

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Introducción a MacOS y XCode

Introducción a MacOS

El sistema operativo Mac OS X fue **diseñado por Apple**.

Apple Inc. fue fundada en **1976** por **Steve Wozniak**, **Steve Jobs** y **Ron Wayne**, aunque la cara más conocida de Apple siempre fue **Steve Jobs**, ya que era quien anunciaba todos los productos y se encargaba del **marketing** y desarrollo de negocio de la empresa. No obstante, tanto Steve Wozniak como Ron Wayne fueron igual de relevantes y cruciales que Steve Jobs.

Apple creó el Apple I, un pequeño ordenador que ofrecían a sus, aún, pocos clientes, llegando a unas 200 ventas, aproximadamente. Fue con el dinero que obtuvieron con el Apple I con el que consiguieron desarrollar el Apple II para seguir con el Apple III, el **Apple III+** (que **corregía gran cantidad de desaciertos** de su antecesor), y el **Lisa**.

Pero no fue hasta la época de **1980** que no desarrollaron el **primer Macintosh**, el cual ya contaba con un sistema operativo con **interfaz gráfica**. Fue aquí cuando nació el **sistema operativo de Apple**.

Este fue el comienzo de los Macs, desarrollados íntegramente por la compañía Apple, la cual decidió llevar una fortísima política de **encapsulación y hermetismo**, y de desarrollar sus **propios ordenadores y sus sistemas** operativos, teniendo fuertemente acaparadas todas sus **patentes**.

Fue en el año **1999**, con la aparición de **MacOS X**, cuando decidieron dejar de desarrollar sus sistemas operativos desde cero y empezar a **basarse en UNIX** para ello.

Los primeros ordenadores Macintosh utilizaban un procesador llamado **PowerPC**, que, en aquella época, era de los que mejores prestaciones presentaban, hasta que **Intel** lo sobrepasó, y Apple, viendo esto, decidió pasar a desarrollar sus ordenadores con procesadores de Intel.

Los procesadores de los nuevos MacBook

Ya hemos comentado que Apple fabrica sus ordenadores, tanto de sobremesa como portátiles, basados en microprocesadores Intel.

Todo esto va a cambiar a partir de la salida de su **nuevo sistema operativo**, el llamado **Big Sur**, el cual estará optimizado para sus **propios procesadores**, dejando de colaborar con **Intel** después de tantos años.

Versiones de iOS y MacOS X

El **sistema operativo** que Apple usa para sus **smartphones** se llama **iOS**.

Apple suele anunciar sus **nuevas versiones** del sistema operativo **iOS** coincidiendo con el anuncio de algún **nuevo smartphone**. Estos lanzamientos se hacen en una **conferencia llamada WWDC**, celebrada en **California**, EEUU.

Las **versiones** que podemos encontrar hasta la fecha del sistema operativo iOS son las que podemos ver en la siguiente tabla.

Versiones del sistema operativo iOS

Versión	Fecha de salida
iOS 1	Junio 2007
iOS 2	Junio 2008
iOS 3	Junio 2009
iOS 4	Junio 2010
iOS 5	Octubre 2011
iOS 6	Septiembre 2012
iOS 7	Junio 2013
iOS 8	Septiembre 2014
iOS 9	Septiembre 2015
iOS 10	Septiembre 2016
iOS 11	Septiembre 2017
iOS 12	Junio 2018
iOS 13	Septiembre 2019
iOS 14	Septiembre 2020
iOS 15	Junio 2021
iOS 16	Septiembre 2022

Al igual que iOS, el sistema operativo MacOS X está desarrollado en versiones, las cuales son:

