Entornos de programación móviles

Julián David Morillo Pozo

PID_00176754

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

El desarrollo de aplicaciones móviles es el proceso por el cual se desarrolla un *software* para dispositivos móviles (como *smartphones* o *tablets*). La forma de distribución de estas aplicaciones puede variar, las aplicaciones pueden venir preinstaladas en los teléfonos o pueden ser descargadas por los usuarios desde *app stores* (tiendas de aplicaciones) y otras plataformas de distribución de *software*.

En este módulo veréis los diferentes entornos de programación para aplicaciones móviles existentes. Comenzaremos por una revisión histórica de su evolución. Después, revisaremos los diversos lenguajes de programación que se pueden utilizar en estos entornos y, por último, estudiaremos a fondo algunos de los entornos más populares.

Objetivos

Con el estudio de este módulo pretendemos que consigáis los objetivos siguientes:

1. Que conozcáis y comprendáis el concepto de entorno de programación en el ámbito del desarrollo de aplicaciones para dispositivos móviles.

2. Que identifiquéis los diversos entornos de programación existentes y conozcáis sus arquitecturas, las características de los sistemas operativos usados y las técnicas de programación requeridas en cada uno de ellos.

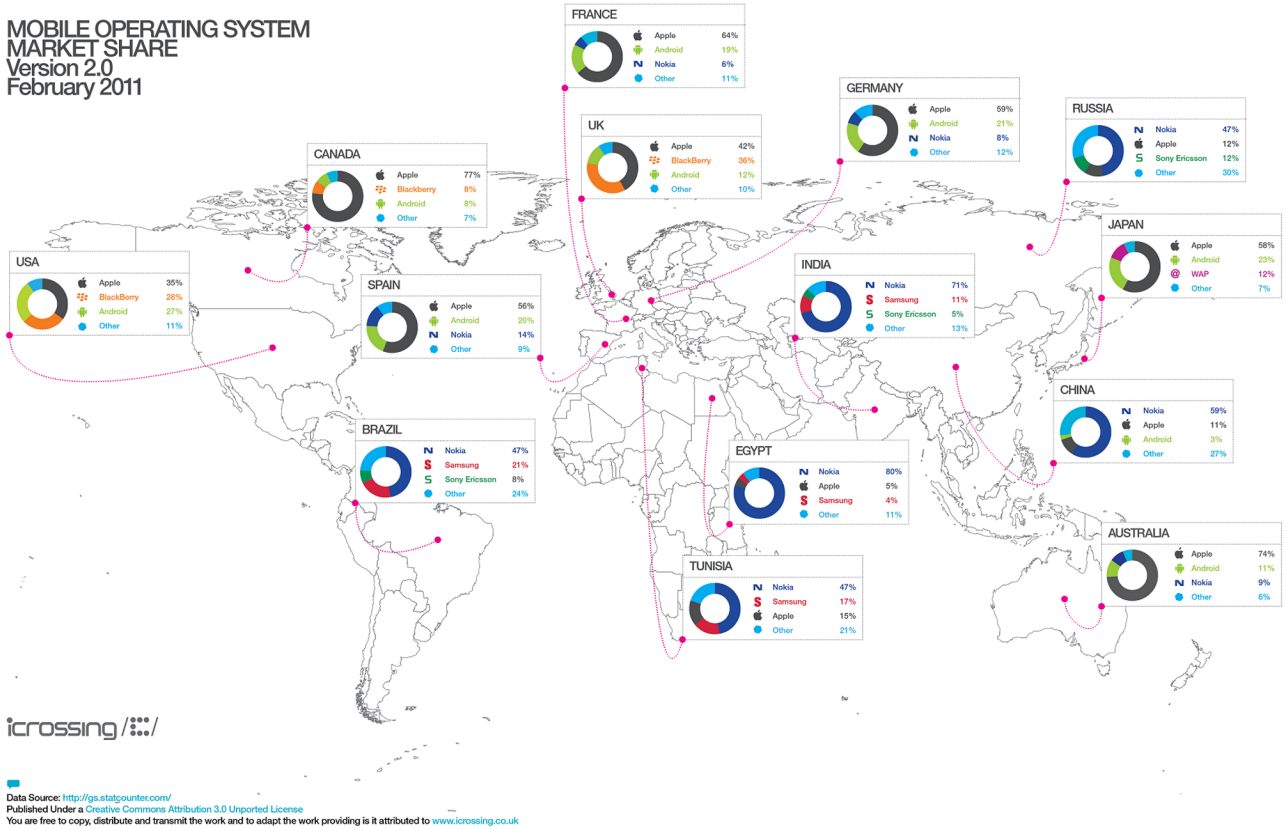
3. Que comprendáis que los entornos no funcionan de forma aislada, sino que coexisten.

4. Que seáis capaces de elegir el entorno de programación idóneo según los requerimientos de la aplicación móvil que pretendáis desarrollar.

1. Historia y evolución de los entornos de programación móviles

La industria de los dispositivos y las aplicaciones móviles es un entorno en constante cambio. Durante el 2010 vimos cómo Nokia abandonó Symbia y quedó a la espera de que MeeGo y su asociación con Windows les relanzaran en el mundo de los *smartphones*. También asistimos al espectacular crecimiento de Android, que ha pasado por delante del iOS de Apple y de BlackBerry y se ha convertido en la segunda plataforma (por detrás de Symbian).

En lo que respecta al mercado de los fabricantes de equipos originales vimos más movimientos en el 2010 que en los diez años anteriores. Apple y RIM adelantaron a algunos de los fabricantes tradicionales (Sony Ericsson, Motorola, LG) y reclamaron un puesto en el *top* 5. Según algunas estimaciones, ZTE podría unirse a ellos pronto. La siguiente figura ofrece una visión general sobre cómo está el mercado de las plataformas móviles en todo el mundo. Por países, destaca el dominio de Apple en Estados Unidos y en diversos países de Europa (como España), así como la cada vez más destacada presencia de Android. Nokia arrasa en India, en China y en otras potencias emergentes.

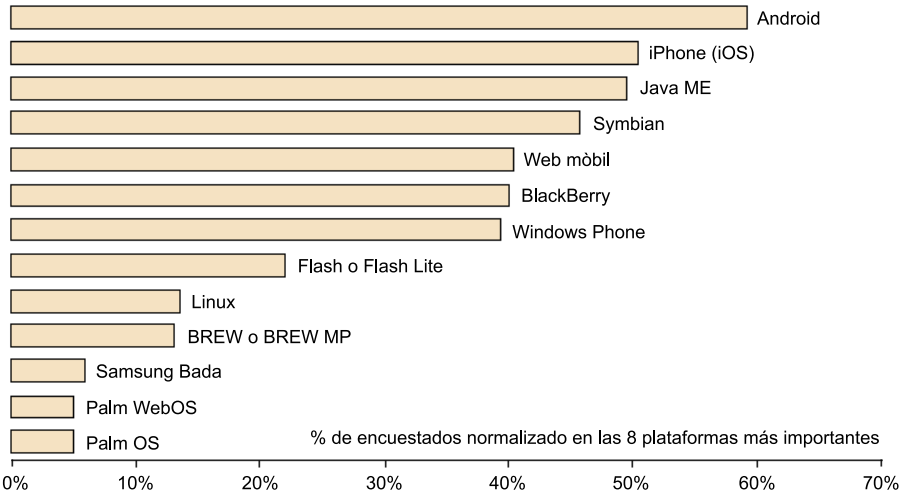


Cuotas de mercado de las diferentes plataformas móviles

La lucha por la supremacía de las plataformas móviles está candente. Android y iPhone, así como BlackBerry o Nokia, son varios de los actores que más destacan.

Por lo tanto, las principales medidas para el desarrollo de aplicaciones móviles han cambiado mucho y lo seguirán haciendo. Una de ellas es la popularidad de las diferentes plataformas entre los desarrolladores. En los últimos tiempos se ha producido una migración en las preferencias de los desarrolladores, que los ha movido desde la "vieja guardia" (Symbian, BlackBerry y Java) hacia los nuevos reyes del sector: iOS y Android. Según algunos estudios, cerca del 60% de los desarrolladores han desarrollado aplicaciones para Android, tal como se puede ver en la figura siguiente. El iOS de Apple ocupa el segundo lugar (con más del 50%), seguido por Java ME, que se encuentra en tercera posición. Así, podemos ver cómo los desarrolladores cambian el foco de su atención hacia unas plataformas y abandonan otras.

Plataformas más usadas por los desarrolladores de aplicaciones móviles en el 2010



Fuente: Mobile Developer Economics 2010 and Beyond. Producido por VisionMobile. Patrocinado por Telefónica Developer Communities. Junio 2010. Licenciado bajo licencia Creative Commons Attribution 3.0. Cualquier uso o remezcla de este trabajo debe conservar esta nota.

