



Programar en Mac

C-Objective • Cocoa • iPhone/iPad SDK | el manual en español

Un manual de



AppleNeXt



Introducción

Vivimos tiempos de moda para los usuarios del Mac, los 'Macusers'. Cuando Apple lanzó por primera vez el Mac OS X dio comienzo a una nueva era para la programación en Mac y para el mundo de Apple. Al margen de la sólida estructura basada en UNIX, el Mac OS X abrió el camino a nuevas funcionalidades y posibilidades que para los Macusers eran tan sólo un sueño hace pocos años.

Gracias a este nuevo sistema operativo, Apple fijó su mirada en los 'developers' (desarrolladores o coloquialmente, programadores) y desde entonces, no ha hecho más que profundizar en esa relación.

El elemento principal de esta fructífera relación entre Apple y los programadores es Cocoa, el modo de programación en Mac (un conjunto de API's) basado en el lenguaje de programación C-Objective, que a su vez es una potente evolución del Lenguaje C.

Nuestro objetivo con este manual es aprender a utilizar Cocoa para programar en Mac. Pero antes de nada, sepamos que es Cocoa.



Mac OS X es un sueño para cualquier programador

Programar en Mac nunca ha sido tan sencillo y accesible como tras la aparición de Mac OS X. Para los principiantes, nuestros queridos y simpáticos amigos de Apple tuvieron la acertada idea de poner a nuestra disposición los 'Developer Tools', las herramientas para programación. Apple nos ofrece de forma gratuita el Xcode y las demás herramientas para programar en Mac que podemos descargar de la página web de la ADC (Apple Developer Connection), de la que hemos hablado ya en AppleNeXt.com.

Descargando e instalando el Xcode tendrás acceso de forma inmediata a una amplia colección de utilidades, herramientas, vídeos, documentación y ejemplos de código para comenzar a programar tus aplicaciones para Mac, iPhone, iPod Touch o iPad. Como hemos dicho en tantas ocasiones, basta con que sepas programar en Mac, para que también sepas programar para iPhone, iPod Touch y el nuevo y revolucionario iPad. ¡Toda una ventaja!

Antiguamente, un kit de herramientas como las que podrás descargar en la ADC de Apple costarían muchos miles de euros. Sin embargo, ahora Apple te las dará gratis.



En las versiones anteriores de Mac OS X, el kit de herramientas para desarrolladores venía en un CD separado. Pero las nuevas versiones no. Normalmente tendrás la opción de instalarlos con el DVD de instalación de tu Mac, pero si no quieres andar buscando el DVD en el baúl de los recuerdos, podrás descargarlo en internet. ¿Cómo? Tan sencillo como dándote de alta como miembro gratuito de la Apple Developer Connection: developer.apple.com

De hecho, aunque tengas un disco con las herramientas de programación, te conviene echar un vistazo a la ADC para ver si tu versión está actualizada, puesto que Apple suele renovar esas herramientas y mejorarlas cada poco tiempo. Conviene que hagas de la ADC tu página de referencia para programar porque será la mejor manera de que estés al día.

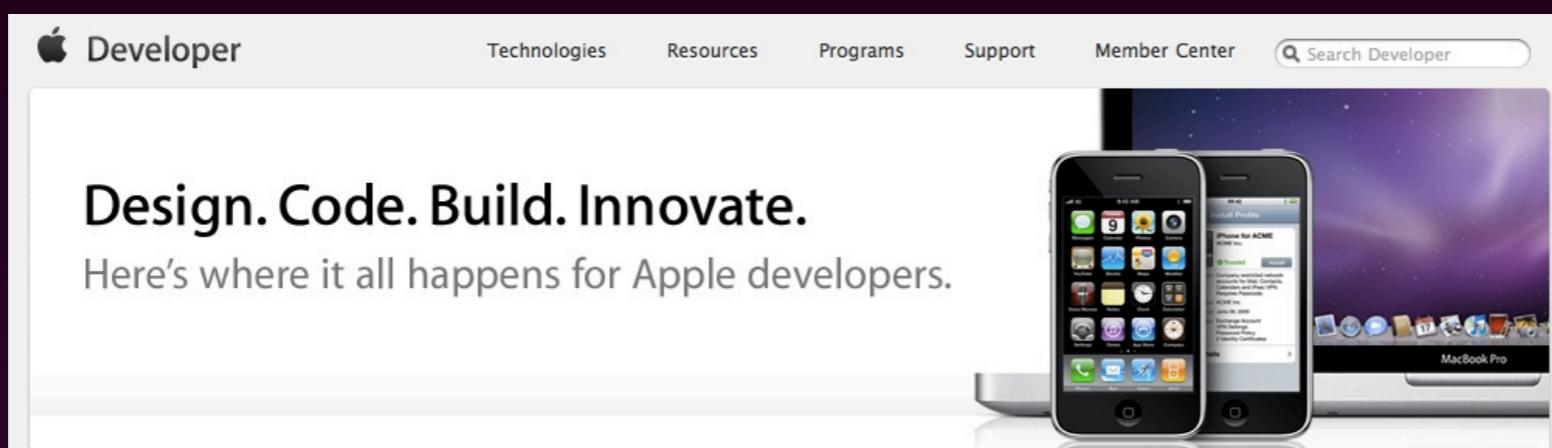
Aunque el Xcode y las demás herramientas para programar en Mac sean gratuitas no por ello significa que sean de peor calidad. Más bien, todo lo contrario. Con el [Xcode de Macintosh](#) podrás sacar mucho provecho a las [siguientes posibilidades](#):

- [Escribir códigos de programación](#) en una amplia variedad de lenguajes diferentes: C, Objective-C, Python, Ruby, Java, y como no, AppleScript.
- [Crear atractivos interfaces](#) siguiendo las líneas del Human Interface Guidelines de Apple. Para eso utilizarás la herramienta llamada 'Interface Builder' que viene incluido en el kit de herramientas.
- [Programar interesantes aplicaciones](#), muchas de las cuales podrás utilizar en nuevos códigos sin la necesidad de volverlos a escribir.

