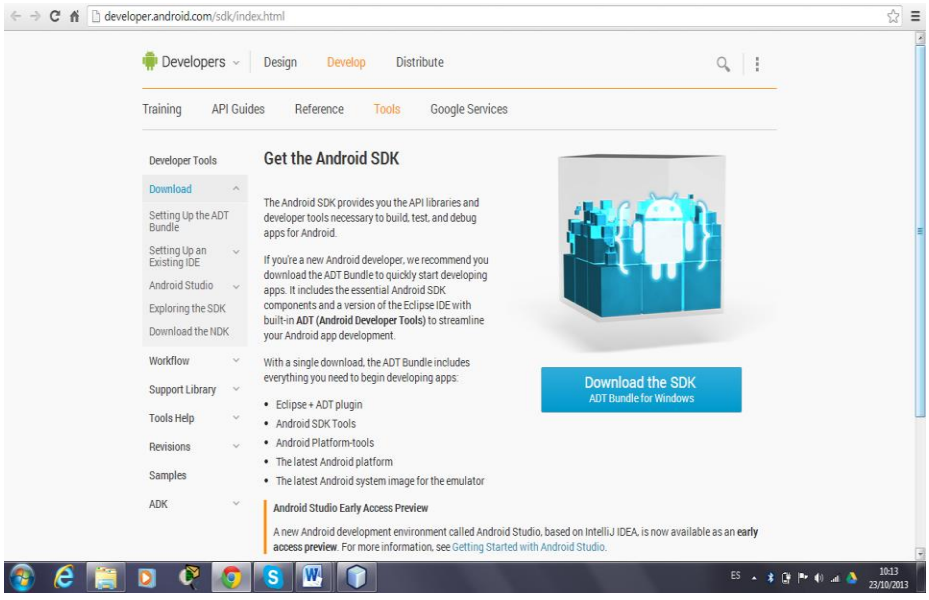


Figura C.1: Página de descarga del Android SDK

Luego de que se haya descargado el Android SDK:

1. Ejecutamos el instalador y pulsamos Next, si se detecta el instalador JDK les va aparecer la siguiente ventana. (Ver Figura C.2)

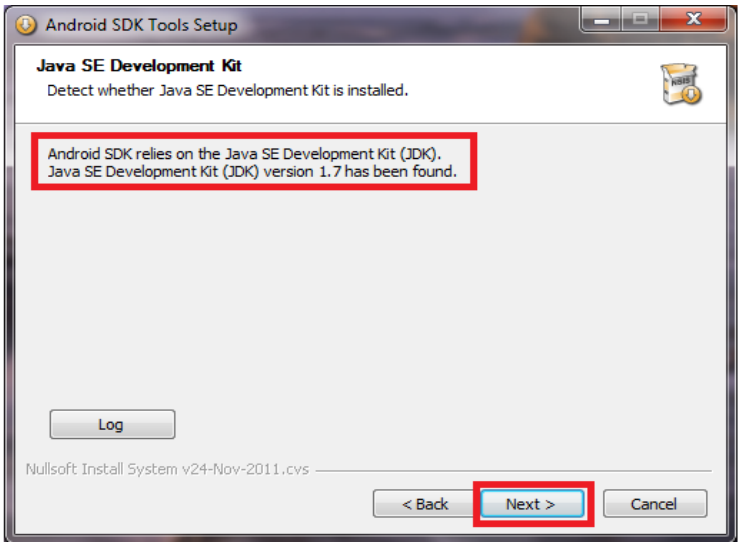


Figura C.2: Instalación del Java SE Development Kit

Si no se encontró el instalador del JDK lo que debemos hacer es seleccionar manualmente la ruta donde se descargó el instalador. (Ver Figura C.3)

2. Seleccione Next hasta que se instale. Una vez instalado le damos clic en continuar y al momento de arrancar el SDK cuando llegue al final del instalador para ejecutar el SDK debemos seleccionar la casilla Start SDK Manager.

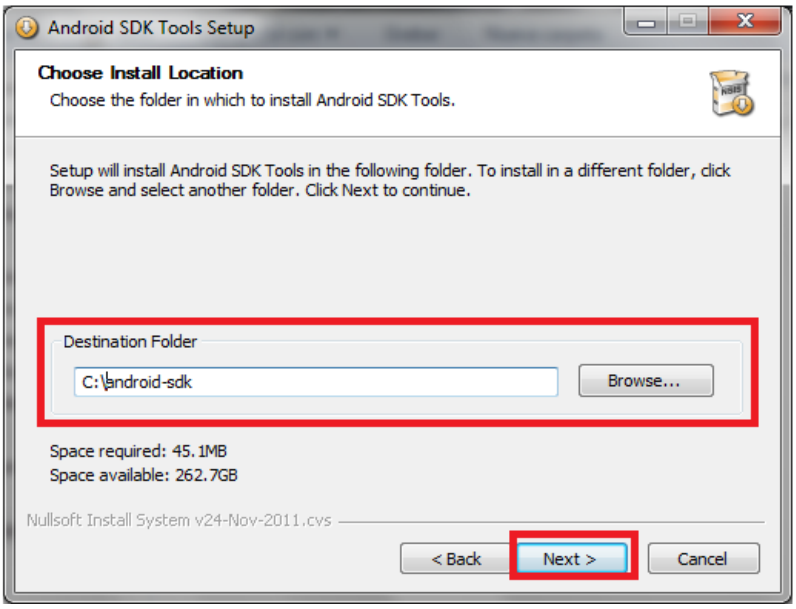


Figura C.3: Selección de la manual de la ruta del instalador JDK

3. Luego se nos abrirá una parte del SDK llamada Android SDK Manager el cual se encontrara vacío, esperaremos a que la barra busque en Internet lo que podemos descargar. (Ver Figura C.4)
4. Una vez que termine de buscar en Internet nos aparecerá lo que se muestra en la Figura C.5 y seleccionamos las casillas que están dentro del marco lila.
5. Seleccione Install 7 packages
6. Nos aparecerá una ventana donde nos indica si aceptamos los términos de la licencia, dar clic en Acept All y luego en Install.
7. Ahora procedemos a instalar el plugin para eclipse denominado ADT para de esta manera vincular el SDK con Eclipse. Vamos a la barra de menú > help > Install New Software.

