



**DOCUMENTO DE INVESTIGACION SOBRE EL ANALISIS DE LAS CARACTERISTICAS
DE LOS DISPOSITIVOS MOVILES INTELIGENTES (SMART PHONES)**



SMART PHONES

Diciembre 2008

© Copyright Diciembre 2008 por Belatrix Software Factory BSF S.A.

www.belatrixsf.com

businessdevelopment@belatrixsf.com

+1 (617) 608-1413

Índice

1. Introducción	3
1.1 Acerca de Belatrix Software Factory	3
2. Cuotas de Mercado	3
2.1 Cuota de mercado respecto a las ventas	4
2.1 Comportamiento del mercado de EEUU en el 2008	5
3. Sistemas operativos para dispositivos móviles	5
3.1 Symbian	6
3.1.1 Desarrollando en Symbian OS	6
3.2 Mac OS (iPhone)	7
3.2.1 Desarrollando para iPhone	9
3.3 RIM (Research in motion)	11
3.3.1 Desarrollando para RIM	12
3.4 Windows Mobile	12
3.4.1 Desarrollando para Windows Mobile	12
3.5 Linux	13
3.5.1 Desarrollando para Android	15
4. Conclusión	18

1. Introducción

Belatrix Software Factory como empresa líder en tecnologías de software viene desarrollando un profundo y extenso conocimiento sobre plataformas móviles por lo que ha encarado un proceso de investigación y desarrollo sobre estas tecnologías para ampliar aún más ésta área en la empresa.

El presente documento contiene una investigación y un análisis sobre los principales sistemas operativos que se utilizan en los dispositivos móviles inteligentes (SmartPhones), indicando sus cuotas de mercado correspondientes. También se incluyen las características técnicas requeridas y los conocimientos (skills) para lograr desarrollos de aplicaciones que sean soportadas por los SmartPhones con los distintos SDK (Kits de desarrollo de software) disponibles.

¿Qué es un Smartphone?:

Un Smartphone o dispositivo móvil inteligente, es un teléfono celular con prestaciones superiores a las típicas, a menudo comparadas con las prestaciones de una PC. Sin embargo no hay un acuerdo entre los fabricantes acerca de esta definición, para algunos se trata de un teléfono que corre un sistema operativo completo e identificable, que provee una interface estándar y una plataforma para desarrollo de aplicaciones. Para otros es simplemente un teléfono móvil con funcionalidades avanzadas como: e-mail, Internet y/o un teclado integrado. A los efectos de este documento se tendrá en cuenta la primera definición, ya que nos interesa conocer los sistemas operativos de cada plataforma y los procesos de desarrollo y publicación de software para cada una.

1.1 Acerca de Belatrix Software Factory

Belatrix Software Factory es una compañía argentina dedicada al outsourcing o tercerización de servicios de desarrollo de software, fundada en el año 1.993 y completamente orientada desde el año 2.001 a la provisión de servicios de tercerización internacional a clientes de diversos países tales como Estados Unidos, Canada y varios países de Europa y Escandinavia.

Los servicios que presta son:

- Desarrollo de software a medida.
- Testing y aseguramiento de la calidad de software.
- Diseño gráfico.
- Soporte remoto de sistemas.

Para contactar a Belatrix Software Factory:

e-mail: businessdevelopment@belatrixsf.com

Teléfono: +1 (617) 608-1413, interno 600 para el área comercial.

Web: www.belatrixsf.com

2. Cuotas de Mercado ¹

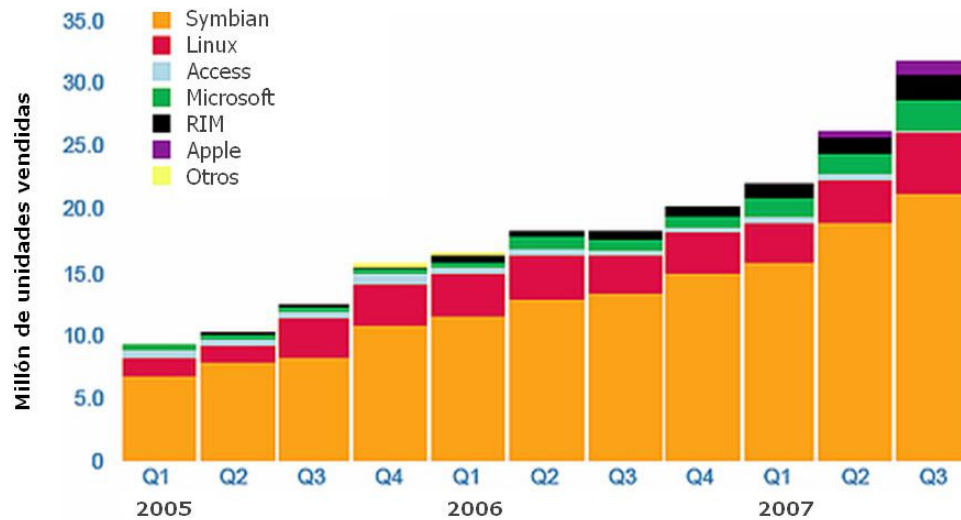
El mercado de los sistemas operativos para smartphones mundial es dominado por Symbian que dispone de una cuota de mercado del 57,1%, RIM (BlackBerry) es quien le sigue con una

¹ Datos a noviembre de 2008. Principales Fuentes:
Wikipedia. (www.wikipedia.com)
Reuters (<http://www.reuters.com/article/technologyNews/idUSTRE4A58OK20081106>).

cuota del 17.4% y Windows Mobile es el tercero con el 12% del mercado mundial. Los sistemas basados en Linux tienen el 7.3% de cuota de mercado, aunque este número es la suma de una serie bastante numerosa de distintas plataformas con grandes diferencias entre sí, que utilizan mismo sistema operativo para el kernel. El sistema operativo de Apple, Mac OS, dispone a nivel mundial de un 2.8% de cuota de mercado aunque es de destacar que casi todas las ventas de iPhones (único smartphone con MacOS) se han realizado en Estados Unidos, donde se ubica en el segundo puesto de ventas siendo superado solo por RIM.