Versiones del sistema operativo MacOS X

1. Mac OS X **10.0 (Cheetah)**
2. Mac OS X 10.1 (**Puma**)
3. Mac OS X 10.2 (**Jaguar**)
4. Mac OS X 10.3 (**Panther**)
5. Mac OS X 10.4 (**Tiger**)
6. Mac OS X 10.5 (**Leopard**)
7. Mac OS X 10.6 (**Snow Leopard**)
8. Mac OS X 10.7 (Lion)
9. Mac OS X 10.8 (**Mountain Lion**)
10. **Mac OS X 10.9 (Mavericks)**
11. Mac OS X 10.10 (**Yosemite**)
12. Mac OS X **10.11 (El Capitán)**
13. Mac OS X 10.12 (**Sierra**)
14. **Mac OS X 10.13 (High Sierra)**
15. Mac OS X 10.14 (**Mojave**)
16. Mac OS X 10.15 (Catalina) (hasta Catalina son consecutivas, desde 10.0 a 10.15)
17. Mac OS X 11.6 (Big Sur)
18. Mac OS X 12.4 (**Monterey**)

¿Qué versión de MacOS X es mejor para desarrollar?

Es normal que cuando queremos adentrarnos en un sistema operativo que no conocemos, investiguemos un poco las distintas versiones que posee. Ocurre lo mismo con los sistemas operativos GNU/Linux y Windows, que tienen varias versiones e incluso distribuciones, en el caso de GNU/Linux.


MacOS X es algo distinto. Todas las versiones que existen no atienden a diferentes sistemas operativos, como ocurre con Windows XP, 7, 10..., sino que se corresponden a un único sistema operativo que ha ido **evolucionando** a lo largo de los años.

Debido a esto, lo más recomendable para empezar a usarlo y desarrollar aplicaciones será tener el **último sistema operativo que esté disponible**, ya que, además, las **nuevas actualizaciones** de las aplicaciones **no serán compatibles** con sistemas operativos de un **tiempo atrás**, ya que se quedarán **obsoletos** y no serán compatibles con ellas.










Atajos de teclado en MacOS

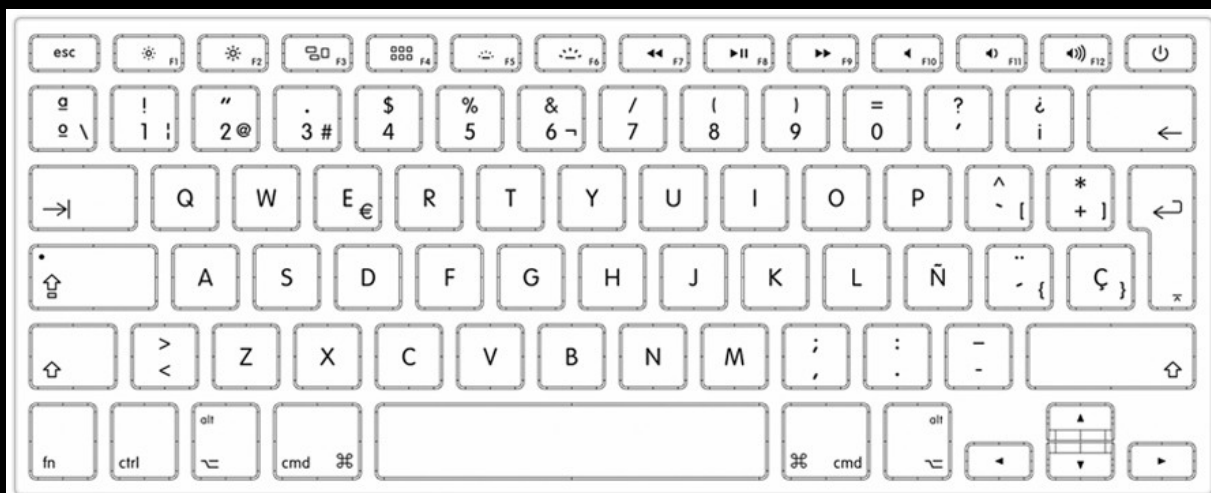
Todos conocemos los típicos atajos de teclado que nos ayudan a realizar las tareas de una forma mucho más simple, como el tan recurrido “control + c” para copiar o “control + z” para deshacer.

El sistema operativo MacOS X también posee una serie de atajos, pero **no son los mismos** a los que estamos acostumbrados si usamos GNU/Linux o Windows.