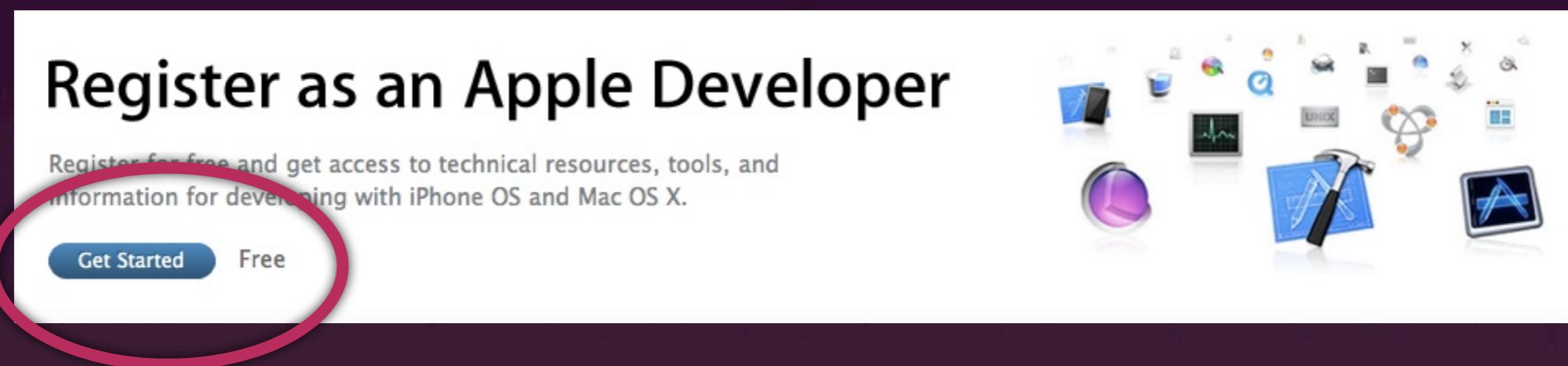


Por lo tanto:

1. Entrar en la Apple Developer Connection (ADC): developer.apple.com



2. Registrarse como usuario: Podrás elegir uno de los programas de programadores: Gratuito





3. Descargar el Kit de herramientas para Desarrollo de Software

Te habrás dado cuenta que Kit para Desarrollo de Software en inglés es: *Software Development Kit*, es decir, el **SDK**, que te sonará de algo, ¿no?



El SDK del iPhone es el mismo para iPad e iPod Touch. Además incluye el Xcode, que es la aplicación necesaria para programar en Mac OS X. Por lo tanto, descargando la última versión del iPhone SDK no necesitarás nada más para empezar a programar para Mac.



iPhone SDK

Aunque ya lo hemos explicado en AppleNeXt.com, para los despistados vamos a recordar qué es el iPhone SDK.

Se trata de un kit de herramientas necesarias para la Programación en Mac (en concreto para programar Apps para el iPhone, pero las herramientas para el iPhone son las mismas que para Mac OS X). Descargando el iPhone SDK podrás programar tanto para iPhone como para Mac OS X, iPad o iPod Touch. Durante este manual nos referiremos a él como iPhone SDK o iPad SDK indistintamente.

El iPhone SDK se puso a disposición de los desarrolladores un año después del lanzamiento del primer iPhone, el 8 de marzo de 2008. Además se anunció el lanzamiento de la AppStore, la principal tienda de Apps.