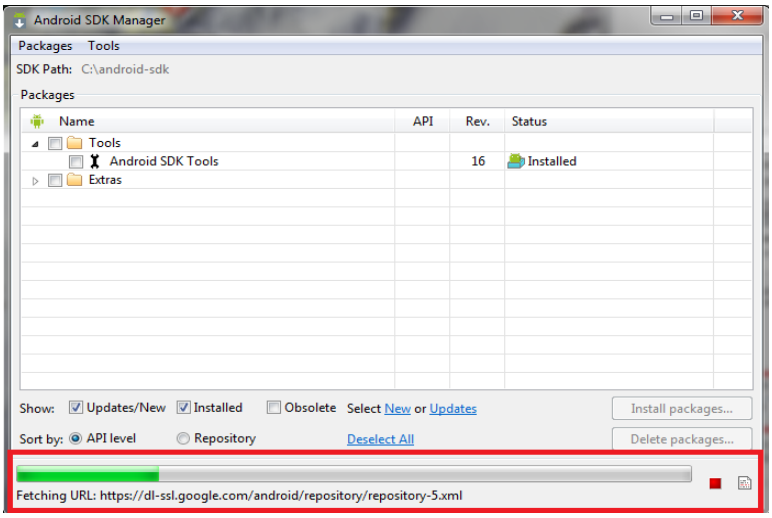


Figura C.4: Android SDK Manager

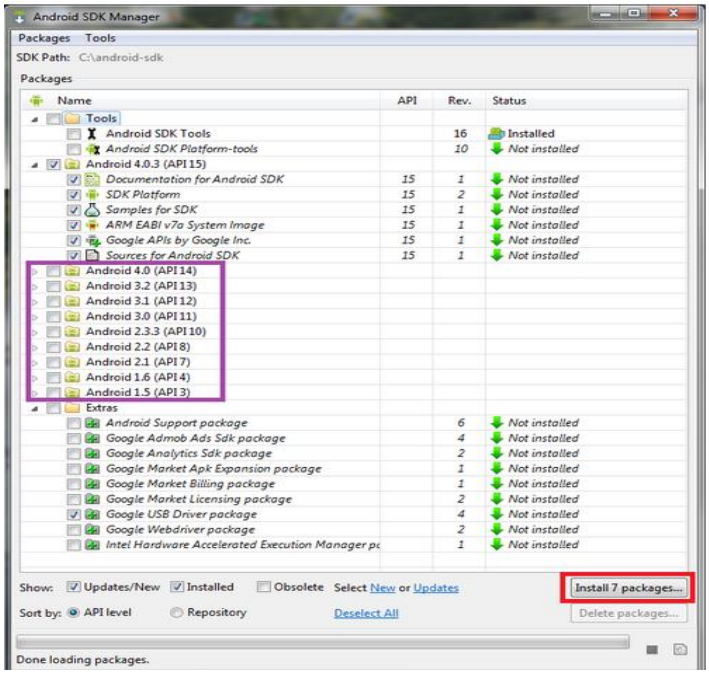


Figura C.5: Selección de casillas para instalación de las versiones de Android

8. Pulsamos en Add y nos aparecerá una nueva ventana, en Name colocamos el nombre que queramos poner a nuestro plugins y en Location https://dl-ssl.google.com/android/eclipse/. (Ver Figura C.6)