Para empezar, los teclados de Apple no son iguales que los teclados “normales” que todos estamos acostumbrados a utilizar, ya que estos disponen de una **tecla especial** llamada “**comando**”, cuyo símbolo es , que se usará para los atajos en este tipo de ordenadores.

Una lista de los atajos más comunes que vamos a usar es la siguiente:

- Comando () + C: Copiar (equivalente a Cntrl + C)
- Comando () + X: Cortar (equivalente a Cntrl + X)
- Comando () + V: Pegar (equivalente a Cntrl + V)
- Comando () + Z: Deshacer (equivalente a Cntrl + Z)
- Comando () + R: Esta combinación nos permitirá arrancar desde el **sistema recuperación** de MacOS.
- Opción (): Esta combinación nos permitirá arrancar el **Gestor de arranque**, que permite elegir **otros discos o volúmenes de arranque** si están disponibles.
- Mayúsculas (): Arranca en **modo seguro**.
- Comando () + S: Esta combinación nos permitirá arrancar en **modo de usuario único**. Es una combinación de teclas que requiere macOS **High Sierra** o una versión **anterior**.
- T: Arranca en modo de **disco de destino**.
- Comando () + V: Arranca en **modo detallado**.



Cerrar aplicaciones correctamente en MacOS X

Vamos a ver cómo podemos cerrar una aplicación correctamente en nuestro sistema operativo Mac.

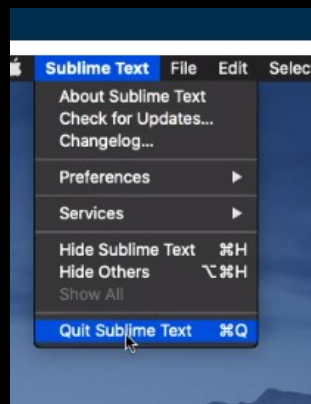
Aquí podemos ver que tenemos el escritorio y abajo nuestro **dock** con las aplicaciones que tenemos con acceso directo. Si pulsamos sobre alguna de ellas, por ejemplo, Sublime Text, esta aplicación se abrirá normalmente.

Si estamos acostumbrados a sistemas operativos como **Windows y GNU/Linux**, lo normal en estos sistemas operativos es cerrar la aplicación mediante el **botón de cerrar y se cerrará definitivamente**, pero en el sistema operativo Mac esto no es así.

En el sistema operativo Mac, si cerramos la aplicación directamente desde su botón de cerrar, ocurre lo siguiente: vemos que Sublime Text tiene un **punto debajo**, eso quiere decir que **se sigue ejecutando en segundo plano**. Con esto, lo que conseguimos es que si volvemos a seleccionar que **se abra**, lo haga **mucho más rápido**.



Si queremos cerrar la aplicación completamente, tendremos que venir a este **menú superior izquierdo**, que será el menú de nuestra aplicación que tenemos abierta. Seleccionar el nombre de la aplicación y la última opción siempre será cerrar la aplicación ('Quit' seguido del nombre del programa). Si pulsamos aquí, podemos ver que la aplicación se ha cerrado y debajo no tiene ningún punto blanco que indique que se está ejecutando en segundo plano.



Lenguajes de programación para aplicaciones en smartphones iOS

Ya sabemos, de unidades anteriores, que para desarrollar aplicaciones en Android, podemos utilizar los lenguajes Java y Kotlin, además del XML para la implementación de las interfaces gráficas.

Para el desarrollo de las aplicaciones para iOS, también podremos elegir entre dos lenguajes de programación, como son **Objective-C** y **Swift**, aunque **en el núcleo** de estos dispositivos se encuentra **Cocoa Touch**, la cual no es más que la **API** que **permite desarrollar las aplicaciones** para los dispositivos del sistema operativo de Apple.

Arquitectura de Apple

Objective-C		Swift	
Foundation	UIKit	...	
Cocoa Touch			
IOS			

Como podemos intuir de la imagen, la **API de Cocoa Touch** debe ser **enorme** y llena de **funcionalidades**, ya que es la encargada de hacer que **todo funcione** correctamente.