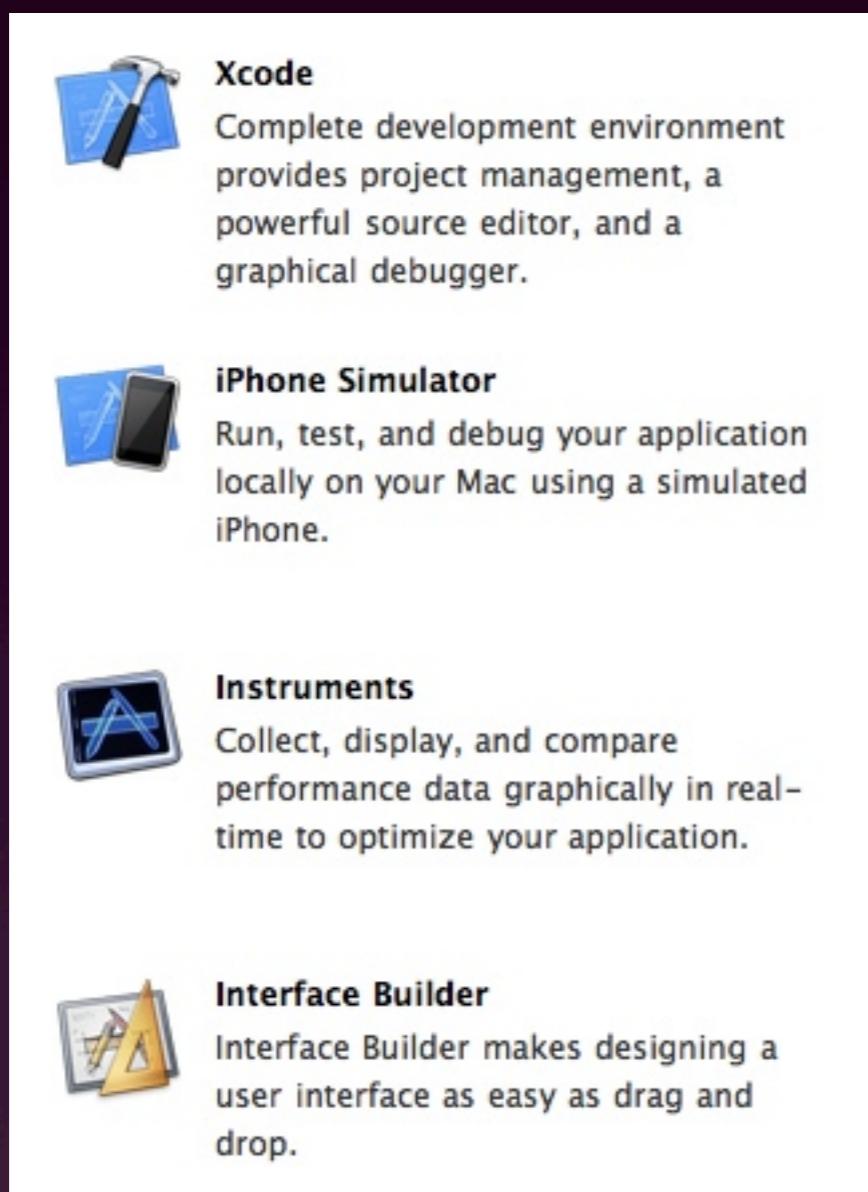
Una **App** es el nombre coloquial con el que se denominan las aplicaciones de iPhone, iPad e iPod Touch.

La **AppStore** no es sólo la tienda de las Apps del iPhone y el iPad, sino la única plataforma legal para introducirlas en el mercado y para descargarlas a tu iPhone.

Por lo tanto, a partir de ahora, cualquier App que programes para tu iPhone o iPad deberás lanzarla a través de la AppStore, de la que ya hemos hablado en AppleNeXt.com.



El iPhone SDK consta de 4 herramientas principales:



Xcode
Complete development environment provides project management, a powerful source editor, and a graphical debugger.

iPhone Simulator
Run, test, and debug your application locally on your Mac using a simulated iPhone.

Instruments
Collect, display, and compare performance data graphically in real-time to optimize your application.

Interface Builder
Interface Builder makes designing a user interface as easy as drag and drop.

1. **Xcode**: Es la herramienta principal para el desarrollo de software en Mac. Como ya hemos dicho, servirá para programar tanto en iPhone, iPod Touch e iPad como para Mac OS X.
2. **iPhone Simulator**: Una vez hayas terminado la programación de tus aplicaciones podrás probarlas en un simulador del iPhone desde la pantalla de tu Mac.
3. **Instruments**: Es un conjunto de herramientas útiles para sacar el máximo partido a tus nuevas Apps. Además ofrece una línea de tiempo para los datos programados.
4. **Interface Builder**: Esta joya es la encargada del diseño de los atractivos interfaces de usuario que tendrán tus Apps, basados en las Human Interfaces Guidelines oficiales de Apple.



Cocoa

'Cocoa' es el nombre que reciben las APIs o el conjunto de frameworks para Programar para Mac. Es, para que nos entendamos, un modo de programación basado en el lenguaje Objective-C (como ya sabemos, una evolución que potencia las posibilidades del lenguaje C).



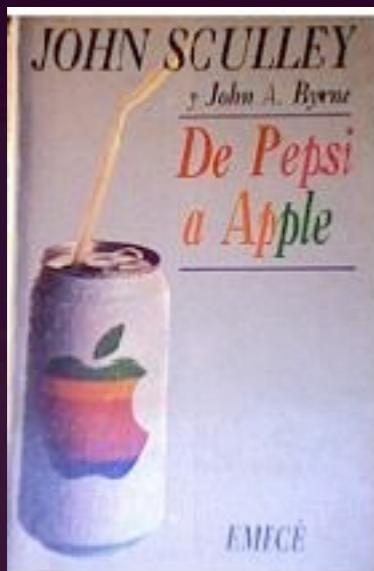
'Cacao Maravillao...' Cocoa es la palabra inglesa para denominar el Cacao.

Originariamente, Cocoa era el nombre de una aplicación para niños con la que podían crear sus primeros proyectos multimedia. La marca fue adquirida por Steve Jobs y a partir de entonces se refiere en el ámbito informático para el método de programación en Mac.



Cocoa: un poco de historia

Érase una vez, un par de chicos llamados Steve fundaron una compañía en un garaje a la que llamaron Apple. La compañía tuvo mucho éxito y creció rápidamente y se decidió poner en manos de un importante ejecutivo que provenía de una de las compañías más importantes del mundo. Éste se llamaba John Sculley y llegó a Apple convencido por la famosa frase de Steve Jobs: “¿Quieres dedicarte toda tu vida a vender agua azucarada o prefieres venir a hacer historia?”.



Tras algunos problemas sobre el control de la compañía, Steve Jobs –el padre de la criatura– decidió ceder e irse. Pero lejos de quedarse con los brazos cruzados, fundó una nueva compañía informática NeXt Computer.

