



Introducción al entorno de programación de Mac OS X

Acerca de este documento

Mac OS X es el nuevo sistema operativo de Apple. La X de 10 no debe entenderse como una evolución de Mac OS 9, sino como un sistema operativo totalmente distinto, diseñado desde cero, y con una base distinta.

La historia de Mac OS X, empezó con NeXTSTEP, un SO diseñado con fines experimentales y de investigación, creado por Steve Jobs tras abandonar Apple. Cuando se abrió el código fuente de este sistema operativo paso a llamarse OpenStep, y después, tras comprarlo Apple como la base para su nuevo sistema operativo, se le volvió a cambiar el nombre por Rhapsody, y por último, cuando se terminó de retocar para comercializarlo paso a llamarse Mac OS X.

En este reportaje vamos a estudiar las principales características que ofrece este SO de cara al programador.

Nota legal

Este reportaje ha sido escrito por Fernando López Hernández para MacProgramadores, y de acuerdo a los derechos que le concede la legislación española e internacional el autor prohíbe la publicación de este documento en cualquier otro servidor web, así como su venta, o difusión en cualquier otro medio sin autorización previa.

Sin embargo el autor anima a todos los servidores web a colocar enlaces a este documento. El autor también anima a cualquier persona interesada en aprender a programar en Mac OS X a bajarse o imprimirse este reportaje.

Madrid, Enero del 2006

Para cualquier aclaración contacte con:

fernando@DELITmacprogramadores.org

Tabla de contenido

API totalmente orientada a objeto	4
Programando el API.....	4
Cocoa.....	5
Carbon	5
Java	6
Objective-C++	7
Herramientas de programación.....	8
Xcode.....	9
Interface Builder	10
Desarrollo de aplicaciones	11
REALBasic	11
AppleScript	12
Automator	13
Darwin.....	15
Conclusión	15

API totalmente orientada a objetos

A diferencia de las API de otros sistemas operativos tradicionales como Windows, UNIX u OS/2, Mac OS X tiene una API totalmente orientada a objetos, con un diseño realmente bueno, que vamos a ir explicando.

Como ha comentado varias veces Steve Jobs, una API totalmente orientada a objetos es una apuesta arriesgada ya que exige a los programadores un mayor nivel de formación, aunque tiene la ventaja de que el tiempo de desarrollo se reduce mucho. Esto pudo ser un inconveniente a principios de los 90, cuando se empezó a desarrollar NeXTSTEP, pero actualmente la mayoría de las herramientas de programación que se están diseñando usan metodologías de programación orientadas a objetos.

Una apreciación curiosa, como consecuencia de esta clara orientación a objetos, y de la reducción del coste de desarrollo que implica, es que en el mundo de Mac OS X la mayoría de las aplicaciones están siendo desarrolladas por programadores individuales, a diferencia de los grandes equipos de trabajo que suelen desarrollar las aplicaciones más conocidas en otros entornos.

Programando el API

En Mac OS X, aunque da soporte a muchos lenguajes de programación, los lenguajes más usados para trabajar directamente con su API son dos: Objective-C y Java.

Objective-C es un superconjunto de C que amplía a éste para convertirle en un lenguaje orientado a objetos. Es decir, tiene la misma raíz que tuvo C++, pero incluye conceptos más avanzados que C++ como son los protocolos o interfaces, y es mucho más dinámico que C++, en el sentido de que toda la toma de decisiones se realiza en tiempo de ejecución, y se dejan pocas decisiones para el compilador. Esto da al programador mucha más flexibilidad en la creación de estructuras de datos dinámicas y de objetos polimórficos.

Aunque no formaba parte del NeXTSTEP original, Apple ha introducido un segundo lenguaje mucho más popular: Java. En principio, la función de Java no es la programación de aplicaciones en red como podría pensarse, ya que Objective-C permite programar las mismas aplicaciones, sino la adopción de estándares y la posibilidad de exportar e importar aplicaciones a otros entornos fácilmente.

A diferencia de otros sistemas operativos Mac OS X trae instalado Java "de fábrica" con lo cual siempre podemos ejecutar una aplicación Java sin necesidad de instalar previamente la máquina virtual.

Cocoa

La API de programación de Mac OS X se llama Cocoa (chocolate, suponemos que para complementar el café que tomaban los programadores de Java) y como dijimos antes es una API totalmente orientada a objetos, es decir formada básicamente por clases. A esta API se la considera un framework en el sentido de que sus clases ayudan al programador a montar el esqueleto de su aplicación.

Una característica importante de esta API es que es accesible tanto desde Objective-C como desde Java, lo cual es muy útil ya que reduce considerablemente el tiempo de aprendizaje del framework. Hay que tener en cuenta que en la actualidad las librerías de programación son tan grandes, que su aprendizaje nos lleva mucho más tiempo que el aprendizaje del lenguaje de programación en sí.

