Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

ANÁLISIS DE PLATAFORMAS POPULARES DE DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

Luis Miguel Pérez Vásquez
Asesorado por el Ing. Mario Joaquín Trujillo Ramírez.

Guatemala, noviembre de 2011

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ANÁLISIS DE PLATAFORMAS POPULARES DE DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

LUIS MIGUEL PÉREZ VÁSQUEZ
ASESOR ING. MARIO JOAQUÍN TRUJILLO RAMÍREZ.

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, NOVIEMBRE 2011

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Juan Álvaro Díaz Ardavín
EXAMINADOR	Ing. César Fernández Cáceres
EXAMINADOR	Ing. Pedro Pablo Hernández Ramírez
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

ANÁLISIS DE PLATAFORMAS POPULARES DE DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha agosto de 2009.

Luis Miguel Pérez Vásquez

AGRADECIMIENTOS A:

Dios	Por ser mi fuente de sabiduría y fuerza.
Mis padres	Por su apoyo. Especialmente a mi madre.
Mis hermanos	Muchas gracias.
Esposa e hijo	Por su amor.
Ing. Mario Joaquín Trujillo Ramírez	Por su asesoramiento, revisión y corrección en este trabajo.
La Facultad de Ingeniería de Universidad de San Carlos de Guatemala	Por forjar mi conocimiento.
Todos mis familiares y amigos	Me brindaron su apoyo. Principalmente a la familia Pérez Pérez.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Auge tecnológico Expansión de la tecnología, mayor impulso de cierta tecnología.

Red analógica Señal cuya magnitud se representa mediante variables continúas.

Interfaz En ciencias computacionales es el enlace entre un programa y el usuario.

Persistencia Forma de mantener siempre disponibles datos en un sistema.

IDE De las siglas en inglés: *Integrated Development Environment*, se traduce en: entorno de desarrollo integrado, el que contiene herramientas para facilitar el desarrollo de aplicaciones.

Plataforma Base sobre la cual es posible desarrollar aplicaciones para dispositivos con recursos limitados.

Toolkit Colección de herramientas integradas que permiten automatizar un conjunto de tareas de alguna de las fases del ciclo de vida del sistema.

Dispositivo	Elemento de *hardware* conectado a un circuito que es capaz de recibir y ejecutar instrucciones de computadora.
Telecomunicación	Toda trasmisión, emisión o recepción de signos, señales, circuitos, imágenes, sonidos o informaciones de cualquier naturaleza por hilo.
Framework	Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.
Arquitectura	Es la esencia en la construcción y permite definir la estructura que tendrá una aplicación o proyecto de *software*.
Market	Es un espacio en la red, que permite reunir a vendedores y clientes permitiendo comercializar productos utilizando internet como medio.
Núcleo	Se refiere al centro de una aplicación, *framework* o sistema operativo, en la que gira la estructura que se ha definido.
App Store	Tienda de aplicaciones virtual de *Apple* para aplicaciones iOS.
Android Market	Tienda de aplicaciones virtual del sistema operativo *Android*.

RESUMEN

La presente investigación, tiene como objetivo ayudar a los desarrolladores independientes empresas o personas aventureras, en la elección de la plataforma de desarrollo para dispositivos móviles. En la primera parte se clasifican las diferentes representaciones de dispositivos móviles vigentes, con el fin de ampliar los enfoques que pueda presentar la solución a desarrollar.

La segunda parte del trabajo permite tener un panorama de los conceptos generales de análisis de sistemas con enfoque para dispositivos móviles, los lineamientos generales de análisis y diseño; además conocer la variedad de herramientas que se pueden utilizar para la fase de desarrollo de una solución.

La tercera parte se enfoca en la plataforma de desarrollo Java, siendo pionera en el medio y de la presencia que esta tiene en los objetos móviles cotidianos.

La cuarta parte se centra en la plataforma *Android*, que desde su presentación ha sido la plataforma más aceptada por los usuarios de dispositivos móviles.

La quinta parte profundiza sobre la plataforma iOS que sin duda es la plataforma que revolucionó el paradigma de sistemas móviles elegantes y funcionales.

La sexta parte presenta la plataforma Blackberry OS, la cual forma parte importante en el mercado de dispositivos móviles.

Se finaliza en la séptima parte donde se muestran los análisis de las plataformas analizadas en los capítulos anteriores; permite tener un panorama amplio de plataformas para poder elegir la correcta para una idea-solución.

OBJETIVOS

General

Analizar las plataformas de desarrollo vigentes utilizadas diariamente.

Específicos

1. Conocer la documentación proporcionada por las empresas que gestionan las plataformas para desarrollar aplicaciones en los dispositivos móviles a estudiar.

2. Aplicar un análisis de perfil y monetario, de las aplicaciones desarrolladas en las plataformas estudiadas.

3. Describir los principales tipos de desarrollo que se pueden realizar para dispositivos móviles; así como, la oportunidad de los desarrolladores independientes y empresas de aplicaciones para dispositivos móviles en el media actual.

4. Detallar los aspectos generales importantes de arquitectura de las plataformas para poder conocer las ventajas y desventajas de estas plataformas; así como explotar el soporte que puedan proporcionar las API’s, y entornos de desarrollo de aplicaciones.

5. Presentar las especificaciones básicas de recursos para desarrollar en las plataformas de desarrollo para móviles populares. Realizar un esquema comparativo general entre las plataformas populares.

INTRODUCCIÓN

"En la actualidad, el ser humano se moviliza rápido. Se vive en un mundo en el que el estrés se alza como claro dominado, un mundo en el que la pérdida de un segundo en el tiempo puede ser vital en muchos sentidos. Un mundo en el que el hombre pretende llegar a todas partes por medio de la tecnología pero, ante todo, se trata un mundo en el que el ansia de satisfacer todos estos requerimientos nos ha llevado a reducir al hombre y a la máquina en un mismo ente, como dos partes de un todo…"[1]

En la actualidad el uso de dispositivos móviles es común en la vida cotidiana, la mayor parte de personas utiliza un dispositivo electrónico, estos dispositivos tienen la capacidad de ejecutar aplicaciones útiles a las necesidades, lo que se hace indispensable para las actividades diarias.

En principio los dispositivos se diferenciaban por la potencia de su *hardware*, sin darle importancia a la plataforma que administraba dicho *hardware*, regularmente contaban con una agenda, calendario, marcados rápidos. Sin embargo el usuario percibía esa diferencia sólo en el costo y tamaño del dispositivo, es aquí donde la importancia de la plataforma fue notable, los dispositivos incrustaron un sistema operativo el cual en conjunto con el *hardware* disponible permitiría ejecutar nuevos tipos de aplicaciones que hasta el momento parecían estar fuera del alcance.

[1] BLANCO LÓPEZ, Jorge. Inteligencia artificial. http://www.aepia.org/. 11-10-2010.

Este cambio hizo posible diferenciar empresas que fabrican *hardware* y *software*, permitió a empresas y desarrolladores crear aplicaciones de todo tipo explotando las ventajas y disminuyendo las desventajas de las diferentes plataformas en la variedad de dispositivos para ofrecer un mejor servicio. Esta descripción de las distintas plataformas vigentes en el mercado tecnológico son los aspectos que abarca esta investigación.

1. DISPOSITIVOS MÓVILES

Regularmente se denomina dispositivo móvil a cualquier segmento de telefonía móvil, con ciertas capacidades, sin embargo, un dispositivo móvil puede ser cualquier objeto electrónico que se ajuste a las siguientes características básicas:

- Reducido tamaño para poder ser transportado.
- Capacidad de procesamiento y almacenamiento de datos.
- Incorpora elementos de entrada y salida básicos (por ejemplo: teclado, pantalla, cámara).

Un dispositivo electrónico móvil, se define a un grupo heterogéneo que cumpla, supere o diversifique las características mencionadas con anterioridad, además se debe destacar la plataforma de aplicaciones que proporciona la administración de recursos del dispositivo.

1.1. Clasificación de los dispositivos móviles

La clasificación se presenta difícil de definir, dado que está sujeta a diferentes valores y no hay un estándar de clasificación de una determinada línea.

El aumento de dispositivos fue notable a partir de la década de los años 90, en donde empiezan a comercializar dispositivos con prestaciones similares, ante esta situación los fabricantes agregaron diferentes características en sus dispositivos, lo que dificulta clasificar los dispositivos móviles.

Clasificaremos a los dispositivos en base a la función principal para lo que fueron creados, pero cabe mencionar que esta clasificación no es estándar, la pertenencia de una categoría específica no implica que el dispositivo no pueda ofrecer características propias de otro segmento.

Los dispositivos móviles pueden clasificarse de la siguiente manera:

1.1.1. Dispositivo de comunicación

El objetivo principal de este dispositivo es ofrecer un servicio de comunicación. Estos servicios pueden ser: llamadas de voz, mensajes de texto, mensajes multimedia, correo electrónico, acceso WAP, etc. Como terminales de ejemplo se pueden mencionar a los siguientes teléfonos inteligentes: iPhone, Samsung Galaxy S, Blackberry, etc.).

1.1.2. Dispositivo de computación

Son los dispositivos que ofrecen mayores capacidades de procesamiento de información, y ofrecen periféricos más sofisticados similares a un ordenador de sobremesa. Dentro de este grupo podemos mencionar *laptop*, *notebook*, PDA's ya que pueden ofrecer capacidades grandes de procesamiento.

1.1.3. Reproductor multimedia

Este dispositivo ha sido específicamente diseñado para la reproducción de formatos digitales, sean estos de audio, video o imágenes. Los más frecuentes son reproductores MP3, iPod.

1.1.4. Capturador multimedia

Este tipo de dispositivos facilitan la grabación de información digital en formatos específicos y optimizados para estas funciones. Dentro de este segmento podemos mencionar las cámaras digitales, cámaras de video digital.

El trabajo por parte del proveedor debe estar limitado a tareas específicas bien definidas y calendarizadas de mantenimiento, al final la nube debe auto-administrarse con la ayuda de a sus usuarios.

1.1.5. Consola portátil

Este dispositivo se define, porque ejecuta programas potentes en gráficos y multimedia, se expandieron de tal forma que junto con los teléfonos marcaron una época. Actualmente son uno de los dispositivos más rentables para las empresas, lo que los lleva a formar una guerra comercial para expandirse en este sector. Podemos mencionar a Nintendo y Sony.

1.1.6. Dispositivos híbridos

En esta categoría podría clasificarse muchas características de los dispositivos anteriores, tienen gran potencia de procesamiento, pueden adaptarse periféricos y es utilizado en ambientes personales de trabajo, este tipo de dispositivos se presentan como la tecnología que revolucionará la forma de comunicarnos y organizar la vida. En esta categoría se puede citar a las *Tablets* (iPad, Samsung *Galaxy* Tab, Motorola Xoom).

1.2. Teléfono inteligente (*Smartphone*)

Es un término en inglés cuya traducción al castellano es “Teléfono Inteligente”, y se refiera a la forma en que la tecnología móvil ha dado un avance notable, mejorando sus características y prestaciones que permiten comparaciones con equipos de sobremesa. Dentro de las principales características está la gran capacidad de procesamiento y almacenamiento de datos, redes de telecomunicaciones, pantalla táctil, GPS, periféricos potentes. Cuenta con varias aplicaciones que explotan los recursos de *hardware*, estas aplicaciones pueden ser: navegadores web, clientes de correo, acceso a redes sociales, aplicaciones ofimáticas, reproductores multimedia, así como tienda de aplicaciones.

Una de sus principales desventajas con respecto a los equipos de sobremesa, es su reducida pantalla, así como el bajo rendimiento de su batería cuando se someten a uso constante, lo que obligan a tener presente al desarrollar aplicaciones de *software* para estos dispositivos. Estos dispositivos hacen uso de las redes de telecomunicaciones para poder transmitir datos, descarga de aplicaciones, descargas multimedia y las funciones básicas del teléfono móvil como llamar y mensajes de texto.

Para poder administrar los recursos de *hardware*, las aplicaciones y otras configuraciones es necesario aclarar varios conceptos que utilizan los dispositivos móviles, los cuales se definen a continuación:

1.3. Sistema operativo

Conjunto de programas destinados a permitir la administración de recursos del dispositivo. Su finalidad es gestionar el *hardware* del dispositivo

desde los niveles más básicos, permitiendo también la interacción con el usuario.

1.4. *Middleware*

Es un tipo de *software* que se sitúa entre el sistema operativo y las funciones de red del dispositivo en sí, proporcionando una interfaz de programación de aplicaciones (API), que facilita la programación y el manejo de aplicaciones distribuidas.

.

2. DESARROLLO Y OPORTUNIDAD DEL DESARROLLADOR PARA MÓVILES

2.1. Telecomunicaciones, *hardware*, *software*

El término telecomunicación (del prefijo griego tele, “distancia” o “lejos”, “comunicación a distancia”) es toda transmisión, emisión o recepción de signos, señales, datos, imágenes, voz, sonidos o información de cualquier naturaleza que se efectúa a través de cables, radioelectricidad, medios ópticos físicos e inalámbricos.

Las telecomunicaciones se utilizan para hacer negocios, comercio y un sinfín de actividades que requieren estar conectados de forma inmediata. Esta comunicación se vale de dispositivos inalámbricos, pues son fáciles de transportarlos y tienen un costo relativamente bajo.

Las empresas de telecomunicaciones se enfocan principalmente al mejoramiento de infraestructura y mantenimiento de éstas, así como mejorar la rapidez de transmisión en las comunicaciones en general.

Las empresas que fabrican *hardware* se enfocan en mejorar el desempeño en las capacidades de procesamiento y almacenamiento, periféricos y mejora en el desempeño de los recursos en general.

Las empresas que proveen las plataformas de aplicaciones, son las que permiten a las entidades desarrollar aplicaciones que luego el sistema base convertirá a aplicaciones ejecutables para el *hardware* que esta soporta.

Con la llegada de los *Smartphone* se ha superado la barrera impuesta en los años anteriores en donde únicamente un sector cerrado tenía licencia para desarrollar aplicaciones, aquí aparece la oportunidad a desarrolladores independientes y empresas de crearse un espacio en este segmento de la industria de aplicaciones móviles.

2.2. Aparición de desarrolladores

Antes de la aparición de los teléfonos inteligentes las empresas de telecomunicaciones crearon servicios que permitieron mejorar las comunicaciones con los dispositivos móviles, pero realmente con esta revolución llegaron de dispositivos como el iPhone de *Apple*, dando prioridad al desarrollo y diseño de las aplicaciones de *software*, manejando con buen desempeño las prestaciones de *hardware* y ofrecer una mejor sensación al usuario y no limitar al dispositivo al valor puramente del *hardware* y funciones básicas como se venía haciendo anteriormente (realizar llamadas, mensajes de texto, agencia, contactos).

La dinamización del sector supone una gran exigencia tanto a operadores como a fabricantes, así como una inversión a todas luces enorme, por lo que las empresas creadoras de las plataformas de aplicaciones, han optado mayoritariamente por dejar a programadores externos el desarrollo de aplicaciones para los usuarios.

Así, las empresas se encargan de posibilitar que los desarrolladores independientes dispongan de entornos de programación, tutoriales y tiendas online, donde distribuir o comercializar sus aplicaciones. De esta forma, los desarrolladores disponen de herramientas y sistemas para reducir el proceso de investigación, desarrollo y pruebas en estos dispositivos.

Esta carrera por incentivar a los desarrolladores se realiza a todos los niveles. Las empresas creadoras de sistemas operativos móviles, como: *Apple* con el iOS, Google con *Android*, RIM con BlackBerry OS, entre los que más destacan, confían en que la penetración de sus sistemas operativos entre los dispositivos móviles sea un gancho suficiente para que los desarrolladores decidan crear nuevas aplicaciones específicas para ellos. Así cuantas más aplicaciones existan para un determinado sistema operativo, más demanda habrá de él entre fabricantes y operadores.

2.3. Plataforma de desarrollo

En informática, una plataforma de desarrollo es el entorno de *software* común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo; sin embargo, también es posible encontrarla ligada a una familia de lenguajes de programación o a una interfaz de programación de aplicaciones (API por sus siglas en inglés).

Se está familiarizado con sistemas operativos de escritorio y únicamente se instala las dependencias de la plataforma de desarrollo y luego se utiliza los entornos integrados de desarrollo, el cual se define a continuación.

2.4. Entorno de desarrollo integrado (IDE)

Es un programa informático compuesto por un conjunto de herramientas de programación. En este caso los entornos de desarrollo se dedican exclusivamente a un solo lenguaje de programación, aunque existen varios entornos que se pueden utilizar para varios lenguajes que se definen en la instalación del mismo.

Los IDE's (por sus siglas en inglés) proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, C#, Delphi, *Visual Basic*, etc.

2.5. Tienda virtual de aplicaciones

Servicio que permite realizar comercio entre vendedores y clientes, que tiene como principal forma de hacer las transacciones por medio de internet. Los vendedores ponen a disposición productos y servicios, los cuales describen el propósito y que son categorizados para ser hallados fácilmente por los usuarios. Los vendedores pueden ser empresas, o desarrolladores independientes, que realizan aplicaciones que pueden ser gratis o de pago.

El desarrollador debe registrarse en la tienda virtual para poder publicar aplicaciones. Cuando desea publicar una aplicación, esta manda a la tienda virtual la aplicación para que revise el contenido de la aplicación y le ofrece los términos del uso del *software*, este regularmente especifica la cantidad que la tienda añadirá al costo de la aplicación por concepto de publicación.

Cuando los clientes compren las aplicaciones, la tienda virtual le acreditará la cantidad de dinero que obtuvieron en un cierto periodo de tiempo especificado entre ambos, a una cuenta registrada previamente por el desarrollador.

2.6. Proceso de desarrollo de aplicaciones

Los procesos y métodos de desarrollo de aplicaciones dependen de cada entidad, sin embargo mencionaremos las etapas clásicas de desarrollo y que son bastante aceptables en el desarrollo de aplicaciones móviles:

2.6.1. Análisis

Extraer los requisitos de un producto de *software* es la primera etapa para crearlo. Se requiere de cierta habilidad y experiencia en la ingeniería de *software* para reconocer requisitos incompletos, ambiguos o contradictorios. En las aplicaciones móviles se debe comprender el sector de mercado que se desea competir, así como la forma de licenciamiento del producto.