NeXt juntó a un pequeño equipo de ingenieros brillantes. Este pequeño equipo dio lugar a un nuevo ordenador... y a un nuevo Sistema Operativo... y a una nueva empresa... y también a un conjunto de herramientas de programación.

Ese Sistema Operativo estaba a años luz por encima de cualquier otro OS en el mundo de la informática, pero desafortunadamente la enorme expectación que generó no se tradujo en ventas y la fábrica tuvo que cerrar en 1993. NeXt Computer se convirtió en NeXt Software, dedicada exclusivamente a ese nuevo Sistema Operativo.



El sistema operativo y el kit de herramientas continuaron bajo el nombre de **NeXtSTEP**, que aunque no fuese comercializado en ningún ordenador desde entonces, tuvo mucho éxito en la comunidad científica, los programadores y agencias de investigación. Se desarrollaban nuevas aplicaciones cada semana y los científicos se dieron cuenta que NeXtSTEP les permitía implementar sus ideas con mayor velocidad que cualquier otro sistema operativo.

¿Qué era realmente ese sistema operativo?

NeXt decidió utilizar UNIX como raíz del sistema operativo investigando a partir del código fuente de BSD Unix de la Universidad de California en Berkeley. ¿Por qué UNIX? UNIX se rompía mucho menos que Microsoft Windows o el primitivo Sistema de Macintosh y además incorporaba poderosas posibilidades en materia de redes (Internet y compañía).

La aplicación al sistema de ventanas que NeXt hizo con UNIX fue conocida como OpenSTEP y estaba basado en el lenguaje Objective-C, una evolución del lenguaje C que reducía considerablemente los posibles errores de código (llamados ‘bugs’) y hacía su aprendizaje muy sencillo.

La estabilidad y seguridad de NeXtSTEP lo hicieron muy popular entre las agencias de inteligencia, de seguridad y muchas de sus funcionalidades fueron incorporadas a Solaris y Windows NT. Sin embargo OpenSTEP nunca fue utilizado en un Mac, salvo la primera versión de Mac OS X Server, conocido como Rhapsody en 1999.



...y NeXtSTEP se convirtió en Mac OS X

Apple había estado trabajando muchos años tratando de alcanzar un sistema operativo similar al logrado por NeXt Software. Estos esfuerzos fueron conocidos como el 'Proyecto Copland'. El proyecto no cuajaba y finalmente los directivos de Apple decidieron dejarlo de lado y optaron porque el siguiente sistema operativo de Apple sería encargado a una compañía externa.

¡¡¡Y aquí entró en acción NeXt!!! Los directivos de Apple estudiaron todos los sistemas operativos y prefirieron NeXtSTEP, el sistema basado en Cocoa que había creado la empresa NeXt, fundada por Steve Jobs, el fundador de Apple.

Apple compró la totalidad de la compañía NeXt Software en diciembre de 1996. Y NeXtSTEP pasó a convertirse en Mac OS X.

- Steve Jobs funda Apple.
- Le echan.
- Los nuevos directivos son incapaces de crear un Sistema Operativo basado en UNIX.
- Steve Jobs funda NeXT.
- NeXT crea el mejor Sistema Operativo del mundo.
- Apple compra NeXT y Steve Jobs vuelve a Apple.



Acabarás adorando Cocoa

Es cierto: acabarás adorando Cocoa y la programación en Mac. Pero no inmediatamente. Primero deberás aprender las bases y para comenzar deberás conocer las herramientas que vamos a utilizar.

Ya hemos hablado del iPhone SDK que contiene las herramientas para programar cualquier aplicación para cualquier dispositivo de Apple. Además de las 4 herramientas básicas, y sólo para tu información, te comentamos que existe también el [GCC](#) (GNU C Compiler, es decir, el compilador) y el [GDB](#) (GNU Debugger) para localizar cualquier error (recordemos que se llaman 'bugs' y el proceso de corrección se conoce como 'Debug').

Quizá ahora te suene a chino, pero es impresionante que una herramienta como Xcode sea capaz de incorporar todos los recursos en una misma App: las imágenes, sonidos... **Tú únicamente deberás escribir tu código en Xcode, y el propio Xcode lo compilará junto a todos los recursos y lanzará la nueva aplicación para Mac, iPhone, iPad...**

Otra de las maravillas de Cocoa es el Interface Builder, que no sólo es un 'GUI builder' (literalmente: Constructor de Interfaces gráficos para usuario; en cristiano: un creador de interfaces). Es mucho más: porque permite la creación de objetos y la asignación de funciones a esos objetos, muchos de los cuales son 'UI Elements' como botones, campos para texto... y otros tantos serán partes de las 'Clases' que crearás.

No te asustes, simplemente te cuento esto para que sepas que Xcode y Interface Builder son las mejores herramientas de programación que existen y ambas juntas dan lugar a **Cocoa**.



Ya te sabes la historia.
Ya has descargado las herramientas.

Ahora, comienza la acción.

NOTA: Este es un manual creado por **AppleNeXt.com**. Como parte de los servicios que prestamos, el **profesor Emilius B. Starking** estará encantado de resolver las dudas que te surjan mientras aprendes a Programar en Mac. Para ello, deberás acceder a la web de AppleNeXt.com y consultar la sección “Resuelve tus Dudas”, en la que el Prof. Starking publicará las respuestas.