Los lenguajes de programación propiamente dichos, **Objective-C** y **Swift**, son los que nos van a permitir **“controlar” la API de Cocoa Touch** para que haga lo que necesitemos. (Se podría decir que Cocoa Touch es al sistema Mac lo que es “C” a los Sistemas Microsoft y Linux.)

En un principio, las aplicaciones se desarrollaban solo en Objective-C, aunque hace relativamente poco tiempo, se introdujo la posibilidad de poder desarrollar en **Swift**, un lenguaje mucho **más evolucionado y de alto nivel que el Objective-C**. Esto no quiere decir que Objective-C no se use en la actualidad, todo lo contrario, todavía hay multitud de aplicaciones que se desarrollan o están desarrolladas en este lenguaje, por la simple razón de que al llevar tanto tiempo utilizándose, hay gran cantidad de tareas que están desarrolladas en él y se pueden reutilizar.

No obstante, el **futuro** del desarrollo de aplicaciones para smartphones Apple es el lenguaje **Swift**, ya que ofrece muchas más **ventajas respecto a Objective-C**. Algunas de ellas:

- El código es mucho más **conciso**.
- Gestiona **automáticamente** la **memoria**.
- Ofrece un **tipado fuerte** de datos.

Instalación de XCode

XCode es el entorno de desarrollo oficial que nos ofrece Apple para poder desarrollar aplicaciones tanto para el sistema operativo **MacOS X** como para sus **smartphones**. Tenemos que tener en cuenta que no podremos desarrollar aplicaciones para los sistemas operativos de Apple **en ningún otro programa que no sea XCode**.

Para poder instalar XCode, necesitamos cumplir los siguientes requisitos:

1. Un **ordenador Mac** con la versión del sistema operativo actualizada a **la más reciente**.
2. Una **cuenta** creada y validada de **iTunes**.
3. La aplicación **App Store** instalada en nuestro Mac, la cual ya viene por defecto en la instalación, y con la **cuenta de iTunes** con la sesión **iniciada**.

Cuando cumplamos los requisitos anteriores, ejecutamos la App Store.

Una vez iniciada, la localizamos con el buscador de apps mediante su nombre: XCode.

Una vez que hayamos seleccionado XCode en la App Store, deberemos pulsar en "Obtener" para que comience la descarga del programa y su instalación.

El proceso de **instalación** de XCode es **muy largo**, ya que este entorno de desarrollo es bastante completo y deberá descargarse en su totalidad antes de poder instalarse.

¿Dónde instalar XCode?

Ya conocemos ciertas características sobre la compañía Apple, desarrolladora de los iPhones, MacBook, etc., y un aspecto que no deja indiferente a nadie que quiera iniciarse en su sistema operativo, o desarrollar apps para sus dispositivos móviles, es el hermetismo del que se dispone en la compañía.

No podremos instalar sus sistemas operativos en otros PC que no sean su marca, por incompatibilidades de hardware. Estos problemas no solo se limitan al aspecto del sistema operativo, sino que con sus aplicaciones ocurre lo mismo, y XCode no es una excepción.

Si queremos instalar XCode para aprender a desarrollar aplicaciones para los smartphones de Apple, no podremos hacerlo a menos que dispongamos de un ordenador Apple, ya que XCode no está disponible para ningún otro sistema operativo.

Creación de un proyecto en XCode

Una vez que ya tenemos instalado XCode en un Mac, estamos en condiciones de crear un nuevo proyecto. Para ello pulsamos sobre la opción

“Create a new Xcode Project”

y seleccionamos el tipo de app

“Single View App”

que es una aplicación vacía.

Aquí tendremos que cumplimentar los siguientes [datos](#):

- **Product Name:** Este es el nombre con el que vayamos a nombrar a la app.
- **Organization Name:** Este es el nombre del **desarrollador** de apps; en nuestro caso, podemos poner nuestro nombre.
- **Organization Identifier:** Este es el **identificador** del desarrollador de apps.
- **Language:** Lenguaje en el que va a estar programada la app. Podremos elegir entre Objective-C y Switt. Nosotros elegiremos siempre Swift.