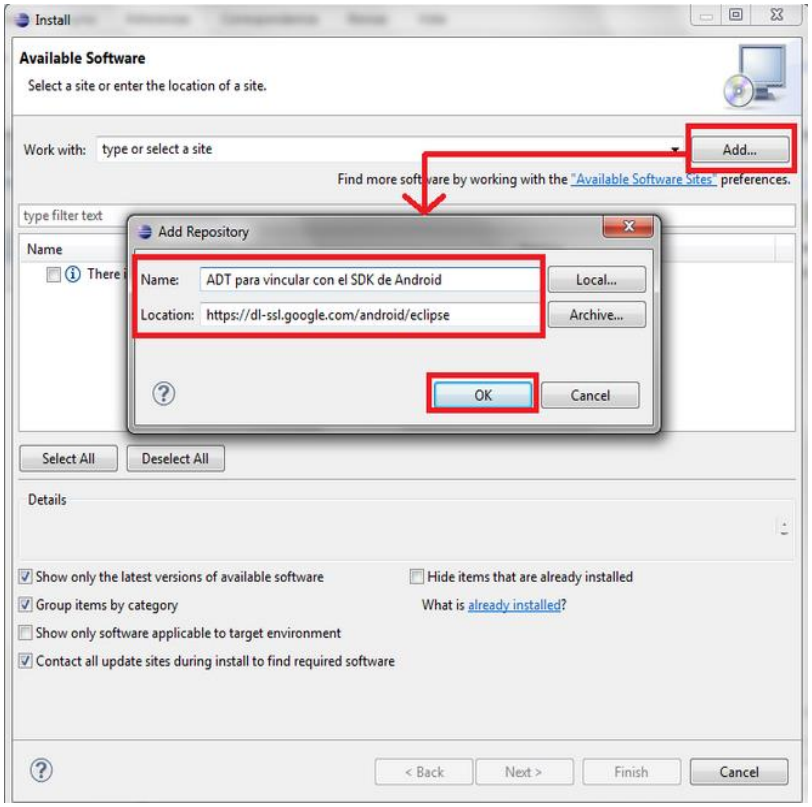


Figura C.6: Instalación del Android Developer Tools

9. Seleccionar la casilla de “Developer Tools” para que se marquen todas y pulsamos en siguiente. (Ver Figura C.7)

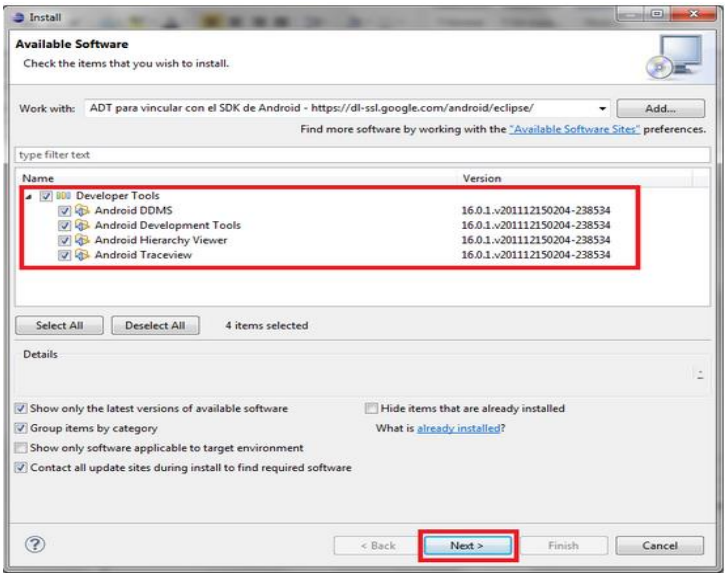


Figura C.7: Selección del Developer Tools

10. Presionamos Next y aceptamos la licencia y pulsamos finalizar. (Ver Figura C.8)

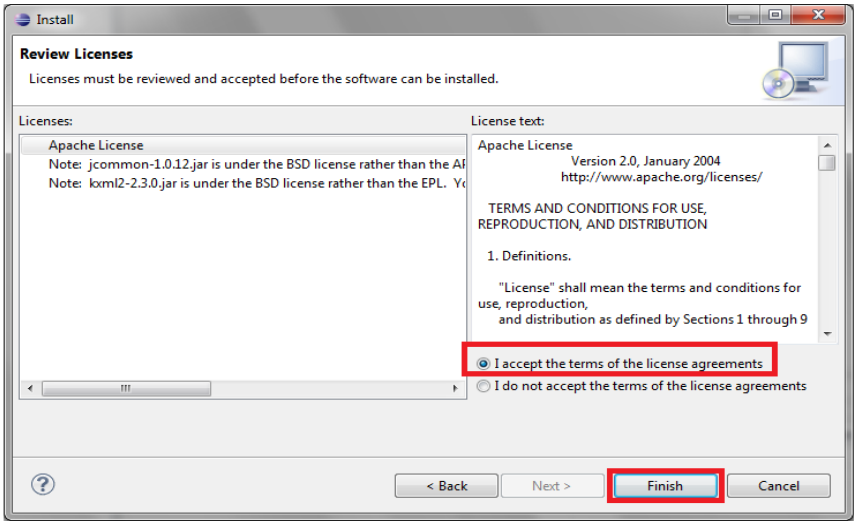


Figura C.8: Términos de aceptación de la licencia Apache

11. Al finalizar nos pedirá reiniciar Eclipse para guardar los cambios. Si todo se instaló correctamente al dar clic en Windows que se encuentra en la barra de menú de Eclipse, deberá aparecer lo que se encuentra señalado con amarillo (Ver Figura C.9) y si es así ya estamos listos para programar nuestras aplicaciones en Android. [27]

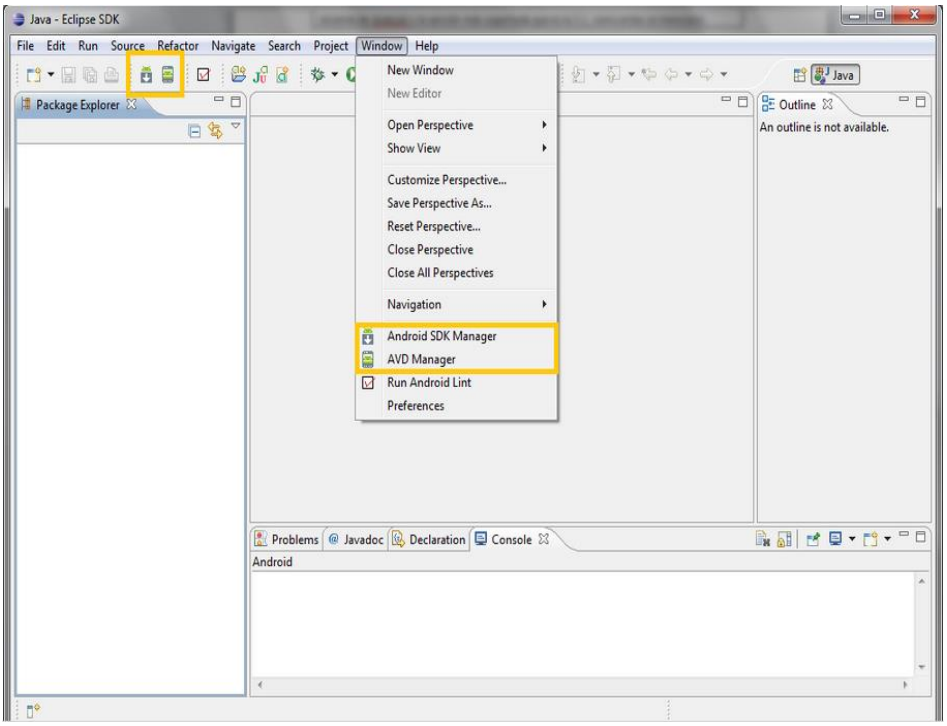


Figura C.9: Instalación correcta y finalizada del SDK

ANEXO D
Código de las Aplicaciones: Calculadora, Texto en pantalla y Tic Tac Toe

Aplicación Hola Mundo

El código en Android sería el siguiente:

```
public class HolaMundo extends Activity {
  public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        TextView texto = new TextView(this);
        texto.setText("Hola Mundo");
        setContentView(texto);
  }
}
```

El código en JME es:

```
public class HolaMundo extends MIDlet {
  // Componentes del interfaz de usuario
  private Display display;
  private Form form;

  //Constructor de la clase.
  public HolaMundo() {
    // Cogemos el display
    display=Display.getDisplay(this);
```

```
    // Creamos el form
    form = new Form ("Mi primer MIDlet");
    form.append("Hola Mundo!!!\n");
  }
  //Método que iniciar la ejecución del midlet
  protected void startApp() {
     display.setCurrent(form);
  }
  //Método que interrumpe la ejecución del midlet
  protected void pauseApp() { }

  //Método que se llama cuando se finaliza el midlet
  protected void destroyApp(boolean incondicional) {
    display = null;
    form = null;
  }
}
```

Aplicación de una calculadora

El código en Android es:

```
public class MainActivity extends Activity implements OnClickListener{

        // Creación de botones: suma, resta, dividir, multiplicar
        private Button botonSumar, botonRestar, botonMult, botonDiv;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);//muestra en pantalla los   componentes
de la aplicacion
```

```
                botonSumar= (Button)findViewById(R.id.buttonSuma);//encuentra el boton
creado en activity_main
                botonSumar.setOnClickListener(this);//establece un evento para cuando el
boton es oprimido

                botonRestar= (Button)findViewById(R.id.buttonResta);
                botonRestar.setOnClickListener(this);

                botonMult= (Button)findViewById(R.id.buttonMult);
                botonMult.setOnClickListener(this);

                botonDiv= (Button)findViewById(R.id.buttonDiv);
                botonDiv.setOnClickListener(this);
@Override
        public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is // present.
                getMenuInflater().inflate(R.menu.main, menu);
                return true;
        }

        public void onClick(View vista){
                TextView temp1, temp2;
                TextView resultado = (TextView)findViewById(R.id.textView1);
                temp1= (TextView)findViewById(R.id.editText1);// encuentra las cajas de texto
creados en activity_main
                temp2 = (TextView)findViewById(R.id.editText2);
                try{
                        //obtiene los valores de la caja de texto
                        float a = Float.parseFloat(temp1.getText().toString());
                        float b = Float.parseFloat(temp2.getText().toString());//y los convierte en
una variable tipo float
                        if(botonSumar.isPressed()){// si el usuario oprime el boton sumar
```

```
// establece el texto en el TextView para ser mostrado por pantalla
                        resultado.setText("resultado:"+(a+b));
                    }
                    else if(botonRestar.isPressed()){// si el usuario oprime el boton restar
                        resultado.setText("resultado:"+(a-b));

                    }
                    else if(botonMult.isPressed()){// si el usuario oprime el boton multiplicar
                        resultado.setText("Resultado: "+(a*b));
                    }
                    else{// si el usuario oprime el boton dividir

                        // si el dividendo es cero lanza una excepcion
                        if(b == 0) throw new DivisionPorCeroException("No se puede
dividir por cero");
                        else
                            resultado.setText("Resultado: "+(a/b));
                    }

                }catch(DivisionPorCeroException e){// clase creada para evitar dividir por cero
                    resultado.setText(e.getMessage());

                }catch(Exception e) {
                    resultado.setText("Se debe introducir solo numeros");
                }

        }

        private class DivisionPorCeroException extends Exception{

            private String texto;
            public DivisionPorCeroException(String texto){
```

```
                this.texto = texto;
            }

            public String getMessage(){

                return texto;
            }
        }
}
```