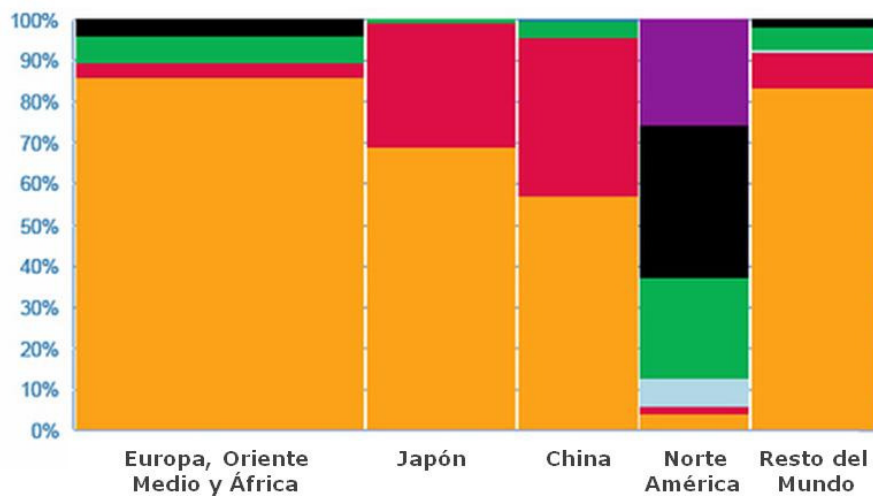
Open Handset Alliance's Android es un reciente sistema operativo para smartphones desarrollado por Google y T-Mobile, que debutó en el mercado con el HTC G1 el 22 de Octubre de 2008 y ha ganado el 4% del mercado de ventas en Estado Unidos en menos de un cuatrimestre. Es de destacar que el HTC G1 se vende como "versión para desarrolladores" lo que denota la falta de madurez del sistema operativo.

2.1 Cuota de mercado respecto a las ventas



El mercado de móviles de EEUU tiene la particularidad de tener una distribución totalmente diferente a la del resto del mundo, lo que puede verse en el siguiente gráfico (extraído de una publicación de symbian), otra particularidad es el mercado de Japón y China donde es importante la cantidad de móviles con sistemas operativos basados en distribuciones de Linux, sistema operativo que tiene muy poca presencia en el resto del mercado mundial:

Q3=Ventas de Smart Phones por vendedor de sistema operativo y por región



La aparición del iPhone 3G, entre el 2do y 3er cuatrimestre del 2007, impulso las ventas de este smartphone en los Estados Unidos en forma sorprendente, en parte debido al respaldo de una fuerte campaña publicitaria y al uso intensivo de la pantalla Multi Touch.

2.1 Comportamiento del mercado de EEUU en el 2008

El dispositivo BlackBerry, que fabrica Research in Motion Ltd, aumentó su participación en el mercado estadounidense de teléfonos multiuso durante el primer trimestre, mientras que el iPhone de Apple Inc perdió terreno, según un reporte de International Data Corporation (IDC).

Según el informe, la participación de RIM en el mercado estadounidense de teléfonos avanzados (smartphones), creció a un 44,5 por ciento en el primer trimestre, desde un 35,1 por ciento del cuarto trimestre año pasado.

En tanto, la cuota de mercado del iPhone de Apple retrocedió a un 19,2 por ciento desde un 26,7 por ciento del cuarto trimestre.

Motorola Inc, el principal fabricante de teléfonos en Estados Unidos, también vio disminuir sus ventas a un 2,6 por ciento en el mercado de teléfonos multiuso, desde un 7,5 por ciento en el cuarto trimestre.

Palm Inc. vio avanzar su participación a un 13,4 por ciento desde un 7,9 en el cuarto trimestre. Mientras, la de Samsung Electronics Co Ltd's subió a un 8,6 por ciento, desde un 5,1 por ciento, indicó IDC. La participación de High Tech Computer Corp's (principal fabricante de móviles con Windows Mobile) bajó a un 4,1 por ciento desde un 7,9 por ciento en el trimestre anterior y finalmente Android hizo su aparición en el mercado de EEUU con un dispositivo que se vende como "Versión para desarrolladores" ganando un 4% de las ventas del último trimestre de este año.

3. Sistemas operativos para dispositivos móviles

A continuación, se detallan las características más relevantes de los principales sistemas operativos para móviles:

3.1 Symbian

Es un sistema operativo propietario diseñado para teléfonos móviles, con librerías asociadas e interface de usuario. Desciende de Psion EPOC y corre exclusivamente en procesadores ARM, es producido por Symbian Ltd, sociedad formada por Nokia (47.9%), Ericsson (15.6%), Sony Ericsson (13.1%), Panasonic (10.5%), Siemens AG (8.4%) y Samsung (4.5%).

Es un Sistema Operativo basado en ROM que ha sido diseñado para ahorrar batería. Symbian está basado en un micro kernel, una mínima porción del sistema tiene privilegios de kernel, el resto se ejecuta con privilegios de usuario y es tarea del kernel manejar las interrupciones y prioridades. En Symbian, cada aplicación corre en sus propios procesos y tiene acceso solo a su propio espacio de memoria. Este diseño hace que las aplicaciones para Symbian sean orientadas a "single threads" y no "multi threads". Algo para destacar es que el sistema posee componentes que permiten el diseño de aplicaciones multiplataforma, esto es diferentes tamaños de pantalla, color, resolución, teclados, etc. La mayoría de estos componentes han sido diseñados en C++.

El diseño del sistema operativo permite que los aparatos con Symbian puedan estar en funcionamiento constante sin necesidad de ser reseteados, preservando la información del usuario y funcionando correctamente (probado en laboratorio). Aunque esto último se está comprometiendo debido a la complejidad de los últimos equipos con Symbian y a la multitud de programas externos al SO.

Si bien no es un software Open Source, las APIs disponen de documentación pública y hasta la versión 8.1 cualquiera podía desarrollar software para Symbian, desde la versión 9.1 se introducen el Framework de Seguridad y algunas funcionalidades que exigen a los desarrolladores firmar digitalmente sus aplicaciones. Si bien funcionalidades básicas (Como la escritura de archivos) pueden ser habilitadas por el usuario, otras más avanzadas (Como el acceso a los dispositivos multimedia) necesitan certificación y firma mediante el programa de firmas de Symbian (Symbian Signed), que usan empresas independientes o fabricantes de teléfonos para la certificación.