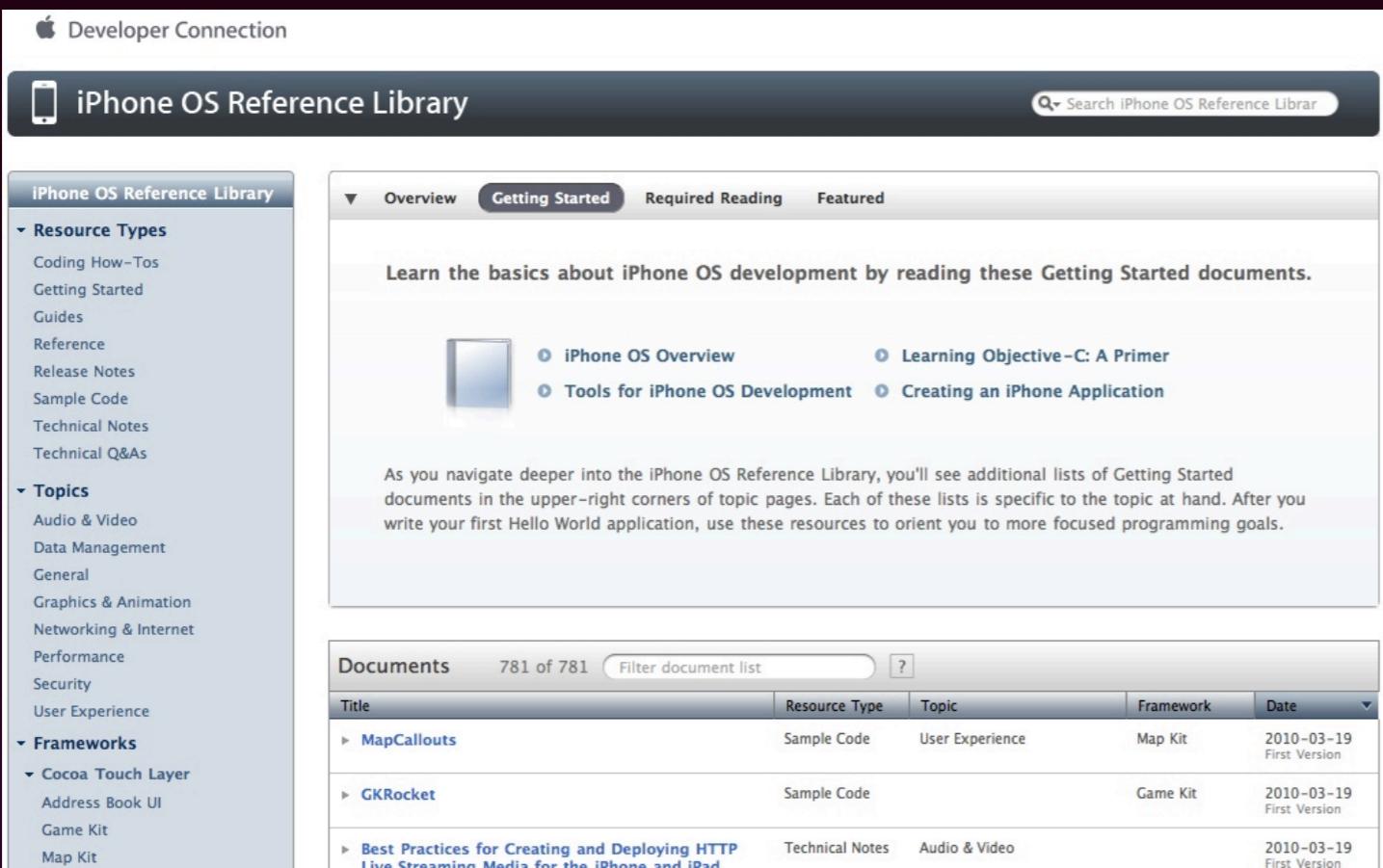
Puesta a punto del Xcode

Doy por hecho que te has decidido a comenzar a aprender cómo Programar en Mac con AppleNeXt.com. Bien, supongo que ya serás miembro (gratuito o no) de la Apple Developer Connection (developer.apple.com) y si no es así, será que te has saltado alguno de los pasos anteriores. También imagino que ya habrás descargado e instalado el iPhone SDK, ¿no?

Una vez tengas descargado e instalado el iPhone SDK lo más inmediato que debes hacer es localizar Xcode y abrirlo. En tu disco duro habrá aparecido una nueva carpeta llamada 'developer' y en ella podrás localizar el Xcode dentro de la carpeta 'Applications'.

Macintosh HD > Developer > Applications > Xcode

Ahora es el momento de descargar la documentación ([iPhone Reference Library](#)). Esto durará un momento y te ahorrará tiempo mientras programas tus aplicaciones.



The screenshot shows the iPhone OS Reference Library interface. On the left, there's a sidebar with categories like Resource Types, Topics, and Frameworks. The main content area has tabs for Overview, Getting Started (which is selected), Required Reading, and Featured. Under the Getting Started tab, it says "Learn the basics about iPhone OS development by reading these Getting Started documents." It lists four items: iPhone OS Overview, Learning Objective-C: A Primer, Tools for iPhone OS Development, and Creating an iPhone Application. Below this, a note explains that as you navigate deeper, you'll see additional lists of Getting Started documents. At the bottom, there's a table titled "Documents" showing 781 items, with columns for Title, Resource Type, Topic, Framework, and Date.