La librería Cocoa está dividida en dos kits:

Application Kit Este kit contiene todas las clases relacionadas con la interfaz gráfica (ventanas, botones, cajas de texto, etc.). Pero el Application Kit es mucho más que un conjunto de componentes de interfaz gráfica, también nos da acceso a OpenGL, o tiene clases para gestión de documentos (mediante el patrón vista-documento usado también en otros muchos frameworks de programación), o por ejemplo, también trae clases que encapsulan fuentes, colores, impresión, corrección ortográfica o drag and drop.

Foundation Kit Este kit contiene todas las demás clases no visuales como son las clases de acceso a ficheros, de programación multihilo, de trabajo en red, de utilidad, de gestión de cadenas, etc.

Carbon

Carbon es el nombre de la API que mantiene compatibilidad con la antigua API de Mac. Para facilitar la migración al nuevo SO a las aplicaciones Mac actuales, Apple ha puesto esta API en Mac OS X. Se trata de un API programable basada en funciones C.

Los nuevos programadores que lleguen a Mac OS X no deben de estudiar esta API, ya que con el tiempo Apple acabará dejando de darla soporte y luego quitándola.

La estrategia es la misma que la que aplicó Microsoft cuando migro de Win16 a Win32, pero con la diferencia de que Win32 está ahí para quedarse, y Carbon acabará siendo sustituida por Cocoa.

Java

Por un lado la máquina virtual de Apple dispone de todas las librerías estándar del Java 2 SDK 1.4.2 (AWT, Swing, Servlets, JSP, J2EE...), y de hecho está certificada por Sun como una máquina virtual 100% pure Java. Esto permite ejecutar aplicaciones Java hechas en otros entornos en Mac OS X, y viceversa.

Pero además desde Java podemos acceder a todas las clases de la API Cocoa (véase Figura 1), lo cual permite realizar aplicaciones Java prácticamente tan rápidas como sus correspondientes aplicaciones Objective-C.

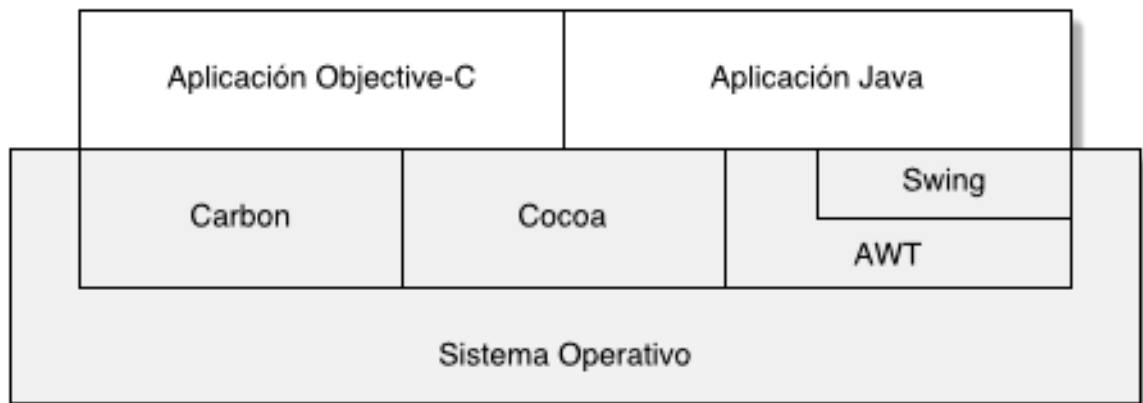


Figura 1: Java puede acceder directamente a Cocoa, además de disponer de las librerías gráficas estándar AWT y Swing.

De todos es conocida la lentitud que caracteriza a las aplicaciones gráficas hechas con Swing. Pero este problema desaparece al utilizar Cocoa, la API nativa de Mac OS X.

No sólo las aplicaciones Java sobre Cocoa se ejecutan muy rápidamente, sino que además la implementación de la máquina virtual de Apple está mucho más optimizada que en otros SO como Windows o Linux, esto es debido a que la máquina virtual de Apple accede directamente a las capas de rederning del SO también llamadas Core Services (p.e. el window server), que son capas del SO a las que el programador de aplicaciones no accede directamente, sino que accede siempre a través de la API del SO (véase figura 2). Esta es una ventaja que se debe al hecho de que la máquina virtual de Apple esté implementada por ingenieros de Apple. En otros SO como Windows, Sun tiene que conformarse con acceder a los servicios de Windows a través de la API de Win32.