El certificado "TrustCenter ACS Publisher ID Certificate" es requerido para que los desarrolladores puedan firmar sus aplicaciones con el software "Test House", proceso que no es gratis (el costo del certificado ronda los u\$s200 anuales). Symbian Signed provee certificación y firmado gratis solo para aplicaciones freeware mediante el software Cellmania.

3.1.1 Desarrollando en Symbian OS

Symbian cuenta con cinco interfaces de usuario o plataformas para su sistema operativo, las denominadas Serie 60, Serie 80, Serie 90, UIQ y MOAP. La mayoría de los móviles utilizan la Serie 60, todos los de Sony Ericsson trabajan bajo UIQ, así como Motorola.

El lenguaje nativo de Symbian OS es el C++ aunque no en una implementación estándar. Existen múltiples SDKs (Software Development Kit) para el desarrollo de aplicaciones, siendo los principales UIQ y S60. Algunos fabricantes ofrecen SDKs propios o extensiones a los SDK para sus productos o para familias de productos que se pueden bajar de los sitios web.

Los SDK oficiales contienen documentación, los headers, las librerías necesarias para compilar un software Symbian, emuladores basados en Windows y un compilador. Hasta la versión 8 se incluye como compilador GCC, la versión 9 usa una nueva ABI

(application binary interface) y requiere un compilador distinto.

La programación en C++ para Symbian requiere el uso de técnicas especiales como descriptores o CleanupStack, esto puede hacer que programas relativamente simples sean más difíciles de implementar que en otros entornos. Actualmente las técnicas de programación necesarias para desarrollar en Symbian hacen que los programas sean propensos a errores en rutinas de bajo nivel en lugar de errores en las funcionalidades específicas de la aplicación.

El primer IDE oficial y comercial para Symbian, Codewarrior, fue reemplazado durante el 2006 por Carbide c++ un IDE basado en Eclipse desarrollado por Nokia que se ofrece en tres versiones:

Carbide.c++ OEM Edition for device creation users.

Carbide.c++ Professional Edition for developers working with preproduction devices.

Carbide.c++ Developer Edition for application development on production phones.

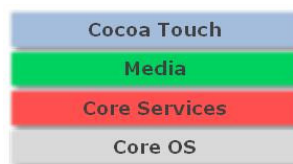
Todas estas versiones son "free of charge" según se informa en la página web "forum.Nokia.com" desde donde se puede descargar tanto el IDE como el SDK s60. Microsoft Visual Studio 2003 y 2005 también es soportado como IDE de desarrollo a través del plug in Carbide.vs.

Muchos de los dispositivos con Symbian OS además de C++ pueden ser programados en OPL, Python, Visual Basic, Simkin y Perl, así como en Java ME.

El plug in para Visual Studio de AppForge llamado "AppForge CrossFire" que permitía programar en un dispositivo Symbian en Visual Basic, VB.NET o C# dejó de estar disponible en el 2007 cuando Oracle compró la propiedad intelectual de AppForge y anuncio que no tiene planeado vender o prestar soporte a los viejos productos de esta empresa. Finalmente también existe una versión del IDE de Borland para Symbian, así como herramientas o técnicas desarrolladas por la comunidad para programar en Linux o MacOS.

3.2 Mac OS (iPhone)

El sistema operativo de iPhone/iPod Touch se ha basado, como no podía ser de otra manera, en su hermano mayor Mac, conformado por la siguiente arquitectura de capas:



Core OS, *Core Services* y *Media* son una copia exacta del código fuente de Mac OSX y la única que ha sufrido algunas variaciones ha sido Cocoa, ya que en Mac OSX, Cocoa se limitaba al uso del ratón y teclado y es por ello que para adaptarse al uso de iPhone y su pantalla táctil haya sido modificada, y como resultado se ha llamado ahora Cocoa Touch.

CORE OS: Los pilares de este sistema operativo de iPhone residen aquí y lo forman:

1. el kernel de OSX,
2. sistema de bibliotecas
3. la pila TCP / IP
4. sockets
5. seguridad

6. gestión de energía
7. keychain
8. certificados
9. sistemas de archivos
10. Bonjour

CORE SERVICES: Son los servicios básicos del sistema y consta de:

1. colecciones
2. agenda de direcciones
3. redes
4. acceso a archivos
5. sqlite (base de datos)
6. core location
7. net services
8. threading
9. preferencias
10. utilidades de URL

MEDIA: Capa de gestión multimedia (gráficos, audio, videos, etc) se compone de:

1. Core Audio
2. openAL,
3. mezclador de sonido,
4. grabación de audio
5. reproducción de video
6. manejo de jpg, png, tiff, pdf
7. quartz (2d)
8. Core Animation
9. OpenGL ES

Cocoa Touch, el framework de desarrollo para iPhone/iPod Touch:

1. eventos y controles multi touch
2. soporte del acelerómetro
3. vista jerárquica
4. localización de aplicaciones
5. alertas
6. vista web
7. selector de contactos

8. selector de imágenes
9. soporte de la cámara

3.2.1 Desarrollando para iPhone

El sistema operativo del iPhone es, en esencia, el mismo que usan las Mac desde hace casi diez años, un Unix BSD súper cargado. El sistema operativo fue desarrollado por NeXT, empresa que también desarrollo un novedosísimo (para entonces) framework de programación orientado a objetos. Ese framework evolucionó en lo que hoy conocemos como Cocoa, específicamente Cocoa Touch en el caso del iPhone, la base sobre la que se construye cualquier aplicación nativa para el iPhone/iPod Touch.

Para que el SDK del iPhone compile código ARM que es la arquitectura del sistema real es necesario firmar la aplicación, lo que requiere la certificación de Apple. Una nueva firma es necesaria para distribuir la aplicación y por supuesto, si queremos que esta esté disponible en el appStore (página oficial de distribución de aplicaciones para el entorno MacOS) deberemos solicitar la aprobación de Apple.