El código de calculadora en JME es:

```
//La función getOpciones devuelve las opciones (suma, resta, multiplicación y división) asociadas al evento.
public Form getOpciones() {
    if (Opciones == null) {

        Opciones = new Form("Opciones", new Item[]{getA(), getB(), getR()});
        Opciones.addCommand(getSum());
        Opciones.addCommand(getResta());
        Opciones.addCommand(getMultiplicacion());
        Opciones.addCommand(getDivision());
        Opciones.addCommand(getExitCommand());
        Opciones.setCommandListener(this);
    }
    return Opciones;
  }

//Funciones que retornan el Item asociado a cada llamado de la función opciones
   public Command getSum() {
    if (sum == null) {
```

```
            sum = new Command("suma", Command.ITEM, 0);
        }
        return sum;
    }

    public Command getResta() {

        if (resta == null) {
            resta = new Command("resta", Command.ITEM, 0);
        }
        return resta;
    }
            public Command getMultiplicacion() {
      if (multiplicacion == null) {
       multiplicacion = new Command("multiplicacion", Command.ITEM, 0);
        }
        return multiplicacion;
    }

    public Command getDivision() {
        if (division == null) {
            division = new Command("division", Command.ITEM, 0);
        }
        return division;

    }

//Funciones que retornan el número ingresado por el usuario
    public TextField getA() {
        if (a == null) {
                a = new TextField("Ingrese primer numero", null, 32, TextField.ANY);
            a.setInitialInputMode("null");
        }
```

```
        return a;
    }

 public TextField getB() {
        if (b == null) {
                b = new TextField("Ingrese segundo numero", null, 32, TextField.ANY);
        }

       return b;
    }

//Función que retorna el resultado de la operación
  public StringItem getR() {
        if (r == null) {
           r = new StringItem("Resultado", null);
        }
        return r;
    }

//Función que controla la pantalla y la interacción con el usuario
  public Display getDisplay() {
        return Display.getDisplay(this);
    }

//Procedimiento que se encarga de cerrar el MIDlet
  public void exitMIDlet() {
        switchDisplayable(null, null);
        destroyApp(true);
        notifyDestroyed();
    }

//Procedimiento que pone en marcha el MIDlet
```

```
public void startApp() {
    if (midletPaused) {
        resumeMIDlet();
    } else {
        initialize();
        startMIDlet();
    }
    midletPaused = false;}

//Procedimiento que pausa ya que no hay actividades en segundo plano
public void pauseApp() {
    midletPaused = true;
  }

//Código que realiza las 4 operaciones básicas (Suma, Resta, Multiplicación y división)
public void commandAction(Command command, Displayable displayable) {

        valor1= Double.parseDouble( a.getString());
        // se obtiene el string escrito por el usuario luego se realiza
    // un cast y se lo convierte el valor a double y finalmente
    //  se le asigna ese valor a la variable local valor1

     valor2= Double.parseDouble( b.getString());
    if (displayable == Opciones) {
        if (command == division) {  //valida que tipo de operación    básica selecciono en usuario
                                                    if (valor2==0) // con esta sentencia no permite
                                                                                una división para cero
            r.setText("VALOR2 INGRESADO INCORRECTO");
          else{
                    resultado= valor1 / valor2;// se realiza la división //y se la guarda en la variable
                                                                                resultado
```

```
                r.setText(String.valueOf(resultado));}
                                    // envía el contenido de la variable resultado a       r y lo presenta
                                    en pantalla
        } else if (command == exitCommand) {
            exitMIDlet();
                                } else if (command == multiplicacion) {                    // se ejecuta
                                todo el codigo que se encuentra a c    continuación si el usuario
                                selecciona la opción multiplicación
             resultado= valor1 * valor2;
            r.setText(String.valueOf(resultado));
        } else if (command == resta) {
            resultado= valor1 - valor2;
            r.setText(String.valueOf(resultado));
        } else if (command == sum) {
            resultado= valor1 + valor2;
            r.setText(String.valueOf(resultado));
        }
    }

}
```