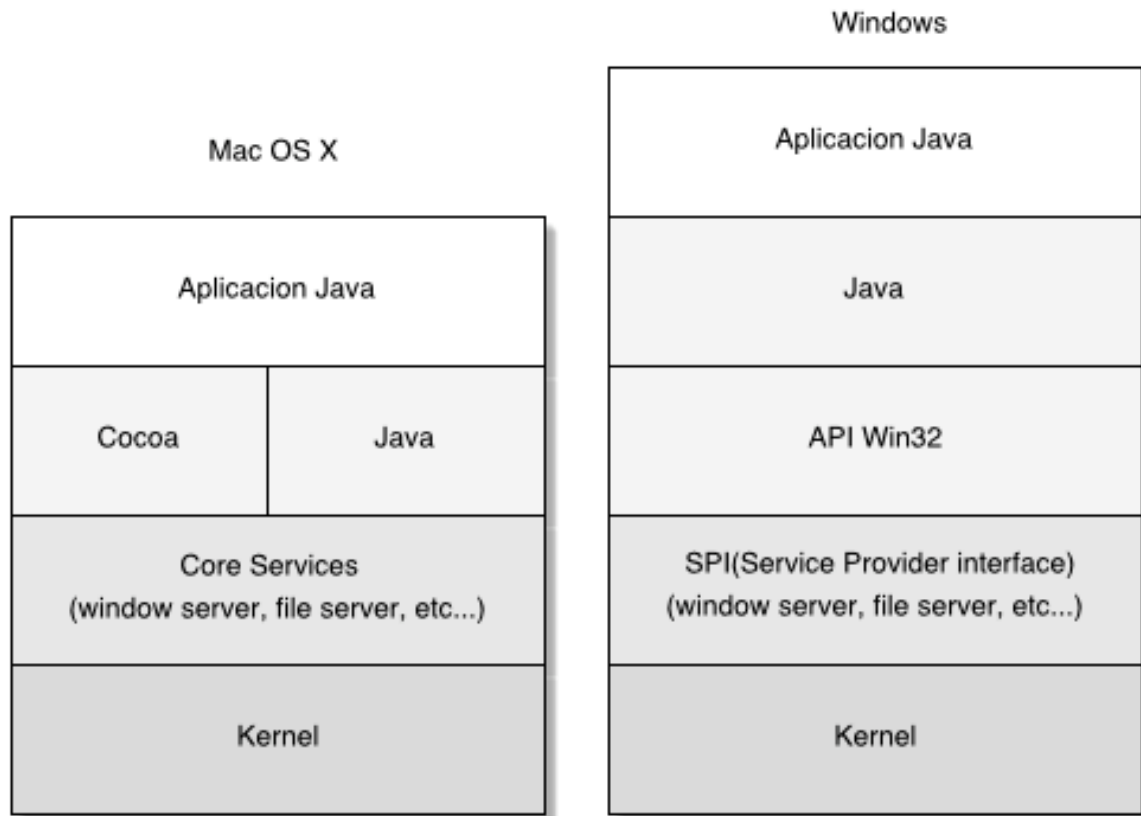


Figura 2: La máquina virtual de Apple no pasa por la API, a diferencia de otras máquinas virtuales como la de Sun para Windows.

No sólo las librerías de interfaz gráfica son más rápidas, sino que también el acceso a ficheros o a red está optimizado, ya que la máquina virtual Java de Apple no utiliza los Core Services para realizar esta tarea, sino que accede directamente al núcleo del SO para solicitar estos servicios (véase Figura 3).

Objective-C++

Objective-C++ es un lenguaje que surgió en los tiempos de NeXTSTEP. Este lenguaje permite mezclar código fuente C++ y Objective-C en el mismo fichero, así como llamar a objetos C++ desde Objective-C y viceversa. Esto permite al programador Objective-C utilizar todas las librerías de C++ existentes, y al programador C++ acceder a todas las librerías de Cocoa. Los ficheros Objective-C++ llevan la extensión `.mm` ó `.M` (extensión antigua que se mantiene por compatibilidad).