2.6.2. Diseño

Se refiere a la representación del mapa del proyecto, la integración de infraestructura, desarrollo de aplicaciones, bases de datos y herramientas gerenciales, que requieren de capacidad y liderazgo, además de proyectar a futuro, solucionando los problemas de hoy. Esta parte del proceso de creación de *software* involucra notaciones de diagramas siendo la más aceptada el lenguaje unificado de modelado (UML).

2.6.3. Programación e implementación

Reducir un diseño a código puede ser la parte obvia del trabajo de realizar un producto de *software*, pero no necesariamente es la que demanda mayor trabajo y ni la más complicada. La complejidad y la duración de la etapa está relacionada al o a los lenguajes de programación utilizados, así como al diseño previamente realizado. La implementación consiste en proporcionar a los usuarios el producto funcionando para que puedan verificar que el producto hace lo que tiene que hacer. En el caso de las plataformas de aplicaciones para dispositivos móviles proporcionar un entorno de desarrollo completo para codificar el proyecto para dicha plataforma.

2.6.4. Pruebas

Consiste en comprobar que el *software* realice correctamente las tareas indicadas en la especificación del problema. Regularmente las plataformas de aplicaciones ofrecen herramientas de simulación que permiten depurar y corregir las aplicaciones creadas.

2.7. Pasos generales para construir aplicaciones

Cuando se desee desarrollar una aplicación para dispositivos móviles debemos tener en cuenta los requerimientos y saber enfocar las herramientas a utilizar, ya que se puede desarrollar una aplicación web o una aplicación nativa, obviamente con resultados distintos no solo por la aplicación en sí, sino por tiempo y esfuerzo.

2.7.1. ¿Qué es una aplicación web?

Se trata de aplicaciones normales desarrolladas en Lenguaje de Marcado de Hipertexto (HTML por sus siglas en inglés), en el cual se define la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. Además a una estructura de marcado se puede dar presentación al formato por medio de hojas de estilo en cascada (CSS por sus siglas en inglés) que en conjunto se construyen del lado del servidor y son vistas en el navegador de los dispositivos móviles. Las aplicaciones web para dispositivos móviles pueden ser desarrolladas en cualquier plataforma de desarrollo sea estándar, empresarial o móvil, por ejemplo: lenguaje de Programación Interpretado (PHP por sus siglas en inglés), Java, Visual Studio, otros.

2.7.2. ¿Qué es una aplicación nativa?

En la mayoría de dispositivos móviles, los usuarios están más acostumbrados a este tipo de aplicaciones, y son todas aquellas que fueron descargadas por medio de una tienda virtual o cualquier otro repositorio y que fueron instaladas directamente en el dispositivo, no es requisito la utilización de navegadores dado que no dependen de esto. Estas aplicaciones son desarrolladas con el SDK que proporcionar las plataformas de desarrollo de aplicaciones.

2.7.3. Diferencias y ventajas

- Al desarrollar aplicaciones web, no se depende delas tiendas virtuales para su distribución y tampoco pasan por revisiones de comprobación.
- Al crear una aplicación nativa, se tiene acceso a la API de la plataforma y se pueden utilizar los controles que provee la misma.
- Una aplicación nativa requiere conocimientos específicos de la plataforma para la cual se trabaja, mientras que para una aplicación web no es necesario.
- Con la aplicación web, no se necesita registrarse como desarrollador autorizado, sino es de libre acceso, mientras que con las aplicaciones nativas es necesario el registro además de tener la opción de cobrar por descarga.

Una vez se tenga claro el tipo de aplicación a desarrollar se deben tener en cuenta los siguientes pasos:

2.7.4. La idea

El primer paso y la tarea más complicada de desarrollar aplicaciones es generar una idea, esta nos permite visualizar el contenido y el grupo de usuarios a quien va dirigida la aplicación. Ayuda a crear un objetivo pues es común tener una idea no muy clara y empezar a desarrollar y al poco tiempo no sabemos que es lo que estamos haciendo, ¿alguna vez les ha pasado?, para que esto no suceda debemos de plasmarla y definirla de forma detallada, de manera que permita agregar detalles en el proceso de construcción.

2.7.5. Investigación

El segundo paso es investigar aplicaciones relacionadas con la aplicación, para ello podemos utilizar las tiendas virtuales que es donde se encuentran disponibles las aplicaciones existentes, por medio de un filtro podemos obtener la información que necesitemos. Una vez que obtenemos esta información podemos analizar quienes han tenido ideas similares, es muy probable que te lleves una sorpresa, puede resultar que no uno sino varios desarrolladores han hecho exactamente lo mismo que se pensó y con más funcionalidades que no se habían contemplado.

Muchas veces las aplicaciones con éxito, no son las más innovadoras, sino las que logran cumplir el objetivo para el cual son diseñadas, las que agregan calidad en el producto y superan las expectativas del cliente que las utiliza.

2.7.6. Análisis y diseño

El tercer paso es diseñar y definir las funcionalidades de la aplicación, este es uno de los pasos más importantes puesto que de un mal diseño puede llevar al fracaso el producto. Para realizar el diseño se debe delimitar al sector al cual va dirigido la aplicación, y no perder el enfoque principal de la aplicación. Al diseñar aplicaciones, se debe investigar las tecnologías con la cual se va a desarrollar y que se ajusta a las necesidades.

2.7.7. Desarrollo

El cuarto paso es desarrollar la aplicación, básicamente es codificar la parte del diseño, agregando aspectos visuales y configuraciones que hagan que la aplicación se integre de forma correcta y optima a la plataforma para la cual se trabajó.

2.7.8. Publicación

El último paso una vez ya haya sido probado por el equipo, es publicarla en la(s) tienda(s) virtual(es) que tenga a disposición la plataforma para los clientes determinen la calidad del producto.

2.8. ¿En qué plataforma debo de realizar mi aplicación?

Cada plataforma contiene muchas ventajas y desventajas, el detalle de algunas plataformas se detallará en los siguientes capítulos; sin embargo se sugiere que se realice la aplicación en una de las plataformas populares para que se pueda medir el grado de aceptación de dicha aplicación, si esta posee una buena aceptación, se analiza si es factible desarrollarla para otras

plataformas. Los detalles de las plataformas más populares se prestan en los siguientes capítulos.

3. PLATAFORMA DE DESARROLLO JAVA

Esta plataforma se hizo muy popular en los principios de los 90's, su objetivo era ser implementado en cada dispositivos móvil que tuviera acceso a la red, recursos limitados, proporcionando un conjunto de interfaces para poder administrar mejor las características del *hardware*.

El nombre que le denominó Java a este importante proyecto fue: Java 2 Micro *Edition* (J2ME). En principio se conceptualizó al igual que el lenguaje estándar de Java, con un conjunto de interfaces de programación de aplicaciones (API) y una máquina virtual, el cual proporcionó soluciones. Seguido a la base obtenida en la primera versión, se impulsó su desarrollo con la versión de configuración J2ME CLDC 1.0, expandiendo aún más las librerías de la API, aunque todavía contaba con deficiencias. Las mejoras en su desarrollo permitió en Julio del año 2000 la primera implementación con perfil, llamado *Mobile Information Device Profile* (MIDP), su enfoque principal era para teléfonos móviles y paginadores.

En esta última entrega del producto J2ME fue considera aceptado en la comunidad de desarrolladores para dispositivos móviles, teniendo un gran crecimiento e incluso es la base para otras plataformas de ejecución de aplicaciones.

Las principales áreas de incursión de J2ME se centra en la creación de juegos para teléfonos con recursos limitados, sin embargo esto no limita a los desarrolladores a competir con otras plataformas ya implantadas para móviles. Java ha sido una de las plataformas que más han sabido explotar los

desarrolladores al construir juegos y aplicaciones de gama baja, permitiendo encontrar buenas sensaciones entre los usuarios.

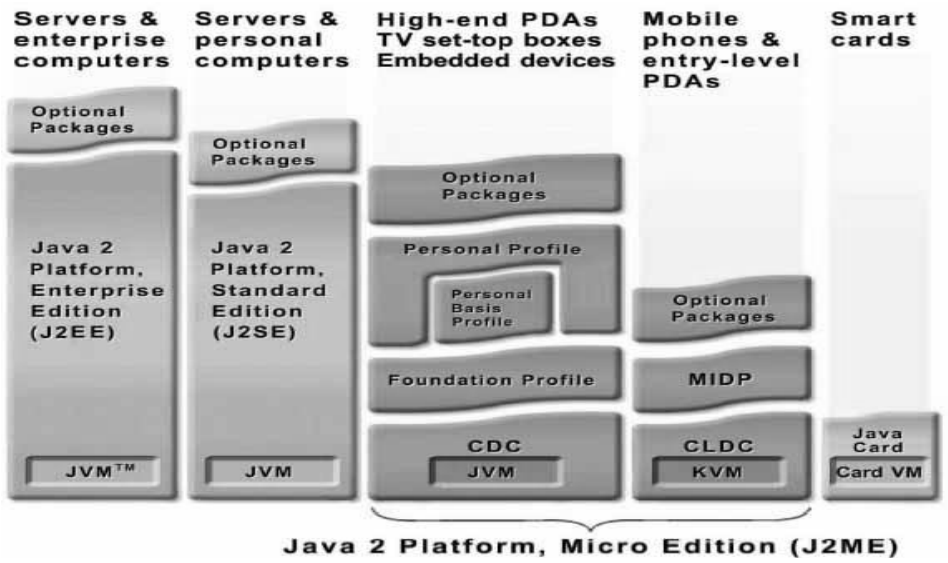
3.1. Diagrama de arquitectura Java 2 *Micro Edition*

J2ME está formada por un conjunto de interfaces de programación estándar que permiten que las aplicaciones desarrolladas se beneficien de las características multiplataforma y que permiten la distribución de aplicación a varios dispositivos.

La arquitectura se puede dividir en dos grandes bloques que dependen del tipo de dispositivo y las características de estos que dependen de la familia y recursos que pueden proporcionar a la plataforma.

El siguiente diagrama nos permite visualizar la forma en que se encuentra la arquitectura de J2ME con respecto a las otras plataformas de Oracle.

Figura 1. **Diagrama de Arquitectura J2ME**



Fuente: www.java.com (06-10-2011).

Para poder tener un entorno de ejecución Java para J2ME que cumpla los requisitos de un rango amplio de dispositivos y mercado objetivo es necesario que se componga de:

- Kilo Virtual *Machine* (KVM)
- Configuraciones
- Perfiles
- Paquetes opcionales

Cada combinación de estos elementos se optimiza para la memoria, potencia de proceso y capacidades de E/S de una categoría de dispositivos. J2ME está constituido por los siguientes componentes:

3.1.1. Kilo Virtual *Machine* (KVM)

Una máquina virtual, es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial el cual es generado por el compilador del lenguaje que lo utilice. La *Kilo Virtual Machine* corresponde a la máquina virtual más pequeña desarrollada por *Sun Microsystem*. Para una configuración estándar, requiere entre 50 y 80 Kbytes de memoria, y dado su bajo requerimiento de memoria dinámica para funcionar, su requerimiento final no debería superar los 180 Kbytes.

3.1.2. Configuraciones

La configuración define las mínimas librerías Java y las capacidades de la máquina virtual. Actualmente ya están disponibles la especificación CLDC 1.1, para dispositivos con un total de memoria entre 128 Kb y 512 KB, conectividad a la red y alimentación limitada, y la especificación CDC. Proporciona la funcionalidad básica para un conjunto de dispositivos que comparten características similares, tales como gestión de memoria o conectividad de la red. En la actualidad existen dos configuraciones J2MEy se detallan a continuación:

3.1.2.1. Connected Limited Device Configuration (CLDC).

Esta configuración está diseñada para dispositivos con conexión de red intermitente, procesadores lentos y memoria limitada como son teléfonos móviles, asistentes personales (PDA’s). Está orientado a dispositivos que cumplan las siguientes características:

- Procesador: 16 bit / 16 Mhz o más.
- Memoria: 160 Kb – 512 Kb de memoria total disponible para la plataforma Java.
- Alimentación: alimentación limitada, a menudo basada en batería.
- Trabajo de red: conectividad a algún tipo de red, con ancho de banda limitado habitualmente.

La especificación CLDC se ha desarrollado dentro de Java *Community Process*, en conjunto con 500 patrocinadores que representan a las industrias de fabricantes de dispositivos inalámbricos, proveedores de servicios y terminales de punto de venta (POS). También proporciona la implementación del CLDC HotSpot, disponible para usos comerciales bajo licencia.

Esta máquina virtual está orientada a la nueva generación de dispositivos con una cantidad de memoria disponible mayor. La CLDC RI es adecuada para dispositivos que cumplan las siguientes características:

- Procesador: 32 bits.
- Memoria: mayor a 1 MB de memoria total disponible para la plataforma.
- Alimentación: alimentación limitada, a menudo basada en batería.
- Trabajo en red: conectividad a algún tipo de red, con ancho de banda limitado habitualmente.

3.1.2.2. Versiones de CLDC

- CLDC 1.1 (JSR 139): CLDC 1.1 es una versión de la especificación CLDC 1.0 e incluye nuevas características como son punto flotante o soporte a referencias débil, junto con otras mejoras. CLDC 1.1 es compatible con

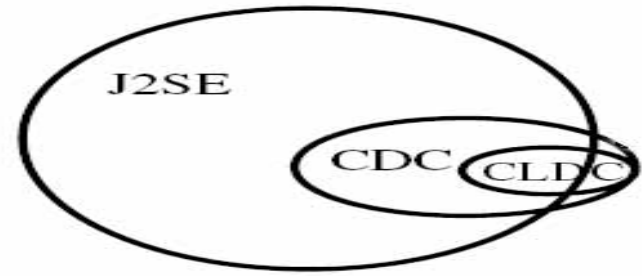
versiones anteriores y sigue soportando dispositivos pequeños o con recursos limitados.

- CLDC 1.3 (JS30).
- CLDC HotSpot Implementación: es una máquina virtual muy optimizada que presenta una diferencia de rendimiento muy alta frente a la KVM. Incluye características que soportan una ejecución más rápida de aplicaciones y una gestión de recursos más eficientes, manteniendo los requisitos en cuanto a plataforma de ejecución.

3.1.2.3. Características de las JVM con CLDC

La máquina virtual para CLDC soporta un subconjunto de funcionalidad de J2SE, además de incorporar una funcionalidad propia tal y como detalla el siguiente diagrama:

Figura 2**.** **Diagrama de Subconjunto CLDC vs J2SE**



Fuente: www.java.com (06-10-2011).

3.1.2.4. Cambios entre máquina virtual J2ME y J2SE

- Tipos de datos: J2ME no incluye los tipos numéricos decimales flotantes y dobles, ya que la mayoría de los dispositivos CLDC no tienen unidad de coma flotante debido fundamentalmente a que es una operación costosa.
- Pre-verificación: la verificación del código en J2ME se hace fuera del dispositivo, con objeto de reducir la carga de la máquina.
- Inclusión de ficheros: “descriptor” y “manifiesto” al empaquetar ficheros J2ME, conteniendo información sobre las aplicaciones que incluyen.
- Nueva Biblioteca gráfica adaptada a los dispositivos con memorias de poco tamaño y pantallas también pequeñas.
- No existe un método principal como entrada para la ejecución de la función. Este se sustituye por el método “*startapp*”.
- La recolección de basura se hace de manera manual y no automática como en el J2EE.
- No soporta la finalización de instancias de clases.
- El soporte de gestión de errores es limitado, debido a las exigencias que impone en los dispositivos a nivel de memoria.
- Por motivos de seguridad se eliminan las siguientes características:
    - Java *Native Interface* (JNI).
    - Cargadores de clase definidos por el usuario.
    - *Reflection* (Reflexión).

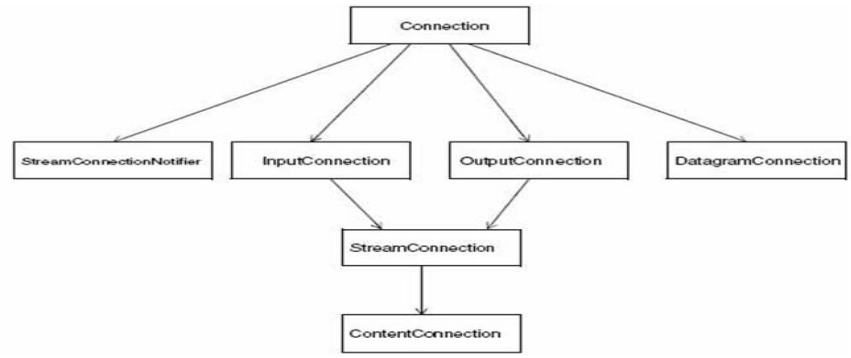
3.1.2.5. Clases especificas a CLCD

Se definen seis interfaces básicas de conectividad, las que enumeramos a continuación:

- Un dispositivo básico de entrada serie.
- Un dispositivo básico de salida serie.
- Un dispositivo de comunicaciones orientadas a datagrama.
- Un dispositivo de comunicación orientada a circuito (TCP).
- Un mecanismo de notificación para informar a un servidor de conexiones cliente servidor.
- Una conexión básica a un servidor Web.

Las clases específicas de CLDC se ilustran en la siguiente figura:

Figura 3. **Jerarquía de Interfaces especificas a CLDC**



Fuente: www.java.com (06-10-2011).

3.1.2.6. *Conected Device Configuration* (CDC)

Esta configuración se ha desarrollado para dispositivos con 2MB o más de memoria disponible para la plataforma, incluyendo RAM y memoria flash o

ROM. Estos dispositivos requieren todas las características y funcionalidades de la JVM e incluyen teléfonos móviles de última generación, asistentes personales, terminales de punto de venta (TPVs), permitiendo tantas redes con conexión intermitente o de alta velocidad, sin cablear. La plataforma CDC para J2ME incorpora:

- Una máquina virtual de Java completa compatible con J2SE 1.3.1
- Bibliotecas de clase e Interfaces de programación mínimas para que el sistema funcione.
- Soporte completo para carga de clases, gestión de subprocesos (*threads*) y mecanismos de seguridad.