Aplicación de tres en raya

El código en Android:

```
package com.example.tiv;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
```

```
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {
        // Crea los botones
    Button btn00,btn01,btn02,btn10,btn11,btn12,btn20,btn21,btn22;

    //Dialogo Builder construido para mostrar un error o una alerta al jugador
    AlertDialog.Builder alert,error;

    // Etiqueta para ser mostrada en pantalla
    TextView playerLbl,header;

  @Override
  public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setClickEvent();
  }

  // Cuando un boton es oprimido
  public void setClickEvent(){

    btn00=(Button)findViewById(R.id.btn00);
    btn01=(Button)findViewById(R.id.btn01);
    btn02=(Button)findViewById(R.id.btn02);
    btn10=(Button)findViewById(R.id.btn10);
    btn11=(Button)findViewById(R.id.btn11);
```

```
    btn12=(Button)findViewById(R.id.btn12);
    btn20=(Button)findViewById(R.id.btn20);
    btn21=(Button)findViewById(R.id.btn21);
    btn22=(Button)findViewById(R.id.btn22);
    btn00.setOnClickListener(new Clicker());
    btn01.setOnClickListener(new Clicker());
    btn02.setOnClickListener(new Clicker());
    btn10.setOnClickListener(new Clicker());
    btn11.setOnClickListener(new Clicker());
    btn12.setOnClickListener(new Clicker());
    btn20.setOnClickListener(new Clicker());
    btn21.setOnClickListener(new Clicker());
    btn22.setOnClickListener(new Clicker());

}

// Para repetir el juego
public void resetButton(){
        // Quita las etiquetas de los botones
    btn00.setText("");
    btn01.setText("");
    btn02.setText("");
    btn10.setText("");
    btn11.setText("");
    btn12.setText("");
    btn20.setText("");
    btn21.setText("");
    btn22.setText("");
    playerLbl.setText("Player1");

    header=(TextView)findViewById(R.id.header);
    header.setText("Tic Tac Toe : New Game");
```

```
    }

    // Crea el dialogo
    public void createDialog(){
          //Muestra el dialogo de ganador
        alert=new AlertDialog.Builder(this);
        alert.setTitle("Winner");
        alert.setPositiveButton("OK",new DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int which) {

                resetButton();
            }

        });
        //Muestra un error
        error=new AlertDialog.Builder(this);
        error.setTitle("Error");
        error.setMessage("This cell is occupied");
        // Cierra el dialogo
        error.setPositiveButton("OK",new DialogInterface.OnClickListener(){

            public void onClick(DialogInterface dialog, int which) {
            }
        });
    }

    // Verifica quien es el ganador del juego
    public boolean determineWinner()

    {
        createDialog();
if((btn00.getText().toString().equals("" +"X")
```

```
                    && btn01.getText().toString().equals("X") &&
btn02.getText().toString().equals("X"))
                    || (btn00.getText().toString().equals("X") &&
btn10.getText().toString().equals("X")
                            && btn20.getText().toString().equals("X"))
            || (btn10.getText().toString().equals("X") && btn11.getText().toString().equals("X")
                            && btn12.getText().toString().equals("X"))
            || (btn20.getText().toString().equals("X") && btn21.getText().toString().equals("X")
                            && btn22.getText().toString().equals("X"))
            || (btn01.getText().toString().equals("X") && btn11.getText().toString().equals("X")
                            && btn21.getText().toString().equals("X"))
            || (btn02.getText().toString().equals("X") && btn12.getText().toString().equals("X")
                            && btn22.getText().toString().equals("X"))
            || (btn00.getText().toString().equals("X") &&

btn11.getText().toString().equals("X")
                            && btn22.getText().toString().equals("X"))
                    || (btn02.getText().toString().equals("X") &&
btn11.getText().toString().equals("X")
                            && btn20.getText().toString().equals("X"))){
            alert.setMessage("Winner is : Player1");
            alert.show();
            return true;
        }

        if((btn00.getText().toString().equals("0")
                    && btn01.getText().toString().equals("0") &&
btn02.getText().toString().equals("0"))
                    || (btn00.getText().toString().equals("0") &&
btn10.getText().toString().equals("0")
                            && btn20.getText().toString().equals("0"))
            || (btn10.getText().toString().equals("0") && btn11.getText().toString().equals("0")
                            && btn12.getText().toString().equals("0"))
```

```
            || (btn20.getText().toString().equals("0") && btn21.getText().toString().equals("0")
                    && btn22.getText().toString().equals("0"))
            || (btn01.getText().toString().equals("0") && btn11.getText().toString().equals("0")
                    && btn21.getText().toString().equals("0"))
            || (btn02.getText().toString().equals("0") && btn12.getText().toString().equals("0")
                    && btn22.getText().toString().equals("0"))
            || (btn00.getText().toString().equals("0") && btn11.getText().toString().equals("0")
                    && btn22.getText().toString().equals("0"))
                || (btn02.getText().toString().equals("0") &&
btn11.getText().toString().equals("0")
                    && btn20.getText().toString().equals("0"))){

            alert.setMessage("Winner is : Player2");
            alert.show();

            return true;
        }

      return false;
    }

    // Vuelve a verificar el ganador
    public void checkAgain(){
      boolean flag;

      if(!btn00.getText().toString().equals("") && !btn01.getText().toString().equals("")
                && !btn02.getText().toString().equals("") &&
!btn10.getText().toString().equals("")
                && !btn11.getText().toString().equals("") &&
!btn12.getText().toString().equals("")
                && !btn20.getText().toString().equals("") &&
!btn21.getText().toString().equals("")
                && !btn22.getText().toString().equals(""))
```

```
  {
            flag=determineWinner();

            if(!flag){

                alert.setMessage("No One Won , GAME OVER !!!!!");
                alert.show();
            }
  }
}

// Clase que es llamada cuando un boton es oprimido
class Clicker implements OnClickListener{

  boolean flag;

  public void onClick(View v){

       playerLbl=(TextView)findViewById(R.id.playerLbl);

       String lbl=playerLbl.getText().toString();
       Button btn=(Button)v;

       if(btn.getText().toString().equals("")){
            if(lbl.equals("Player1")){

                btn.setText("X");
                playerLbl.setText("Player2");
            }
            else{

                btn.setText("0");
```

```
                playerLbl.setText("Player1");
            }
        }

        else{

            error.show();
        }

        flag=determineWinner();
            // Si no se ha determinado ganador
        if(!flag){
            checkAgain();
        }
    }
  }
}
```

El código en JME:

```
package tresenraya;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class TicTacToe extends MIDlet implements CommandListener {

  // Presenta los elementos en la pantalla
  private Display theDisplay;
  private TicTacToeCanvas canvas;

  // Guarda el usuario que gana si es el player1 o el player2
```

```
private String winner = "";
private boolean gameOver = false;

//Estado de juego y trablero de juego para el tic tac toe
private int whoseturn = 0;
private char board[][] = { {'1' , '2' , '3'} ,
                           {'4' , '5' , '6'} ,
                           {'7' , '8' , '9'} };

// Command
static final Command exitCommand = new Command("Exit", Command.STOP, 1);

public TicTacToe()
{
   // Crea el main de la pantalla
   theDisplay = Display.getDisplay(this);
}

class TicTacToeCanvas extends Canvas
{
  public void paint(Graphics g)
  {
     int width = getWidth();
     int height = getHeight();
     g.setColor( 0, 0, 0 );
     g.fillRect( 0, 0, width, height );
     g.setColor( 255, 255, 255);
     String r1 = board[0][0]+"|"+board[0][1]+"|"+board[0][2]+"\n";
     String r2 = board[1][0]+"|"+board[1][1]+"|"+board[1][2]+"\n";
     String r3 = board[2][0]+"|"+board[2][1]+"|"+board[2][2];
     g.drawString("TIC TAC TOE",15,0, Graphics.LEFT|Graphics.TOP);
     g.drawString(r1,30,15, Graphics.LEFT|Graphics.TOP);
     g.drawString(r2,30,30, Graphics.LEFT|Graphics.TOP);
```

```
    g.drawString(r3,30,45, Graphics.LEFT|Graphics.TOP);
    if (winner != "")
    {
          g.drawString("WINNER: "+winner,0,60,Graphics.LEFT|Graphics.TOP);
    }
    else
    {
          g.drawString("TURN: Player "+(whoseturn+1),0,60,Graphics.LEFT|Graphics.TOP);
    }
}

// Chequea la tecla presionada por el jugador
protected void keyPressed(int keyCode) {
 // Llena con X o con O el tablero
  if (keyCode >= 49 && keyCode <= 57 && !gameOver)
  {
     int x = 0, y = 0;
     keyCode -= 49;
     if (keyCode < 3)
     {
       x = keyCode;
       y = 0;
     }
     else if (keyCode < 6)
     {
       x = keyCode-3;
       y = 1;
     }
     else
     {
       x = keyCode-6;
       y = 2;
     }
```

```
      if (board[y][x] != 'X' && board[y][x] != 'O')
       {
         if (whoseturn == 0)
          board[y][x] = 'X';
         else
           board[y][x] = 'O';

         // turno del siguiente jugador
         whoseturn = 1-whoseturn;

         checkForWinner();

         // pintar la pantalla
         canvas.repaint();
        }
      }
    }
}

protected void startApp() throws MIDletStateChangeException
{
   canvas = new TicTacToeCanvas();

   // Añade el comando exit
   canvas.addCommand(exitCommand);
   canvas.setCommandListener(this) ;

   theDisplay.setCurrent(canvas);
}

protected void pauseApp() { }
```

```
public void destroyApp(boolean unconditional)

{  }
// Manejo de eventos
public void commandAction(Command c, Displayable d)
{
 if (c == exitCommand)
 {
   destroyApp(false);
   notifyDestroyed();
 }
}

private void checkForWinner()
{
   int i,j;
   int numXConsec = 0;   // Numeros Consecutivos para X
   int numOConsec = 0;   //Numeros Consecutivos para O

   boolean tie=true;

   // verifica el tic tac toe
   for (i=0; i<3; i++)
   {
        for (j=0; j<3; j++)
             if (board[i][j] != 'X' || board[i][j] != 'O')
             {
                  tie = false;
                  break;
             }
   }
   if (tie)
   {
```

```
        GameOver(-1);
        return;
    }
    //Chequea horizontalmente el tablero para ver si hay un ganador
    for (i=0; i<3; i++)
    {
        numXConsec = numOConsec = 0;
        for (j=0; j<3; j++)
        {
            if (board[i][j] == 'X')
                numXConsec++;
            else if (board[i][j] == 'O')
                numOConsec++;
            else
                break;
        }
        if (numXConsec > 2)
        {
            GameOver(0);
            return;
        }
        else if (numOConsec > 2)
        {
            GameOver(1);
            return;
        }
    }

    // Chequea el tablero verticalmente
    for (i=0; i<3; i++)
    {
        numXConsec = numOConsec = 0;
        for (j=0; j<3; j++)
```

```
            {
                if (board[j][i] == 'X')
                {
                    numXConsec++;
                }
                else if (board[j][i] == 'O')
                    numOConsec++;
                else
                    break;
            }
            if (numXConsec > 2)
            {
                GameOver(0);
                return;
            }
            else if (numOConsec > 2)
            {
                GameOver(1);
                return;
            }
        }

        if ( ((board[0][0] == board[1][1]) && (board[1][1] == board[2][2]) &&
(board[2][2] == 'X')) ||
        ((board[0][2] == board[1][1]) && (board[1][1] == board[2][0]) &&
(board[2][0] == 'X')) )
        {
            GameOver(0);
            return;
        }
        if ( ((board[0][0] == board[1][1]) && (board[1][1] == board[2][2]) &&
(board[2][2] == 'O')) ||
        ((board[0][2] == board[1][1]) && (board[1][1] == board[2][0]) &&
```

```
(board[2][0] == 'O')) )
        {
            GameOver(1);
            return;
        }
    }

    private void GameOver(int p)
    {
      if (p == -1)
          winner = "Tie!";
      else if (p == 0)
          winner = "Player 1!";
      else
          winner = "Player 2!";
      gameOver = true;
    }}
```