Aunque inicialmente el compilador de Mac OS X no soportaba Objective-C++, a partir de Septiembre del 2001 Apple, tras las continuas peticiones recibidas por parte de los desarrolladores, anunció que lo incluiría y así lo hizo metiéndolo como parte del `gcc` a partir del `gcc 2.95.2`, que venía con Mac OS X 10.1

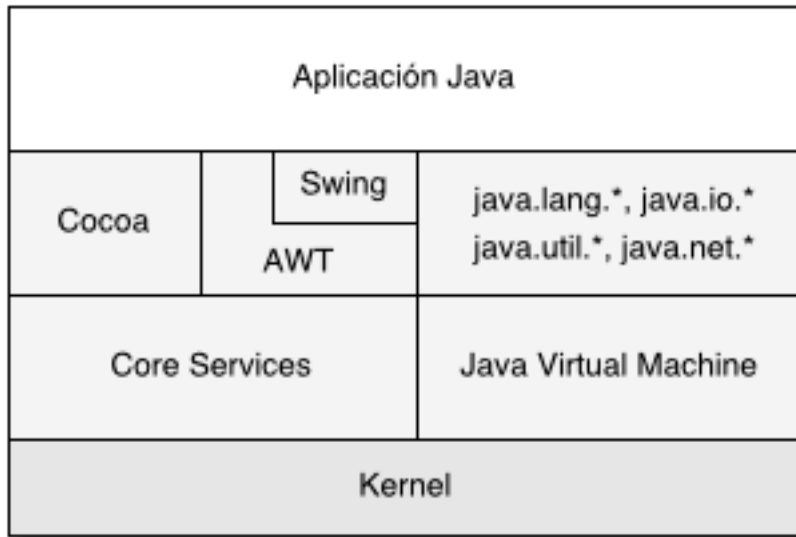


Figura 3: La máquina virtual Java de Apple accede directamente al núcleo de Mac OS X en el acceso a ficheros o red.

Herramientas de programación

Para programar bajo Objective-C todo lo que necesitamos es un viejo conocido, el comando `gcc` de GNU. Esto es así porque Steve Jobs cedió la implementación de Objective-C a GNU cuando se diseñó OpenStep, y actualmente Apple y GNU siguen colaborando para mejorar Objective-C.

Los programadores Java también están de enhorabuena porque pueden programar con los comandos `java` y `javac`, que suelen venir actualmente en casi todos los kits de desarrollo Java.

Aun así Mac OS X tiene todo un kit de herramientas de programación visual e IDE (Integrated Development Environment), que además Apple distribuye gratuitamente con sólo darnos de alta en la ADC (Apple Developer Connection) en <http://developer.apple.com/>

En las llamadas Mac OS X Developer Tools destacan dos aplicaciones: Xcode e Interface Builder, las cuales vamos a comentar con más detalle.

Si este documento le está resultando útil puede plantearse el ayudarnos a mejorarlo:

- Anotando los errores editoriales y problemas que encuentre y enviándolos al sistema de Bug Report de Mac Programadores.
- Realizando una donación a través de la web de Mac Programadores.

Xcode

Xcode, es un IDE en el sentido clásico que nos permite organizar nuestros ficheros por proyectos, compilarlos y depurarlos con todo tipo de ayudas y detalles (véase Figura 4).

Esta herramienta trabaja con ficheros Java (.java), C/C++ (.h, .c, .cpp), Objective-C (.h, .m) y Objective-C++ (.h, .mm, .M), e incluso podemos hacer un programa parte en Objective-C y parte en Java, lo que demuestra el grado de integración que hay entre Objective-C y Java en Mac OS X.