3.1.2.7. Implementación CDC *HotSpot*

La máquina virtual fue optimizada para uso en dispositivos personales móviles. Esta implementación se adapta a las mismas especificaciones de la máquina virtual que requieren las máquinas virtuales Java para J2SE pero tiene unas características específicas tales como rendimiento, soporte de dispositivo, gestión de recursos y características de estabilidad diseñadas para ajustar a las necesidades de los productos de consumo y dispositivos embebidos.

3.1.2.8. Modelos de aplicación

Con CDC se pueden desarrollar y ejecutar distintos tipos de aplicaciones, permite gestionar distintos escenarios y soportar necesidades de usuario, entre los principales se encuentran los modelos de aplicación, aplicaciones independientes, applets, Xlets.

3.1.3. Perfiles

El primer perfil desarrollado se denomina MIDP, que en conjunto está diseñado para operar con el CDLC y permitir la ejecución de aplicaciones Java en dispositivos móviles. En el desarrollo de este perfil participan empresas tales como: Ericsson, Fujitsu, Motorola, Nokia, NTT DoCoMo, *Palm Computer*, RIM, Sony, Siemens y Oracle, entre otros.

En la actualidad existen los siguientes perfiles asociados a J2ME:

- *Mobile Information Device Profile* (MIDP).
- *Mobile Information Device Profile* (MIDP).
- *Personal Profile.*
- *Personal Basic Profile*.

3.1.3.1. *Mobile Information Device Profile* (MIDP)

Ofrece la funcionalidad básica para las aplicaciones móviles, incluyendo la interfaz de usuario, conectividad a redes, almacenamiento local de datos y gestión del ciclo de vida de las aplicaciones. Al combinarlo con la configuración CLDC, MIDP proporciona un entorno de ejecución Java completo que incrementa la capacidad de los dispositivos móviles y que reduce el consumo de memoria y energía.

3.1.3.2. *Foundation Profile* (FP)

Los perfiles CDC están organizados en capas de forma que permitan la agregación según se precise para proporcionar funcionalidad a las aplicaciones para distintos tipos de dispositivos. El FP es el perfil de más bajo nivel para el

CDC. Proporciona una implementación lista para el trabajo en red que se puede emplear en implementaciones embebidas que carece de interfaz de usuario. También se puede combinar con los perfiles *Personal Basic* y personal para los dispositivos que precisan de una interfaz gráfica de usuario (IGU).

3.1.3.3. *Personal Profile* (PP)

El perfil personal, es el perfil para CDC orientado a dispositivos que requieren una GUI completa o capacidad de ejecutar componentes de una aplicación que se ejecuta en un contexto de otro programa, regularmente en un navegador, a estos componentes se denominan *applets* de Internet, y se implementan por ejemplo en: PDAs de gama alta, consola de videojuegos, teléfonos inteligentes, otros.

3.1.3.4. *Personal Basic Profile* (PBP)

El perfil Personal Basic es un subconjunto del perfil Personal y proporciona un entorno de aplicación para dispositivos con conexión que toleran un nivel de presentación gráfica básica o que precisan de conjuntos de herramientas (*toolkits*) gráficas especializadas para aplicaciones específicas.

3.1.3.5. PDA *Profile* (PDAP)

Es similar al MIDP pero diseñado para PDAs que tengan mejores pantallas y memoria de los teléfonos móviles.

3.1.3.6. *Game Profile* (GP)

Ofrece la plataforma para escribir juegos en dispositivos CDC. Debido a sus componentes y a su arquitectura que provee dinamismo a las aplicaciones, convierte a J2ME en una alternativa bastante completa a los desarrolladores y a los usuarios para trabajar con estas herramientas.

3.2. Herramientas para desarrollo

Esta sección pretende mostrar el estado del arte en relación a las aplicaciones, *toolkits* y herramientas asociadas al desarrollo de aplicaciones para J2ME y sus tecnologías asociadas (CDC, CLDC, MIDP).

A continuación se muestra la Tabla 1, la cual identifica los principales *toolkits* y emuladores del lenguaje J2ME, seguido de la Tabla 2, la cual nos muestra los principales entorno de desarrollo integrado para el lenguaje J2ME.

Tabla I. **Toolkits y Emuladores**

Herramienta	Descripción	Compatibilidad	Proveedor	Descarga
J2ME *Wireless Toolkit*	Proporciona herramientas para el desarrollo de aplicaciones para dispositivos con la especificación MIDP.	JTWI Roadmpa1, MIDP 2.0, CLDC 1.1, JSR-82 (Bluetooth)	Oracle	Oracle
Nokia *Developer's* Suite for J2ME	Orientado al desarrollo de aplicaciones para dispositivos Nokia compatibles con J2ME.	MIDP 2.0 y 1.X, CLDC 1.0	Nokia	Nokia
SDKs y herramientas Motorola iDEN	Desarrollo de aplicaciones J2ME orientadas a móviles iDEN compatibles con J2ME.	MIDP 2.0 y 1.X, CLDC 1.0	Motorola	Motorola
Samsung SDK	SDK Java para desarrollo de aplicaciones compatibles con J2ME para Samsung.	MIDP 2.0 ,1.0 y CLDC 1.0	Samsung	Samsung
Sony Ericsson	J2ME SDK para dispositivos de Sony Ericsson.	MIDP 2.0, MIDP 1.0 y CLDC 1.X(3D)	Sony Ericsson	Sony Ericsson

Fuente: elaboración propia.

Tabla II. **Entornos de desarrollos integrados (IDE's)**

Nombre	Proveedor	Descripción	Descarga
Java Studio Mobility	Oracle	IDE para desarrollar aplicaciones que se pueden desplegar en dispositivos móviles compatibles con Java, para MIDP /CLDC.	Oracle
JBuilder X Mobile *Edition*	Borland	IDE para desarrollar aplicaciones que se pueden desplegar en dispositivos móviles compatibles con J2ME y basadas en C++	Borland
Netbeans Mobility Pack	Netbeans	Con NetBeans Mobility Pack, se puede escribir, probar y depurar aplicaciones compatibles con Java.	Netbeans
Eclipse ME J2ME	Sourceforge.net	Proporciona soporte multiplataforma para el desarrollo de midlets J2ME con el IDE.	Eclipse
Code Warrior para Java	Metro Werks	Es un IDE que permite el desarrollo de aplicaciones MIDP. Incorpora compiladores J2ME / MIDP.	Metro Werks

Fuente: www.java.com (06-10-2011).

3.3. Licencia de J2ME

Entre Noviembre de 2006 y Mayo de 2007, *Sun* Microsystems liberó la mayor parte de sus tecnologías Java, pero fue el 22 de Diciembre de 2006 cuando se incluyó J2ME bajo la licencia GNU GPL1, de acuerdo con las especificaciones del Java *Community* Process2, de tal forma que todo J2ME de *Sun* es ahora *software* libre (aunque la biblioteca de clases de Oracle que se requiere para ejecutar los programas Java aún no lo es).

3.4. Versión actual de J2ME

La versión actual es el SDK 3.0, el cual se puede encontrar con más detalle en la página oficial de Java.

4. PLATAFORMA *ANDROID*

Siempre es saludable que en cualquier entorno informático o de comunicaciones se incorpore como alternativa a los desarrolladores una plataforma que permite realizar aplicaciones con *software* libre para dispositivos móviles. Lo más interesante de *Android* es que desde su aparición hace 3 años, ya se está consolidando como una alternativa fuerte para desarrolladores y usuarios, ahora es un desarrollo totalmente fuerte en el sector de las telecomunicaciones e internet.

Android plataforma para aplicaciones móviles, totalmente abierta tanto los fabricantes de dispositivos móviles y para los desarrolladores de aplicaciones que funcionan en tales dispositivos. *Android* incluye básicamente un sistema operativo, middleware y aplicaciones base.

Inicialmente desarrollado por Google y luego en conjunto con Open *Handset Alliance*, *Android* permite a los desarrolladores escribir código gestionado en lenguaje de programación Java y controlar los dispositivos por medio de bibliotecas desarrolladas o adaptadas por Google.

El núcleo de *Android* es Linux Kernel 2.6, a este nivel están implementados todos los drivers de los dispositivos primarios, como son cámaras, puertos USB, teclados. El fabricante de un dispositivo puede agregar sus propios drivers.

La empresa Google ha publicado el Kit de Desarrollo de *Software* (SDK), que provee de herramientas y conjunto de Interfaces de Programación (APIs) necesarias para que los programadores o aventureros en las tecnologías

puedan realizar aplicaciones utilizando la plataforma *Android* por medio del lenguaje de programación Java.

4.1. Funciones principales de *Android*

Las plataformas se definen por funciones que se añadan al sistema y que faciliten la utilización de los dispositivos, administrando las aplicaciones y permitir la utilización óptima de los recursos. Las funciones principales de la plataforma *Android* que podemos mencionar son:

- Página de inicio: muestra las aplicaciones, *Widgets* y accesos directos. También apoyo personalizable con fondos de pantalla y diferentes temas.
- Teléfono: regula las funciones de red telefónica, así como controles de llamada, conferencia, servicios suplementario y fácil integración con contactos agregados.
- Navegador web: es un navegador basado en *WebKit* que soporta lenguaje de marcado de hipertexto (HTML) y lenguaje extensible de marcado de hipertexto (XHTML).
- Correo electrónico: proporciona acceso a servidores de correo electrónico disponibles en Internet basados en protocolos POP3, IMAP4 y SMTP.
- Multimedia: permite la gestión, importación, y reproducción de contenido multimedia en diferentes codificaciones con formatos comunes de video e imágenes (MPGE4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- Utilidades: alarmas, calculadoras, calendarios, cámara, contactos, mensajería instantánea, Mensajes multimedia, ajustes, marcados de voz y muchas otras utilidades.

Android posee un conjunto de buenas prácticas y estándares de desarrollo para aplicaciones que permite la reutilización y reemplazo de

componentes, típico en lenguajes de programación orientados a objetos. La arquitectura de las aplicaciones permite que una aplicación pueda publicar su funcionalidad para que esté disponible y pueda ser utilizada por otras aplicaciones. El mismo mecanismo permite a los componentes ser reemplazados por los usuarios. Los más destacables son:

- Administrador de notificaciones: permite a todas las aplicaciones mostrar alertas personalizables en la barra de estado.
- Administración de actividades: maneja el ciclo de vida de las aplicaciones y provee un comportamiento común en la navegación, manteniendo una pila de recursos que pueden ser recuperados después, el usuario puede utilizar otras aplicaciones y luego recuperar el estado de cualquier actividad.
- Administrador de recursos: provee acceso a los recursos como cadenas, gráficos y archivos, control de acceso a disco de otras aplicaciones.
- Vistas: utilizadas para desarrollar listas, grillas, cajas de texto, botones e incluso un navegador web.

4.1.1. Máquina virtual de *Android*

La máquina virtual de *Android* se llama Dalvik, diseñada por Dan Bornstein con contribuciones de otros ingenieros de Google. Esta máquina virtual permite emular a un ordenador que ejecuta aplicaciones como si de un sistema real se tratase, está máquina virtual está diseñada para dispositivos móviles, debido a la poca memoria y múltiples instancias simultaneas, delegando el control y la gestión de memoria al sistema operativo.

4.1.2. Gráficos

Incorpora variedad de librerías de gráficos 2D y gráficos 3D basados en la especificación estándar Open GL, que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan dichos gráficos.

4.1.3. Sistema gestor de base de datos

Utiliza la base de datos SQLite, para almacenamiento de datos estructurados. Sus principales características son:

- Sistema de base de datos relacional compatible con ACID (transacciones seguras).
- Contenida en una pequeña librería en C.
- Se integra en la aplicación (no es un proceso independiente, ni diferencia entre cliente-servidor).
- Todos los datos se almacenan en un único fichero.

4.1.4. Redes

Utiliza tecnologías de conectividad mediante USB 2.0, Bluetooth 2.0 EDR, EDGE (evolución del GPRS) 3.5G y 802.11 b/g WIFI.

4.1.5. Entornos de desarrollo

El SDK disponible en el sitio oficial de *Android* trae un emulador de dispositivos, que sirve como herramienta para depurar perfiles de memoria, performance, y un plug-in para el Entorno de desarrollo integrado (IDE) Eclipse.

En cuanto al *hardware* soportado por la plataforma, funcionara en cualquier ARM basado en el Kernel Linux. Exige un mínimo de 128Mb de RAM y 256 de memoria flash.

4.1. Arquitectura

Android presenta una arquitectura en capas, las cuales se apoyan en las capas inferiores, anteriormente se comentó que la base de arquitectura de esta plataforma se encuentra en el núcleo Linux.

4.1.1. Kernel

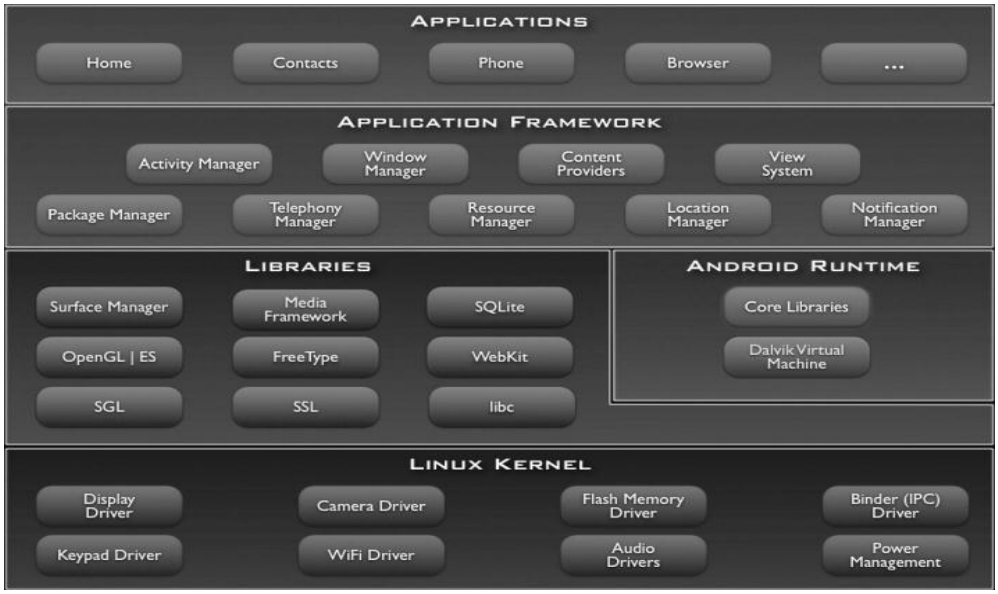
En este nivel están implementados todos los drivers de los dispositivos primarios o periféricos como lo son: cámaras, puertos USB, teclado, video, red inalámbrica, bluetooth, audio, administrador de tareas, acceso y manipulación de disco (E/S). En esta capa los fabricantes también agregan o personalizan los drivers, por ejemplo para las pantallas táctiles.

El esquema que describe la arquitectura de *Android* se muestra a continuación.

4.1.2. Librerías nativas

Sobre el núcleo están las librerías nativas, las cuales ofrecen funciones básicas a otros subsistemas como por ejemplo SQLite (base de datos con soporte SQL), Open GL (gráficos avanzados en 3D), SSL (cifrado de datos), CODECs de medios (audio, video).

Figura 4. **Diagrama de la Arquitectura *Android***



Fuente: www.Android.com (06-10-2011).

4.1.3. Ejecución de *Android*

Toda esta estructura es compartida por la máquina virtual Dalvik que es donde se ejecutan las aplicaciones especificadas. Cada aplicación corre su propio proceso, con su propia instancia en la máquina virtual David, esta máquina virtual ha sido diseñada de tal forma que un dispositivo pueda ejecutar múltiples instancias de forma eficiente. Dalvik está basada en registros y ejecuta clases compilados con Java, que han sido transformadas en formato .dex por la herramienta incluida en la plataforma denominada DX.

Los ejecutables no son clases de Java, sino que las clases de Java son post-procesadas a un formato llamado .dex, el cual es optimizado para que tenga un uso de memoria mínimo, la máquina virtual tiene acceso a las librerías de funciones estándar de Java (*Core Libraries*) por ejemplo Objeto, Cadena, entero, etc., esto es como decir el ambiente de desarrollo para el lenguaje Java.

4.1.4. Estándares de aplicaciones

La máquina virtual tiene acceso a un conjunto de estándares y buenas prácticas para desarrollo de aplicaciones que contienen un gran número de módulos escritos en lenguaje Java para aplicaciones. Se divide en subsistemas como son las notificaciones, actividades, recursos y telefonía.

4.1.5. Aplicaciones

Es el nivel superior, las aplicaciones se ejecutan sobre toda la infraestructura. Estas son 100% lenguaje Java y se pueden encontrar en aplicaciones como cliente de correo, agenda, navegador, reloj.

4.2. Anatomía de una aplicación

De una manera simplificada, una aplicación en *Android* es una colección de componentes de varios tipos. Estos tienen en su bajo nivel acoplamiento entre ellos, de tal manera, que una aplicación podría ser como federación de todos ellos. Estos componentes se ejecutan en el mismo proceso del sistema. Se pueden crear múltiples hilos dentro de tal proceso, y además es posible crear procesos hijos completamente separados del proceso padre.

Estas son las partes más importantes de las APIS (*Application Programming Interface*) de *Android*.