ANEXO E

APIs de las plataformas comparadas

APIs de JME:

java.io: Operaciones de E/S básicas.

java.lang: Provee de clases que son fundamentales para el lenguaje de programación Java.

java.util: Contiene las clases de colección, y las instalaciones de fecha y hora.

javax.microedition.midlet: Define las interacciones entre la aplicación y el entorno en que se ejecuta la aplicación.

javax.microedition.lcdui: La API de interfaz de usuario proporciona un conjunto de características para la implementación de interfaces de usuario para aplicaciones MIDP.

javax.microedition.rms: Almacenamiento persistente en el dispositivo.

javax.microedition.io: Conexión genérica, tiene como misión crear y manejar diferentes conexiones de red (por HTTP, datagramas, sockets). [28]

javax.comm: Es una extensión de Java que permite el acceso mediante software al puerto serial, puerto paralelo, y al modo SPP.

java.text: Clases e interfaces para gestionar textos, número, fechas, etc.

java.net: Clases e interfaces de red (TCP/IP).

javax.microedition.midlet: Ciclo de vida de las aplicaciones (MIDlets).

javax.microedition.lcdui.game: Clases e interfaces de interfaz gráfica para juegos.

java.math: Clases soporte matemático.

java.security: Clases e interfaces de seguridad y gestión de certificados.

javax.microedition.pki: Clases e interfaces de seguridad basada en clave pública.

javax.microedition.media.control: Clases, interfaces para reproducción de sonido.

JSR 256: "Mobile Sensor API": Sirve para manejar el sensor de orientación, y el acelerómetro. [28]

JSR 179 Location API: El API de ubicación permite a los MIDlets acceder a la información y los servicios de ubicación a través de GPS.

JSR 234 API AMMS (API Advanced Multimedia Supplements): Permite crear MIDlets con características multimedia avanzadas. Usted puede acceder a los controles específicos de la cámara, como el brillo, contraste, flash, modos de iluminación y zoom. También puede acceder a las capacidades de procesamiento de audio avanzado y radio.

JSR 82 API Bluetooth: La API de Bluetooth permite utilizar conexiones Bluetooth en tu MIDlet.

JSR 211 CHAPI (Content Handler API): Permite a los MIDlets invocar registros Java ME y las aplicaciones que no son Java para el manejo de contenido.

JSR 211 API M3G (API Mobile 3D Graphics): La API de gráficos 3D Mobile permite a los MIDlets utilizar gráficos en 3D en su interfaz de usuario.

JSR 238 API Internacionalización Mobile: Permite a los MIDlets formatear los elementos de datos de acuerdo a los lugares indicados, gestionar los recursos de aplicaciones y comparar cadenas según las reglas especificadas por la localidad dada. [28]

JSR 135 MMAPI (Mobile Media API): La API de medios de comunicación móvil permite a los MIDlets visualizar imágenes y reproducir audio y video.

JSR Nokia UI API: El Nokia UI API es una extensión de Nokia-propiedad de MIDP. Proporciona una funcionalidad adicional, especialmente para los desarrolladores de juegos, en forma de capacidades de audio y gráficos.

JSR 75 (FC) PDA Optional Package: FC API (FileConnection API): Permite a los MIDlets acceder al sistema de archivos del dispositivo.

API de VoIP: Permite desarrollar MIDlets con funciones de VoIP.

JSR 75 (PIM) PDA Optional Package: PIM API (Personal Information Management API): Permite a los MIDlets acceder a los datos personales almacenados por los usuarios, incluyendo las listas de contactos, calendarios y listas de tareas pendientes.

JSR 226 Scalable 2D Vector Graphics API (M2G API): La API de gráficos vectoriales 2D escalables permite a los MIDlets cargar y representan las imágenes vectoriales 2D externos almacenados en el formato SVG Tiny W3C. El API se puede extender para soportar otros formatos. [28]

JSR 177 SATSA API (Security and Trust Services API): La API permite a los MIDlets utilizar certificados PKI, almacenamiento de datos privados, las operaciones criptográficas y servicios seguros para la autenticación, la autorización y la identificación.

JSR 172 (RPC) y 172 (XML Parser) Web Services API: Los paquetes proporcionan capacidades básicas de procesamiento de XML y permite la reutilización de conceptos de servicios web en el diseño de los clientes de Java ME para los servicios de la empresa.

JSR 205 (WMA 2.0) WMA (Wireless Messaging API): El API de mensajería inalámbrica permite a los MIDlets redactar, enviar y recibir mensajes SMS y MMS. [28]

JSR 257 Contactless Comunication API: Permite conectar dos dispositivos mediante NFC.