Sin lugar a dudas, el cambio más significativo en los últimos tiempos es que la distancia entre Android y iOS, por un lado, y el resto de plataformas, por otro, se está haciendo cada vez mayor. La *app store* de Apple contiene más de trescientas mil aplicaciones, mientras que estimaciones recientes sitúan el número de aplicaciones del Market de Android en unas ciento treinta mil.

Mientras tanto, Nokia ha estado poniendo un esfuerzo considerable en la Ovi Store y, de hecho, ha incrementado su popularidad entre consumidores y desarrolladores, aunque aún le queda un largo camino para alcanzar a los dos gigantes dispensadores de aplicaciones.

Los motivos por los que los desarrolladores se mueven hacia iOS y Android pueden ser varios, pero los más importantes son los que exponemos a continuación. Por un lado, Apple ofrece una plataforma que es relativamente fácil de aprender y de usar, con la que el desarrollador puede diseñar UI[1] muy bue-

(1)UI (*user interface*)

nas. Además, tiene la tienda de aplicaciones más grande y, aunque el problema de la certificación es un inconveniente para algunos, no existen los problemas de portabilidad y fragmentación. Android, por otro lado, ha ido ganando ímpetu en todos los campos asaltando los mercados clave de sus competidores. Por supuesto, tiene muchos inconvenientes derivados de la fragmentación, pero estos se pasan por alto muchas veces debido a la dependencia de muchos fabricantes de esta plataforma.

Otro aspecto importante a comentar es la disparidad que ha habido entre las ventas de dispositivos para cada plataforma y el número de aplicaciones disponibles. Sería de esperar que las plataformas con la mayor penetración en el mercado fueran las que lo dominaran (en términos de aplicaciones), pero nada más lejos de la realidad, por lo menos hasta el 2011.

Si tomamos como referencia el tercer cuatrimestre del año 2010, podéis ver que las dos plataformas con la penetración más baja, iOS y Android, tenían el mayor número de aplicaciones disponibles.

En el lado opuesto, mientras Java ME y Flash Lite han tenido, con diferencia, la mayor penetración en el mercado, apenas se pueden comparar con las nuevas plataformas en cuanto a volúmenes de aplicaciones.

En el cuarto cuatrimestre, este contraste se hizo incluso mayor. Tanto la tienda de Android como la de iOS crecieron prácticamente en cien mil aplicaciones cada una. Windows Phone mostró un crecimiento digno de mencionar (alcanzó cuatro mil aplicaciones en apenas dos meses), aunque le queda un largo camino para convertirse en una amenaza seria para los dos actores principales.

Por lo tanto, vemos que cada vez hay más desarrolladores dedicados a este tipo de aplicaciones. No obstante, debemos señalar que las encuestas realizadas indican que la mayoría reconocen pocos beneficios económicos, mientras que solo un 5% tiene beneficios por encima de sus expectativas.

Si bien es cierto que asistimos a un *boom* de tiendas de aplicaciones, esto no es necesariamente una bendición para los desarrolladores. Muchos de ellos se enfrentan a problemas para que los potenciales usuarios descubran sus aplicaciones, las cuales se quedan enterradas bajo otras miles de aplicaciones. Podríamos decir que es como ir a una tienda de discos con doscientos mil CDs: solo se mira en el *top* 10.

En este sentido, una opción para los desarrolladores es adoptar una estrategia de **escaparate múltiple** mientras se adapta el modelo de beneficio a tiendas de aplicaciones específicas. Por ejemplo, hasta ahora ha sido muy difícil vender contenido en Android, por lo que parece que la mejor estrategia para esta plataforma, por lo menos para darse a conocer, es que la aplicación sea gratis.

Por lo tanto, la conclusión que podemos sacar de todo esto es que hay que estar muy atento para ver cómo evolucionan factores importantes como el desarrollo de aplicaciones, los beneficios, la distribución, la venta al por menor, la portabilidad y la fragmentación, entre otros.

2. Aplicaciones web y aplicaciones nativas

Antes de entrar a fondo en los entornos de programación de aplicaciones para dispositivos móviles, vamos a establecer de forma resumida el ámbito en el que nos vamos a mover. A continuación definiremos términos clave y compararemos las ventajas e inconvenientes de los dos paradigmas de desarrollo más comunes.

Para empezar, vamos a definir qué se entiende por aplicación web y por aplicación nativa. Vamos a considerar las ventajas e inconvenientes de cada una de ellas.

Una **aplicación web** es, básicamente, un sitio web específicamente optimizado para un dispositivo móvil. Las características que definen una aplicación web son las siguientes: la interfaz de usuario se construye con tecnologías web estándar, está disponible en una URL[2] (pública, privada o protegida por una contraseña) y está optimizada para los dispositivos móviles. Una aplicación web no está instalada en el dispositivo móvil.

(2)URL (*uniform resource locator)*: localizador uniforme de recursos

En el caso de la aplicación web, el sitio puede ser cualquiera, desde una web-anuncio de un pequeño negocio estándar a una calculadora de hipotecas o un controlador diario de calorías (el contenido es irrelevante).

Las **aplicaciones nativas**, por el contrario, están instaladas en el dispositivo móvil, tienen acceso al *hardware* (altavoces, acelerómetro, cámara, etc.) y están escritas en algún lenguaje de programación compilado (como, por ejemplo, el Objective-C).

Diferentes aplicaciones tienen diferentes requisitos. Algunas aplicaciones se adaptan mejor a las tecnologías web que otras. Conocer las ventajas e inconvenientes de cada paradigma os ayudará a decidir qué camino es el más apropiado para cada situación.

La principal ventaja del desarrollo de aplicaciones nativas es que se puede acceder a todas las características *hardware* del dispositivo.

A continuación enumeramos los inconvenientes del desarrollo de aplicaciones nativas:

- La aplicación solo funcionará en la plataforma escogida.
- Hay que desarrollarla usando el lenguaje de programación establecido para la plataforma.
- Es más complicado distribuir parches o actualizaciones que solucionen errores.
- El ciclo de desarrollo es más lento.

Objective-C

Para implementar una aplicación nativa para iPhone hay que programar en Objective-C

Las ventajas del desarrollo de aplicaciones web son las siguientes:

- Los desarrolladores web pueden usar sus propias herramientas.
- Se pueden usar los conocimientos de diseño y desarrollo web que ya se tengan.
- La aplicación funcionará en cualquier dispositivo que tenga un navegador web.
- Se pueden solucionar errores en tiempo real.
- El ciclo de desarrollo es más rápido.

Los inconvenientes del desarrollo de aplicaciones web son los siguientes:

- No se puede acceder a todas las características del dispositivo móvil.
- Puede ser difícil conseguir efectos sofisticados en la interfaz de usuario.

Qué aproximación es la mejor en cada caso es un debate interesante. La naturaleza de los dispositivos móviles, que cada vez más están permanentemente conectados, hace que la línea entre aplicaciones web y aplicaciones nativas se difumine. Incluso hay varios proyectos (entre los que PhoneGap es el más notable) que desarrollan soluciones que permiten a los desarrolladores web coger una aplicación web y empaquetarla como una aplicación nativa, ya sea para iPhone o para otra plataforma móvil.

3. Enumeración de los diferentes entornos

Al igual que el sistema operativo de un ordenador, un sistema operativo móvil es la plataforma *software* que determina las funciones y las características disponibles en el dispositivo, como el control de los teclados, la seguridad inalámbrica, la sincronización con aplicaciones, el correo electrónico, los mensajes de texto, etc. El sistema operativo móvil determina también qué aplicaciones de terceras partes se pueden instalar en el dispositivo. Por lo tanto, cada sistema operativo define unos entornos sobre los que podemos crear aplicaciones. En este apartado vamos a hacer un repaso de los más importantes.

Firmware

El sistema operativo de un dispositivo se conoce en inglés como *firmware*.

3.1. Entornos para dispositivos de diferentes vendedores

En este subapartado estudiaremos las plataformas *software* que pueden funcionar en diferentes plataformas *hardware* de diferentes fabricantes. En concreto, explicaremos la historia y las características principales de las siguientes:

- Java ME
- Symbian
- Android
- Windows Mobile
- Qt framework
- BREW.
- Palm OS

3.1.1. Java ME

En 1999, Sun desarrolló una versión de Java especialmente diseñada para dispositivos móviles, Java 2 Micro Edition, basada en una máquina virtual llamada KVM. Esta primera versión solo contenía una única máquina virtual y un único API (inicialmente diseñados para Palm OS), hecho que puso de manifiesto la insuficiencia de esta solución para la gran variedad de dispositivos diferentes que existían. De esta forma, en el 2000 nació la primera versión de una configuración, el *connected limited device configuration* (J2ME CLDC 1.0). Una configuración ofrece el API básico para programar dispositivos, aunque no aporta todas las clases necesarias para desarrollar una aplicación completa. Por lo tanto, la primera configuración no tenía las herramientas necesarias para permitir a los desarrolladores escribir programas para el dispositivo Palm. En julio del 2000 nació la primera implementación de un perfil, concretamente el llamado *mobile information device profile* (MIDP), que no estaba destinado a PDA, sino a teléfonos móviles y a paginadores. A partir de este primer per-

fil, J2ME fue ampliamente aceptado por la comunidad de desarrolladores de dispositivos móviles y se ha ido expandiendo a una gran velocidad hasta la actualidad.