Title	Resource Type	Topic	Framework	Date
MapCallouts	Sample Code	User Experience	Map Kit	2010-03-19 First Version
GKRocket	Sample Code		Game Kit	2010-03-19 First Version
Best Practices for Creating and Deploying HTTP Live Streaming Media for the iPhone and iPad	Technical Notes	Audio & Video		2010-03-19 First Version

<http://developer.apple.com/iphone/library/navigation/index.html>

Cocoa Touch

En el iPhone, iPad e iPod Touch, Cocoa es conocido como Cocoa Touch porque estos dispositivos contienen eventos de tipo 'touch' (táctiles). Sabes de sobra de lo que estoy hablando puesto que llevas mucho tiempo toqueteando tu iPhone o tu iPod Touch y habrás ampliado una foto, girado con tus dedos, abierto una ventana o ajustado el navegador del Safari...

Técnicamente, los 'Touch events' te permitirán programar respuestas para cada vez que el usuario toque la pantalla del iPhone con sus dedos.



Cuando avancemos un poco sabrás lo que son las 'Clases'. Hoy por hoy, simplemente te digo que Cocoa Touch te proporcionará las librerías básicas de 'Clases' que necesitarás para crear tus aplicaciones para el iPhone.

Foundation Framework

Otro de los conceptos que debes conocer para aprender a Programar en Mac son los 'Frameworks'. Los Frameworks son colecciones de código dedicados a una misma tarea. ¿Se entiende, no? Pues bien, para Programar aplicaciones para el iPhone utilizaremos siempre dos tipo de Frameworks:

- Foundation framework: destinado a colecciones de temas estándar y tareas básicas.
- UIKit framework: Destinado al interfaz de usuario.

La comodidad del Foundation framework es que existe una amplia colección de tareas básicas ya programadas que puedes utilizar sin necesidad de programarlas tú mismo desde cero. No te preocupes, Apple también selecciona sus componentes, no los fabrica desde cero. Nosotros no queremos perder tiempo.



Tu primera Aplicación para Mac

Muchos manuales comienzan proporcionándote mucha filosofía sobre la programación. Nosotros hemos querido darte las pinceladas maestras en las páginas anteriores pero no queremos perder más tiempo con eso. Hemos preferido enseñarte a programar tu primera aplicación en Cocoa, y cuando termines te encontrarás en condiciones para toda esa filosofía...

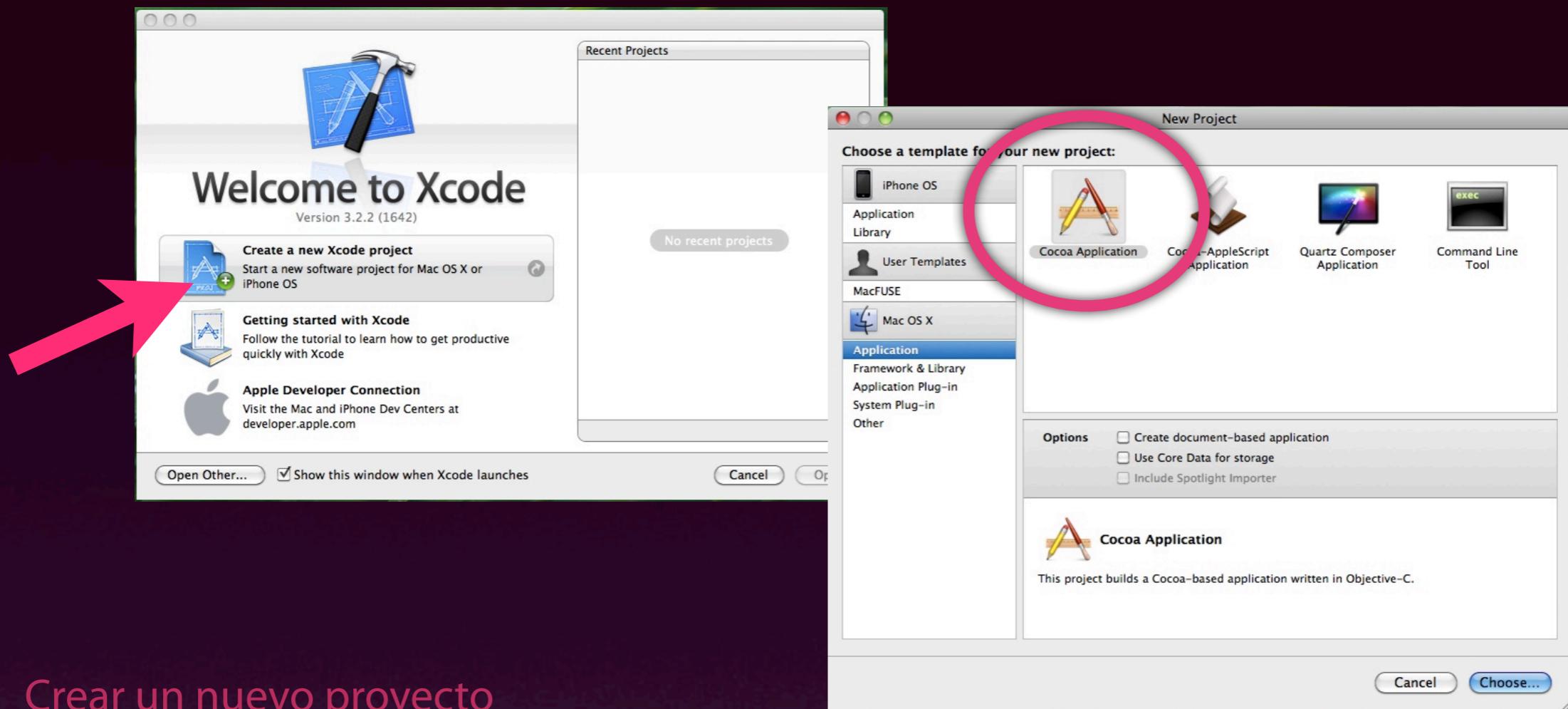
Nuestra primera aplicación será un [generador aleatorio de números](#) utilizando dos botones:

- calcular un número aleatorio durante un tiempo
- y mostrar dicho número.