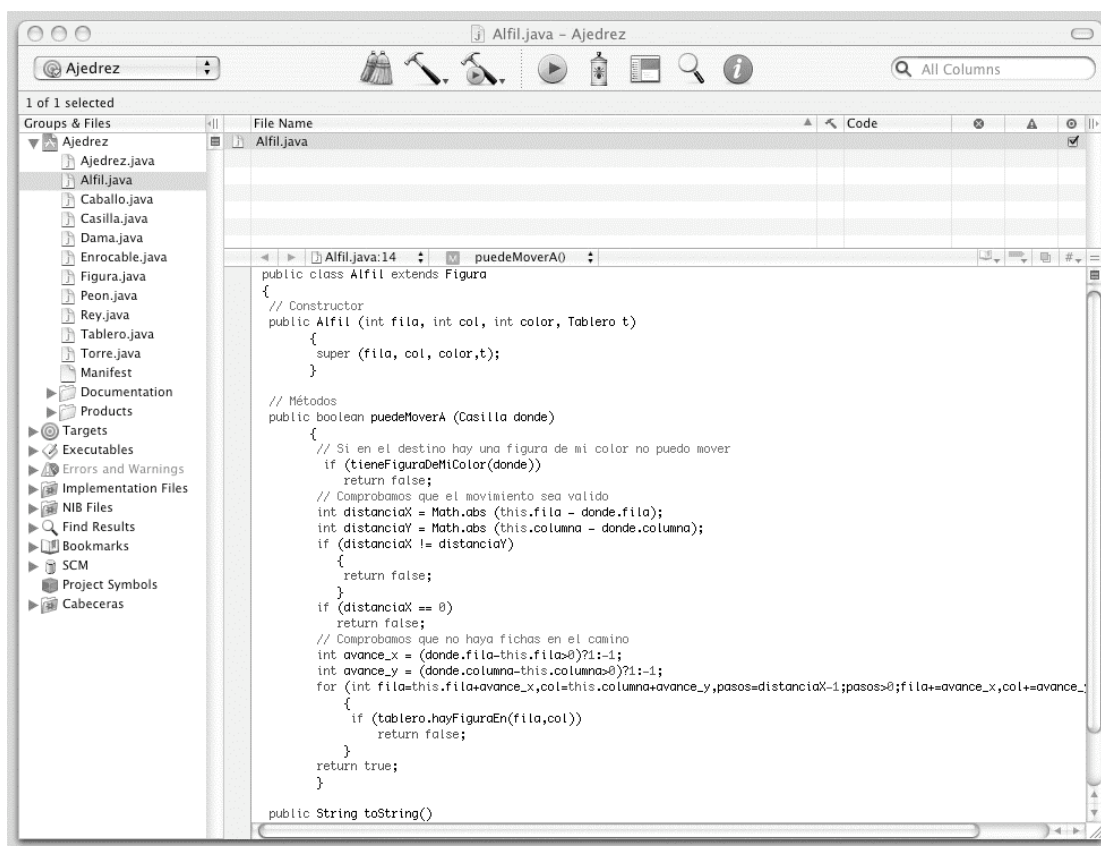


Figura 4: Xcode es el IDE de Apple para Mac OS X.

Interface Builder

Interface Builder (véase Figura 5), es una herramienta de desarrollo visual que nos ayuda a crear rápidamente la parte de la interfaz gráfica de la aplicación. Esta herramienta además de código, genera ficheros `.nib` que son ficheros con una descripción de los componentes de interfaz gráfica, que se cargan al arrancar la aplicación y que usa Cocoa para pintar la ventana y sus componentes en pantalla, es decir, es el equivalente a los recursos en otros sistemas.