El proceso completo para desarrollar, compilar y publicar una aplicación para el iPhone es el siguiente: ²

- 1.** Inscribirse en el iPhone Dev Center, aceptar todas las condiciones legales, y descargar el SDK. Haciendo esto podemos empezar a escribir nuestras aplicaciones, compilarlas y probarlas en el simulador. Pero todavía NO podremos instalarlas en nuestro iPhone.
- 2.** Inscribirse en el programa para desarrolladores. Después de completar todos los formularios y leer más legales, podemos enviar la solicitud y esperar la respuesta hasta el siguiente día laboral. Esto tiene un costo de U\$S99, y con él obtenemos el derecho a ejecutar nuestras aplicaciones en nuestro propio iPhone.
- 3.** Certificados. Por defecto, un iPhone tan solo puede ejecutar aplicaciones firmadas por Apple y como no es posible estar enviándoles el código cada vez que se quiere probar algo, es necesario crear un perfil con el identificador de nuestro teléfono (accesible mediante las Xcode) e introducirlo en un formulario de la web de Apple. Luego para generar el certificado con el que firmar nuestras aplicaciones vamos a "Keychain Access" y en "Asistente para Certificados" generaremos uno nuevo. Si todo ha ido bien ya podremos experimentar nuestras aplicaciones en el iPhone.
- 4.** ¡Desarrollar! Hasta no hace mucho, se prohibía a los desarrolladores hablar sobre el SDK, lo que dificultaba enormemente el desarrollo en sí. Trabajar con cualquier plataforma nueva conlleva dificultades que habitualmente se resuelven en foros y grupos para desarrolladores. Por suerte, Apple ha abandonado esta política tan cerrada y ahora se puede intercambiar códigos, conocimientos, etc.
- 5.** ¡Distribuir! Para distribuir es necesario otro certificado especial de distribución que hay que solicitar siguiendo el mismo proceso de antes. Acceder a Keychain Access, solicitar, aprobar, descargar e instalar. Al igual que en el punto 3, también necesitamos hacer funcionar un nuevo perfil de distribución en Xcode; proceso sobre el que Apple facilita muchas páginas de información con toda clase de capturas en las que se detalla el modo de reconfigurar el proyecto de Xcode para que utilice este certificado de distribución.
- 6.** Enviar la aplicación a Apple a través de un nuevo formulario web con información sobre esta (nombre, descripción, versión...), su icono y algunas capturas de pantalla.

² Fuente: <http://www.mikeash.com/?page=pyblog/the-iphone-development-story.html>

7. Esperar... Una vez enviada, la aplicación pasa a la cola de revisión, sin que sepamos en ningún momento lo que se está haciendo, el estado de la solicitud, o cuanto queda para que termine el proceso. Si la aplicación no es aceptada por el motivo que sea, no habrá ayuda al respecto, solo se puede hacer los arreglos necesarios y volverla a enviar a través de un enlace con el que enviamos nuevas versiones de una misma aplicación. Si los problemas continúan, habrá que esperar una semana para saberlo y ver si la solución que has dado resuelve satisfactoriamente el problema.

8. ¡¡Aceptada!! Ahora queda esperar que dé frutos la aplicación frente de los 10 millones de clientes potenciales que la podrán comprar.

Desarrollando en Cocoa Touch, la plataforma de desarrollo para el iPhone.³

Cocoa es una API escrita en un dialecto especial de C llamado Objective-C. En particular, cuando desarrollamos para el iPhone/iPod Touch, usamos Cocoa Touch. La diferencia básica entre Cocoa y Cocoa Touch es la forma en la que el usuario invoca comandos. En Cocoa tenemos un teclado físico y un mouse o trackpad, en Cocoa Touch usamos un teclado virtual y nuestros dedos directamente sobre la pantalla.

Por encima de la API en sí nos encontramos con las herramientas que forman parte del entorno de desarrollo de la plataforma:

- **Xcode:** Xcode es la pieza esencial del entorno. Es un IDE muy completo, en el que podemos editar código fuente, acceder a un vasto volumen de documentación, y hasta un debugger gráfico. Xcode está construido por numerosos módulos Open Source (como el compilador gcc y el debugger gdb).
- **Instruments:** Instruments sirve para monitorear la aplicación en desarrollo y sintonizar finamente su performance, en un maravilloso entorno gráfico. Instruments está basado en DTrace, una herramienta Open Source desarrollada por Sun Microsystems. Es esencial a la hora de identificar pérdidas de memoria y otros bugs difíciles de rastrear.
- **Dashcode:** Dashcode fue diseñada para desarrollar widgets para el Dashboard de Mac OS X. La versión del SDK de iPhone/iPod Touch es básicamente la misma que la de Mac OS X, y fue incluida para facilitar el desarrollo de aplicaciones Web para el iPhone.
- **Simulator:** el simulador de iPhone permite probar las funcionalidades básicas de la aplicación en desarrollo. El simulador corre un sistema operativo parecido al real. Cuando se trabaja con el simulador, Xcode compila para x86, en lugar de ARM (que es la arquitectura del sistema real). Para compilar código ARM es necesario firmar la aplicación, lo que requiere la certificación de Apple.
- **Interface Builder:** Su uso elemental es el desarrollo de la Interface Humana (GUI), sin embargo es mucho más que eso. IB es donde los diferentes módulos se conectan, y es también donde se le da vida a muchos de los módulos precompilados que se verán en la pantalla.

XCOCODE:

Xcode es el motor que le brinda poder al ambiente integrado de desarrollo de Apple para Mac OS X y para iPhone OS. También es una aplicación que se encarga de la mayoría de los detalles del proyecto desde el inicio hasta el despliegue que permite:

- Crear y manejar proyectos, incluyendo plataformas de especificación,

³ Fuente: <http://www.iphonегurues.com/la-plataforma-de-desarrollo-cocoa-touch/>

requerimientos de objetivo, dependencias, y configuraciones de la estructura.

- Escribir código fuente en editores con características tales como coloreo de sintaxis e indentación automática.
- Navegar y buscar a través de los componentes de un proyecto, incluyendo los archivos de encabezado y de documentación.
- Construir el proyecto.
- Depurar el proyecto de forma local, en el simulador iPhone OS, o remotamente, en un depurador gráfico a nivel de fuente.