Java ME[3] (anteriormente conocida como J2ME[4]) es, por lo tanto, una edición de Java orientada a dispositivos pequeños. Es una versión recortada del Java SE con ciertas extensiones enfocadas a las necesidades particulares de este tipo de dispositivos. Esta tecnología consiste en una máquina virtual y en un conjunto de API[5] adecuados para estos dispositivos.

(3)Java ME (Java Micro Edition)

(4)J2ME (Java 2 Platform, Micro Edition)

(5)API (*application programming interface*)

Esta plataforma produce normalmente aplicaciones portables, aunque algunas veces existen librerías específicas de cada dispositivo (comúnmente usadas para juegos), que las hacen no portables. A pesar de ello, Java ME se ha convertido en una buena opción para crear aplicaciones para teléfonos móviles, ya que se puede emular en un PC durante la fase de desarrollo y luego se pueden cargar fácilmente las aplicaciones en el móvil. Aunque el proceso no sea directo, resulta bastante económico portarlas a otros dispositivos al utilizar tecnologías Java para el desarrollo.

Se usa muchas veces para proporcionar aplicaciones simples en teléfonos móviles de gama baja. Por lo tanto, las aplicaciones (incluyendo sus datos) no pueden ocupar demasiada memoria si se tienen que ejecutar en la mayoría de estos teléfonos. Además, tienen que estar firmadas criptográficamente para poder usar APIs como la de acceso al sistema de ficheros. Esto es relativamente caro y raramente se hace, incluso para aplicaciones comerciales. Java ME se ejecuta sobre una máquina virtual que permite un acceso razonable, pero no completo, a las funcionalidades del dispositivo sobre el que se ejecuta la aplicación. El proceso JSR[6] sirve para incrementar gradualmente la funcionalidad disponible para JavaME, mientras proporciona a los operadores y a los fabricantes la capacidad de prevenir o limitar el acceso al *software* disponible.

(6)JSR (*Java specification requests*)

3.1.2. Symbian

La historia de Symbian comienza en el año 1981. En la siguiente cronología podéis ver la evolución del sistema operativo Symbian:

- **1981**. Psion lanza su primer producto, Flight simulator.
- **1984**. Psion Organiser ve la luz.
- **1990**. SIBO SO (16 bits).
- **1997**. EPOC SO (32 bits).
- **1998**. El nombre de Symbian aparece por primera vez.

- **1999**. EPOC versión 5.
- **2000**. Symbian 6.0.
- **2001**. Symbian 6.1.
- **2003**. Symbian 7.0.
- **2004**. Symbian 8.0.
- **2005**. Symbian 9.0.
- **2008**. Nokia compra Symbian Ltd., la empresa que hay detrás de Symbian OS.
- **2009**. Creación de la *Symbian Foundation*.
- **2010**. Se publica el código fuente de Symbian bajo licencia EPL[7].
- **2011**. Nokia realiza una importante alianza con Microsoft y deja de lado el sistema operativo Symbian, que sería reemplazado por el Windows Phone 7.

(7)EPL (Eclipse Public License)

Symbian es un sistema operativo fruto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Psion, Nokia, Ericsson y Motorola, con el que se pretendía desarrollar y estandarizar un sistema operativo que permitiera a teléfonos móviles de diferentes fabricantes intercambiar información.

El Symbian OS fue durante unos años el sistema operativo estándar para los *smartphones* de la época, ya que más del ochenta y cinco por ciento de los fabricantes de estos dispositivos tenían licencias para usarlo. El Symbian OS estaba diseñado para los requisitos específicos de los teléfonos móviles 2.5G y 3G.

Diseñada desde el inicio para dispositivos móviles, la plataforma Symbian es un sistema operativo de tiempo real, multitarea, específicamente pensada para funcionar bien en sistemas con recursos limitados, así como para maximizar la eficiencia y la vida de la batería y minimizar, de esta manera, el uso de memoria. La Symbian Foundation mantiene el código para la plataforma de *software* libre basada en Symbian OS y las aportaciones de *software* de Nokia, NTT DOCOMO y Sony Ericsson, e incluye las interfaces de usuario S60 y MOAP(S). La plataforma es de código abierto en su totalidad, y la mayoría se proporciona bajo la Licencia Pública de Eclipse.

Popularidad de Symbian OS

Se han vendido más de trescientos millones de unidades basadas en Symbian OS. Durante años, Symbian OS ha gozado de más del 50% de cuota de mercado.

El sistema operativo Symbian incorporó el soporte a pantallas táctiles gracias a UIQ[8]. UIQ es una interfaz de usuario gráfica basada en el uso de un lápiz, que se puede encontrar en teléfonos 2.5G y 3G de las siguientes marcas: Motorola, Sony Ericsson, BenQ y ARIMA. Los teléfonos UIQ utilizan pantallas táctiles con una resolución de 208 a 240 x 320 píxeles y una profundidad de color de 12, 16, 18 o 24 bits, dependiendo de la versión de UIQ o del terminal. Las últimas versiones de UIQ fueron las 3.x.

(8)UIQ (*user interface Quartz*): interfaz de usuario Quartz

3.1.3. Android

Android es una plataforma basada en Linux de la Open Handset Alliance, entre cuyos treinta y cuatro miembros se encuentran Google, HTC, Motorola, Qualcomm y T-Mobile. Por lo tanto, treinta y cuatro de las principales compañías de *software*, *hardware* y telecomunicaciones dan soporte a esta plataforma. El *kernel* de Linux se usa como HAL[9]. La programación de aplicaciones se hace básicamente en Java. Es necesario el SDK[10] específico de Android para desarrollar, aunque se puede usar cualquier IDE[11] Java. El código que sea crítico en cuanto a rendimiento se puede escribir en C, C++ u otros lenguajes de código nativo usando el NDK[12] de Android.

(9)HAL (*hardware abstraction layer*)

(10)SDK (*software development kit*)

(11)IDE (*integrated development environment*)

(12)NDK (*native development kit*)

3.1.4. Windows Mobile

La plataforma Windows Mobile estuvo disponible en una gran variedad de dispositivos de diferentes operadores inalámbricos. Se podía encontrar el *software* Windows Mobile en productos de Dell, HP, Motorola, Palm e i-mate. Los dispositivos con Windows Mobile estaban disponibles para redes GSM o CDMA.

Windows Mobile es una variante de Windows CE para teléfonos móviles. Originalmente, Windows CE se desarrolló para ordenadores de bolsillo y PDA con pantallas táctiles que funcionaban con un *stylus*, y se adaptó posteriormente para su uso en *smartphones* equipados con un teclado. Por lo tanto, los teléfonos se convirtieron en la mayor base de dispositivos instalados con CE, aunque la cuota de mercado ha caído dramáticamente desde la aparición de Android y iPhone. Windows Mobile soporta un subconjunto de la interfaz de programación de Win32 y una GUI[13] simplificada con una ventana en la pantalla a la vez. Las aplicaciones se pueden usar en .NET Compact Framework. Windows Mobile 6.5 introdujo las interfaces estilo iPhone basadas en el contacto con los dedos, mientras que Windows Phone 7 es un rediseño sustancial que usa Silverlight y XNA para interfaces de usuario más ricas.

(13)GUI (*graphical user interface)*

3.1.5. Qt framework

Qt usa estándar C++, pero hace un uso extensivo de un pre-procesador especial llamado MOC[14] para enriquecer el lenguaje. También se puede usar Qt en otros lenguajes de programación utilizando enlaces entre lenguajes. Funciona sobre las principales plataformas y tiene un soporte internacional extenso. Entre las

(14)MOC (*meta object compiler*)

características no relacionadas con la GUI, se encuentra el acceso a bases de datos SQL, el tratamiento de XML, la gestión de *threads*, el soporte de red y una API multiplataforma unificada para la gestión de ficheros.

3.1.6. BREW

BREW se usa para aplicaciones en dispositivos CDMA, aunque también soporta modelos GPRS/GSM. Las aplicaciones se distribuyen mediante una plataforma de contenido BREW y han tenido poca penetración en Europa. BREW puede proporcionar control completo del dispositivo y acceder a toda su funcionalidad. No obstante, el potencial que proporciona el código nativo con acceso directo a las APIs del dispositivo ha provocado que el proceso de desarrollo en BREW haya tenido que ser adaptado, en gran medida, para todos los vendedores de *software* reconocidos. Mientras que el SDK de BREW está disponible de forma libre, ejecutar *software* en *hardware* real de un dispositivo móvil (al contrario que el emulador proporcionado) requiere una firma digital que se pueda generar con herramientas publicadas por varios participantes, esencialmente proveedores de contenido para móviles y Qualcomm. Incluso entonces, el *software* solo funcionará en dispositivos habilitados para test. Para que se pueda descargar en teléfonos normales, el *software* tiene que ser comprobado, probado y recibir la aprobación de Qualcomm mediante su programa de testeo TRUE BREW.