4.2.1. *Android Manifest* (XML)

Es el archivo de control que le dice al sistema que es lo que debe hacer con los componentes de más alto nivel que se han creado (“Actividad”,

“Servicio”, “Receptor de Intentos” y “proveedor de contenidos”). Una vez se hayan seleccionado los componentes que se van a utilizar en la aplicación, es necesario que se listen en el “AndroidManifest.xml”

4.2.2. Actividades

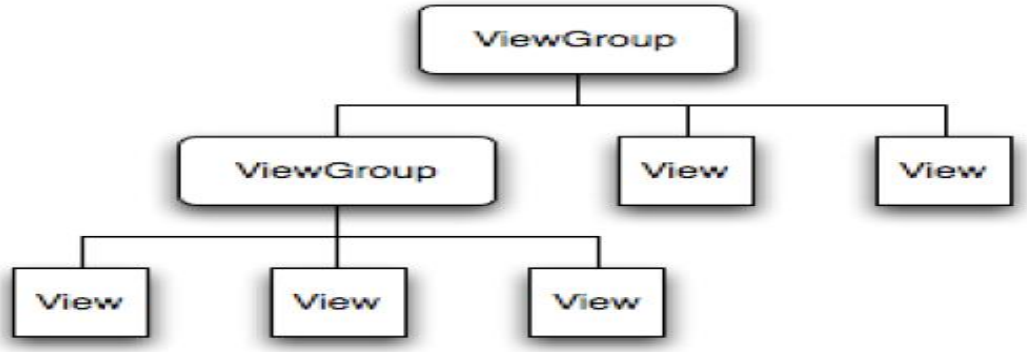
Es un objeto que tiene un ciclo de vida, es un trozo de código que desarrolla algún trabajo en la aplicación. Existen actividades que despliegan la interfaz de usuario, generalmente es el punto de inicio de la aplicación.

4.2.3. Interfaz de usuario y vistas

Las actividades realizan varias acciones, pero estas no se presentan en pantalla. Para conseguir que aparezcan en la pantalla es necesario diseñar la Interfaz Gráfica, Vistas y Vista de Grupos que son las clases que se usan para crear la interfaz entre el usuario y la plataforma *Android*. La vista es un objeto que sabe cómo dibujarse asimismo en la pantalla. Las interfaces de usuario esta conformadas por un árbol de vistas. Las vistas son estructuras de datos cuya característica contiene los datos de la capa e información del área rectangular de la pantalla. Una vista tiene: *layout*, *drawing*, *focus change*, *scrolling*.

Una vista de grupo es un objeto especial de una vista cuya función es contener y controlar la lista de vistas y de otros grupos de vistas. Lo anteriormente descrito se representa en la siguiente figura:

Figura 5. **Árbol de Nodos de elementos de Interfaz de Usuario**



Fuente: www.Android.com (06-10-2011).

Para añadir el árbol a la pantalla, la actividad llama al método setContentView() y pasa una referencia al objeto nodo principal. Una vez que el sistema *Android* ha referenciado el objeto nodo principal ya puede trabajar directamente con el nodo para anular, medir y dibujar el árbol. Cuando la actividad está activa y recibe el foco, el sistema lo notifica y le pide al nodo principal medidas y dibuja el árbol. El nodo principal entonces pide que sus nodos hijos se dibujen a sí mismos, a partir de ese momento cada nodo viewgroup del árbol es responsable de pintar sus hijos directos.

Cabe mencionar un tip para explotar ya que *Android* puede dibujar un árbol tan simple o complejo, se pueden desarrollar *Widgets* que son objetos *View* que ayudan a definir la Interfaz de usuario que existen pre-implementados como son botones, seleccionadores, campos de texto, e incluso controles como relojes, calendarios, control de *zoom*, etc. La mejor forma de definir *layouts* es mediante ficheros XML. Cada elemento del XML puede ser un view o viewgroup. Los layouts predefinidos que proporciona *Android* son: LenarLayout, RelativeLayout, AbsoluteLayout, TableLayout, GridLayot.

La interfaz de usuario define los eventos para interaccionar con el usuario. Se trata de definir los eventos de usuario van a reaccionar a cada uno de los elementos o Widgets de la interfaz gráfica.

4.2.3.1. Intentos

Es un mensaje que representa la “intención” de hacer algo. Por ejemplo si desea escribir información en campo de texto, debe expresar su “Intento” para ver los caracteres en el campo, y luego trasladársela al sistema. Posteriormente, el sistema ubicará una pieza de código que sepa como manipular el Intento y la ejecuta. Los intentos también pueden ser usados para difundir eventos interesantes a través del todo el sistema.

4.2.3.2. Servicios

Es un cuerpo de código que se ejecuta en segundo plano (*background*). Se ejecuta dentro de su propio proceso o en el contexto del proceso de otra aplicación, dependiendo de lo que se necesite. Otros elementos se unen a la petición del servicio y solicitan sus métodos a través de los llamados Controles de Procedimiento Remoto (RPC).

4.2.3.3. Notificaciones

Es un pequeño ícono que aparece en la barra de estado. Permite a los usuarios interactuar con este para recibir información, un ejemplo de estas notificaciones son: SMS, Correo de voz, aplicaciones que utilizan este mecanismo.

4.2.3.4. *Content-Provider*

Es un almacén de datos que provee acceso a datos en el dispositivo. Por ejemplo: datos de la lista de contactos, las aplicaciones pueden definir *Content-Provider* específicos para exponer datos propios de una aplicación.

4.3. Bloques de construcción de una aplicación *Android*

Una aplicación *Android* siempre utiliza alguno o varios bloques de construcción descritos anteriormente, más profundamente son:

- Actividades.
- Intentos (*Receiver*).
- Servicios.
- Proveedor de contenido.

4.3.1. Actividades

Es básicamente la parte visual de una aplicación, es el bloque más usado en las aplicaciones *Android*. Es una pantalla individual en cada aplicación. Cada actividad se implementa como una clase que hereda de *Activity*, lo que hará que las clases se desplieguen en la interfaz de usuario, compuestos de vistas y responder a eventos. Por ejemplo, una aplicación de mensajería podría usar una pantalla para mostrar el listado de contactos y en una segunda pantalla para escribir el mensaje al contacto seleccionado y otras pantallas para cambiar la configuración.

Android usa una clase llamada *Intent* para moverse de una pantalla a otra. Un *intent* describe lo que una aplicación desea hacer. Las dos partes más

importantes de la estructura de datos de un *Intent* son la acción y los datos sobre los cuales se actuará. Los valores típicos para la acción son MAIN, VIEW, PICK, EDIT, etc. Por ejemplo, para ver la información de contacto de una persona podría ser necesario crear un *Intent* con la acción VIEW y los datos definidos como una URI (Uniform Resource Identifer) que representa a una persona.

4.3.2. *Intent-Receiver*

Es un bloque importante en una aplicación por los siguientes aspectos:

- Los *Intent-Receiver* no despliegan una interfaz de usuario, sin embargo ellos pueden utilizar el administrador de notificaciones para avisar al usuario que algo interesante está pasando.
- No es necesario que este ejecutándose una aplicación para que sus *Intent-Receiver* puedan ser llamados. Si una aplicación no estuviera ejecutándose, el sistema la puede activar cuando uno de sus *Intent-Receivers* sea activado.
- Las aplicaciones también pueden transmitir sus propios *Intents* a otras aplicaciones.

4.3.3. *Service*

Es una aplicación que se mantiene activada por un largo tiempo y no despliega una interfaz de usuario. Por ejemplo, un programa que reproduce archivos mp3 desde una lista de música, mientras el usuario realiza otras actividades. En este caso el programa podría iniciar un servicio y de esa forma reproducir la música sin necesidad de usar la pantalla.

El sistema mantendrá el servicio ejecutándose hasta que finalice. Es posible conectarse a Servicio y activarlo sino estuviera ejecutándose. Cuando la aplicación está conectada al servicio, la comunicación entre aplicación y servicio se realiza a través de la interfaz de Servicio disponible. Por ejemplo, siguiendo con el reproductor mp3, esta interfaz podría permitir hacer pausa, saltar o retroceder una nueva canción.

4.3.4. *Content-Provider*

Es un servicio que le ofrece la capacidad a las aplicaciones de comunicarse con otras maneras internas a *Android*. Estas pueden almacenar datos en archivos, en base de datos (SQLite) o cualquier otro mecanismo. El *content-Provider* es de utilidad cuando los datos de la aplicación deben ser compartidos con otras aplicaciones. El *Content-Provider* es una clase de Java que implementa un conjunto estándar de métodos para que otras aplicaciones almacenen o recuperen el tipo de datos que el *Content-Provider* manipula.

Android viene con un paquete de *Content-Provider* con tipo de contenidos comunes como Audio, imágenes, videos o contactos.

Cada objeto alojado en un *Content-Provider* tiene un identificador único, cada búsqueda o consulta, devuelve un objeto cursor que se puede mover de elemento en elemento y dentro de ellos de campo a campo para obtener la información buscada de cada uno de ellos.

Cada *Content-Provider* puede estar compuesto de varias tablas de datos, cada una de ellas identificadas con una URI. Se necesitan tres aspectos para realizar búsquedas en un *Content-Provider*:

- La URI que identifica la tabla dentro del p*rovider*.
- Los nombres de los campos que quiere recibir.
- Los tipos de datos de esos campos.

4.4. Paquete de desarrollo (SDK)

Actualmente Google tiene publicado el SDK 3.0. Este paquete de desarrollo incluye las APIs y herramientas necesarias para desarrollar las aplicaciones utilizando Java como lenguaje de programación y testear el código respectivamente. Las librerías adjuntas son compatibles con los siguientes entornos de desarrollo: Eclipse, JDK5 y 6, *Android development tool plugin* y Apache Ant.

Google ofrece 3 versiones del SDK, para Windows, Mac OSX (Intel) y Linux (x86). La forma más rápida de desarrollar es a través del *plug-in* de Eclipse ADT. Este *plug-in* incluye un asistente para empezar en nuevos proyectos que ayuda a crear los ficheros básicos para comenzar el desarrollo.

También incluye un editor de código *Android* que sirve de ayuda para escribir archivos con código XML válido para crear el archivo Manifest.xml y otros recursos. Además el SDK cuenta con un emulador que permite “agregar” componentes de *hardware*, y que este emulador puede integrarse fácilmente con el IDE Eclipse.

4.5. Licencia de *Android*

Posee doble licencia GPLV2 (Kernel de Linux) y Apache 2.0 (Aplicativos) las cuales fueron pensadas para brindar mayor flexibilidad y oportunidades de

negocio al permitir a los desarrolladores la creación de aplicaciones manteniendo sin afectar con esto la licencia del kernel de Linux.

5. PLATAFORMA iOS

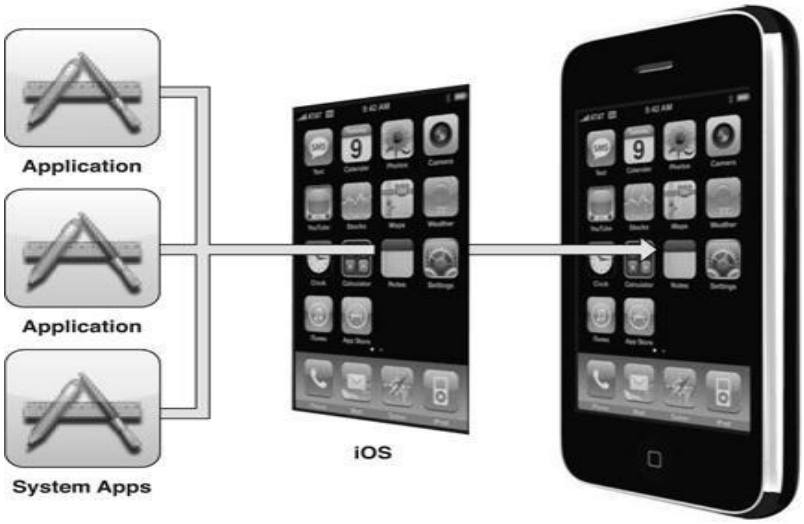
iOS es el sistema operativo que se ejecuta en el iPad, iPhone, iPod Touch. Este sistema operativo gestiona el *hardware* del dispositivo y proporciona las tecnologías necesarias para implementar las aplicaciones nativas. El sistema operativo también incluye varias aplicaciones del sistema como: teléfono, correo, navegador que son servicios estándar para el usuario

El SDK contiene herramientas e interfaces necesarias para desarrollar, instalar, ejecutar y probar aplicaciones nativas. Las aplicaciones se construyen utilizando los *frameworks* del sistema junto con el lenguaje Objective-C y se ejecutan directamente en iOS. A diferencia de las aplicaciones web, las aplicaciones nativas están instaladas físicamente en un dispositivo y por lo tanto siempre está disponible para el usuario, incluso cuando el dispositivo está en modo offline. Las aplicaciones residen junto con otras aplicaciones del sistema y datos del usuario que sincroniza con el ordenador por medio de iTunes.

5.1. Arquitectura iOS

La arquitectura de iOS es similar a la arquitectura básica que se encuentra en Mac OS X. El nivel más alto, actúa como intermediario entre el *hardware* subyacente y las aplicaciones que aparecen en la pantalla, como se muestra en la figura siguiente:

Figura 6. **Nivel Intermedio Arquitectura iOS**

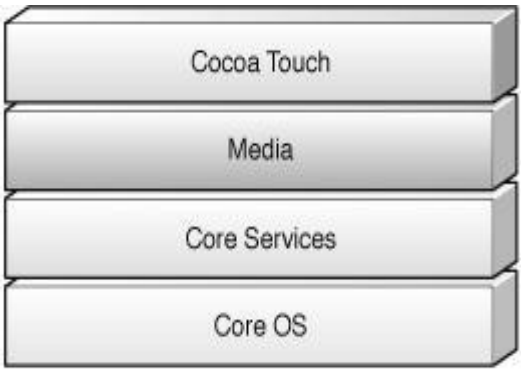


Fuente: http://developer.apple.com/library/ios/#documentation (06-10-2011).

Las aplicaciones creadas rara vez se comunican con el *hardware* directamente. En cambio las aplicaciones se comunican a través de un conjunto de interfaces del sistema definidas que protegen las aplicaciones de los cambios del *hardware*. Esta abstracción hace que sea fácil crear aplicaciones que trabajen constantemente en los dispositivos con capacidades de *hardware* diferente.

La aplicación de tecnologías iOS se puede ver como un conjunto de capas que se muestran en la figura siguiente:

Figura 7. **Arquitectura de Capas de iOS**



Fuente: http://developer.apple.com/library/ios/#documentation (06-10-2011).

Como se observa en la figura anterior en un recorrido ascendente, en las capas más bajas del sistema están los servicios fundamentales y tecnologías en que se basan todas las aplicaciones, las capas de nivel superior contienen servicios y tecnologías más sofisticados.

Al escribir código, se debe preferir los *frameworks* de alto nivel sobre los *frameworks* de bajo nivel siempre que sea posible. Los *frameworks* de alto nivel proporcionan abstracciones orientadas a objetos para construcciones de bajo nivel. Estas abstracciones por lo general hacen que sea mucho más fácil escribir código, ya que reducen la cantidad de código que tienen que escribir y encapsular características potentes y complejas, tales como tomas de corrientes, sockets y redes.

5.2. ¿Qué se encuentra en el SDK de iOS?

El SDK de iOS viene con todas las interfaces, herramientas y recursos necesarios para desarrollar aplicación en iOS desde un ordenador Macintosh basado en Intel. *Apple* ofrece la mayoría de sus interfaces de sistemas en

paquetes especiales llamados *frameworks*. Un *framework* es un directorio que contiene una biblioteca dinámica compartida y los recursos (como archivos de encabezado, imágenes, aplicaciones de ayuda.) necesarios para apoyar esta biblioteca. Para utilizar *frameworks*, se deben enlazar a la aplicación del proyecto de la misma manera que cualquier otra biblioteca compartida.

Además de los *frameworks*, *Apple* también ofrece tecnologías en forma de librerías estándar. Debido a que iOS está basado en UNIX, muchas de las tecnologías que forman los niveles inferiores del sistema operativo se derivan de las tecnologías de código abierto. Las interfaces de estas tecnologías, están disponibles en la biblioteca estándar y directorios de la interfaz. Algunos otros componentes claves del SDK incluyen:

- Xcode Tools: herramienta que apoyan el desarrollo de aplicaciones iOS, incluyendo la aplicación clave.
- XCode: es un entorno de desarrollo integrado (IDE) que gestiona proyectos de aplicación y permite editar, compilar, ejecutar y depurar el código. XCode se integra en muchas otras herramientas y es la principal que se utiliza durante el desarrollo.
- *Interface Builder*: es una herramienta que se utiliza para montar la interfaz de usuario visualmente. Los objetos de la interfaz que se crean se guardan en un tipo especial de archivos y se carga en la aplicación en tiempo de ejecución.
- Instruments: realiza un análisis de rendimiento en tiempo de ejecución. Se utiliza *instrument* para reunir información sobre el comportamiento de la aplicación además de identificar los problemas potenciales.
- Simulador iOS: es una aplicación de Mac Os X que simula tecnología de las capas de IOS, lo que permite testear y depurar las aplicaciones iOS de forma local en un equipo Macintosh basado en Intel.

- Librerías iOS para desarrolladores: es la referencia y documentación conceptual que explica las tecnologías iOS y el proceso de desarrollo de aplicaciones.

Aunque se pueden ejecutar aplicaciones iOS en el simulador, y las herramientas del SDK permiten ejecutar y depurar aplicaciones directamente en un dispositivo conectado. El simulador es ideal para crear y probar aplicaciones de forma rápida, pero no es un sustituto de prueba en dispositivos real.

¿Qué tipo de aplicaciones se pueden crear para iOS?, iOS soporta dos tipos de aplicaciones:

- Aplicaciones nativas.
- Aplicaciones web.

El SDK de iOS soporta la creación de aplicaciones nativas que aparecen en la pantalla principal del dispositivo. No es compatible con la creación de otros tipos de código, como drivers, *frameworks* o librerías dinámicas. Si se desea integrar código de un *framework* o librería dinámica se debe vincular estáticamente el código en un archivo ejecutable de la aplicación en la construcción de su proyecto.

Las aplicaciones Web utilizan una combinación de HTML, hojas de estilo(CSS) y código JavaScript para ejecutar aplicaciones interactivas que se alojan en un servidor web, se transmiten a través de la red y se ejecutan dentro del navegador web Safari. Las aplicaciones nativas, por el contrario, se instalan directamente en el dispositivo y pueden funcionar sin la presencia de una conexión de red.