Xcode construye proyectos desde código fuente escrito en C, C++, Objective-C, y Objective-C++. Este genera ejecutables de todos los tipos soportados en Mac OS X, incluyendo herramientas de línea de comando, marcos de trabajo, plug-ins, extensiones kernel, colecciones (bundles), y aplicaciones. (Para iPhone OS, sólo son posibles aplicaciones ejecutables.) Xcode permite una personalización casi ilimitada de herramientas de construcción y de depuración, paquetes ejecutables (incluyendo lista de información de propiedades y colecciones localizadas), construir procesos (incluyendo archivos copiar, archivos script, y otras fases de construcción), y la interfaz de usuario (incluyendo editores de código separados y multi-vistas). También soporta varios sistemas de manejo de código fuente como CVS, Subversion, y Perforce-permitiéndole añadir archivos a repositorios, efectuar cambios, obtener versiones actualizadas, y comparar versiones.

Xcode está diseñado especialmente para desarrollo en Cocoa. Cuando creamos un proyecto, Xcode configura su ambiente de desarrollo inicial usando plantillas de proyectos correspondientes a tipos de proyectos Cocoa: aplicación, aplicación basada en documentos, aplicación Core Data, herramienta, colección, marco de trabajo, y otros. Para compilar software Cocoa para Mac OS X, Xcode usa el compilador GNU (gcc), y para depurar ese software, usa el depurador de nivel fuente GNU. Ambos gcc y gdb han sido usados en el desarrollo Cocoa desde que Cocoa era NeXTSTEP, y a lo largo de los años ha sido refinado, extendido, y afinado para soportar la compilación y la depuración de binarios Cocoa.

3.3 RIM (Research in motion)

BlackBerry RIM SDK (BlackBerry Research In Motion Software Development Kit) es el entorno para construir aplicaciones para los dispositivos inalámbricos BlackBerry. El lenguaje de programación utilizado es J2ME, esta es la plataforma de java para dispositivos móviles. El uso de los dispositivos Blackberries es popular en el segmento de negocios y no tanto en el segmento de hogares o público general, por ello el SDK está orientado fuertemente al desarrollo de este tipo de aplicaciones.

Además de las librerías estándar CLDC y MIDP, RIM provee algunas librerías muy útiles como: `net.rim.device.api` que provee acceso a funcionalidades específicas para el hardware RIM. Dentro de este paquete hay clases para monitorear los puertos COM, la radio, el teclado o el "thumb wheel". EL paquete `net.rim.device.api.util` incluye construcciones muy útiles como vectores ordenados, vectores clonables, buffers de datos y otros. La librería `net.rim.device.api.system.mobitex` es usada para obtener información de la red Mobitex (red usada por los dispositivos BlackBerry).

El SDK también incluye un IDE que soporta debugging y el emulador necesario para esto.

3.3.1 Desarrollando para RIM

RIM provee un sistema operativo propietario, multitarea para los dispositivos BlackBerrys que hace uso intensivo de los dispositivos de entrada como el teclado y el trackball. El sistema operativo provee soporte para MIDP 1.0 y 2.0 (a partir de la versión 4) así como para WAP 1.2.

Cualquier desarrollador puede escribir software usando las APIs propietarias de BlackBerry, pero para que una aplicación pueda acceder a ciertas funcionalidades restringidas debe estar firmado digitalmente con un certificado asociado a una cuenta de desarrollador en RIM, este proceso garantiza la procedencia de las aplicaciones, no así la calidad del código.

3.4 Windows Mobile ⁴

Windows Mobile es el sistema operativo de Microsoft destinado a dispositivos móviles. La evolución de Windows Mobile, que se puede encontrar en Wikipedia, refleja la evolución de los dispositivos de bolsillo durante los últimos años. Básicamente se originó en una versión del kernel de Windows para sistemas embebidos que fue creciendo hasta ser un sistema operativo para ordenadores de bolsillo Pocket PCs (PPC) y smartphones.

PocketPC es un estándar de Microsoft que impone varios requisitos al hardware y al software de dispositivos móviles.

Cualquier dispositivo que sea clasificado como un PocketPC debe:

- Ejecutar el sistema operativo Microsoft Windows CE o Windows Mobile (versión PocketPC)
- Tener un conjunto de aplicaciones en ROM
- Incluir una pantalla sensible al tacto
- Incluir un dispositivo apuntador, llamado stylus o estilete
- Incluir un conjunto de botones de hardware para activar aplicaciones
- Estar basado en un procesador compatible con el STRONGARM

Algunas de las aplicaciones que se incluyen con estos dispositivos son versiones reducidas de: Outlook, Internet Explorer, Word, Excel, Windows Media Player, etc..

Pocket PC Phone Edition además de los requisitos anteriores incluye radio/s para telefonía celular.

Un SmartPhone para Microsoft es un Pocket PC Phone Edition generalmente sin "touch screen".

3.4.1 Desarrollando para Windows Mobile

Microsoft provee en forma gratuita el "Windows Mobile 6 Professional and Standard Software Development Kits Refresh" SDK que incluye todo lo necesario para el desarrollo de aplicaciones en la plataforma Windows Mobile, pero para instalarlo necesitaremos, como se indica en la sección de requisitos de la página de descargas, el Microsoft Visual Studio 2005 Standard Edition o superior (No está soportado el Visual Estudio Express Editions), esto implica que para desarrollar en WM sea necesario adquirir una licencia de Visual Studio 2005 o superior.

⁴ Fuentes: <http://javiercancela.com/2007/10/19/introduccion-al-desarrollo-de-aplicaciones-para-telefonos-moviles-windows-mobile>

Una vez descargado e instalado el SDK tendremos todo lo necesario para realizar aplicaciones para WM6 incluyendo emuladores (imágenes ROM) para los distintos tamaños de pantalla de los dispositivos en mercado.

Se pueden desarrollar dos tipos de aplicaciones para Windows Mobile: con código nativo o con código administrado (managed code). Llamamos código nativo al código C++ que utiliza directamente la API de Windows Mobile, y código administrado al que utiliza las clases del .NET Compact Framework con C# o VB.Net. (Windows Mobile es la única plataforma móvil importante que no soporta J2ME).