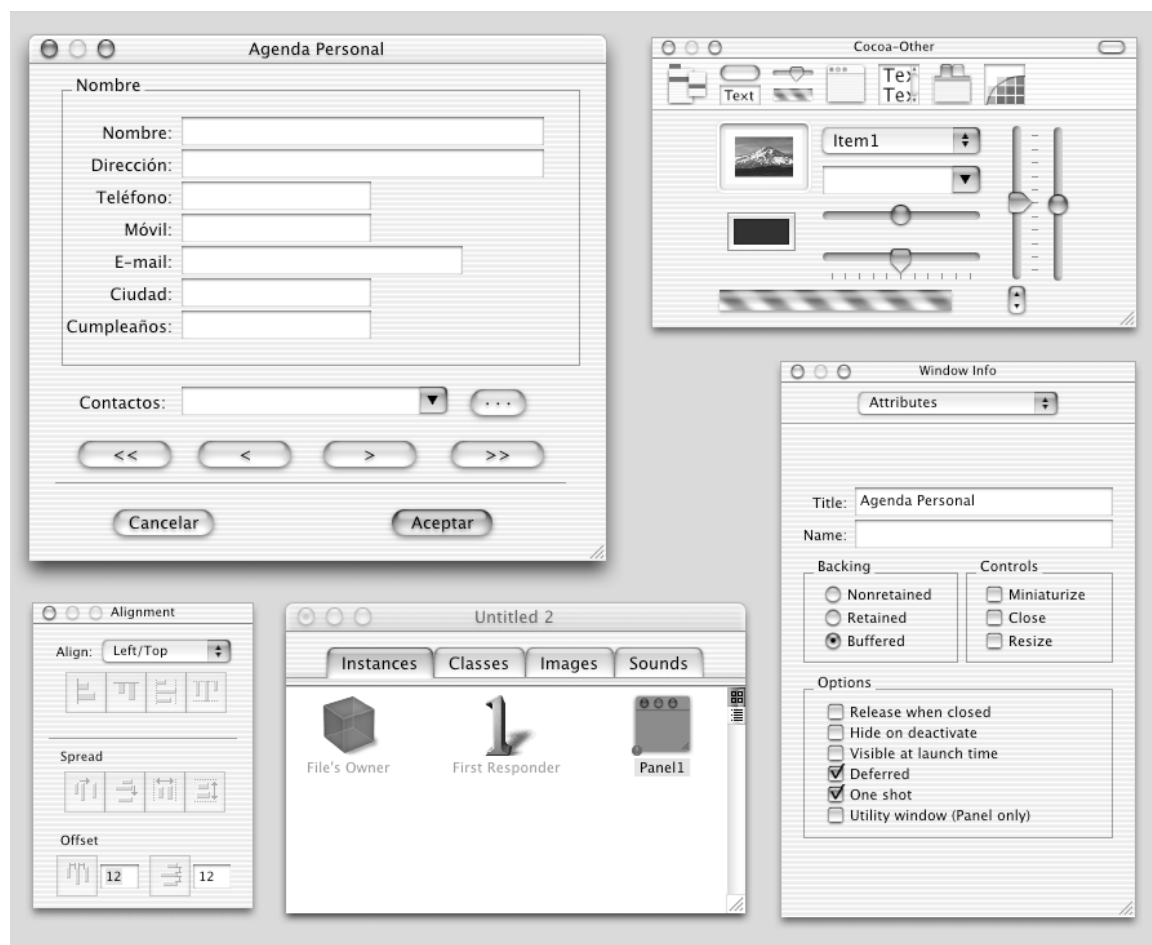


Figura 5: InterfaceBuilder permite crear visualmente la interfaz gráfica de una aplicación.

Desarrollo de aplicaciones

Programar la API de un sistema operativo es una forma de tener acceso a toda la funcionalidad que promociona ese sistema operativo. Pero, si lo que queremos es productividad, necesitamos trabajar con herramientas de programación de más alto nivel, y sólo en casos excepcionales hacer una llamada al API, pidiéndole realizar una operación un poco más extraña que el lenguaje de alto nivel no proporciona.

En este apartado vamos a comentar algunas herramientas de programación de alto nivel de que dispone Mac OS X.

REALBasic

Si lo que quiere es desarrollar aplicaciones de gestión, posiblemente la mejor opción sea programar con REALbasic. Un lenguaje sencillo y un entorno orientado a componentes que se van colocando en un formulario, y que permiten relacionar fácilmente el contenido de los componentes con el de una base de datos (véase Figura 6). Actualmente REALbasic es un entorno de programación muy completo, y permite realizar casi cualquier tipo de aplicación, con lo que también resulta viable usarlo para desarrollo de otro tipo de aplicaciones.

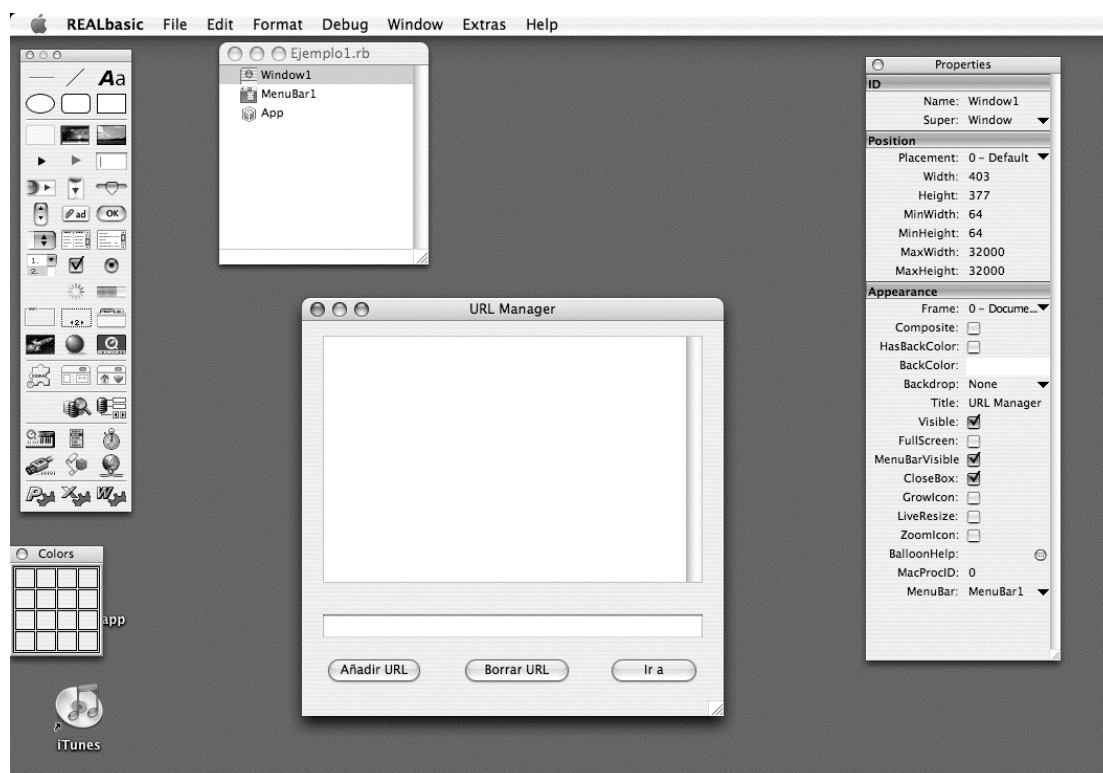


Figura 6: REALbasic permite crear visualmente la interface gráfica de una aplicación.