3.1.7. Palm OS

Desde la aparición del primer Palm Pilot (en 1996), la plataforma Palm OS ha proporcionado a sus dispositivos móviles herramientas de negocio esenciales, así como la capacidad de acceder a Internet o a una base de datos central corporativa mediante una conexión inalámbrica.

El Palm OS tuvo una gran aceptación empresarial en el importante mercado de EE. UU. basada en las Palm PDA.

Palm webOS es el sistema operativo móvil propietario (evolución de Palm). Funciona sobre un *kernel* Linux que soporta multitarea. Se lanzó con Palm Pre y Pixi. Ahora es propiedad de Hewlett Packard.

3.1.8. Flash lite

Se usa en dispositivos que soportan el reproductor Flash lite.

3.1.9. *Microbrowser*

Los entornos basados en el concepto *microbrowser* proporcionan una funcionalidad limitada mediante una interfaz web.

3.2. Desarrollo multiplataforma

En este subapartado vamos a describir *frameworks* que permiten desarrollar aplicaciones que funcionen tanto en iPhone OS como en Android.

3.2.1. Titanium Mobile

Titanium es un *framework* de código abierto que permite desarrollo multiplataforma. Se puede desarrollar una aplicación que funcione en dispositivos móviles (iOS, Android, RIM) o plataformas de escritorios (OSX, Windows).

Todo el código fuente de la aplicación se escribe en Javascript, CSS y HTML5. Esto es positivo, ya que no es necesario aprender lenguajes complejos como Objective-C o C++.

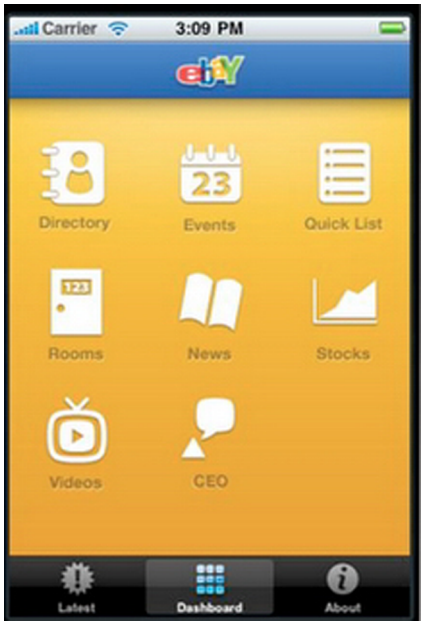
Titanium es extensible, se puede extender el *framework* añadiendo módulos propios en Objective-C o en Java para el caso de Android.

Nota

Existe una gran cantidad de documentación para Titanium.

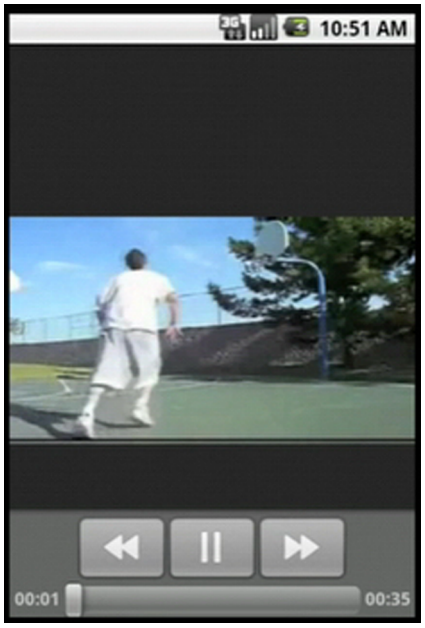
Con Titanium, un desarrollador se puede beneficiar del uso de:

1) Interfaces nativas:



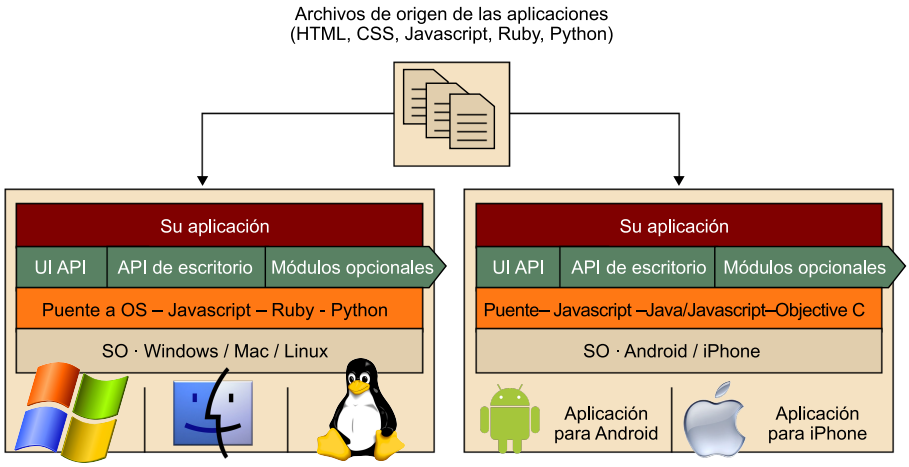
Interfaz nativa

2) Aplicaciones multimedia:



Aplicación multimedia

3) Entorno móvil y de escritorio:



Titanium permite desarrollar tanto aplicaciones móviles como de escritorio

4) Lenguaje Javascript:

> Todo esto es posible gracias a que Titanium tiene un puente que traduce el código Javascript al código equivalente Objective-C o Java en tiempo de ejecución.

3.2.2. PhoneGap

PhoneGap permite desarrollar aplicaciones para Android mediante tecnologías web como HTML, CSS y JavaScript, y puede convertir esas aplicaciones web en aplicaciones nativas Android. De hecho, PhoneGap soporta múltiples

(15)MIT (Massachusetts Institute of Technology)

plataformas (como Android, iPhone, Palm, Windows Mobile y Symbian), así que se puede usar el mismo código fuente para crear aplicaciones para múltiples plataformas. Pese a que se venden como "herramientas de tecnología web", lo que ofrecen PhoneGap u otros *frameworks* similares como Titanium es acceso al *hardware* de la máquina (se pueden hacer aplicaciones en HTML y JavaScript que usen la cámara, la brújula o el acelerómetro). PhoneGap es, además, libre bajo licencia MIT[15].

jQuery i jQTouch

PhoneGap permite desarrollar aplicaciones con librerías de JavaScript como jQuery/jQTouch

Por lo tanto, PhoneGap es una solución de código abierto diseñada para dar acceso JavaScript a los desarrolladores web a características populares de los dispositivos móviles como la cámara, el GPS, el acelerómetro o las bases de datos SQLite locales sin necesidad de tener que escribir aplicaciones completas. El objetivo es hacer más fácil el desarrollo de aplicaciones móviles.

Para conseguir esto, el *framework* PhoneGap actúa como un puente entre las aplicaciones web y los dispositivos móviles. Permite a los desarrolladores envolver aplicaciones web dentro de una aplicación nativa, lo que hace el desarrollo más fácil para aquellos que no están familiarizados con Objective-C y Cocoa.

3.3. Entornos para dispositivos de un vendedor único

Las siguientes plataformas *software* solo funcionan en plataformas *hardware* de un fabricante específico:

1) BlackBerry. Blackberry tiene soporte para correo electrónico, teléfono móvil, mensajes de texto, envío de faxes, navegación por Internet y otros servicios de información inalámbricos, así como una interfaz táctil. Los dispositivos Blackberry disponen de serie de un teclado QWERTY optimizado para utilizarlo tecleando con los pulgares. Cuando aparecieron, los dispositivos Blackberry cogieron pronto una posición dominante en el mercado norteamericano de los *smartphones*. Para Blackberry son importantes el BES[16] y el BlackBerry MDS[17].

(16)BES (*BlackBerry enterprise server*)

(17)MDS (*mobile data system*)

2) iOS de Apple. El SDK para iPhone y iPod usa Objective-C, que está basado en el lenguaje de programación C. En su momento, solo estaba disponible en Mac OS 10.5+ y era la única forma de escribir una aplicación para iPhone. Además, Apple tiene que verificar todas las aplicaciones antes de que se puedan alojar en el *app store*, el único canal de distribución para las aplicaciones para iPhone y iPod *touch*. No obstante, se pueden lanzar aplicaciones para iPhones pirateados no aprobadas por Apple mediante el instalador Cydia. Este sistema se usa también para el *tablet* iPad.