¿Las diferencias entre ambas? El código nativo es más rápido y ocupa menos, además de proporcionar acceso a algunas características del hardware que son inaccesibles desde el Compact Framework. Sin embargo, en la mayor parte de los casos desarrollar código administrado es la mejor opción. El tamaño del ejecutable es cada vez menos importante, y si la velocidad es un factor crítico siempre se puede optar por programar en código nativo las partes de la aplicación que supongan un cuello de botella. Por lo demás, el desarrollo en .NET resulta mucho más fácil y cómodo.

El desarrollo de aplicaciones para Windows Mobile presenta como inconvenientes la falta de alternativas al Visual Studio y el consiguiente desembolso económico necesario para adquirir una licencia. Por otra parte, el lado positivo se encuentra tanto en la calidad de las herramientas disponibles (el propio Visual Studio, los emuladores, la SDK y su documentación...) como en la activa comunidad de desarrolladores existente y agrupada en torno al portal de desarrollo de Microsoft, la MSDN.

3.5 Linux

En los últimos años, muchas compañías relacionadas con el mundo de la movilidad, fabricantes y desarrolladores de software y aplicaciones han optado por utilizar Linux como base para sus sistemas operativos. Japón y China son los países donde Linux móvil está más implantado, llegando a conseguir una cuota del 40% del mercado.

En la actualidad, hay un buen número de iniciativas de adaptación de Linux a terminales móviles y a continuación, se revisaran brevemente las plataformas Linux más populares:

LiMo (Linux Mobile) Foundation:

Es una alianza fundada por Motorola, NEC, NTT DoCoMo, Panasonic Mobile Communications, Samsung Electronics, y Vodafone en enero de 2007 para desarrollar la plataforma LiMo, una plataforma basada en Linux para dispositivos móviles. Desde entonces, nuevos miembros se han unido a la fundación, entre los que se encuentran Infineon Technologies, Mozilla, SFR y Verizon Wireless. LiMo anunció a finales de marzo de 2008 el lanzamiento de LiMo Release 1 (sistema operativo básico y sin aplicaciones que ya está disponible en dos teléfonos comerciales, el Razr 2 y el Rokr E8 de Motorola). A principios de 2009 se espera el lanzamiento de la Release 2, que mejorará la portabilidad, y sus capacidades multimedia.

OpenMoko:

Es un proyecto para crear una plataforma para smartphones usando software libre. Utiliza el núcleo de Linux, junto con un entorno gráfico de usuario construido con el servidor X.Org, el toolkit GTK+ y el gestor de ventanas Matchbox. Está basado en el framework de OpenEmbedded y el sistema de paquetes ipkg. OpenMoko se anunció en 2006 por sus fundadores: First International Computer (FIC). El primer

smartphone en el que funciona OpenMoko es el Neo1973, fabricado por FIC.

MOTOMAGX:

Es un sistema operativo para móviles desarrollado por Motorola. Este sistema es una combinación de Linux y Java. La plataforma MOTOMAGX tiene una arquitectura modular que incluye los siguientes componentes:

- Sistema operativo Linux. Este sistema operativo incorpora paquetes de la comunidad de código abierto y componentes adicionales para satisfacer los requerimientos de los dispositivos móviles. Los primeros productos basados en MOTOMAGX son el MOTOROKRTM Z6, MOTORAZR2 V8, MOTOTM U9 y el recientemente anunciado MOTOROKR E8.
- Plataforma de librerías y servicios middleware. Proporciona servicios a la plataforma y gestiona el ciclo de vida de la aplicación, las interacciones de las aplicaciones y la seguridad de la plataforma.
- Entorno de aplicación (Java ME, WebUI, native Linux). Los entornos de aplicación incluyen las APIs y servicios necesarios.
- Aplicaciones. Las aplicaciones proporcionan al usuario experiencia a través de interacciones con el resto de componentes.

Access Linux Platform (ALP):

Es un sistema operativo basado en Linux para smartphones capaces de ejecutar aplicaciones PALM OS. Esta plataforma es de la empresa Access, que adquirió en 2005 los derechos de Palm OS. En verano de 2008, el operador móvil Orange había anunciado el lanzamiento del primer dispositivo comercial, el Samsung i800, proyecto que finalmente fue suspendido.

ARM Linux Mobile Platform:

ARM y seis empresas más, entre las se encuentran Texas Instruments, Samsung, Marvell o Mozilla están creando una plataforma software estándar basada en código abierto de Linux especialmente diseñada para dispositivos móviles. La previsión es poder presentar los primeros dispositivos Linux durante el 2009.

Qtopia:

Es una plataforma de aplicaciones para dispositivos móviles que utilizan Linux como sistema operativo, desarrollada por la empresa Trolltech. Existen dos categorías de Qtopía, una libre, bajo licencia GPL (Opie), y otra comercial; así como dos ediciones, una para teléfonos móviles y otra para PDAs. Qtopia se instala en numerosos dispositivos móviles de Sharp Corporation de la línea de productos Zaurus, que incluye más de diez modelos. También la empresa Archos lo incluye en el PMA430, un dispositivo multimedia. La edición para teléfonos móviles se espera que esté pronto disponible en numerosos aparatos.

Android:

Es una plataforma software basada en Linux para dispositivos móviles que incluye un sistema operativo, middleware y aplicaciones clave. Esta plataforma está

siendo desarrollada por Google y el Open Handset Alliance. Permite a los desarrolladores escribir código en Java usando librerías de software desarrolladas por Google, pero no soporta programas en código nativo.

La plataforma Android fue anunciada en noviembre 2007 con el Open Handset Alliance, un consorcio de 34 compañías de hardware, software y tele comunicaciones dedicadas al avance de estándares abiertos para dispositivos móviles. Android proporciona un paquete completo de software a todos los niveles:

- Un kernel linux que sirve como base de la pila de software y se encarga de las funciones más básicas del sistema: gestión de drivers, seguridad, comunicaciones, etc.
- Una capa de librerías de bajo nivel en C y C++.
- Un framework para el desarrollo de aplicaciones, dividido en subsistemas para gestión del sistema como el "package manager"; gestión del hardware del teléfono anfitrión ("telephony manager") o acceso a APIs sofisticadas de geolocalización o mensajería XMPP. También incluye un sistema de vistas para manejar el interfaz de usuario de las aplicaciones, que incluyen posibilidad de visualización de mapas o renderizado html directamente en el interfaz gráfico de la aplicación.
- Una suite de aplicaciones (navegador, agenda, gestión del teléfono).