APIs de Android:

android.util: El 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: El paquete de sistema operativo permite el acceso a los servicios básicos como el paso de mensajes, la comunicación entre procesos y funciones de reloj.

android.graphics: La API de gráficos, es el suministro de las clases de bajo nivel como lienzos de apoyo, colores y las primitivas de dibujo. También le permite dibujar sobre lienzos.

android.text: Las herramientas de procesamiento de texto para mostrarlo y analizarlo. [24]

android.database: Contiene clases para explorar los datos devueltos a través de un proveedor de contenido.

android.database.sqlite: Contiene las clases de gestión de bases de datos SQLite que una aplicación podría utilizar para gestionar su propia base de datos privada.

android.gesture: Proporciona clases para crear, reconocer, cargar y guardar gestos.

android.content: 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.content.pm: Contiene clases para acceder a información sobre un paquete de aplicaciones, incluyendo la información sobre sus actividades, permisos, servicios, firmas y proveedores. [29]

android.content.res: Contiene clases para el acceso a los recursos de aplicaciones, tales como los archivos RAW de activos, colores dibujables, medios de comunicación u otros otros archivos en el paquete, además de importantes detalles de configuración de dispositivos (orientación, tipos de entrada, etc) que afectan a la forma de la aplicación se puede comportar.

android.view: Proporciona clases que exponen clases básicas de interfaz de usuario que se encargan de diseño de la pantalla y la 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 de alto nivel que proporciona acceso a los controles de mapas que usted puede utilizar en su aplicación. Incluye el control MapView así como la superposición y la clase MapController utilizados para anotar y controlar dichos mapas.

android.app: Contiene clases de alto nivel que encapsulan el modelo general de la aplicación Android.

android.provider: Para facilitar el acceso a los desarrolladores a determinados proveedores de contenidos estándar, el paquete proveedor ofrece clases para todas sus distribuciones. [29]

android.telephony: Las API´s de telefonía le dan la posibilidad de interactuar directamente con el dispositivo móvil, permitiéndole realizar, recibir y controlar las llamadas del teléfono, su 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: Contiene las clases de API de marco que definen los servicios basados en la localización y afines Android haciendo uso de GPS, torres de telefonía móvil y puntos de acceso Wi-Fi.

Google Location Services API: Parte de Google Play Services, proporciona un marco más potente y de alto nivel que automatiza tareas como la ubicación elección de proveedor y la administración de energía.

Geofencing API: Esta API me permite la configuración de los límites geográficos en torno a lugares específicos y recibir notificaciones cuando el usuario entra o sale de esas áreas.

android.media: Proporciona clases que administran diversas interfaces de comunicación de audio y video. [29]

android.opengl: Proporciona una interfaz estática OpenGL ES y los servicios públicos. Programación en 3D.

android.hardware: Es compatible con las características de hardware, tales como la cámara y otros sensores.

android.hardware.usb: Proporciona soporte para comunicarse con el hardware periférico USB que se conectan a los dispositivos con Android.

android.bluetooth: Proporciona clases que administran la funcionalidad Bluetooth, tales como la exploración de dispositivos, la conexión con los dispositivos, y la gestión de la transferencia de datos entre dispositivos.

android.accessibilityservice: Las clases de este paquete se utilizan para el desarrollo de los servicios de accesibilidad que proporcionar información alternativa o aumentada para el usuario.

android.app.admin: Proporciona funciones de administración de dispositivos a nivel de sistema, lo que permite crear aplicaciones de seguridad con reconocimiento de que son útiles en los entornos empresariales y en los que los profesionales.

android.app.backup: Contiene la copia de seguridad y restaurar la funcionalidad disponible para las aplicaciones.

android.net: Tiene clases que ayudan con acceso a la red.

android.media.audiofx: Proporciona clases que administran los efectos de audio implementadas en el marco de los medios de comunicación. [29]

android.media.effect: Proporciona clases que permiten aplicar una variedad de efectos visuales a las imágenes y videos.

android.net.rtp: Proporciona APIs de RTP, permitiendo a las aplicaciones la gestión de datos interactivos en streaming.

android.net.sip: Proporciona acceso a la funcionalidad de SIP, como realizar y recibir llamadas VoIP utilizando SIP.

android.net.wifi: Proporciona clases para manejar la funcionalidad Wi-Fi en el dispositivo.

android.net.wifi.p2p: Proporciona clases para crear P2P con conexiones Wi-Fi Direct.

android.nfc.tech: Estas clases proporcionan acceso a las características de una tecnología de la etiqueta, que varían según el tipo de etiqueta que se digitalizan. [29]

android.os.storage: Contiene clases para el servicio de almacenamiento del sistema, que gestiona los sistemas de archivos activos binarios conocidos como blobs binarios opacos.

android.preference: Proporciona clases que administran preferencias de la aplicación y ejecutar la interfaz de usuario de las preferencias.

android.security: Proporciona acceso a algunas instalaciones de los subsistemas de seguridad de Android.

android.service.textservice: Proporciona clases que permiten crear correctores ortográficos de una manera similar a la del sistema general de métodos de entrada.

android.support.v7.media: Contiene las API que controlan el enrutamiento de los canales y control multimedia del dispositivo actual a altavoces externos y dispositivos de destino.

android.telephony.cdma: Proporciona APIs para utilizar las funciones de telefonía CDMA específicos.

android.telephony.gsm: Proporciona APIs para utilizar las funciones de telefonía GSM-específicos, como los mensajes SMS de texto / datos / PDU. [29]

android.text.method: Proporciona clases que controlan o modifican de introducción del teclado.

android.text.style: Proporciona clases que se utilizan para ver o cambiar el estilo de un fragmento de texto en un objeto View.

android.app.Activity y android.app.ListActivity: Controlan el ciclo de vida de la aplicación.

android.graphics.drawable.Drawable: Sirve para definir la interfaz de usuario de la aplicación de bajo nivel.

android.widget.ArrayAdapter y android.widget.ListAdapter: Sirve para definir la interfaz de usuario de la aplicación de alto nivel.

android.graphics: Proporciona herramientas de gráficos de bajo nivel, tales como telas, filtros de color, puntos y rectángulos que permiten dibujar directamente en la pantalla.

android.graphics.drawable.shapes: Contiene clases para dibujar formas geométricas. [29]

ANEXO F

Tabla de tamaños de pantalla de las plataformas Android y JME

Teléfono	Plataforma	Tamaño de Pantalla	Ancho/Alto	PPI
Galaxy SIII	Android	4.8	720 x 1280	306
Galaxy SII	Android	4.3	480 x 800	219
Google Nexus 4 by LG	Android	4.7	768 x 1280	320
Galaxy Nexus	Android	4.6	720 x 1280	316
Galaxy S4	Android	5	1920 x 1080	441
Galaxy Note II	Android	5.5	720 x 1280	267
Galaxy S	Android	4.0	480 x 800	233
HTC One	Android	4.7	1080 x 1920	468
Droid Razr	Android	4.3	540 x 960	256
Droid	Android	3.7	480 x 854	265
Droid 3 & 4	Android	4.0	540 x 960	280
Google Nexus 10 by Samsung	Android	10.1	2560 x 1600	300
Samsung Galaxy Tab	Android	7.0	600 x 1024	171
Samsung Galaxy Tab 2	Android	7.7	600 x 1024	170
Samsung Galaxy Tab 10.1	Android	10.1	800 x 1280	149
Samsung Galaxy Tab2 10.1	Android	10.1	800 x 1280	149
Google Nexus 7 by Asus	Android	7.0	800 x 1280	216