Realmente lo que nos interesa es que aprendas a utilizar un *input* del usuario para que obtenga un *output*. Así tu cabeza comenzará a entender el lenguaje de la programación.

NOTA: Te [recomendamos que coloques la aplicación Xcode en el dock](#) de tu Mac para que sea más rápido llegar a él, puesto que a partir de ahora lo vamos a utilizar bastante.



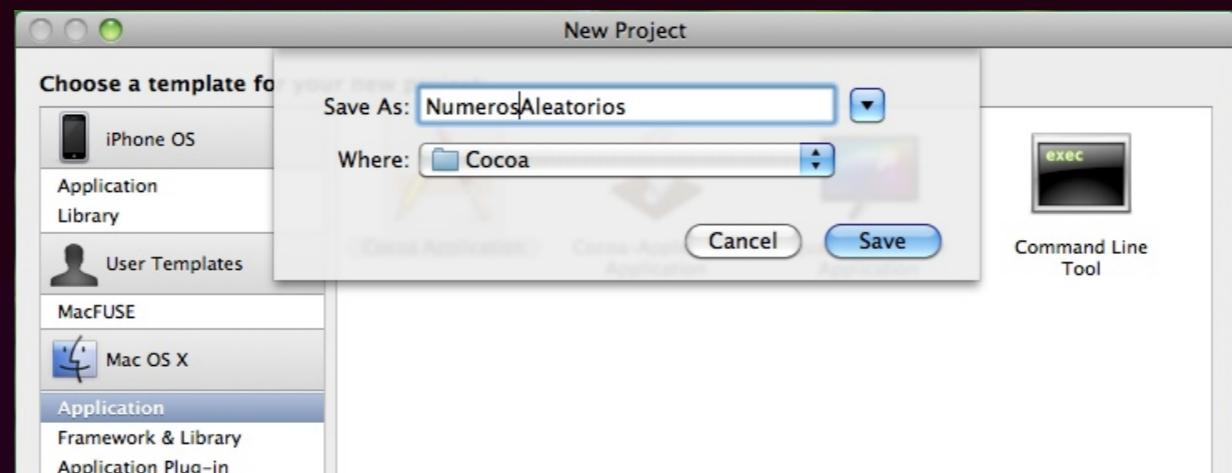


Crear un nuevo proyecto

Como ya hemos dicho antes, una de las ventajas de Xcode es que agrupa todo el trabajo y los recursos de tu aplicación. Esto lo hace en un directorio llamado 'Project Directory'. Pues bien, lo primero que haremos es crear un nuevo proyecto con el esqueleto que todas las aplicaciones deben tener por defecto.

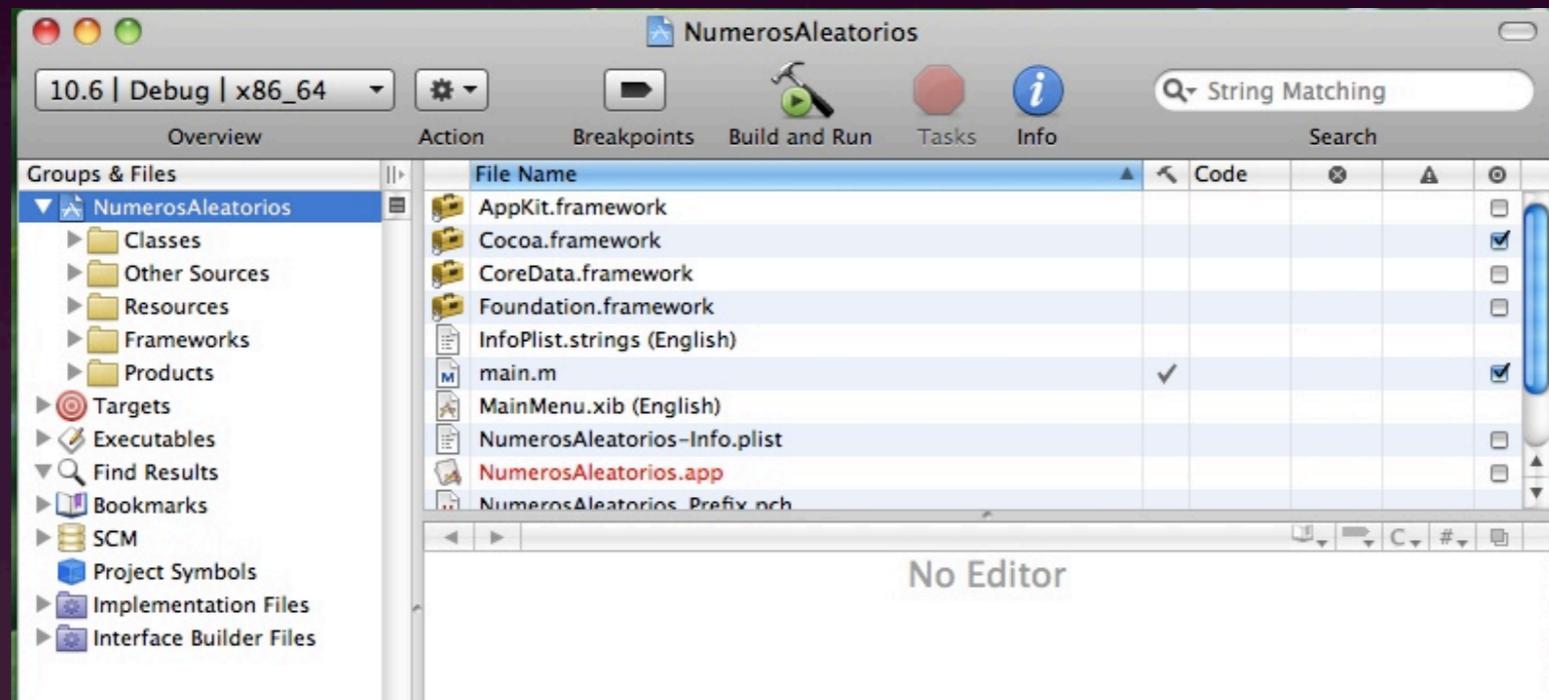
En la página de bienvenida de Xcode selecciona '**Create a new Xcode project**'.