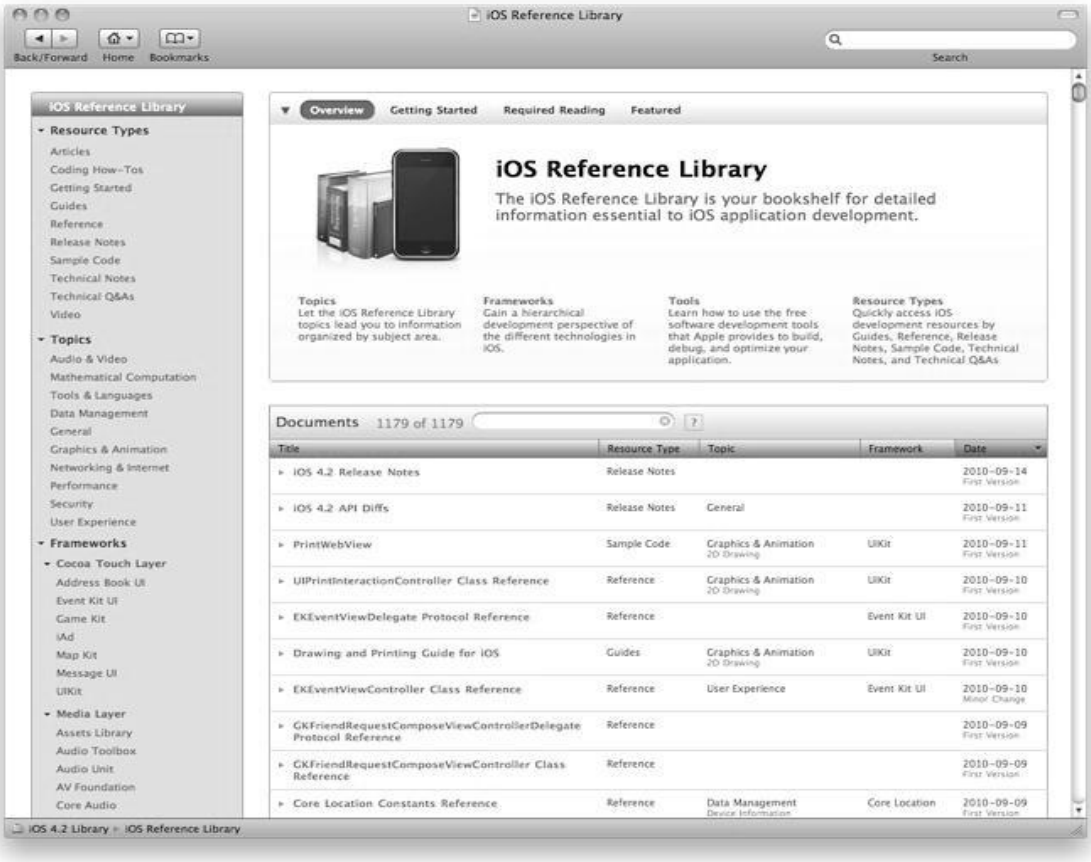
5.3. ¿Cómo utilizar las librerías de desarrollo?

Las librerías de desarrollo de iOS contienen documentación, ejemplos de código, tutoriales y otra información que se necesite para escribir aplicaciones iOS. Debido a que las librerías de desarrollo contienen miles de páginas de documentación, que van de alto nivel a nivel principiante, es difícil comprender cómo encontrar información en un paso en el proceso de desarrollo.

La biblioteca de desarrollo utiliza algunas técnicas para organizar el contenido que debería hacer más fácil para navegar entre ella.

Para acceder a las librerías de desarrollo de iOS en una página web desde XCode se selecciona Ayuda→Documentación de Desarrollo, en la ventana que se muestra se pueden realizar búsquedas y permite visualizar documentos que se pueden marcar para ser consultados. Además se puede filtrar por medio del campo de búsqueda el cual desplegará la información de los documentos solicitados. La muestra de la variedad de documentación que se encuentra en la ayuda de la librería de desarrollo se muestra en la siguiente figura:

Figura 8. **Librerías de Desarrollo de iOS**



Fuente: http://developer.apple.com/library/ios/#documentation (06-10-2011).

5.4. Capas de arquitectura iOS

5.4.1. Capa *Cocoa Touch*

La capa de *Cocoa Touch* contiene los *frameworks* claves para la creación de aplicaciones iOS. Esta capa define la infraestructura de aplicaciones básicas y el soporte a tecnologías claves como multitarea, entrada basada en el contacto, notificaciones *push*, y muchos servicios del sistema de alto nivel. Al diseñar aplicaciones, se debe investigar las tecnologías en esta primera capa para ver si se ajustan a sus necesidades.

Las siguientes secciones describen varios conceptos claves disponibles en la capa *Cocoa Touch*, esta sección define las presentaciones de alto nivel.

- *Multitasking*: las aplicaciones creadas usando SDK 4.0 de iOS o posteriores) no se termina cuando el usuario presiona el botón de inicio, sino que cambian de un contexto de ejecución en segundo plano. El soporte multitarea definido por el marco de interfaz de usuario denominado UIKit ayuda a la transición de aplicaciones desde y hacia el estado de *background* sin problemas. Para preservar la vida de la batería, la mayoría de aplicaciones se suspenden por el sistema poco después de entrar en *background*. La solución de suspensión se mantiene en la memoria pero no se ejecuta ningún código. Este comportamiento permite que una aplicación para reanudar rápidamente cuando se requiera sin consumir energía de la batería interna. Sin embargo, las aplicaciones pueden permitir que continúe funcionando en segundo.
- Una aplicación puede solicitar una cantidad finita de tiempo para completar una tarea importante.
- Una aplicación puede declararse como el soporte a los servicios específicos que requieren de tiempos regulares de ejecución en segundo plano.
- Una aplicación puede utilizar las notificaciones locales para generar alertas de usuarios en los horarios establecidos, o no se ejecuta la aplicación.
- Impresión: introducido en iOS 4.2, el soporte de impresión UIKit permite a las aplicaciones enviar el contenido de forma inalámbrica a impresoras cercanas. En su mayor parte UIKit hace todo el trabajo pesado asociado a la impresión. gestiona la interfaces de impresión, trabaja con la aplicación para representar el contenido para imprimir, y se encarga de la programación y ejecución de los trabajos de impresión en la impresora.

Los trabajos de impresión presentados por la aplicación se entregan al sistema de impresión, que gestiona el proceso de impresión actual. Los trabajos de impresión de todas las aplicaciones en un dispositivo están en espera y se imprimen en primero entrar, primero en imprimir. Los usuarios pueden obtener el estado de los trabajos de impresión desde la aplicación.

- Protección de Datos**:** introducido en iOS 4.0, la protección de datos permite a las aplicaciones que trabajan con información confidencial del usuario aprovechando el cifrado integrado. Cuando la solicitud designa a un archivo especifico como protegidos, el sistema almacenará el archivo en disco en un formato codificado. Mientras el dispositivo está bloqueado, el contenido del archivo es inaccesible tanto para su aplicación y para cualquier intruso potencial. Sin embargo, cuando el dispositivo está bloqueado por el usuario, una clave de descifrado secreto permite que una aplicación tenga acceso al archivo.
- Servicio de Notificaciones *Push*: introducido en iOS 3.0, proporciona una manera de alertar a los usuarios de una nueva información, incluso cuando las aplicaciones no están en ejecución. El uso de este servicio, puede enviar notificaciones de texto, añadir una tarjeta de identificación con el ícono de la aplicación, o desencadenar alertas audibles en los dispositivos de los usuarios en cualquier momento.

Existen varias presentaciones en esta capa, sin embargo solo se mencionan las que se consideran importantes para llamar la atención del desarrollador.

5.4.2. Capa Media

Contiene medios de comunicación de gráficos, audio y video y tecnologías orientadas a la creación de experiencia multimedia. Las

tecnologías en esta capa se han diseñado para hacer más fácil la creación de aplicaciones que se lean y reproduzcan de forma correcta. Algunas tecnologías son:

- Tecnologías de Gráficas: los gráficos de alta calidad son parte importante en las aplicaciones de iOS. La forma más simple y eficiente al crear una aplicación es utilizar imágenes predeterminadas junto con las vistas estándar del *framework UIKit* y dejar que el sistema haga dibujo. Sin embargo, puede haber situaciones donde necesita ir más allá de gráficos simples. En estas situaciones, puede utilizar las siguientes tecnologías para administrar el contenido gráfico de la aplicación:
  - *Core* de gráficos (también conocido como *Quartz*) se encarga del vector nativo 2D y renderizado basado en imágenes.
  - *Core Animation***:** proporciona soporte avanzado para la animación de puntos de vista y otros contenidos.
  - OpenGL: proporciona soporte para 2D y 3D utilizando interfaces aceleradoras por *hardware*.
  - Núcleo de Texto: proporciona un sofisticado diseño de texto y motor de renderizado.
- Tecnología de Audio: están diseñadas para ayudarle a proporcionar una rica experiencia de audio para los usuarios. Esto incluye la posibilidad de reproducir audio de alta calidad de audio, registro de alta calidad, y activar la función de vibración en determinados dispositivos. El *framework* media *player* ofrece un fácil acceso a la biblioteca de iTunes del usuario y soporte para la reproducción de pistas y listas de reproducción.
- Tecnologías de Video: iOS ofrece varias tecnologías para reproducir su contenido basado en video. También permite utilizar estas tecnologías para capturar video e incorporarlo a su aplicación.

5.4.3. Core Services

Esta capa contiene los servicios del sistema fundamentalmente todas las aplicaciones en uso. Incluso si no se utiliza estos servicios, muchas partes del sistema se construyen en la parte superior de ellos. A continuación se describen las presentaciones principales:

- Bloque de Objetos: introducido en iOS 4.0, los objetos de bloques son un lenguaje de nivel C, construido para incorporar C y código *Objective*-C. Los bloques son particularmente útiles como las devoluciones de llamada o en lugares donde se necesita una manera fácil de combinar tanto código a ejecutar y datos asociados.
- *Grand Central Dispatch*: es una tecnología BSD que se utiliza para administrar la ejecución de las tareas en la aplicación. GCD combina un modelo de programación asincrónica con un núcleo altamente optimizado para ofrecer una cómoda alternativa a *threading*. GCD ofrece tipos de tarea tales como leer y escribir descriptores de ficheros, temporizados y el control de señales y eventos del proceso.
- SQLite: Desde la aplicación, se pueden crear archivos de base de datos local y manejar las tablas y registros de los archivos. La colección está diseñada para uso general, pero sigue siendo optimizado para proporcionar un acceso rápido a los registros de base de datos.

5.4.4. *Core* OS

Contiene características de bajo nivel que la mayoría de otras capas se basan. Incluso si se usa estas tecnologías directamente en las aplicaciones, lo más probable que es utilizado por otros *frameworks*. Y en situaciones donde es

necesario tratar explícitamente con la seguridad o comunicarse con un acceso a *hardware* externo los *frameworks* utilizan esta capa.

El nivel de sistema como el *core* del entorno, los controladores y las interfaces de UNIX a bajo nivel están a cargo del sistema operativo. El *core* en si está basado en Mach y es responsable de todos los aspectos del sistema operativo. Gestiona el sistema de memoria virtual, hilos, sistemas de archivos, red y comunicación entre procesos. Los drivers en esta capa también proporcionan la interfaz entre el *hardware* disponible y los marcos del sistema. Por razones de seguridad, el acceso al núcleo y los controladores se limitan de los *frameworks* del sistema y aplicaciones.

iOS provee un conjunto de interfaces para el acceso a muchas características de bajo nivel del sistema operativo. Su aplicación tiene acceso a estas funciones a través de las librerías LibSystem. Las interfaces están basadas en C y proporcionar soporte para las siguientes:

- *Threading* (Hilos POSIX)
- Redes (BSD sockets)
- Acceso al *File-System*.
- Estándares de Entrada/Salida
- Bonjour o Servicios DNS
- Información local.
- Posicionamiento de memoria.
- Computación matemática.

5.5. Herramientas de desarrollo iOS

Para desarrollar aplicaciones para iOS, se necesita un ordenador Macintosh con procesador Intel y las herramientas Xcode. Xcode es la suite de herramientas de desarrollo de *Apple* que proporciona apoyo a la gestión de proyectos, edición de código, ejecutables de construcción, depuración a nivel de fuentes, optimización del performance y más.

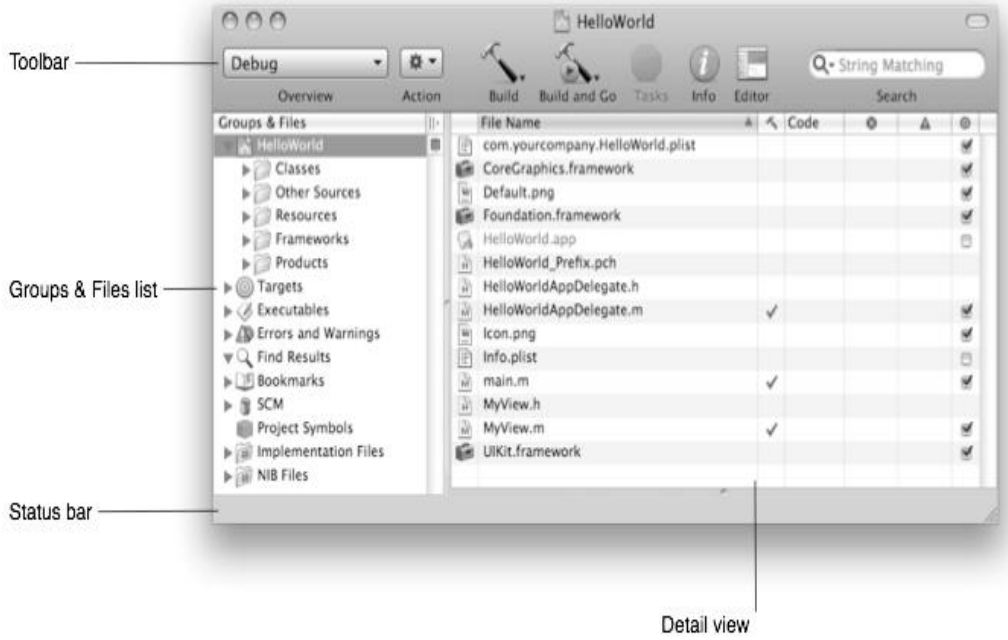
5.5.1. XCode

El enfoque de las experiencias de desarrollo es la aplicación XCode, XCode es un entorno de desarrollo integrado (IDE) que proporciona todas las herramientas necesarias para crear y gestionar proyectos iOS y código fuente, genera un archivo ejecutable, ejecuta y depura el código, ya sea en el simulador iOS o en un dispositivo. XCode incorpora una serie de características para hacer fácil el desarrollo de aplicaciones iOS, incluyendo algunas como las siguientes:

- Sistema de gestión de proyectos para la definición de productos de *software*.
- Un entorno de edición de código, que incluye características tales como color de sintaxis, completado de código e indexación de símbolos.
- Un visor de documentos avanzados para la visualización y búsqueda de documentos de *Apple*.
- Un parser sensible al contexto para ver la información sobre los símbolos de código seleccionado.
- Un avanzado sistema de construcción con la comprobación de dependencias y construcción de evaluación de la regla de compiladores GCC, soporte de compiladores C, C++, Objective-C, Objective-C ++.

Para crear una nueva aplicación en iOS, se empieza por crear un nuevo proyecto en XCode. El proyecto gestiona toda la información asociada a la aplicación, incluyendo los archivos de origen, la configuración de generación, y las normas necesarias para poder unir todas las piezas. El corazón de todo proyecto de XCode es la ventana del proyecto. Esta ventana proporciona acceso rápido a todos los elementos claves de la aplicación. La ventana se muestra en la figura siguiente:

Figura 9. **Ventana del IDE XCode**

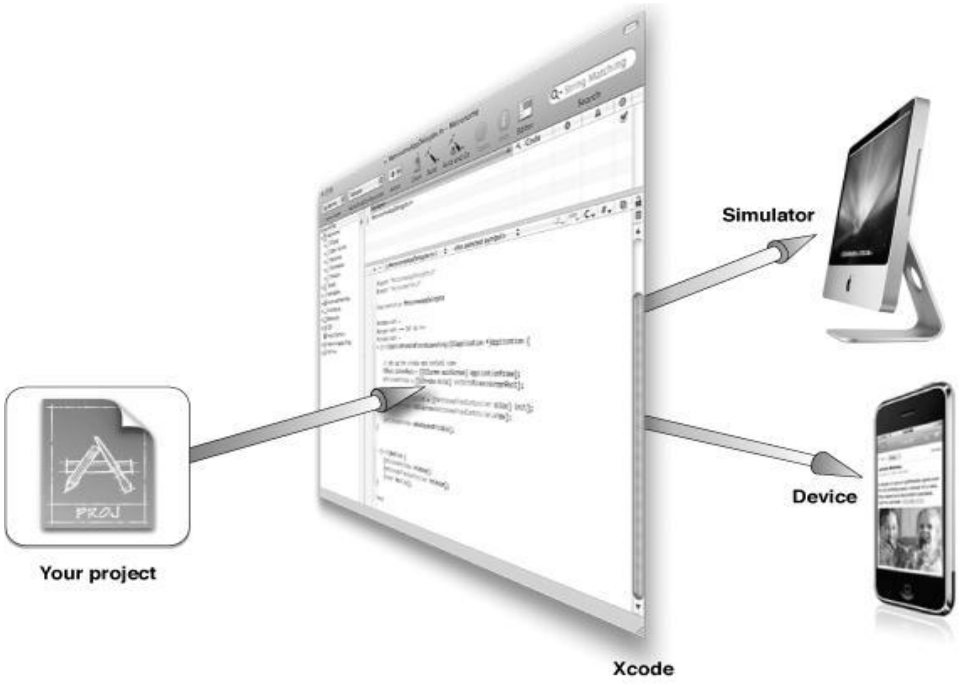


Fuente: http://developer.apple.com/library/ios/#documentation (06-10-2011).

En la barra de herramientas, puede acceder a herramientas de uso común y comandos. Y en el panel de detalles, se puede configurar un espacio para trabajar en el proyecto. Otros aspectos de la ventana del proyecto proporcionar información contextual sobre el proyecto.