4. Lenguajes de programación

Como ya hemos visto, hay dos clases principales de aplicaciones para dispositivos móviles: las aplicaciones nativas y las aplicaciones web. Un tercer caso de paradigma sería el marcado por Java. Para este caso, muchos de los nuevos móviles soportan alguna versión de MIDP[18], y el desarrollo en este entorno es bastante sencillo. La instalación, no obstante, es algo más complicado. En general, se instalan aplicaciones mediante enlaces en Internet, pero algunos operadores o fabricantes ponen límites a las aplicaciones que se pueden instalar en el móvil.

[18]MIDP (*mobile information device profile*)

En cuanto a las aplicaciones nativas, depende del sistema operativo del móvil. Para muchos, desarrollar aplicaciones nativas puede costar dinero (para herramientas y SDKs), y también hay problemas en lo que respecta a la manera en que se distribuyen las aplicaciones. La instalación y la depuración de errores varían en función del sistema operativo.

Por lo tanto, el lenguaje de programación que se use vendrá probablemente dictado por el dispositivo y la plataforma para la que se desea desarrollar una aplicación, además de por la aplicación que se desea crear.

A continuación enumeramos los diferentes lenguajes con los que se pueden desarrollar aplicaciones nativas para diferentes plataformas:

- Si se quiere hacer una aplicación para iPhone o iPod touch, se usará **Objective-C**.
- Si se quiere hacer una aplicación para Android, se usará **Java**.
- Si se quiere hacer una aplicación para BlackBerry, se usará **Java Micro Edition**.
- Si se quiere hacer una aplicación para Symbian OS, se puede usar **C++**, **Java** o **.NET *Compact Framework***.
- Si se quiere hacer una aplicación parar Windows Mobile, las opciones son **Visual C++**, **Visual C#**, **Visual Basic**, **JScript** y **ASP.NET**.

Así, la plataforma o el dispositivo dictarán qué lenguajes de programación se pueden utilizar. Si se quiere desarrollar para una plataforma que permite tanto C++ como Java, entonces el tipo de aplicación que se planea desarrollar podría dictar qué lenguaje es la mejor opción.

A continuación se muestra una compilación de los lenguajes de programación más populares para dispositivos móviles.

4.1. Lenguajes de programación para Windows Mobile

Se puede escoger entre varias opciones de lenguajes de programación a la hora de desarrollar aplicaciones para dispositivos con Windows Mobile. En este subapartado describiremos brevemente cada una de estas opciones.

4.1.1. Visual C++

Se conoce a C++ como un lenguaje de desarrollo nativo, debido a que interactúa directamente con el *hardware* de un dispositivo Windows Mobile sin que intervenga ninguna otra capa (al contrario que Visual C#, por ejemplo). Programar usando C++ puede ser un desafío, ya que no es un lenguaje fácil de aprender. Algunos errores en un programa C++ pueden, potencialmente, bloquear todo el dispositivo.

Errores en un programa C ++

Acceder a memoria que ha sido liberada u olvidarse de liberar memoria en un programa C++ puede bloquear un dispositivo Windows Mobile

Las ventajas de usar Visual C++ son las siguientes: la velocidad de ejecución, el tamaño de la aplicación y la flexibilidad. Las aplicaciones escritas en C++ se ejecutan muy rápido y consumen los recursos mínimos.

Juegos de acción

Los juegos de acción en tiempo real son buenos ejemplos de programas que se benefician de la velocidad de ejecución de C++

Lectura recomendada

Una buena forma de aprender Visual C++ es investigar la *Visual C++ Express Edition* de Visual Studio (gratuita), ver el vídeo de entrenamiento, los *webcasts*, y leer la documentación. Aunque la *Express Edition* de Visual Studio no permite desarrollar aplicaciones para Windows Mobile, casi todo lo que aprendáis sobre desarrollo de aplicaciones se puede aplicar directamente a dispositivos móviles.

Las aplicaciones Visual C++ pueden interaccionar con el dispositivo Windows Mobile llamando a las APIs Win32. Estas APIs son funciones que realizan acciones concretas, como hacer que se oiga un sonido o dibujar un botón en la pantalla. Hay miles de APIs (Windows Mobile soporta un subconjunto del conjunto completo de APIs Win32 para escritorio) y están documentadas en la sección titulada *Windows mobile features (native)* del MSDN[19] de Microsoft. Cuando se navega por esta sección, hay que tener cuidado con el hecho de que algunas APIs solo están disponibles para Windows Embedded CE, una plataforma que está relacionada (pero separada) de Windows Mobile.

[19]MSDN (*Microsoft developer network*)

Si se tiene experiencia desarrollando para Windows usando Visual C++, la transición a Windows Mobile no es especialmente complicada. Sería necesario aprender a instalar y usar las herramientas específicas y luego aprender a utilizar las características específicas del dispositivo, lo que permitirá explotar las capacidades de los dispositivos.

Para empezar una aplicación Visual C++, hay que arrancar Visual Studio, seleccionar Archivo > Nuevo > Proyecto y seleccionar Smart device en el nodo Visual C++.

Si se es nuevo tanto programando como con Windows Mobile, sería una buena idea empezar con Visual C# y hacer la transición a Visual C++ entonces.

4.1.2. Visual C# y Visual Basic

Visual C# y Visual Basic .Net son lenguajes de desarrollo más sencillos que Visual C++. No solo son relativamente fáciles de aprender, sino que además tienen soporte para el .NET Compact Framework.

.NET Compact Framework

.NET Compact Framework es una librería de clases que realizan gran cantidad de tareas usadas frecuentemente en programación para simplificar el desarrollo de aplicaciones

Las herramientas de desarrollo para C# y Visual Basic .NET incluyen un diseñador completo de interfaz de usuario, WYSIWYG[20]. Podéis arrastrar y colocar botones y otros controles directamente en la ventana de la aplicación, y entonces hacer doble clic para acceder al código que hay debajo. Este sistema hace que crear interfaces de usuario para las aplicaciones sea extremadamente rápido y fácil.

[20]WYSIWYG (*what you see is what you get*)

Como parte de la librería Compact Framework, hay disponibles clases extra que cubren desde las estructuras de datos hasta la intercepción de mensajes de texto. Para hacer uso de las características específicas de Windows Mobile, se proporcionan un conjunto de clases extra. Estas clases proporcionan acceso a las características del dispositivo como, por ejemplo, la lista de contactos o la cámara.

Si se tiene experiencia desarrollando aplicaciones para Windows usando Visual C#, la transición debería ser relativamente sencilla. El Compact Framework es un subconjunto del .NET Framework, por lo que el código puede necesitar de ligeras modificaciones para algunas funcionalidades.

Visual C# es una buena forma de aprender programación. Para aprender todo lo necesario tanto de Visual C# como de Visual Basic, podéis acudir al MSDN de Microsoft.

Para empezar una aplicación Visual C# o Visual Basic .NET, hay que arrancar Visual Studio, seleccionar Archivo > Nuevo > Proyecto y seleccionar Smart device en el nodo correspondiente al lenguaje que nos interese.

4.1.3. JScript

El navegador web incluido en los dispositivos Windows Mobile (Internet Explorer Mobile) soporta JScript. JScript es un superconjunto del lenguaje conocido como JavaScript. Los programas JScript son ficheros de texto plano que ejecuta el navegador web. Pueden estar incrustados en una página HTML o almacenados en ficheros separados.

Una aplicación JScript se ejecuta dentro del navegador web y usa la ventana del navegador web para la entrada y salida de información. Es posible hacer uso de técnicas de programación AJAX[21] para proporcionar un grado de interacción con el usuario y comunicarse con un servidor remoto. Debido a la naturaleza de JScript, las aplicaciones no pueden acceder a datos locales que no sean simplemente *cookies*, lo que introduce algunas limitaciones.

[21] AJAX (*asynchronous JavaScript and XML*): JavaScript asíncrono y XML

Herramienta de desarrollo

No se necesita ninguna herramienta de desarrollo especial: un editor de texto es suficiente para crear una aplicación JScript. Podemos guardar el programa de forma local o acceder a él desde un servidor web.

4.1.4. ASP.NET

Mientras que JScript es una solución del lado del cliente para programas aplicaciones de Internet, ASP.NET es una solución del lado del servidor. Con ASP.NET se pueden escribir aplicaciones en C# o Visual Basic .NET que residan en un servidor web y realicen tareas complejas, como crear controles de interfaz de usuario y acceder a bases de datos. ASP.NET aísla las características del dispositivo de la aplicación y facilita la ejecución de una aplicación en varios tipos de dispositivos diferentes.

5. Ejemplos de entornos

En este apartado describiremos dos entornos de forma más detallada: iPhone / iOS y Android.

5.1. iPhone / iOS

iOS comprende el sistema operativo y las tecnologías que se usan para ejecutar aplicaciones de forma nativa en dispositivos como iPad, iPhone y iPod touch. Aunque comparte una herencia común y muchas tecnologías de base con el Mac OS X, iOS se diseñó para satisfacer las necesidades de un entorno móvil, donde las necesidades de los usuarios son ligeramente diferentes. Si se han desarrollado previamente aplicaciones para Mac OS X, se encontrarán muchas tecnologías familiares, pero también tecnologías que solo están disponibles en iOS, como el soporte a interfaz táctil o al acelerómetro.