Las aplicaciones Android están programadas en Java, pero no corriendo sobre Java ME, sino sobre Dalvik, una máquina virtual Java desarrollada ex profeso por Google y optimizada para dispositivos embebidos.

Android permite a los desarrolladores escribir código manejado en un lenguaje similar a Java que utiliza librerías java desarrolladas por Google, pero que no soporta desarrollos en código nativo. Casi toda la plataforma Android fue liberada en 2008 bajo la licencia "Apache free-software and open-source". El primer teléfono que usa Android es el HTC G1 Dream y se vende como "Versión para desarrolladores".

En resumen, aparte de las plataformas Linux mencionadas anteriormente se encuentran otras como el proyecto Ubuntu Mobile and Embedded que también intenta ser una versión del sistema operativo Ubuntu Linux adaptado para funcionar en dispositivos móviles.

Si bien muchas marcas participan en los proyectos de desarrollo de sistemas operativos para móviles basados en Linux la cuota de mercado de este sistema operativo es solo significativa en Japón y China (utilizado por algunos dispositivos Motorola muy populares en estos países) donde es la competencia directa de Symbian con un market share de aproximadamente 40% si se suman los móviles de las distintas arquitecturas basadas en Linux.

Ahora mismo hay una gran expectativa ante la plataforma de Google, Android. Lo que se espera es que, con Android, exista un estándar abierto de plataforma móvil que evite la fragmentación de los sistemas operativos y dispositivos. La pregunta que surge es: ¿Podrá Google, con su potente plataforma, Android, acabar con la fragmentación e imponerse ante los grandes gigantes de los dispositivos móviles?

3.5.1 Desarrollando para Android

Se puede desarrollar una aplicación Android con las mismas herramientas utilizadas

para desarrollar en Java. Las "Android core libraries" proveen las funcionalidades necesarias para construir impresionantes y poderosas aplicaciones para dispositivos móviles.

El "Android SDK" soporta Windows XP, Windows Vista, Linux y Mac OS y utiliza como IDE de desarrollo a Eclipse mediante el plug in "Android Development Tools (ADT)" aunque es posible utilizar Apache Ant para entornos Mac o Linux

Anatomía de una aplicación Android ⁵:

Las aplicaciones Android están constituidas a partir de la combinación de los siguientes bloques:

- **Activity:**

Una "actividad" es el bloque más usado en las aplicaciones "Android". Generalmente, una "actividad" es una pantalla individual en tu aplicación. Cada "actividad" se implementa como una clase que hereda de "Activity", lo que hará que la clase despliegue una UI compuesta de "Views" y responda a eventos. Lo común es que una aplicación consista en múltiples pantallas. 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 ver y/o cambiar la configuración. Cada una de estas pantallas debe ser implementada como una "Activity". La navegación entre las pantallas se hace iniciando una nueva "Activity". En algunos casos, una "Activity" podría devolver un valor a la "Activity" anterior, por ejemplo la "Actividad#1" podría permitir seleccionar una foto y esta foto sería devuelta a la "Activity" que hizo el llamado a "Actividad#1".

Cuando una pantalla es abierta, la previa es puesta en pausa y agregada al "History Stack". Posteriormente, el usuario puede navegar hacia pantallas previas invocando las pantallas almacenadas en el "History Stack". Las pantallas también pueden ser removidas del "History Stack" cuando resulta inapropiado su almacenamiento. Android mantiene un "History Stack" por cada una de las aplicaciones que son activadas desde la "Home Screen".

- **Intent y IntentFilter:**

Android usa una clase especial 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. Los datos son expresados como una URI. 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 que representa a esa persona.

Una clase relacionada con "Intent" es "IntentFilter". Si bien un "Intent" es una solicitud para realizar algo, un "IntentFilter" es una descripción de lo que intenta hacer una "Activity" (o de lo que intenta recibir). Una "Activity" que es capaz de desplegar la información de contacto de una persona podría declarar que sabe cómo tratar la acción "VIEW" cuando es aplicada a los datos que representan a la persona. Una "Activity" declara sus "IntentFilter"s en el archivo "AndroidManifest.xml".

⁵ Fuente: <http://celutron.blogspot.com/2007/11/manos-la-obra-anatoma-de-una-aplicacin.html>

La navegación de pantalla a pantalla es ejecutada a través de la resolución de intentos. Para navegar hacia adelante, una "Activity" llama "startActivity(myIntent)". Entonces, el sistema examina todos los "IntentFilter"s que existen para las aplicaciones instaladas y toma aquellas "Activity" cuyos "IntentFilter" mejor se acercan a "myIntent". La nueva "Activity" es informada del "Intent", lo que causa que sea activada. El proceso de resolución de "Intent" ocurre en tiempo de ejecución cuando "startActivity" es llamado, lo cual tiene dos beneficios claves:

Una "Activity" puede reutilizar funcionalidades de otras componentes con sólo hacer un una solicitud en la forma de "Intent".

Una "Activity" puede ser reemplazada en cualquier momento por una nueva "Activity" que tenga un "IntentFilter" equivalente.

- **IntentReceiver:**

Se puede utilizar un "IntentReceiver" cuando queramos programar una aplicación para que se ejecute como respuesta a un evento. Por ejemplo, cuando recibes una llamada, cuando la red de datos está disponible o cuando es media noche. Los "IntentReceiver"s no despliegan UI, sin embargo ellos podrían utilizar el "NotificationManager" para avisar al usuario que algo interesante está pasando. Los "IntentReceiver"s son registrados en el archivo "AndroidManifest.xml", pero también pueden ser registrados programáticamente con el uso de "Context.registerReceiver()". La aplicación no requiere estar corriendo para que sus "IntentReceiver"s puedan ser llamados. Si la aplicación no estuviera corriendo, el sistema la puede activar cuando uno de sus "IntentReceiver"s sea gatillado. Las aplicaciones también pueden transmitir sus propios "Intent" a otras aplicaciones a través de "Context.broadcastIntent()".