Cuando se genera la aplicación en XCode, se tiene la opción de construirlo para el simulador iOS o para un dispositivo conectado. El simulador ofrece un entorno local para probar las aplicaciones para asegurarse de que se comportan básicamente de la manera deseada. Cuando esté pulido el comportamiento básico de la aplicación, se puede dar la instrucción a XCode para construir la aplicación y ejecutarla en un dispositivo basado en iOS conectado a su ordenador. Esto se puede ilustrar en la siguiente figura:

Figura 10. **Ejecutando un proyecto en XCode**

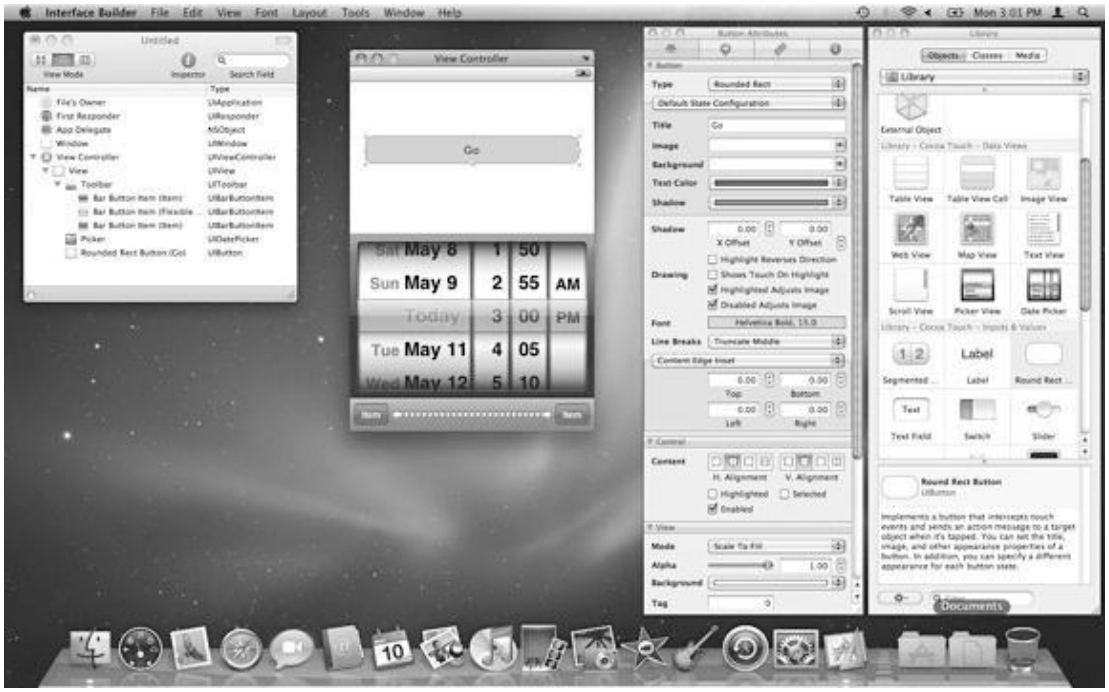


Fuente: http://developer.apple.com/library/ios/#documentation (06-10-2011).

5.5.2. *Interface Builder*

Es la herramienta de uso para ensamblar la interfaz de la aplicación de usuario, usando Interface *Builder* se ensambla la ventana de la aplicación arrastrando componentes pre-configurados en ella, como se muestra en la siguiente figura.

Figura 11. **Construyendo Interfaces iOS usando *interface Builder***



Fuente: http://developer.apple.com/library/ios/#documentation (06-10-2011).

Los componentes incluyen controles estándar del sistema, tales como *switchs*, campos de texto y botones, y también puntos de vista personalizado para representarlos en puntos de vista de aplicación. Una vez colocados los componentes en la superficie de ventana, se pueden posicionar arrastrando a su alrededor, configurar sus atributos, establecer las relaciones entre los objetos y el código. Cuando la interfaz está definida define se guarda el contenido en un archivo, que es un formato de archivo de recursos personalizados.

5.5.3. *Instruments*

Para asegurarse de que se ofrece la mejor experiencia al usuario con el *software* realizado, el entorno de *instrument* le permite analizar el rendimiento

de las aplicaciones iOS mientras se ejecutan en el simulador o dispositivo. *Instrument* reúne los datos de la aplicación en ejecución y presenta los datos en una pantalla gráfica llamada el punto de vista temporal, Puede recopilar datos sobre el uso de memoria de la aplicación, la actividad del disco, la actividad de red y rendimiento gráfico. La vista de línea de tiempo puede mostrar todos los tipos de información, lo que le permite correlacionar el comportamiento general de la aplicación, no solo el comportamiento en un área específica. El punto de vista temporal se ilustra en la siguiente figura.

Figura 12. **Tuneando la aplicación usando Instruments**

Fuente: http://developer.apple.com/library/ios (06-10-2011).

iOS es una plataforma de aplicaciones bastante explotada y estable, lo que le permite posicionarse bien en el ámbito de dispositivos móviles, es una plataforma “cerrada” pero con una gran arquitectura que permite realizar aplicaciones sin necesidad de utilizar las capas más bajas del sistema.

6. PLATAFORMA BLACKBERRY OS

BlackBerry OS es un sistema operativo móvil desarrollado por Research In Motion para sus dispositivos BlackBerry. Dentro de las características del sistema se encuentra la multitarea y soporte para diferentes métodos de entrada, particularmente *touchpad* y pantallas táctiles.

Permite también gestionar accesos a correo electrónico, navegación web y sincronización con programas como Microsoft Exchange aparte de manejar las funciones usuales del teléfono móvil. Aparte de los dispositivos de la propia marca, otras marcas utilizan el cliente de correo electrónico de BlackBerry como: Siemens, HTC, Sony Ericsson. La mayoría de estos dispositivos cuentan con teclado QWERTY completo.

BlackBerry OS está claramente orientado a su uso profesional como gestor de correo electrónico y agenda. Desde la versión actual, se puede sincronizar el dispositivo con el correo electrónico, calendario, tareas, notas y contactos con Microsoft Exchange Server.

Los desarrolladores independientes también pueden crear aplicaciones para BlackBerry pero en el caso de querer tener acceso a ciertas funcionalidades restringidas necesitan ser firmados digitalmente para poder ser asociados a una cuenta de desarrollo de RIM.

Para realizar aplicaciones que corran en dispositivos BlackBerrys se utiliza el entorno de desarrollo Java (capítulo 3), más un conjunto de APIs diseñadas y

optimizadas que en conjunto permite realizar aplicaciones para la plataforma BlackBerry OS.

6.1. Firma de APIs

Lo primero que debemos saber es que *Research In Motion* (RIM) debe realizar un seguimiento del uso de algunas Interfaces de Programas de Aplicación (APIs por sus siglas en ingles) de BlackBerry en el entorno de desarrollo Java (JDE) para un control en materia de seguridad, esto quiere decir, que ellos velan hasta cierto punto de que la aplicación no se use para tomar información del usuario por ejemplificar un caso.

En la documentación de referencia de las API, se muestran que requieren de firma mediante un ícono de candado o simplemente donde se indique según la librería. Si se utiliza las APIs en las clases Java o código de las aplicaciones, estas deben ser firmadas con una clave, proporcionada por RIM antes que se pueda cargar la aplicación con el archivo .cod en el dispositivo. Si se instala una aplicación sin firma el dispositivo dará un error de seguridad. A continuación se muestra la imagen de algunas APIs que necesitan firma:

Figura 13. **API's que necesitan ser firmadas por RIM**

BlackBerry Java Development Environment API reference

To view the BlackBerry Java Development Environment API Reference for version 4.6.0, visit http://www.blackberry.com/developers/docs/4.6.0api/index.html.

New classes in the BlackBerry Java Development Environment version 4.6.0

- net.rim.blackberry.api.messagelist.ApplicationIcon
- net.rim.blackberry.api.messagelist.ApplicationIndicator
- net.rim.blackberry.api.messagelist.ApplicationIndicatorRegistry
- net.rim.blackberry.api.messagelist.ApplicationMessageFolder
- net.rim.blackberry.api.messagelist.ApplicationMessageFolderRegistry
- net.rim.blackberry.api.messagelist.ApplicationMessageSearchProperties
- net.rim.blackberry.api.mms.MMS
- net.rim.blackberry.api.sms.SMS
- net.rim.device.api.lbs.Locator
- net.rim.device.api.lbs.LocatorException
- net.rim.device.api.notification.NotificationProviderRegistry
- net.rim.device.api.system.Sensor
- net.rim.device.api.ui.decor.Background
- net.rim.device.api.ui.decor.BackgroundFactory
- net.rim.device.api.ui.decor.Border
- net.rim.device.api.ui.decor.BorderFactory
- net.rim.device.api.util.LongVector
- net.rim.device.api.util.TimeZoneUtilities

New interfaces in the BlackBerry Java Development Environment version 4.6.0

- net.rim.blackberry.api.messagelist.ApplicationMessage
- net.rim.blackberry.api.messagelist.ApplicationMessage$Status
- net.rim.blackberry.api.messagelist.ApplicationMessageFolderListener
- net.rim.blackberry.api.mms.SendListener
- net.rim.blackberry.api.pdap.BlackBerryEventList
- net.rim.blackberry.api.pdap.BlackBerryToDoList
- net.rim.blackberry.api.pdap.ListChangeListener
- net.rim.blackberry.api.sms.SendListener
- net.rim.device.api.system.SensorListener

Fuente: http://www.blackberry.com/developers/docs/4.6.0api/index.html (06-10-2011).

6.2. Registro de aplicaciones

El registro y firma de las aplicaciones solo será requerido cuando se necesite correr la aplicación en el dispositivo. La firma no es necesaria para correr la aplicación en el simulador. Para poder firmar las aplicaciones necesitamos US$ 20.00 que se paga por tarjeta de crédito luego de rellenar el formulario, este formulario les dará un pin que no deben olvidar ya que se usará para instalar las *Keys* (Llaves), este pin no es el del dispositivo. El sitio donde se registran es: https://www.blackberry.com/SignedKeys/.

Luego de 4 días aproximadamente recibiremos la respuesta de RIM en tres correos, cada uno tendrá un adjunto, que son los siguientes:

- Client-RCR-1560404037.csi
- Client-RRT-1560404037.csi
- Client-RBB-1560404037.csi

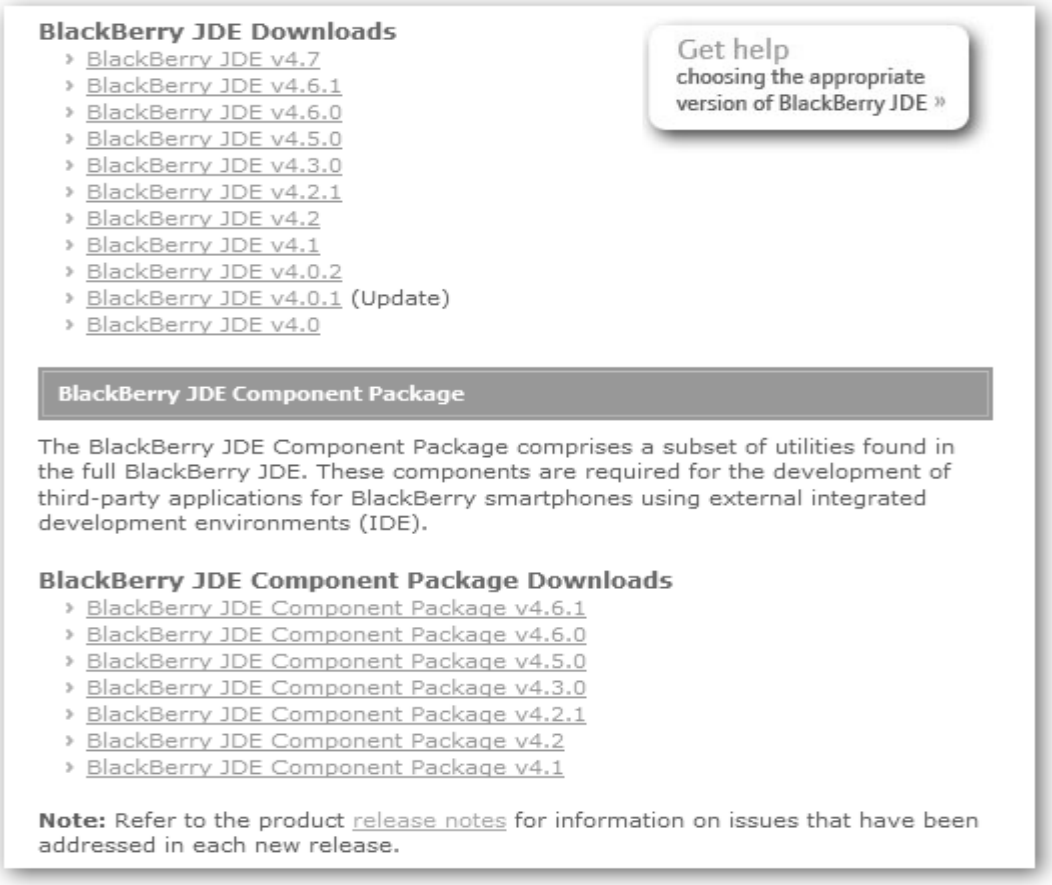
Estos adjuntos debemos descargarlos y ejecutarlos en la computadora donde desarrollaremos la aplicación. Es importante saber que estas firmas son para uso de un computador. En este proceso se les pedirá una contraseña, deben de colocar una para cada llave, y se deben introducir junto con el pin con lo cual no se debe olvidar esta combinación.

6.3. Herramientas de Desarrollo

Dentro de las herramientas para poder desarrollar aplicaciones necesitamos principalmente el JDE (*Java Development Environment*), aunque también se puede desarrollar con el IDE que se prefiera, por ejemplo existe un complemento para JDE en Eclipse. Netbeans, *Visual* C y otros.

A continuación se muestra la imagen y dirección del sitio donde se puede descargar el JDE en sus distintas versiones y otras herramientas para el desarrollo en Java:

Figura 14. **JDE para desarrollar en BlackBerry OS**



Fuente: http://www.blackberry.com/developers/docs/4.6.0api/index.html (06-10-2011).

El JDE principalmente incluye las siguientes herramientas:

- BlackBerry *Integrated Development Environment* (IDE)
- BlackBerry *Smartphone Simulator*.
- Java ME APIs *and* BlackBerry APIs
- Aplicaciones de ejemplo.

El JDE a su vez exige tener algunas otras herramientas para su correcta instalación y uso. Entre ellas están:

- Java SE Development Kit.
- Microsoft Direct X / Open GL.
- En el caso de Windows compatible con Microsoft Windows XP Professional y posteriores.
- Intel Pentium superior a 800Mhz.
- Mínimo 400 MB de RAM.
- Mínimo 500 MB de Disco duro disponible.

Vale la pena mencionar que estas herramientas se deben instalar en el computador a desarrollar, dado que allí tendremos las llaves para firmar aplicaciones.

6.4. Simulador de aplicaciones

Otra de las herramientas que proporciona RIM es el simulador de aplicaciones, que es el entorno donde nos permite depurar las aplicaciones realizadas. Esta herramienta se activa automáticamente al construir y ejecutar la aplicación en desarrollo. En el JDE opción que se encuentra en *Build→Build All* and *Run*, el simulador dependerá dela versión del JDE instalado, por ejemplo la versión 4.60 trae el simulador del Bold, el 4.7 el simulador del Storm y así sucesivamente.

En la figura siguiente se muestra el ejemplo de un simulador versión 4.6.

Figura 15. **Simulador JDE para BlackBerry OS**

Fuente: http://www.blackberry.com/developers/docs/4.6.0api/index.html (06-10-2011).

6.5. Distribuir una aplicación

Luego de probar y depurar los errores en la aplicación procedemos a construir el proyecto para que se generen los archivos .cod, .jar y .jad. Para ello seleccionamos el proyecto a distribuir y luego seleccionamos la opción *Build→Build All* del JDE.

Si la clase requiere firma, debemos de indicarle la llave, mediante la opción "*Requires signing with keys*", esta opción la encontramos en *Build→Request Signatures*…

Una vez seleccionamos esta opción nos aparecerán las clases que necesitan firmas y las que no, en el caso de que requieran firmas se debe solicitar la validación de las credenciales mediante la opción "*Request*", para ello debe de contar con conexión a internet.

Cuando el proceso nos indique debemos de introducir la clave configurada y aceptar para firmar las clases que lo necesiten. La imagen de petición de credencial se muestra a continuación:

Figura 16. **Autenticación de Firmas en Clases para BlackBerry OS**

Fuente: http://www.blackberry.com/developers/docs/4.6.0api/index.html (06-10-2011).

Ahora podemos generar el archivo .alx para la instalación vía Desktop Manager, esto lo realizamos pulsando la opción *Project→Generate ALX File.*

Por último tenemos todos los archivos juntos en la carpeta "com" los podemos copiar para luego subirlos a un servidor para la descarga OTA o para la distribución de la forma que mejor les parezca. Sin olvidar que es necesaria que para la instalación OTA o vía SD se requieren los archivos .jad y .cod, para la instalación vía Desktop Manager se requieren de los archivos .alx y .cod.

6.6. Comparación de plataformas de desarrollo para móviles

Para tener un panorama breve de las diferentes plataformas de desarrollo se presenta a continuación un esquema comparativo entre las plataformas estudiadas anteriormente, estas incluyen a las plataformas: J2ME, *Android*, iOS, BlackBerry OS.

Tabla III. **Cuadro comparativo Plataforma de Desarrollo para móviles**

Descripción	**J2ME**	***Android***	**iOS**	**BlackBerry OS**
Independencia de Sistema Operativo Anfitrión	SI	SI	NO	SI
IDE propietario	*Sun Wireless Tookit*	SDK *Android*	Xcode	No Aplica
Toolkits y Emuladores	SI	SI	SI	SI
Licencia y Distribución de *Apps*	GPLV1	GPLV2	De Pago	De Pago
Multitasking	NO	SI	SI (>iOS 4.0)	SI
Notificaciones Push	NO	SI	SI (>iOS 4.2)	SI
Gráficos	Open GL	Open GL	Quartz	Open GL
Tienda de Aplicaciones	NO	*Android Market*	*App Store*	*App World*
Enfoque de Aplicaciones	Juegos	Juegos, Multimedia, Utilidades, Empresarial	Juegos, Multimedia, Utilidades, Empresarial.	Multimedia, Empresarial, Gestión de Mensajería.