El SDK de iOS contiene el código, la información y las herramientas necesarias para desarrollar, probar, ejecutar, depurar errores y adaptar aplicaciones para iOS. Las herramientas Xcode proporcionan el entorno básico para editar, compilar y depurar errores en el código. Xcode también proporciona el punto de lanzamiento para probar las aplicaciones en un dispositivo iOS y en un simulador iOS (una plataforma que imita el entorno básico iOS, pero se ejecuta en un ordenador Macintosh local).

Este subapartado proporciona una descripción detallada de las características básicas que se pueden encontrar en iOS.

5.1.1. Visión general del sistema iOS

iOS (conocido como iPhone OS antes del 2010) es el sistema operativo para dispositivos móvil de Apple. Originalmente desarrollado para el iPhone, se ha ido extendiendo para dar soporte a otros dispositivos Apple (como el iPod touch, el iPad y Apple TV). Apple no da licencias para la instalación de iOS en *hardware* de terceras partes. En enero del 2011, el *app store* de Apple contenía más de trescientas mil aplicaciones iOS, que se habían descargado colectivamente más de diez billones de veces. En el último trimestre del 2010, tenía el 16% de cuota del mercado de los sistemas operativos para *smartphones* (en unidades vendidas), y era el tercero por detrás del Android de Google y Symbian. En el 2010 se llevó el 59% del consumo web móvil (sin incluir el iPad) en Norteamérica.

La interfaz de usuario del iOS se basa en el concepto de manipulación directa mediante la utilización de gestos multicontacto. Los elementos de control de la interfaz consisten en deslizadores, interruptores y botones. La respuesta a las peticiones del usuario es inmediata y proporciona una interfaz fluida. La interacción con iOS incluye gestos como "tocar fuerte", "tocar de forma más débil", "sujetar" y "soltar", que tienen definiciones específicas en el contexto del sistema operativo iOS y su interfaz multicontacto.

Algunas aplicaciones usan los acelerómetros internos para responder cuando se sacude el dispositivo (un resultado común es el comando *deshacer*) o rotar en tres dimensiones (un resultado común es cambiar de orientación vertical a horizontal y viceversa).

iOS está derivado del Mac OS X, con el que comparte la fundación Darwin y es, por lo tanto, un sistema operativo parecido a Unix (por naturaleza).

En iOS hay cuatro capas de abstracción:

- la capa Core OS
- la capa Core Services
- la capa Media
- la capa Cocoa Touch

Espacop d'iOS

El sistema operativo iOS usa escasos quinientos megabytes de la capacidad de almacenamiento del dispositivo, dependiendo del modelo.

5.1.2. Historia del sistema iOS

El sistema operativo apareció con el iPhone en el *Macworld Conference & Expo* en enero del 2007 y fue lanzado en junio de ese año. Al principio, los mensajes de marketing de Apple no especificaban un nombre diferente para el sistema operativo; simplemente decían que el "iPhone ejecuta OS X". Inicialmente, no soportaba aplicaciones de terceras partes. Steve Jobs argumentaba que los desarrolladores podían programar aplicaciones que "se comportarían como aplicaciones nativas en el iPhone". En octubre del 2007, Apple anunció que se estaba desarrollando un SDK nativo y que planeaban ponerlo "en las manos de los desarrolladores en febrero". En marzo del 2008, Apple lanzó la primera versión beta junto con un nuevo nombre para el sistema operativo: iPhone OS.

Las grandes ventas de los dispositivos móviles de Apple encendieron el interés en el SDK. El mes de septiembre anterior, Apple había lanzado el iPod touch, que tenía la mayoría de las capacidades del iPhone no relacionadas con la telefonía. Además, Apple vendió más de un millón de iPhones durante las vacaciones del 2007. En enero del 2010, Apple anunció el iPad, un dispositivo con una pantalla más grande que el iPhone y el iPod touch, diseñado para navegar por Internet, por contenidos multimedia y para lectura de iBooks.

En junio del 2010, Apple renombró iPhone Os como iOS. El nombre IOS había sido usado por Cisco durante una década para su IOS[22], usado en *routers* Cisco. Para evitar pleitos potenciales, Apple pagó la licencia para usar la marca IOS de Cisco.

(22)IOS (*internetwork operating system*): sistema operativo de interconexión de redes

5.1.3. Historia de las versiones del sistema iOS

La versión 4, anunciada en abril del 2010, presentaba multitarea, correo electrónico organizado en hilos y varias características orientadas a los negocios. En el WWDC[23] 2010, Apple anunció que iPhone OS se había renombrado a iOS. Apple pagó la licencia para la marca iOS a Cisco Systems (que posee IOS), la misma compañía con la que Apple había tenido una disputa sobre la marca iPhone. Apple lanzó el iOS el 4 en junio del 2010, tres días antes que el iPhone 4, para reducir la carga en los servidores de Apple. iOS 4 fue la primera versión del sistema operativo que era una actualización gratuita para el iPod touch; Apple cobraba 9,99$ para actualizaciones anteriores. Apple anunció previamente que los usuarios de iPad con *software* 3.x recibirían una actualización gratuita de la siguiente versión importante (4.x).

(23)WWDC (Conferencia Mundial de Desarrolladores de Apple)

iOS 4.0.1 incluía un arreglo para el indicador de la fuerza de la señal recibida. Se lanzó en julio de 2010, el día antes de que Apple realizara una conferencia de prensa para explicar su respuesta a los muy publicitados problemas de la antena del iPhone. Apple también lanzó iOS 3.2.1 para el iPad, que mejoraba la conectividad WiFi, la reproducción de vídeo y el copiar y pegar de archivos PDF[24] adjuntos del *tablet*, además de otras actualizaciones.

(24)PDF (*portable document format*)

En agosto del 2010 se lanzó el iOS 4.0.2 para iPhone y iPod touch, y el iOS 3.2.2 para el iPad, para arreglar algunas vulnerabilidades de seguridad.

En septiembre del 2010 se lanzó el iOS 4.1 para el iPhone y iPod touch; la actualización arreglaba algunos errores detectados por los usuarios, mejoraba la duración de la batería y añadía una nueva característica llamada Game Center, que permitía a los jugadores jugar partidas con otros jugadores, subir puntuaciones altas y desbloquear logros, y añadía soporte inicial para el iPod touch 4th Generation y la Apple TV 2G. iOS 4.1 también añadía fotografía HDR[25], una característica que solo el iPhone 4 era capaz de usar. El iOS 4.1 también añadía una nueva característica, llamada Ping, una herramienta de descubrimiento y red social de música.

(25)HDR (*high dinamic range*): alto rango dinámico

Game Center

Apple acabó quitando el Game Center del iPhone 3G debido a informes de bajo rendimiento.

En noviembre del 2010 se lanzó el iOS 4.2 para los desarrolladores. Nunca se lanzó al público, ya que se encontró un *bug* en la parte de WiFi en la edición limitada. Finalmente, lo que Apple hizo fue lanzar el iOS 4.2.1 al público.

El iOS 4.2.1 se lanzó en noviembre de 2010 con soporte para todos los dispositivos Apple A4, tercera y segunda generación de dispositivos, con la exclusión del Apple TV. Proporcionaba soporte inicial de iOS 4.x al iPad, además de AirPlay y AirPrint a todos los dispositivos compatibles. Además, contiene cambios menores en la aplicación YouTube y modifica la animación de multitarea.

El iOS 4.2.5 se lanzó como versión demo para la versión CDMA del iPhone 4. Esta variante iPhone 4 estaba disponible para los clientes de Verizon Wireless en USA. Esta versión tenía ligeros cambios específicos para la versión CDMA del móvil en la interfaz de usuario.

La versión beta del iOS 4.3 se lanzó a los desarrolladores en enero de 2011.

5.1.4. Características del sistema iOS

Siempre que se enciende el dispositivo o se presiona el botón *Home*, se presenta la pantalla principal con iconos de aplicaciones y un receptáculo en la parte inferior donde los usuarios pueden colocar las aplicaciones usadas con más frecuencia. La pantalla tiene una barra de estado a lo largo de la parte superior para mostrar datos como la hora, el nivel de batería y la potencia de señal. El resto de la pantalla se dedica a la aplicación actual.

Con el iOS 4 llegó la introducción de un sistema de carpetas simple. Se puede arrastrar cualquier aplicación y soltarla encima de otra para crear una carpeta. Una vez hecho esto, se pueden añadir otras diez aplicaciones a la carpeta mediante el mismo procedimiento (una carpeta puede gestionar hasta doce aplicaciones en iPhone y iPod touch y hasta veinte en iPad). Se selecciona un título para la carpeta de forma automática en función del tipo de aplicaciones que hay dentro, pero el usuario puede también editar el nombre.