Otra gran ventaja de este entorno es que a partir del mismo código fuente puede generar ejecutables para Mac OS X, Microsoft Windows y Linux. Esta capacidad lo hace ideal para el desarrollo de aplicaciones que quieren atacar distintos nichos de mercado.

AppleScript

AppleScript es un lenguaje de programación de script con una sintaxis muy fácil de leer porque utiliza frases similares a las del lenguaje natural. Por ejemplo, una frase escrita en este lenguaje sería:

```
set content of text field "saldo" to 0
```

Aunque el lenguaje es fácil de leer, no lo es tanto a la hora de escribirlo debido a que las frases tienen verbos, artículos, adjetivos, sustantivos, etc.



Figura 7: Script Editor es un pequeño IDE para AppleScript

La principal utilidad de AppleScript es que permite realizar fácilmente comunicación interprocess usando mensajes llamados **Apple events**. La mayoría de las aplicaciones Mac OS X son AppleScript compatibles, lo cual significa que son capaces de responder a Apple events enviados desde un

script o desde otra aplicación. Tanto Carbon como Cocoa proporcionan un framework en el que es fácil hacer a una aplicación scriptable.

Los eventos AppleScript no sólo se pueden enviar entre procesos de la misma máquina, sino entre procesos de distintas máquinas, para lo cual usan el protocolo SOAP (el estándar de comunicación RPC propuesto por XML).

Apple script usa **Open Scriptable Architecture (OSA)** para transmitir los eventos del script a la aplicación. Aunque OSA está pensado para que se puedan enviar eventos desde distintos lenguajes de script (p.e. Python o Perl), actualmente el único lenguaje de script capaz de enviar Apple Events a las aplicaciones es AppleScript.

Para crear, ejecutar, y depurar los scripts se usa un pequeño editor llamado Script Editor (véase Figura 7), aunque también es posible crear scripts con Xcode. De hecho actualmente es posible compilar el fichero de script para generar una aplicación Mac OS X directamente ejecutable.

Automator

Automator es un intento de poder programar sin tener que aprender ningún lenguaje, simplemente encadenando pequeños módulos, llamados **acciones**, que realizan operaciones concretas, y que agrupándolos en serie resuelven un problema. A esta agrupación de tareas se le llama **workflow**. Una vez que hemos creado un workflow podemos guardarlo y, o bien generar una aplicación Mac OS X ejecutable, al hacer doble click en ella, o bien generar un plug-in que se ejecute desde otra aplicación (p.e. Finder).

Para crear un workflow se usa una herramienta llamada Automator, que vemos en la Figura 8. La idea es muy similar a la de la redirección de pipes en los shell de script: Cada acción recibe unos datos de entrada, realiza una operación con ellos, y devuelve otros datos de salida que se pasan a la siguiente acción. Existen muchos tipos de datos que pueden entrar/salir de las acciones, y además existen reglas de conversión que permiten convertir entre datos parecidos.

La principal innovación de Automator es que consigue hacer todo esto sin usar sentencias de control de flujo (*if*, *while*, etc.), sino únicamente conectando visualmente las acciones.

Aunque en el momento de escribir este reportaje todavía no hay muchas acciones (en torno a 300 acciones), se espera que en breve vayan surgiendo nuevas acciones que permitan realizar casi cualquier tarea con Automator.

Si una aplicación es AppleScript compatible cualquier desarrollador AppleScript puede realizar acciones Automator que accedan a las operaciones de la aplicación. Sin embargo muchas aplicaciones incluyen su propio paquete

de acciones con el fin de facilitar la interacción entre la aplicación y Automator.

Si quiere contribuir desarrollando acciones puede hacerlo usando tres posibles lenguajes: Cocoa, AppleScript, o un lenguaje de script tradicional como Perl, Python o Bash. Además Apple está ayudando a mantener el sitio www.automatorworld.com donde se están publicando gran cantidad de acciones y workflows.