Fuente: elaboración propia.

7. ANÁLISIS DE PLATAFORMAS DE DESARROLLO DE APLICACIONES

La elección de la plataforma de desarrollo para aplicaciones móviles se debe basar en muchos aspectos. Los principales aspectos pueden ser:

- ¿Plataforma Abierta o Cerrada?
- Perfil del Usuario.
- Mercado
- Herramientas de Desarrollo.

7.1. ¿Plataforma abierta o cerrada?

El mercado de los dispositivos con plataforma para aplicaciones es amplio y sigue en crecimiento. De hecho las tres plataformas estudiadas no representan el 100% de las opciones. Cabe mencionar a Symbian todo un símbolo en el pasado inmediato, y de las opciones que vendrán dentro de poco MeeGO y Windows Phone 7, que seguramente tendrán una parte de este mercado.

Sin embargo puede decirse que las tres plataformas elegidas son los sistemas móviles más significativos en la actualidad.

iOS es un sistema cerrado exclusivo de *Apple*. Esto significa que, salvo excepciones, las personalizaciones, modificaciones, mejoras, parches y actualizaciones solo pueden ser emitidos por la empresa responsable.

Si se desea realizar una aplicación para iOS *Apple* cuenta con una buena documentación así como los pasos puntuales a seguir para desarrollar aplicaciones.

Android en cambio, pese a ser propiedad de Google, es un sistema abierto en cuyo desarrollo participan varias empresas de *hardware* e incluso varias operadoras de telecomunicaciones y una activa comunidad de usuarios.

Descrito así, *Android* y J2ME parecen tener una ventaja sobre iOS. La discusión al respecto es amplia y continua.

Android es una plataforma abierta que se ha fragmentado, el hecho que cada modelo de teléfono, cada operador de telefonía, tenga una versión del sistema operativo, que además puede personalizarse y retocarse siguiendo diversos protocolos, puede resultar perjudicial para el desarrollo de aplicaciones dado que puede funcionar para algunas versiones de teléfono y en otras no. Por ejemplo: Samsung Galaxy S es un modelo de Samsung que implementa *Android*, la ROM y modificaciones del sistema es diferente en Telgua, Movistar, Tigo en caso de Guatemala, seguramente un operador actualizara la versión del sistema antes que otro.

Por otra parte, el carácter abierto permite una gran variedad de *hardware*, varias marcas de teléfonos inteligentes incrustan *Android* por ejemplo LG, HTC, Samsung, Motorola, mientras que los sistemas cerrados solo disponen de una marca.

Una plataforma cerrada como iOS es más difícil ser desarrollador autorizado y los pasos son muy estrictos y costosos, cambio con *Android* o J2ME es fácil y accesible para cualquier desarrollador, iOS como sistema

cerrado cuenta con un sistema para sus líneas de dispositivos, mientras que *Android* no puede controlar la homogeneidad en su sistema, dado que las empresas lo personalizan y añaden funciones propietarias.

En cuanto ha cerrado o abierto, se prefiere un sistema abierto, pero actualmente la ventaja la tienen los sistemas cerrados dado al control de las aplicaciones y la garantía de éstas en sus tiendas de distribución.

7.2. Perfil del usuario

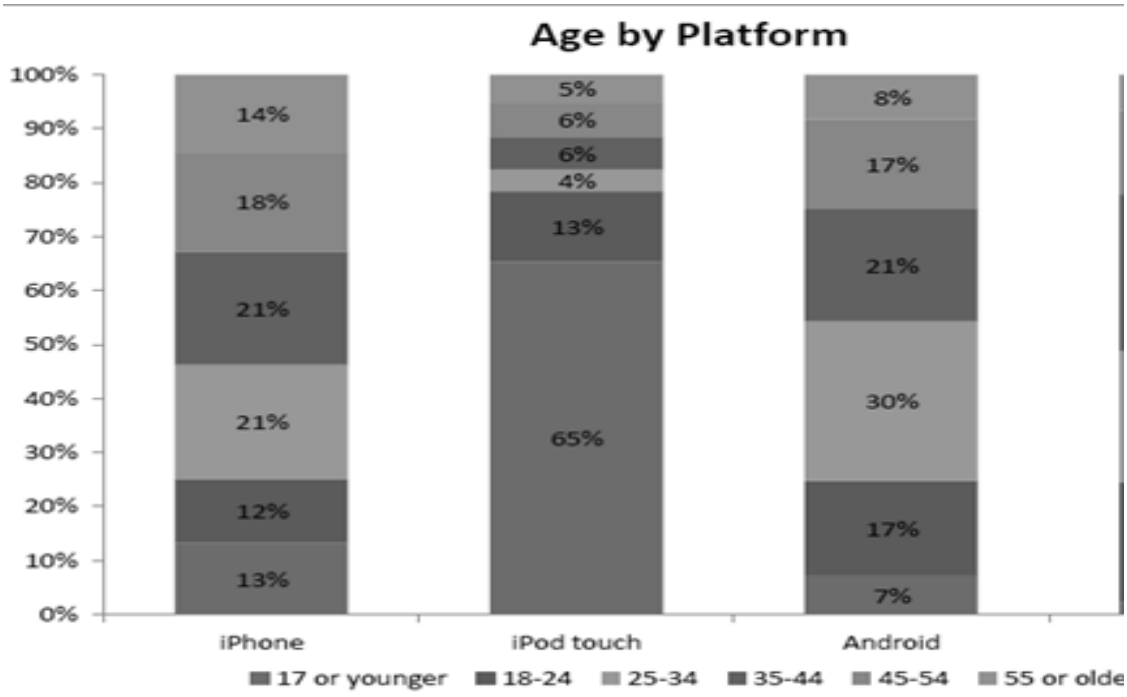
Sería interesante saber cómo los usuarios se ven a sí mismos para poder segmentar de forma más clara los perfiles de los usuarios de las plataformas.

Actualmente existen usuarios que toman al *Smartphone* como herramienta de trabajo. A otras que lo usan únicamente para multimedia y juegos, otros simplemente por servicios de mensajería y servicios de voz.

Las plataformas actuales están diseñadas para todo tipo de aplicaciones, distintos perfiles, es la aplicación a desarrollar que se enfocara a un sector en específico, como también las tiendas virtuales categorizaran la aplicación, por ejemplo: Apps para adultos, jóvenes, todo público. Obviamente, encontramos usuarios de *Android*, iOS, BlackBerry OS en todos los sectores, desde el corporativo hasta los jóvenes estudiantes que aprecian la rapidez de manejo de la herramienta para resolver ciertas tareas y mensajería.

También es válido mencionar las estadísticas muy interesantes realizadas por AdMob para dispositivos móviles correspondientes al año 2010. Las cuales se desglosan de la siguiente forma:

Figura 17. **Rango de edades para uso de plataformas móviles**

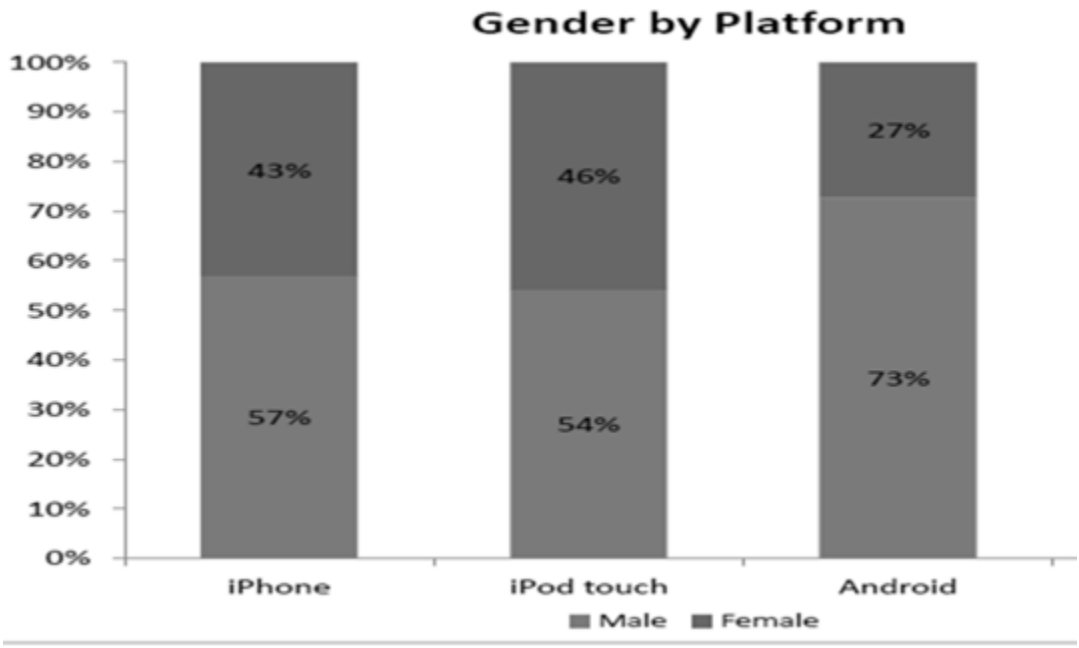


Fuente: http://www.admob.com/ (06-10-2011).

- El 30% de los usuarios de *Android* está entre 25-34 Años, con respecto a iOS están 21% para el mismo rango de edad.
- El 30% de los usuarios de *Android* está entre 25-34 Años, con respecto a iOS están 21% para el mismo rango de edad.
- Los jóvenes prefieren iOS en un 13% y un 7% prefiere *Android*.

Otra gráfica muy interesante es la que muestra las tendencias de hombres y mujeres sobre las plataformas analizadas, como se ve en la siguiente figura.

Figura 18. **Rango de edades para uso de plataformas móviles**



Fuente: http://www.admob.com/ (06-10-2011).

- El 73% de hombres prefiere *Android* mientras que el 57% prefiere iPhone y un 54% prefiere iPod Touch.
- Los iPod touch son populares entre los consumidores que ya utilizan teléfonos inteligentes. 1 de cada 4 usuarios de iPhone en la actualidad posee o tiene intención de comprar un iPod *touch* los próximos meses.

En la actualidad es posible que esta tendencia se mantenga, sin embargo este breve análisis sirve para poder enfocar la aplicación al sector objetivo, el cual finalmente será quien utilizará el producto.

7.3. Mercado

Anteriormente se analizaron 2 formas de enfocar las aplicaciones, pero si el objetivo es monetizar la idea a desarrollar es importante saber en qué forma lo podemos hacer y qué beneficios traen las plataformas para realizar una aplicación.

Los dispositivos móviles estén ocupando el mercado es una afirmación que actualmente no sorprende, tan solo hay que estar atentos a tanta publicidad que generan las empresas de telecomunicaciones (ejemplo: Claro, Tigo, Movistar en Guatemala). En este mercado competitivo de plataforma de aplicaciones destacan *Android*, iOS, BlackBerry OS.

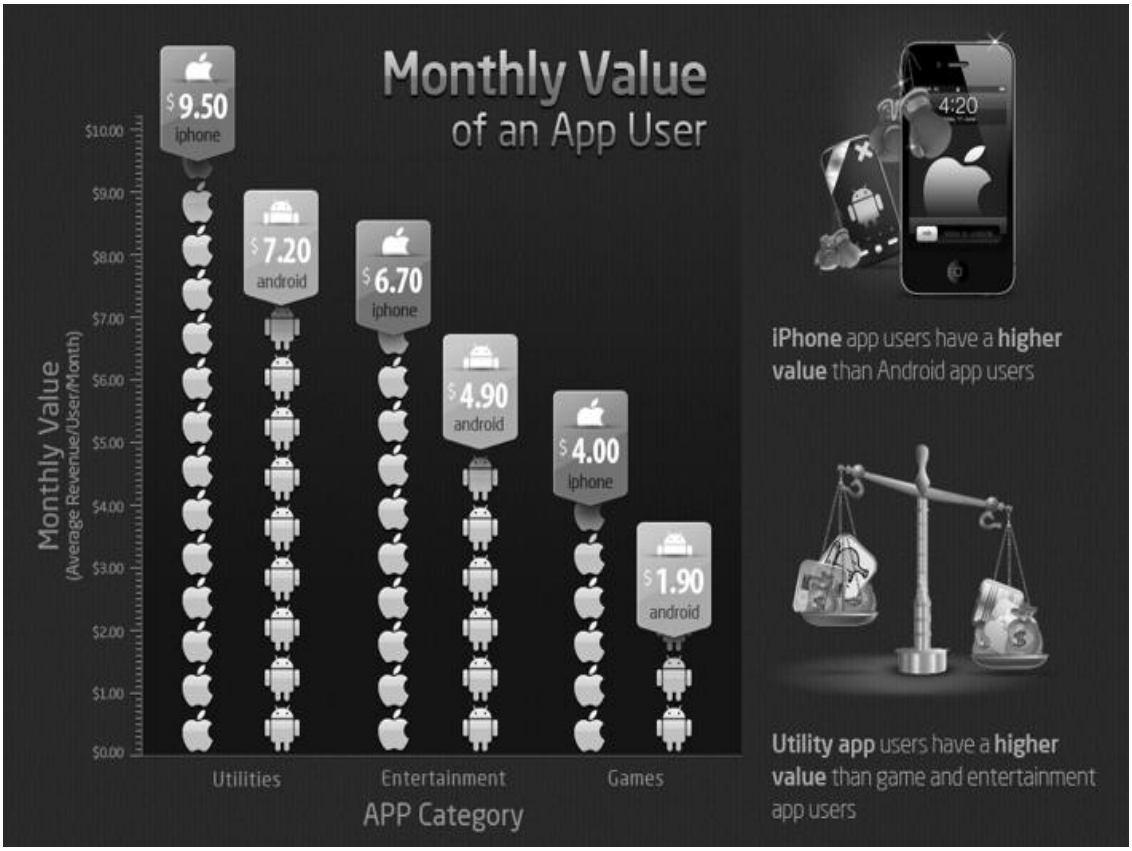
En un análisis realizado por la empresa Mobclix , mediante un estudio comparativo, se han examinado dos de las plataformas más fuertes del mercado *Android* vs iOS. Para ello se han centrado exclusivamente en las aplicaciones gratuitas que se mantienen gracias a la inclusión de publicidad dentro de la aplicación.

Tras consultas las aplicaciones con más de 500,000 descargas de las tiendas de aplicaciones de las dos plataformas, y en las que se han estado más de dos semanas dentro del top 10 englobadas en 3 categorías: utilidades, entretenimiento, juegos, se ha llegado a la conclusión de que el usuario de iOS es más rentable para el desarrollador de este tipo de aplicaciones que el usuario de *Android*.

Estos datos no deben tomarse como absoluto, sin embargo podemos ver la marcada tendencia de los usuarios por utilizar iOS entre la diversidad de plataforma, también es cierto que *Android* es una fuerte competencia, en la

medida en que este sistema madure y encuentre la estabilidad en la fragmentación le hará cara a los movimientos de iOS, a continuación se muestra la ilustración de este análisis.

Figura 19. **Rango de edades para uso de plataformas móviles**



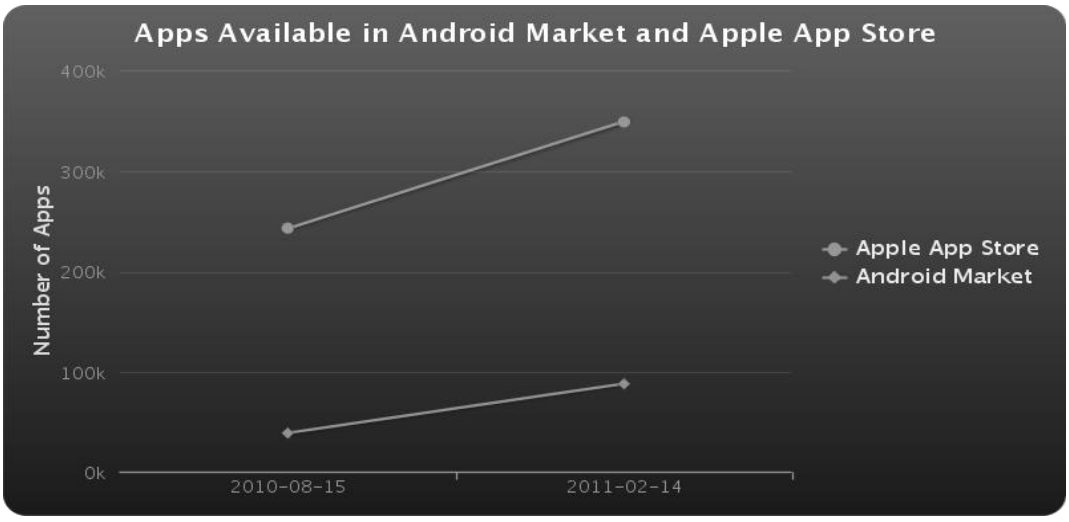
Fuente: http://blog.mobclix.com/ (06-10-2011).

Los resultados son bastante contundentes ya que como se muestra en la figura anterior, los usuarios iOS son más rentables para quienes desarrollan aplicaciones que se mantienen con publicidad incrustada en ellas en las tres categorías, además en algunas categorías como juegos, la diferencia de ingresos medios generados por los usuarios de *Android* e iOS en aplicación gratuitas es el doble.

Los resultados anteriores parecerían que iOS es la plataforma a seleccionar para poder rentabilizar el esfuerzo, sin embargo no es así, dado que *Android* es muy fuerte, y más cuando se lee la gran variedad de empresas que están adoptando esta plataforma de aplicaciones, tanto que apartir del 2009 viene creciendo de forma rápida.

El caso es que *Android* entra con fuerza en el mercado en todos los aspectos incluyendo el de las aplicaciones, en el que *Apple*, con la App *Store* domina el mercado, bueno, más bien por ahora, porque según un estudio de My Look Out , una compañía especializada en seguridad para móviles, el mercado de las aplicaciones para móviles podría dar vuelta a mediados de 2012, momento en el que existirían más aplicaciones para *Android* que para iOS. Esto se ilustra en la siguiente figura.