La pantalla de inicio del iOS contiene aplicaciones por defecto. Algunas de estas aplicaciones no son visibles por defecto. El usuario puede acceder a ellas mediante la aplicación Settings o mediante otro método.

Activación de Nike+iPod

Nike+iPod se activa mediante la aplicación Settings, mientras que AirPrint se activa cuando el usuario imprime un fichero

Todas las utilidades (como notas de voz, calculadora y brújula) están en una carpeta llamada Utilidades en 4.0. Muchas de las aplicaciones incluidas están diseñadas para compartir datos.

Compartición de datos

En el sistema iOS se puede seleccionar un número de teléfono de un correo electrónico y guardarlo como un contacto o marcarlo para hacer una llamada telefónica

El iPod touch mantiene las mismas aplicaciones que están presentes, por defecto, en el iPhone, a excepción de las aplicaciones (anteriores a la cuarta generación) de teléfono, mensajes, brújula y cámara. La aplicación iPod (presente en el iPhone) se divide en dos aplicaciones en el iPod touch: música y vídeos. La fila inferior de aplicaciones se usa para delinear los propósitos principales del iPod touch: música, vídeos, Safari y *app store* (esta configuración se cambió

en la actualización 3.1). Para la cuarta generación de iPod touch, incluye FaceTime y cámara, y la configuración del receptáculo inferior cambia a música, *mail*, Safari, vídeo.

El iPad viene con las mismas aplicaciones que el iPod touch, excluyendo Stocks, Tiempo, Reloj, Calculadora, y la aplicación Nike+iPod. Se proporcionan aplicaciones de música y vídeo por separado, como en el iPod touch, aunque (como en el iPhone) la aplicación de música se llama iPod. La mayoría de las aplicaciones están completamente rehechas para beneficiarse de la pantalla más grande del iPad. La configuración por defecto del receptáculo inferior incluye Safari, *mail*, fotos y iPod.

Multitarea

Antes del iOS 4, la multitarea estaba limitada a una selección de las aplicaciones que Apple incluía en los dispositivos. A Apple le preocupaba que al ejecutar múltiples aplicaciones de terceras partes de forma simultánea se descargara la batería demasiado rápido. A partir del iOS 4, en dispositivos iOS de tercera generación en adelante, hay soporte para multitarea mediante siete APIs:

- audio en segundo plano
- voz sobre IP
- localización en segundo plano
- notificaciones *push*
- notificaciones locales
- finalización de tareas
- cambio rápido de aplicación

Presionar dos veces el botón *Home* activa el intercambiador de aplicación. Entonces aparece una interfaz deslizable desde la parte inferior. Si escogemos el icono correspondiente, se cambia a esa aplicación. A la izquierda hay iconos que funcionan como controles de música. El usuario también puede finalizar aplicaciones.

Game Center

Game Center es una red social de juego multijugador online lanzada por Apple. Permite a los usuarios "invitar a amigos a jugar a un juego, empezar un juego multijugador, controlar los logros y comparar las puntuaciones más altas en un tablón de líderes".

Game Center se anunció durante una presentación de iOS4 en abril del 2010. Se lanzó una versión previa para los desarrolladores registrados de Apple en agosto. Finalmente se lanzó en septiembre del 2010 con al iOS 4.1 en iPhone, iPhone 3GS y iPod touch de la segunda a la cuarta generación. Game Center

hizo su debut público en el iPad con el iOS 4.2.1. No hay soporte para el iPhone 3G y el iPhone original. No obstante, Game Center está disponible de forma no oficial para el iPhone 3G.

5.1.5. Desarrollo de aplicaciones para iOS

Las aplicaciones tienen que estar escritas y compiladas específicamente para iOS y la arquitectura ARM. El navegador web Safari soporta aplicaciones web como otros navegadores. Hay disponibles aplicaciones nativas autorizadas de terceras partes para dispositivos con iOS 2.0 o posterior en el *app store* de Apple.

SDK

En octubre del 2007, en una carta abierta, Steve Jobs anunció que en febrero del 2008 se pondría a disposición de los desarrolladores externos de Apple un SDK. El SDK se lanzó en marzo del 2008 y permite a los desarrolladores hacer aplicaciones para el iPhone y el iPod touch, así como probarlas en un simulador iPhone. No obstante, cargar una aplicación en los dispositivos es solo posible después de pagar al iPhone Developer Program. Desde el lanzamiento de Xcode 3.1, Xcode es el entorno de desarrollo para el iOS SDK. Las aplicaciones iPhone, como el iOS y el Mac OS X, están escritas en Objective-C.

Los desarrolladores pueden poner cualquier precio a sus aplicaciones (por encima de un mínimo) para que se distribuyan en el *app store*, del que recibirán el 70% por cada venta. También pueden optar por lanzar su aplicación de forma gratuita y, de esta manera, no pagan ningún coste para lanzar o distribuir la aplicación, excepto el coste miembro.

5.1.6. *Jailbreaking*

iOS ha estado sujeto a una variedad de diferentes manipulaciones centradas en añadir funcionalidad sin el apoyo de Apple. Antes del debut del *app store* en el 2008, la razón principal para el *jailbreaking* era instalar aplicaciones nativas de terceras partes. Apple dijo que no diseñaría actualizaciones de *software* específicamente para que estas aplicaciones dejaran de funcionar (siempre que no fueran aplicaciones que hicieran desbloqueo de SIM[26]), pero el caso es que con cada actualización de iOS el *jailbreak* parecía dejar de funcionar.

[26] SIM (*subscriber identity module*): módulo de identificación de suscriptor

Desde la llegada del *app store* y las aplicaciones de terceras partes, el objetivo de la comunidad de *jailbreaking* ha cambiado. El principal objetivo del *jailbreaking* es permitir la personalización del dispositivo, usar emuladores y mejoras hechas por la comunidad como multitarea, Adobe Flash Player o acceder al sistema de ficheros del iPhone. La multitarea solo está disponible en dispositivos iOS de tercera generación en adelante, y las aplicaciones en el *app store* no tienen permiso para modificar la apariencia del sistema operativo.

Algunos *jailbreakers* también intentan compartir de forma ilegal aplicaciones de pago del *app store*. Este objetivo ha causado alguna distensión dentro de la comunidad de *jailbreaking*, debido a que no era el objetivo original del *jailbreaking* y es ilegal. Hay también algunos usuarios que se oponen a la censura de contenidos de Apple.

5.1.7. Gestión de derechos digitales

La naturaleza cerrada y propietaria del iOS ha generado críticas, particularmente de abogados de derechos digitales como la Electronic Frontier Foundation, el ingeniero informático y activista Brewster Kahle, el especialista en leyes de Internet Jonathan Zittraion y la Free Software Foundation, que protestó en el evento de presentación del iPad y ha hecho del iPad su objetivo con su campaña *Defective by design*. El competidor Microsoft también ha criticado el control de Apple sobre su plataforma.

En el conflicto están las restricciones impuestas por el diseño del iOS, conocidas como DRM[27], destinadas a bloquear los contenidos que se compran a la plataforma Apple, el modelo de desarrollo (que requiere una suscripción anual para distribuir aplicaciones desarrolladas para el iOS), el proceso de aprobación de aplicaciones centralizado, así como el control general de Apple sobre la plataforma en sí misma. Particularmente en disputa está la capacidad de Apple para inhabilitar o borrar aplicaciones de forma remota.

[27] DRM (*digital rights management*) gestión de derechos digitales

Algunas voces dentro de la comunidad tecnológica han expresado su preocupación por el hecho de que el iOS cerrado represente una tendencia cada vez mayor hacia la visión de Apple de la informática, y hacen notar el potencial de estas restricciones para reducir la innovación en *software*.

No obstante, también hay voces fuera de Apple que han mostrado su apoyo al modelo cerrado del iOS. El desarrollador de Facebook Joe Hewitt, que protestó contra el control de Apple sobre su *hardware* como un "precedente horrible", ha argumentado después que las aplicaciones cerradas en el iPad están relacionadas con las aplicaciones web y proporcionan mayor seguridad.

5.2. Android

Android es una pila de *software* de código abierto para dispositivos móviles que incluye sistema operativo, *middleware* y aplicaciones básicas. Google Inc., compró la empresa desarrolladora inicial del *software*, Android Inc., en el 2005. El sistema operativo de Android está basado en una versión modificada del *kernel* de Linux. Google y otros miembros de la Open Handset Alliance colaboran en el desarrollo y lanzamiento de Android. El AOSP[28] está encargado del mantenimiento y desarrollo de Android.

(28)AOSP (*Android open source project*)

En el cuarto trimestre del 2010, el sistema operativo Android fue la plataforma de *smartphone* más vendida del mundo, destronando al Symbian de Nokia de la primera posición por primera vez en 10 años. Otras fuentes indican que Symbian estaba aún ligeramente por delante en ventas, si se tenían en cuenta algunos teléfonos de modelos antiguos Symbian no Nokia.