Amazon Kindle Fire HD 7"	Android	7.0	800 x 1280	216
Amazon Kindle Fire HD 8.9"	Android	8.9	1200 x 1920	254
Amazon Kindle Fire 1st Gen	Android	7.0	600 x 1024	169
Nook Tablet	Android	7.0	600 x 800	167
Asus Transformer	Android	10.1	1200 x 1920	224
Nokia 5310 XpressMusic	JME	2	240 x 320	200
Nokia E63	JME	2.36	320 x 240	170
Nokia 5300	JME	2.1	240 x 320	190
Nokia Asha 500	JME	2.8	240 x 320	143
Nokia Asha 311	JME	3.0	240 x 400	155
Nokia Asha 503	JME	3.0	240 x 320	133
Nokia 515	JME	2.4	240 x 320	167
Nokia 107 Dual SIM	JME	1.8	128 x 160	114
Motorola EX225 MOTOKEY SOCIAL	JME	2.4	320 x 240	167
Motorola MOTOKEY MINI EX109	JME	2	176 x 144	114
Motorola EX226	JME	2.4	320 x 240	167
Motorola WX306	JME	2.0	176 x 220	282
Motorola EX119	JME	2.4	320 x 240	167
Motorola WX292	JME	1.8	128 x 160	114
Motorola GLEAM+	JME	2.8	240 x 400	167
Motorola MOTOKEY WiFi EX116	JME	2	176 x 220	141
Motorola EX115	JME	2.3	320 x 240	174
Motorola EX112	JME	2.3	320 x 240	174

Nokia 301	JME	2.4	240 x 320	167
Nokia Asha 310	JME	3.0	240 x 400	155
Nokia Asha 210	JME	2.4	320 x 240	167
Nokia Asha 309	JME	3.0	240 x 400	155

BIBLIOGRAFÍA

[1] Nieto, A. (2011) Xataka ANDROID [online] Disponible en http://www.xatakandroid.com/sistema-operativo/que-es-android (5-12-2013)

[2] López, M. (2013) UNO CERO [Online] Disponible en http://www.unocero.com/2013/09/23/la-historia-de-android/ (24-05-2013)

[3] González, R. y Garrido, E. (2013) PODER pda [Online] Disponible en www.poderpda.com/noticias/android-la-breve-historia-del-linux-de-google/ (24-05-2013)

[4] Cosmos (2013) Xataka ANDROID [online] Disponible en http://www.xatakandroid.com/tag/distribucion-de-versiones-android (6-12-2013)

[5] Deitel, P. Deitel, H. y Deitel, A. *"Android How to Program".* Primera Edición. Editorial Prentice Hall, 2012.

[6] Mohsin (2011) Mundial de Mohsin [Online] Disponible en http://worldofmohsin.blogspot.com/2011/02/understanding-android-software-stack.html (26-05-2013)

[7] Meier, R. *"Professional Android 4 Application Development".* Primera Edición. Editorial Wrox, 2012.

[8] Anónimo (2013) Versiones del lenguaje Java [Online] Disponible en http://www.cad.com.mx/versiones_del_lenguaje_java.htm (30-11-2013)

[9] Pérez, A. (2013) AJPD Soft [Online] Disponible en http://www.ajpdsoft.com/modules.php?name=News&file=print&sid=437 (25-05-2013)

[10] Rodríguez, A. (2013) apr [Online] Disponible en http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188 (25-05-2013)

[11] Gálvez, S. y Ortega, D. (2003) *JAVA A TOPE: J2ME (JAVA 2MICRO EDITION)* [Online] Disponible en http://www.lcc.uma.es/~galvez/ftp/libros/J2ME.pdf (06-12-2013)

[12] Prieto, M. (2003) CURSO DE J2ME [Online] Disponible en http://ait.upct.es/asignaturas/ct/practicas/practica2/Curso_J2ME.pdf (25-05-2013)

[13] Campo, C. y Martínez, N. (2013) SOFTWARE EN DISPOSITIVOS MÓVILES [Online] Disponible en http://www.it.uc3m.es/~celeste/docencia/ittt/swc0405/traspas/J2ME_parteI.pdf (25-05-2013)

[14] Sergio Costas Rodríguez (2013) MANUAL DE PROGRAMACION [Online] Disponible en http://www.rastersoft.com/OS2/CURSO/APIEXPL.HTM (25-05-2013)

[15] Muro, L. (2013) Monografías J2ME [Online] Disponible en http://www.monografias.com/trabajos55/licencias-de-software/licencias-de-software.shtml (27-06-2013)

[16] Cancela, L. y Ostos, S. (2013) Software de Comunicaciones [Online] Disponible en https://sites.google.com/site/swcuc3m/home/android/android_javame (22-09-2013)

[17] León, J. (2012) Androideity [Online] Disponible en

http://androideity.com/2012/07/16/5-lenguajes-para-programar-en-android (22-09-2013)

[18] Mr. Geek (2012) OTROGEEK [Online] Disponible en http://www.otrogeek.net/2012/10/emuladores-de-telefonos-moviles.html (22-09-2013)

[19] Anónimo (2013) Developers [online] Disponible en http://developer.android.com/reference/android/nfc/package-summary.html (04-12-2013)

[20] Cosmos (2013) Conectividad Android KitKat [online] Disponible en http://www.xatakandroid.com/sistema-operativo/android-4-4-kitkat-anunciado-oficialmente (04-12-2013)

[21] Anónimo (2013) Developers [Online] Disponible en http://developer.android.com/guide/practices/screens_support.html (30-11-2013)

[22] alejosoft (2008) todoexpertos [Online] Disponible en http://www.todoexpertos.com/categorias/tecnologia-e-internet/programacion/java/respuestas/1811667/ajustar-imagen-a-la-pantalla-del-movil-con-j2me (04-12-2013)

[23] Anónimo (2013) smartGSM [Online] Disponible en http://www.smart-gsm.com/moviles/samsung-galaxy-mini-s5570 (25-05-2013)

[24] Anónimo (2013) smartGSM [Online] Disponible en http://www.smart-gsm.com/moviles/nokia-asha-311 (25-05-2013)

[25] Anónimo (2013) NetBeans [Online] Disponible en https://netbeans.org/community/releases/73/install.html (25-05-2013)

[26] Oracle (2013) Java ME Embedded Guía de introducción para la plataforma Windows [Online] Disponible en http://docs.oracle.com/javame/config/cldc/rel/3.3/win/gs/html/getstart_win32/setup_nbenv.htm (30-11-2013)

[27] Catalan, M. (2013) Geeky Theory Android [Online] Disponible en http://geekytheory.com/tutorial-android-1-instalacion-sdk-y-eclipse/ (30-11-2013)

[28] Desarrolladores de Nokia (2013) Nokia Developer [Online] Disponible en http://developer.nokia.com/Resources/Library/Java/#!supported-apis.html (28-11-2013)

[29] Anónimo (2013) Developers [Online] Disponible en http://developer.android.com/reference/packages.html (28-11-2013)