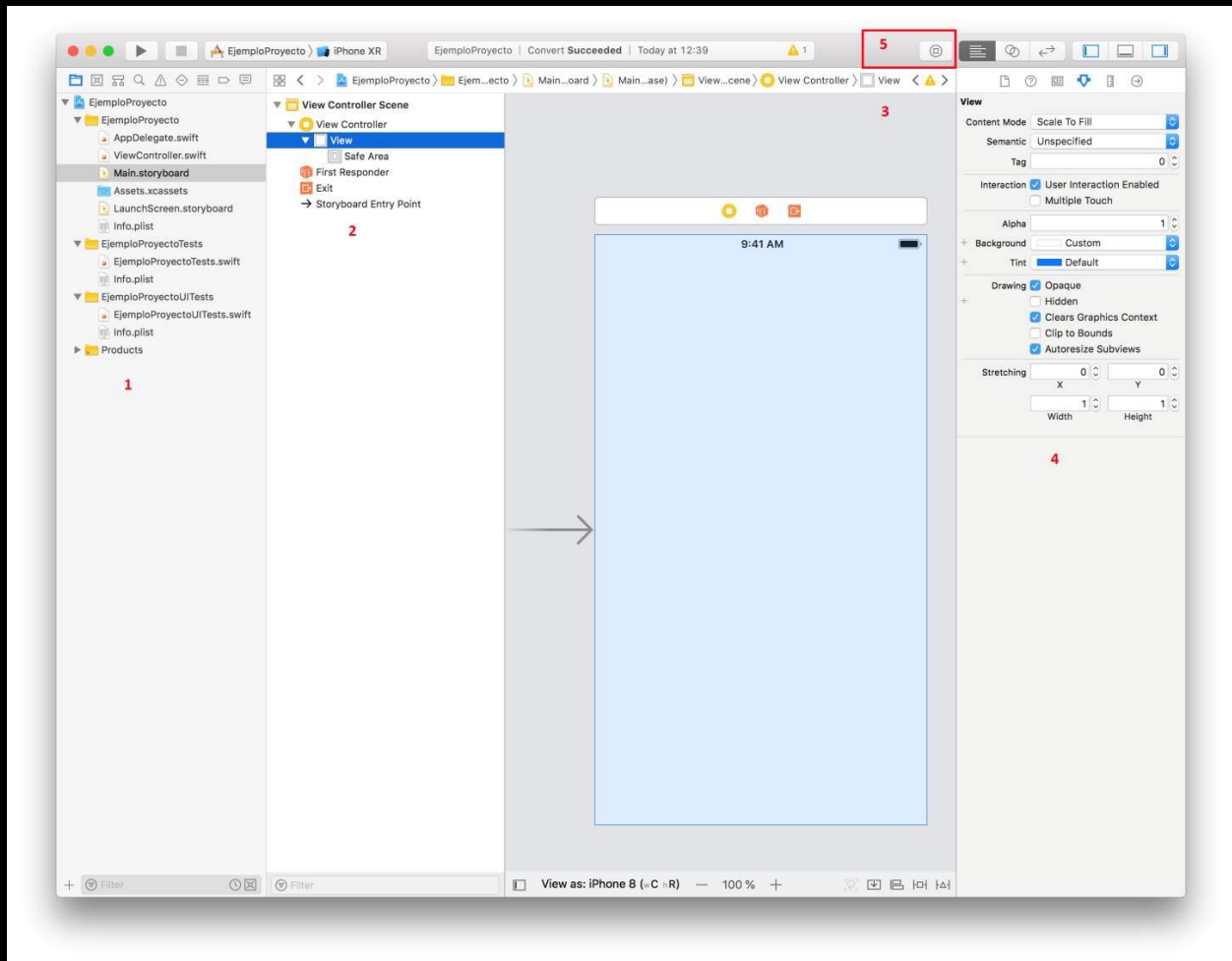
Una vez cumplimentado todo esto, pulsamos el botón

“Next”

y seleccionamos donde queremos guardar el nuevo proyecto.