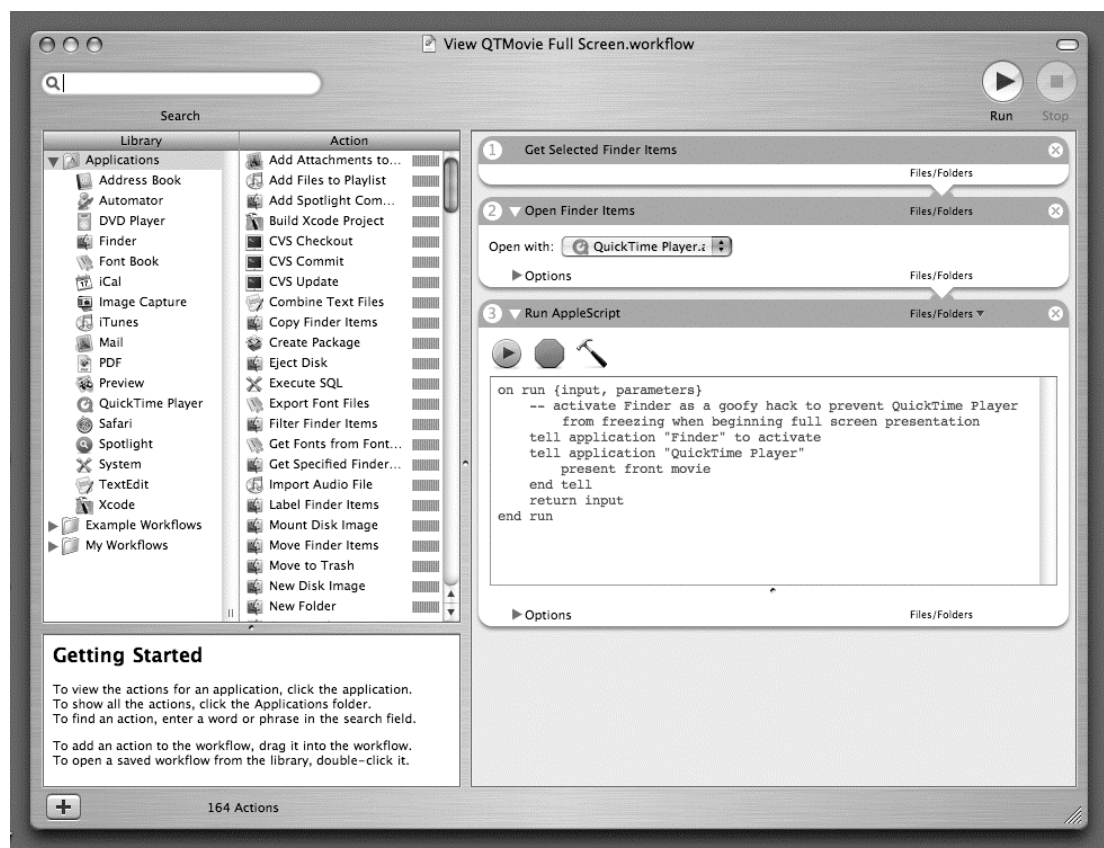


Figura 8: Con Automator podemos crear aplicaciones simplemente encadenando acciones

Darwin

Darwin es el nombre del kernel de Mac OS X. El kernel de Mac OS X, a diferencia del de otros sistemas operativos comerciales, es un proyecto open source mantenido en la web de Apple.

Darwin fue desarrollado por los ingenieros de Apple basándose en dos implementaciones de kernel muy conocidas: Mach 3.0, un microkernel modular desarrollado en la Universidad de Carnegie Mellon, y en el kernel BSD (Berkeley Software Distribution) para lo cual se utilizó el código fuente de FreeBSD 4.4, la versión libre del UNIX de Berkeley.

Mach es el núcleo de Darwin, y realiza las operaciones críticas del núcleo, como son la gestión de memoria, la gestión de procesos e hilos (protección de memoria, scheduling), interrupciones o las operaciones de entrada / salida con los periféricos.

BSD aparece como una capa por encima de Mach que permite a los programadores usar las conocidas llamadas al sistema POSIX de los sistemas UNIX. Estas funciones nos permiten entre otras cosas la gestión de procesos por ID, dar permisos a estos procesos, gestión de señales, o el uso de sockets BSD.

La principal ventaja que aporta BSD es que permite migrar fácilmente aplicaciones UNIX a Mac OS X. Normalmente basta con recompilarlas sin necesidad de hacer más cambios.

Además FreeBSD se caracteriza por ser uno de los sistemas operativos más estables que existen, y por disponer de una pila de TCP/IP superoptimizada que le hace tener un muy buen rendimiento como servidor.

Conclusión

Mac OS X es un gran sistema operativo que acerca muchas nuevas tecnologías tanto al usuario como al programador. Lo mejor es sin duda sus interfaces de programación renovadas, pequeñas, flexibles y bien estructuradas. También es destacable la estabilidad y buen rendimiento del sistema. Respecto al software disponible, el usuario no tendrá dificultad para encontrar software que resuelva los distintos problemas, aunque la cantidad de software sin ser mala no alcanza ni mucho menos a la cantidad de software disponible para Windows. Lo peor sigue siendo un precio demasiado elevado y la pequeña difusión que tiene el Mac en España.

Madrid, Enero del 2006
Fernando López Hernández