Figura 20. **Aplicaciones disponibles en *Android* vs *App Store* 2012**



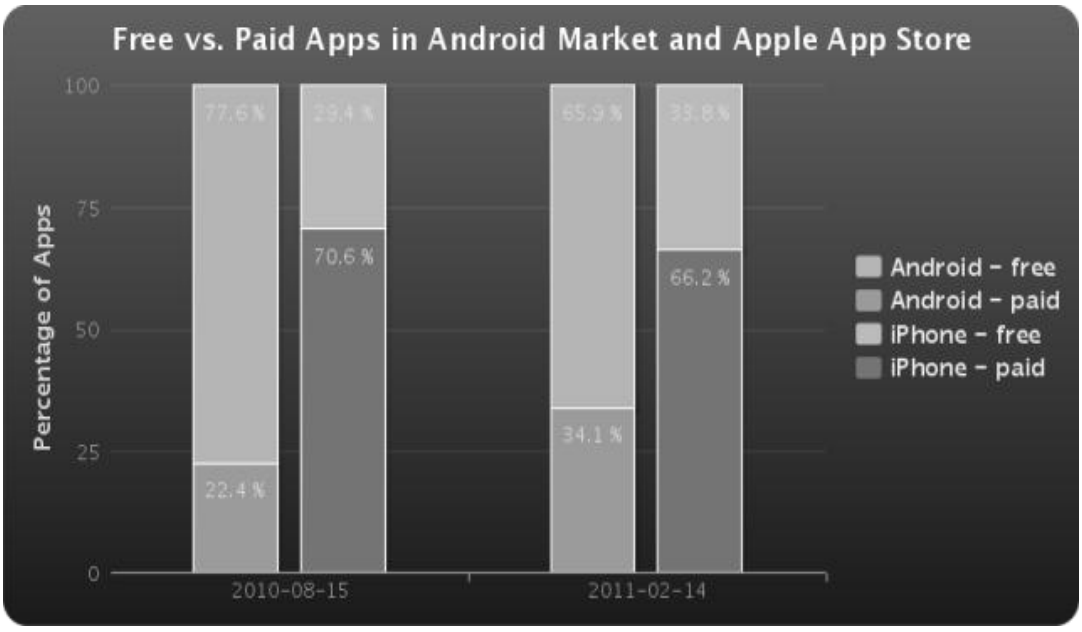
Fuente: http://www.mylookout.com/appgenome (06-10-2011).

La afirmación no deja ser una proyección de los datos actuales, es decir, una predicción pero que, no por ello deja de ser interesante. Esta predicción se

basa en dos datos relativos al crecimiento del número de aplicaciones disponibles en ambas tiendas, por un lado desde agosto de 2010 hasta mediados del mes de febrero 2011, la tasa de crecimiento de las aplicaciones disponibles en *Android* Market es del 127% sin embargo, la del catálogo de *Apple* tan solo ha crecido, en el mismo período un 44%, tasas que son las que sostienen la proyección relativo al año 2012.

Por tanto, según estos datos, aunque la cantidad de aplicaciones que se generan en *Android* es mayor que en iOS, los desarrolladores prefieren este último, ¿Por qué?, pues parece que la cuestión es económica, ya que en la *App Store* tan solo un 34% de las aplicaciones son gratuitas frente al 66% del *Android Market*. Esto se ilustra en la siguiente figura:

Figura 21. **Aplicación de pago y gratis para *Android* & iOS**



Fuente: http://www.mylookout.com/appgenome#platform-wars (06-10-2011).

Según parece, aunque desarrollar para *Android* es más rápido, sobre todo teniendo en cuenta las condiciones que impone *Apple* a los desarrolladores, desarrollar para iOS es un negocio lucrativo, donde los precios de las aplicaciones son mayores que en *Android*.

¿Es cuestión de cantidad o calidad? Con respecto al sistema *Android* tan solo hay que ver sus cifras de crecimiento y la rápida expansión en los dispositivos con este sistema operativo y, precisamente, ha sido esta rápida expansión la que ha disparado el crecimiento en el número de aplicaciones disponibles a los que habría que sumar una mayor libertad en el desarrollo de aplicaciones; y quizás sea eso lo que ha disparado aún más la productividad de los desarrolladores: un control menos estricto.

Sobre la predicción de mediados de 2012, la verdad es que no es algo certero, toda tasa de crecimiento acaba desacelerando hasta llegar a estabilizarse, por tanto, es posible que el número de aplicaciones para *Android* supere al de iOS quizás no tan pronto.

Sin hacer de menos a BlackBerry OS que su ritmo podrá ser menos vertiginoso que *Android* e iOS, pero sus usuarios corporativos, así como sus protocolos de mensajería hacen esta una plataforma bastante interesante para desarrollar y monetizar, sin tomar en cuenta la calidad y estable del *software* de esta plataforma.

7.4. Herramientas de desarrollo

A continuación se desglosa las partes y estructuras de las plataformas investigadas en los capítulos anteriores.

7.4.1. Plataforma J2ME.

Tabla IV. **Especificaciones y requerimientos para J2ME**

Especificación de Recursos de *Software*	
Sistema Operativo (32 y 64 bits)	• Windows XP, Vista, o Windows7 • Linux x86, x64 • Solaris • Mac OS X
SDK	• SDK de Java 2, *Standard Edition* (J2SE SDK), versión 1.4.2 o posteriores.
Entorno de Desarrollo Integrado	• Netbeans 5.0 o posteriores. Se necesitan por lo menos 200Mb de espacio disponible. • Eclipse Mobile Pulsar.
Especificación de Recursos *Hardware*	• 50 Mb de Disco Duro • 64 Mb de Memoria RAM • 166 Mhz CPU • Tarjeta de Sonido • Instalar J2ME *Wireless Toolkit.*
Especificaciones de los Dispositivos.	
Connected Limited Device Configuration (CLDC)	• Corre sobre la KVM • 160 Kb a 512 Kb de memoria total disponible para el entorno de java. • Procesador de 16 o 32 bits. • Soportar conectividad a un tipo de red.

Continuación Tabla IV

MIDP	• Pantalla de al menos 96x54 pixeles. • 128Kb de memoria no volátil para correr componentes MID. • Al menos 8Kb de memoria no volátil para almacenar datos persistentes de las aplicaciones. • 32 Kb de memoria volátil para correr Java. • Conectividad inalámbrica a redes.
Lenguaje de Programación	• Medio, si se posee fundamentos de C++, o principios de programación orientada a objetos.
Costo	• Sin Costo.

Fuente: elaboración propia.

7.4.2. Plataforma *Android*

Tabla V. **Especificaciones y requerimientos para *Android***

Requisitos de *Software*	• Windows XP (32 bits) o Vista(32-64 bits), Windows 7 x86 & x64 • Mac OS X 10.4.8 o posteriores (x86) • Linux (Probado con Linux Ubuntu Hardy Heron).
IDE Soportados	• Eclipse IDE 3.4 (Ganymede). • Complemento JDT Eclipse • *Android Development Tools Plugin*.

Continuación Tabla V

Requisitos Mínimos de *Hardware*.	• Para los paquetes bases del SDK se necesitan 600MB de espacio en disco disponible. • Para cada plataforma descargada en el SDK un espacio adicional de 100MB.
Requisitos Mínimos del Dispositivo.	• 32 MB de RAM • 32 mn de memoria Flash • Procesador de 200 Mhz Online
Costo	• Sin Costo.

Fuente: elaboración propia.

7.4.3. Plataforma iOS

Tabla VII. **Especificaciones y requerimientos para iOS**

Requisitos de *Software*	Mac Os X *Leopard* 10.6.4
IDE Soportados	XCodeVersion 4.1
Plataforma	Intel
Costo	• $99.00 por suscripción a *Developer Center*. • Certificado para desarrollo. • Certificado para distribución.

Fuente: elaboración propia.

7.4.4. Plataforma BlackBerry OS

Tabla VIIII. **Especificaciones y requerimientos para BlackBerry OS**

Requisitos de *Software*	• Windows XP (32 bits) o Vista(32-64 bits), Windows 7 x86 & x64 • Mac OS X 10.4.8 o posteriores (x86) Linux (Probado con Linux Ubuntu Hardy Heron).
IDE Soportados	Netbeans 5.0, Eclipse.
Plataforma	Intel
Costo	• $20.00 por suscripción a *Developer* Center. • Certificado para desarrollo. • Certificado para distribución.

Fuente: elaboración propia.

Cada plataforma proporciona gran documentación y recursos para poder desarrollar, algunos son de pago y otros gratis, cada plataforma también proporciona la forma de distribuir la aplicación (Ejemplo: *Android* Market, *App Store*, *App World*), en esta sección es fácil decidirse por *Android* o J2ME, sin embargo es de contemplar otros aspectos como los mencionados anteriormente, lo que si es cierto que muchos desarrolladores trabajan mejor en la plataforma que se sientan más cómoda y que ofrezca mayor fluidez en los ciclos de desarrollo.

CONCLUSIONES

1. Las tecnologías móviles se desarrollan a pasos agigantados. Hace poco era difícil encontrar dispositivos que realizaran tareas complejas como un organizador personal, multimedia, navegación web.

2. Dado al aumento de las capacidades de los móviles como desarrolladores, se tiene una gran oportunidad al incursionar al medio de desarrollo de aplicaciones para móviles, dado que constituyen gran rentabilidad para las empresas de desarrollo de *software*, ya que plantea un modelo de negocios e intercambio digital abierto para distribuir las aplicaciones realizadas.

3. Las plataformas de Desarrollo constituyen la interfaz entre la idea del desarrollador y el usuario del dispositivo. Estas plataformas están diseñadas para explotar las capacidades de los dispositivos electrónicos, es por ello que cambian a un ritmo acelerado al mismo tiempo que se logra aumentar las capacidades de los móviles.

4. Se aprecia que cada plataforma de desarrollo tiene sus propias características, su arquitectura, e interfaces que permiten realizar el proceso de programación de forma cómoda. Cada plataforma tiene características distintas, algunas tienen más ventajas, por ejemplo: documentación, recursos digitales como tutoriales, portales web.

5. Para preparar una buena aplicación se necesita de una buena idea, tomar en cuenta el sector para el que se diseñará, la monetización que se quiere

obtener y la elección de la plataforma de desarrollo para transformar la solución de la aplicación a los dispositivos que los usuarios puedan utilizar de forma fácil y sencilla.

6. Las estadísticas proporcionan una idea del sector al cual puede dirigirse la aplicación, los recursos que estas brindan y la facilidad de manejar los ciclos de desarrollo, son de gran ayuda para tomar la decisión de programar en una plataforma específica.

7. Los aspectos generales que desarrollan aplicaciones para plataformas móviles, muestran las ventajas y desventajas de cada uno. Los enfoques que pueden tomar los desarrollos así como la forma de monetizar que plataforma es la mejor.

8. Llegar a este punto y decidirse por una plataforma en específico y no dar oportunidad a otra es arriesgado pues ambas a pesar que son fuertes no tienen el mismo enfoque. El punto importante está expresado en el capítulo dos “La Idea”, esto sirve de base para decidirse que plataforma usar, los aspectos de desarrollo, diseño y pruebas no son tan críticos como lo puede ser el sector del mercado que se distribuirá, la monetización que se desee obtener. Si bien es cierto que iOS genera ganancias no asegura que sea la más rentable, pues hay aplicaciones exitosas en *Android* y *BlackBerry* OS que superan a muchas de iOS (ejemplo: AngryBirds y BB *Chat*). También existen aplicaciones gratis en ambas plataformas con publicidad incrustada como elemento externo de monetización.

9. El criterio por gusto, la comodidad, el conocimiento, la seguridad que se tenga en cierta plataforma, puede ser tema de discusión. Anteriormente

en *Android*, tanto las políticas de restricción, la forma de distribución era compleja, pero con la llegada del *Android Market*, la distribución es sencilla, llegando a ser un paso trivial y de valor para el desarrollo. Tampoco se descarta la posibilidad de desarrollar para iOS, porque esta plataforma ofrece mayor rango de monetización. Queda la pregunta abierta, ¿en qué plataforma se desarrolla?

RECOMENDACIONES

1. El tener definida la idea de la aplicación a realizar, se debe estimar el tiempo en que tarda implementar esta idea con la plataforma; luego tomar el tiempo para el área de pruebas en los dispositivos reales, dado que los simuladores vienen configurados con opciones que muchos dispositivos reales aún no tienen.

2. Al elegir una plataforma se debe de considerar si se conoce el lenguaje de programación previamente, esto facilitará en gran medida el tiempo de desarrollo y la curva de aprendizaje será rápida.

3. Contar con un equipo *hardware-software* disponible para realizar el proceso de desarrollo.

4. Apoyarse de las herramientas de análisis y diseño como el caso de los diagramas de modelado UML y hacer el diseño previo con diagramas.

5. En la configuración de los dispositivos reales, verificar que cumplan con las dependencias que pide la plataforma, que sean de la misma versión o que soporta el entorno de ejecución de las aplicaciones.

6. Verificar que las versiones de las herramientas de desarrollo sean compatibles con los ambientes que se necesita. Siempre es recomendable ver las especificaciones del dispositivo y los requisitos mínimos que cumplan para la ejecución de la aplicación.

7. Los recursos digitales disponibles en la web, permiten comprender los entornos de desarrollo y mejores prácticas para hacer *software*.

BIBLIOGRAFÍA

1. *APPLE* INC. Library iOS *and* Navigation. [en línea] enero de 2011. [ref. de 12 de febrero de 2010]. Disponible en Web: <http://developer.apple.com/library/ios/navigation/index.html>.

2. *BLACKBERRY*. Documentacion oficial de API's BlackBerry OS. [en línea]. 25 de mayo de 2010. [ref. de 12 de enero 2011]. Disponible en Web: < http://www.blackberry.com/developers/docs/4.6.0api/index.html >

3. CANCELA JAVIER, Introducción al desarrollo de aplicaciones para telefonos moviles. [ref. de 08 de noviembre 2010]. Disponible en Web: <http://javiercancela.com/2007/10/17/introduccion-al-desarrollo-de-aplicaciones-para-telefonos-moviles-symbian/.>

4. *DEVELOPER ANDROID*. Administrador de Notificaciones para *Android*. [en línea]. 29 de diciembre de 2006 [ref. de 02 de diciembre de 2010]. Disponible en Web: <http://developer.android.com/reference/android/app/NotificationManager.html>.

5. _____. Administrador de Recursos para *Android*. [en línea]. 29 de diciembre de 2006 [ref. de 02 de diciembre de 2010] . Disponible en Web: <http://developer.android.com/guide/topics/resources/resources-i18n.html>.

6. _____. Proveedor de contenidos para *Android*. [en línea]. 29 de diciembre de 2006 [ref. de 02 de diciembre de 2010] . Disponible en Web: <http://developer.android.com/guide/topics/providers/content-providers.html>.

7. _____. Vistas para apps *Android*. [en línea]. 29 de diciembre de 2006 [ref. de 02 de diciembre de 2010] . Disponible en Web: <http://developer.android.com/guide/tutorials/views/index.html>.

8. GRASIA. Arquitectura J2ME. [en línea]. Abril de 2009 [ref. de 02 de octubre de 2010]. Disponible en Web: <http://grasia.fdi.ucm.es/j2me/_J2METech/CLDC.html>.

9. _____. *Datasheet Java 2 Platform, Micro Edition Connected Device Configuration* (CDC). [en línea]. Abril de 2009 [ref. de 02 de octubre de 2010]. Disponible en Web: <http://grasia.fdi.ucm.es/j2me/docs/j2me_cdc.pdf>.

10. _____. *Documentation Connected Limited Device Configuration, Specification Version 1.0a.* [en línea]. Abril de 2009 [ref. de 02 de octubre de 2010]. Disponible en Web: <http://grasia.fdi.ucm.es/j2me/docs/CLDCSpecification1.1.pdf.>.

11. _____. *Documentation Connected Limited Device Configuration, Specification Version* 1.1. [en línea]. Abril de 2009 [ref. de 02 de octubre de 2010]. Disponible en Web: <http://grasia.fdi.ucm.es/j2me/docs/CLDCspec10a.pdf> .

12. KRONOX. Historia de *Android*. [en línea]. 08 de abril de 2008 [ref. de 12 de septiembre de 2010]. Disponible en Web: <http://kronox.org/2009/06/09/historia-de-android/>.

13. MAILXMAIL. Técnicas de Desarrollo de computación móvil. [en línea]. 08 de abril de 2008 [ref. de 12 de septiembre de 2010]. Disponible en Web: <http://www.mailxmail.com/curso-tecnicas-desarrollo-computacion-movil-orientado-pda/procesos-desarrollo-aplicaciones-moviles>.

14. MSDN. Windows Mobile. [en línea]. 08 de abril de 2008 [ref. de 12 de septiembre de 2010]. Disponible en Web: <http://developer.windowsmobile.com/>.

15. Oracle. Java 2 Micro *Edition*. Mobile. [en línea]. 08 de abril de 2008 [ref. de 12 de septiembre de 2010]. Disponible en Web: <http://java.sun.com/javame/index.jsp>.

16. PARDO KUKLINSKY HUGO. "Planeta Web 2.0 Inteligencia Colectiva o medios *FastFood* ". Computer - Septiembre, 2007.

17. ZONA BLACKBERRY. Ejemplos de desarrollo de aplicaciones para Blackberry OS. [en línea]. 08 de abril de 2008 [ref. de 12 de septiembre de 2010]. Disponible en Web: < http://zonablackberry.com.ve/forum/desarrollo-de-aplicaciones-y-temas-para-blackberry/8441-guia-basica-para-el-desarrollo-de-aplicaciones-blackberry-con-imagenes.html>.

18. _____. Servicios de descarga para blackberry OS. [en línea]. 08 de abril de 2008 [ref. de 12 de septiembre de 2010].

Disponible en Web: < http://zonablackberry.com.ve/forum/aplicaciones-originales-y-de-terceros/326-descargador-de-archivos-ota-para-blackberry-ota-downloader-gratis.html>
.