Una vez creado el proyecto, pulsamos en **Main.storyboard** y podremos ver una pantalla como la de la siguiente figura:

Proyecto vacío en Xcode



1. En esta área (de la izquierda), tenemos la **estructura** con todos los ficheros .swift y carpetas que componen nuestro proyecto.
2. En esta zona (a continuación a la derecha de la estructura), se sitúan los **elementos gráficos** que componen cada una de las vistas de nuestra app.
3. Esta sección contiene las vistas de nuestra app en “**vista diseño**”.
4. En el margen **derecho**, se sitúan las **propiedades de los elementos** gráficos y vistas de nuestra app.
5. En esta posición (visible el cambiador de posición en la parte superior), tenemos la **paleta de elementos** gráficos que podemos insertar en nuestra app.

Creación y ejecución de un proyecto en XCode

Vamos a ver cómo podemos crear un proyecto de Xcode y sus partes.

1. **Abrir Xcode:** Para abrir Xcode, simplemente necesitaremos hacer clic en su icono y el programa se abrirá.

2. **Crear un nuevo proyecto:** Una vez abierto Xcode, vamos a crear un nuevo proyecto siguiendo estos pasos:

- Pulsamos en "Crear un **nuevo proyecto**" y seguimos el asistente.
- Elegimos la opción de "**Aplicación vacía**".
- Introducimos el nombre de nuestra aplicación (por ejemplo, "Aplicación Prueba").
- Seleccionamos el lenguaje (en este caso, Swift) y la interfaz (Storyboard o SwiftUI). Nosotros elegiremos Storyboard.
- Introducimos la información de la organización y pulsamos en "Siguiente".
- Elegimos la ubicación donde queremos guardar el proyecto (por ejemplo, en el escritorio), luego pulsamos en "Crear".

3. **Explorar el proyecto:** Una vez creado el proyecto, podemos explorar sus partes:

- Abrimos el "Main Storyboard" para ver la interfaz gráfica del proyecto, que inicialmente será una pantalla vacía.
- Podemos ver tres partes principales:
 - El árbol de archivos y carpetas del proyecto.
 - La interfaz gráfica donde previsualizamos el diseño de la aplicación.
 - El panel de propiedades de los componentes de la interfaz.

4. **Editar el código:** Para editar el código, simplemente hacemos clic en el archivo correspondiente y se abrirá en el editor de código, que estará en Swift.

5. **Ejecutar la aplicación:** Para ejecutar la aplicación en un emulador, seguimos estos pasos:

- Seleccionamos el emulador en el que queremos ejecutar la aplicación (por ejemplo, iPhone 8).
- Pulsamos el botón de ejecutar y se abrirá un emulador donde se instalará y ejecutará la aplicación.
- Esperamos a que se inicie y cargue la aplicación en el emulador.

¡Listo! Ahora tenemos nuestra aplicación ejecutándose en el emulador.

Reflexión

En el mundo de las aplicaciones móviles, ya sabemos que existen dos gigantes: Android y Apple.

Las aplicaciones Android ya las dominamos de las unidades anteriores, pero si nos estamos planteando dedicarnos a ello profesionalmente, ¿qué ocurre con las aplicaciones para Iphone?, ¿merecerá la pena desarrollar aplicaciones para este sistema, dado el tiempo que requerirá aprender, y la inversión que se necesitaría llevar a cabo?

Es verdad que ambos sistemas operativos ofrecen un nicho de mercado muy amplio por separado, pero también cabe destacar, que cuantos más servicios pueda ofrecer una empresa de desarrollo, más posibilidades de negocio tendrá. Por tanto, si nos estamos planteando **crear una empresa de desarrollo de apps para móviles**, sería muy recomendable saber hacerlo para las **dos plataformas**.

Por otra parte, si no es a través de una empresa propia, sino que salimos al mundo laboral a buscar trabajo en otras empresas, un **programador polivalente**, que domine **los dos sistemas**, tendrá muchas **más posibilidades** de encontrar un empleo, así que podemos deducir que estaría bien dominar las dos marcas, aunque estemos **especializados más en una** que en otra.