- **Service:**

Un "Service" es una aplicación que se mantiene activada por un largo tiempo y no despliega una UI. Un ejemplo es 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 con "Context.startService()" y de esa forma reproducir la música sin necesidad de usar la pantalla. El sistema mantendrá el servicio de reproducción de música corriendo hasta que finalice. Es posible conectarse a un "Service" haciendo uso del método "Context.bindService()". Una vez que la aplicación está conectada al servicio, la comunicación entre la aplicación y el servicio se realiza a través de la interfaz que el "servicio" expone. Para el ejemplo del reproductor mp3, esta interfaz podría permitir hacer pausa o saltar a una nueva canción.

- **Content Provider:**

Las aplicaciones pueden almacenar sus datos en archivos, una base de datos "SQLite" o cualquier otro mecanismo. Sin embargo, un "Content Provider" es de utilidad cuando los datos de tu aplicación deben ser compartidos con otras aplicaciones. Un "Content Provider" es una clase que implementa un conjunto estándar de métodos para que otras aplicaciones almacenen o recuperen el tipo de datos que el "Content Provider" manipula.

4. Conclusión

Si bien no hay acuerdo entre los fabricantes acerca de la definición o nombre de los dispositivos móviles, es necesario diferenciar los smartphones con pantalla sensible al tacto (Touch screen) y los que no disponen de ella. Esta división se hace necesaria debido a las diferencias que supone el manejo de un dispositivo móvil adaptado para usar una sola mano y generalmente 2 botones personalizables y un control de desplazamiento (puede ser una rueda un track ball o un botón de 5 direcciones) y el manejar un dispositivo construido para usar las dos manos y seleccionar las acciones desde cualquier área de pantalla ya sea con los dedos o un puntero especial tipo lápiz.

Lo que sí es un hecho, es que un Smartphone, posee un sistema operativo identificable y kits de desarrollo disponibles ya sean estos comerciales o sin cargo. Con respecto a las Cuotas de Mercado, el mercado norteamericano de dispositivos móviles, tiene una distribución muy diferente al resto del mundo y con algunas particularidades a tener en cuenta:

En el resto del mundo el líder indiscutido (con cuotas de mercado siempre superiores al 50%) es Symbian, mientras que en los EEUU este tiene una escasa participación.

El principal sistema operativo en el mercado de EEUU es RIM, un sistema que está pensado para aplicaciones de negocios y que ganó su cuota de mercado gracias a la tecnología de push e-mail, la cual ya está disponible en los dispositivos que corren Windows Mobile 6 y en el iPhone, sus principales competidores.

El mercado de móviles de EEUU continuó en crecimiento durante el 2008 (casi 10 puntos) mientras que los mercados japoneses, europeos y chinos tuvieron retrocesos.

Existe una gran expectativa en torno a Android, que respaldado por Google y con solo un equipo en el mercado (en versión para desarrolladores) logró ventas equivalentes al 4% del mercado de EEUU en un trimestre. En la comunidad de Linux se espera que este sistema sirva para unificar el fragmentado mercado de móviles basados en Linux (con arquitecturas escasamente compatibles entre sí) ya que es de uso libre. Al respecto recientemente Motorola (el mayor fabricante de móviles de EEUU) anunció que abandonaba Symbian (consorcio dominado por Nokia desde el Q4 de 2008) y planeaba construir móviles basados en Android para el año 2009. Recordemos que la cuota de mercado de estos móviles (basados en Linux) en los mercados asiáticos es de aproximadamente 40%.

El sistema operativo para dispositivos Palm denominado Access que solo tiene presencia de mercado en los EEUU fue reemplazado durante el presente año por Windows Mobile según lo anunciara la empresa luego de sacar al mercado sus primeros móviles con este sistema operativo.

Los costos de los entornos de desarrollo y de los SDK:

Tanto iPhone como RIM y Symbian son gratuitos, no así para Microsoft que requiere una licencia de Visual Studio 2005 para la instalación de su SDK.

Todas las aplicaciones que requieran hacer uso de los recursos considerados "no básicos" (acceso a recursos multimedia y de geo-localización) deben ser firmadas con certificados cuyo costo oscila en u\$s200 por año. Una excepción son las aplicaciones para iPhone que requieren de un certificado para poder ser instaladas en el teléfono de pruebas y un segundo certificado para distribuirlos y que se puedan instalar en cualquier iPhone, estos certificados son otorgados por Apple y tienen un costo de u\$s 99 cada uno.

Finalmente, para el 2009, con respecto a los sistemas operativos, se espera el lanzamiento de Windows Mobile 7, que según lo anunciado por Microsoft hará un uso mucho más intensivo de las pantallas táctiles y los sistemas de posicionamiento global (GPS) e incluirá algunas

características del iPhone y Android como el renderizado directamente en pantalla. Google ya anunció la salida al mercado global de su móvil en versión final y sin restricciones de operadores para el primer semestre de 2009.

Síntesis General:

Dispositivo	SDK	S.O.	Navegador	Costo Aprox. (Argentina)
BlackBerry	<ul style="list-style-type: none"> • Plugin para VS .Net C# (Gratis) • Java (Gratis) 	<ul style="list-style-type: none"> • RIM 	<ul style="list-style-type: none"> • OperaMini • Minuet 	<ul style="list-style-type: none"> • Sin Touch:\$1100 • Con Touch:\$2400
HTC	<ul style="list-style-type: none"> • Visual Estudio .NET (desde U\$S 240) • Adroid (Gratis) 	<ul style="list-style-type: none"> • <u>Sin touch screen</u>: W. Mobile SmartPhone • <u>Con touch screen</u>: W. Moble Profesional y Android 	<ul style="list-style-type: none"> • iExplorer • OperaMini • Android 	<ul style="list-style-type: none"> • Sin Touch:\$900 • Con Touch:\$1700
iPhone	<ul style="list-style-type: none"> • Cocoa Touch (Gratis) 	<ul style="list-style-type: none"> • MAC OSX 	<ul style="list-style-type: none"> • SAFARI 	<ul style="list-style-type: none"> • con Touch:\$2000