Android tiene una gran comunidad de desarrolladores programando *apps*[29], que extienden la funcionalidad de los dispositivos. En el 2010 había alrededor de doscientas mil *apps* disponibles para Android. El *Android Market* es la tienda "en línea" gestionada por Google mediante la que también se pueden descargar *apps* de sitios de terceras partes. Los desarrolladores programan principalmente en el lenguaje Java y controlan el dispositivo mediante librerías Java desarrolladas por Google.

(29)apps: *application programs*

Lanzamiento de Android

La llegada de la distribución Android, en noviembre del 2007, se anunció con la fundación de la Open Handset Alliance, un consorcio de setenta y nueve compañías de *hardware*, *software* y telecomunicaciones, con el objetivo de desarrollar estándares abiertos para dispositivos móviles. Google lanzó la mayor parte del código Android bajo la licencia Apache, una licencia de *software* libre y código abierto.

La pila de *software* del sistema operativo Android consiste en aplicaciones Java que se ejecutan en un *framework* de aplicaciones basado en Java y orientado a objetos encima de librerías base Java, que se ejecutan en una máquina virtual Dalvik, la cual realiza compilación JIT[30]. También hay librerías escritas en C que incluyen el gestor de superficie, el OpenCore Media Framework, el sistema gestor de base de datos relacional SQLite, la API gráfica 3D OpenGL ES 2.0, el WebKit layout engine, el motor gráfico SGL[31], SSL, y Bionic libc. El sistema operativo Android consiste en doce millones de líneas de código que incluyen tres millones de líneas de XML, 2,8 millones de líneas de C, 2,1 millones de líneas de Java y 1,75 millones de líneas de C++.

(30)JIT (*just in time*)

(31)SGL (*scene graph library*)

5.2.1. Historia de Android

En octubre del 2003, Andy Rubin, Rich Miner y otros fundaron Android Inc. en Palo Alto, California (EE. UU).

En palabras de Rubin, el objetivo era desarrollar "dispositivos móviles más elegantes que tuvieran más en cuenta la localización y las preferencias de sus dueños".

Entre otros empleados iniciales importantes se incluyen Andy McFadden, que trabajó con Rubin en WebTV, y Chris White, que lideró el diseño y la interfaz de WebTV antes de ayudar a fundar Android.

Rubin, cofundador de Danger Inc., Miner, cofundador de Wildfire Communications Inc. y vicepresidente de tecnología e innovación en Orange, y los otros empleados iniciales llevaron una considerable experiencia en la industria inalámbrica a la compañía. A pesar de los logros obvios del pasado de los fundadores y de los primeros empleados, Android Inc. funcionó de forma reservada y simplemente admitió que estaba trabajando en *software* para teléfonos móviles.

En agosto del 2005, Google adquirió Android Inc. Los empleados principales de Android Inc., entre los que se encontraban Andy Rubin, Rich Miner y Chris White, permanecieron en la compañía después de la adquisición.

En el momento de la adquisición, debido al poco conocimiento que se tenía sobre el trabajo de Android Inc., se conjeturó que Google estaba planeando entrar en el mercado de los teléfonos móviles.

En Google, el equipo liderado por Rubin desarrolló una plataforma para dispositivos móviles basada en el *kernel* de Linux. Google puso en el mercado la plataforma para los fabricantes de dispositivos móviles y los operadores con la premisa de proporcionar un sistema flexible y actualizable. Google alineó a una serie de socios dedicados a componentes *hardware* y *software* e indicó a los operadores que estaba abierto a varios grados de cooperación por su parte.

Las especulaciones sobre la intención de Google de entrar en el mercado de las comunicaciones móviles continuaron durante diciembre de 2006. Informes de la BBC[32] y The Wall Stret Journal indicaron que Google quería su sistema de búsqueda y sus aplicaciones en teléfonos móviles y que estaba trabajando duro para conseguirlo. Algunos medios de comunicación escritos y "en línea" publicaron rápidamente rumores de que Google estaba desarrollando un dispositivo con la marca de fábrica Google. Algunos especularon que, mientras Google definía especificaciones técnicas, estaba mostrando prototipos a fabricantes de teléfonos móviles y operadores de red.

(32)BBC (British Broadcasting Corporation)

En septiembre del 2007, InformationWeek cubrió un estudio de Evalueserve que indicaba que Google había registrado varias patentes de aplicaciones en el área de la telefonía móvil.

En noviembre del 2007, se anunció a sí misma Open Handset Alliance, un consorcio de varias compañías, entre las que se encuentran Texas Instruments, Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group,

Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel y T-Mobile. El objetivo de la Open Handset Alliance es desarrollar estándares abiertos para dispositivos móviles. El mismo día, la Open Handset Alliance también anunció su primer producto, Android, una plataforma para dispositivos móviles construida sobre la versión 2.6 del *kernel* de Linux.

En diciembre del 2008 se unieron catorce nuevos miembros, entre los que se encontraban PacketVideo, ARM Holdings, Atheros Communications, Asustek Computer Inc., Garmin Ltd., Softbank, Sony Ericsson, Toshiba Corp. y Vodafone Group Plc.

Excepto durante breves periodos de actualización, Android ha estado disponible bajo una licencia de *software* libre de código abierto desde octubre de 2008. Google publicó todo el código fuente (incluyendo las pilas de red y telefonía) bajo una licencia Apache. Google también mantiene pública la lista de problemas revisados para que cualquiera pueda verla y comentarla.

5.2.2. Historia de las versiones de Android

Android ha visto varias actualizaciones desde su lanzamiento original. Estas actualizaciones para el sistema operativo base normalmente arreglan fallos y añaden nuevas funcionalidades. Generalmente, cada nueva versión del sistema operativo Android se desarrolla bajo un nombre código basado en un artículo de postre.

Las versiones más recientes de Android son:

- 2.0/2.1 (**Eclair**), que mejoró la interfaz de usuario e introdujo soporte a HTML5 y Exchange ActiveSync 2.5.

- 2.2 (**Froyo**), que introdujo mejoras de velocidad con la optimización del JIT y el motor Chrome V8 JavaScript, y añadió soporte para Adobe Flash.

- 2.3 (**Gingerbread**), que refinaba la interfaz de usuario, mejoraba el teclado *software* y las características del copiar y pegar, y añadía soporte para NFC.

Glosario

AJAX *f* AJAX (*asynchronous JavaScript and XML*), JavaScript asíncrono y XML, es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*rich Internet applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo cual aumenta la interactividad, la velocidad y la usabilidad en las aplicaciones.

Dalvik *f* Dalvik es la máquina virtual que utiliza la plataforma para dispositivos móviles Android. Dalvik ha sido diseñada por Dan Bornstein con contribuciones de otros ingenieros de Google.

jailbreak *m* *Jailbreak* (en español, escaparse de la cárcel o, más literalmente, rompecárcel) es un proceso que permite a los usuarios de los dispositivos iPhone, iPod touch, iPad y Apple TV de todas las generaciones ejecutar aplicaciones distintas a las alojadas en *app store*, el sitio oficial de descarga de programas para estos dispositivos.

licencia Apache *f* La licencia Apache (Apache *license* o Apache *software license* para versiones anteriores a 2.0) es una licencia de *software* libre creada por la Apache Software Foundation (ASF). La licencia Apache (con versiones 1.0, 1.1 y 2.0) requiere la conservación del aviso de *copyright* y el *disclaimer*, pero no es una licencia *copyleft*, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.

licencia MIT *f* La licencia MIT es una de tantas licencias de *software* que ha empleado el Instituto Tecnológico de Massachusetts (MIT, Massachusetts Institute of Technology) a lo largo de su historia, y quizás sería más acertado llamarla licencia X11, ya que es la licencia que llevaba este *software* de muestra de la información de manera gráfica. En cualquier caso, tanto MIT como X11 tienen el mismo texto.

MIDP *m* El *mobile information device profile* (MIDP) permite escribir aplicaciones descargables y servicios para dispositivos móviles que se conecten a la red.

webcast *m* Un *webcast* es un fichero multimedia distribuido por Internet mediante tecnologías de *streaming* para distribuir una única fuente de contenido a varios observadores simultáneos. Un *webcast* se puede distribuir en directo o bajo demanda.

Bibliografía

Marco, M. J.; Marco, J. M.; Prieto, J. y otros (eds.) (2010). *Escaneando la informática*. Barcelona: Editorial UOC. ISBN: 978-84-9788-110-4.

Stark, J. (2010). *Building iPhone apps with HTML, CSS, and JavaScript*. Editorial O'Reilly. ISBN: 978-0-596-80578-4.

Enlaces de Internet

http://en.wikipedia.org/

http://www.visionmobile.com/

http://blog.abrahambarrera.me/

http://developer.appcelerator.com/

http://www.phonegap.com/

http://softlibre.barrapunto.com/

http://www.nitobi.com/

http://msdn.microsoft.com/

http://forums.techarena.in/

http://stackoverflow.com/

http://developer.apple.com/