Cuando se abra el panel de proyectos, verás que existen multitud de posibilidades. Esta vez vamos a realizar una aplicación para Mac, por lo tanto seleccionaremos **Cocoa Application** dentro de la lista de **Mac OS X**.



Pon el nombre que quieras a tu nuevo proyecto. Yo he escogido '**NumerosAleatorios**' que es el tema de nuestra primera aplicación para Mac OS X. Pulsa 'Save'.

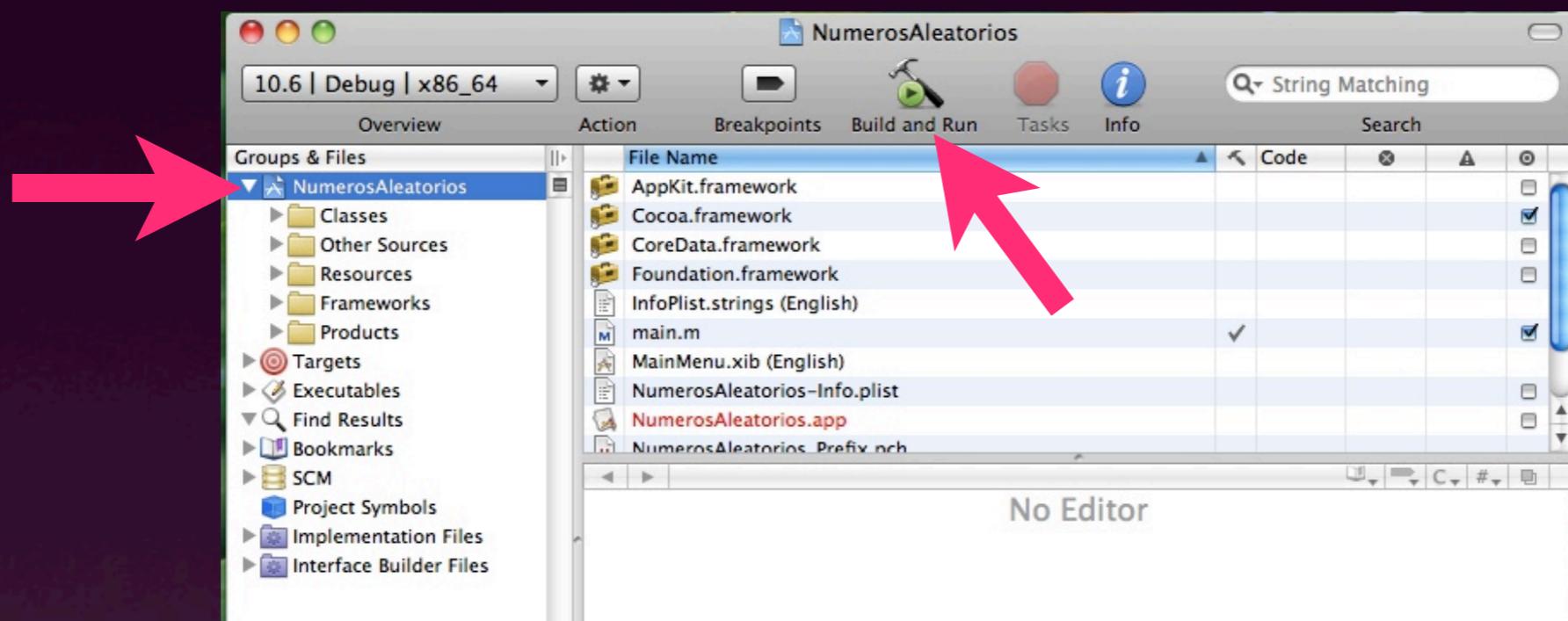
Xcode habrá creado el directorio de tu nuevo proyecto con el esqueleto básico de cualquier aplicación. Lo que nosotros deberemos hacer es **desarrollar ese esqueleto básico a nuestro gusto y después compilarlo todo en una Aplicación que funcione**. Esto es programar.





Echando un vistazo al directorio de tu proyecto podrás ver que existe una visualización global del proyecto en la zona de la izquierda de tu ventana. Cada uno de los elementos de esa parte de la ventana representa un tipo de información diferente que será de utilidad para el programador.

Algunos son archivos, otros son mensajes (como los errores del compilador). Por ahora nos centraremos en los archivos, así que haz click en 'NumerosAleatorios' para ver los archivos que serán compilados dentro de una aplicación.



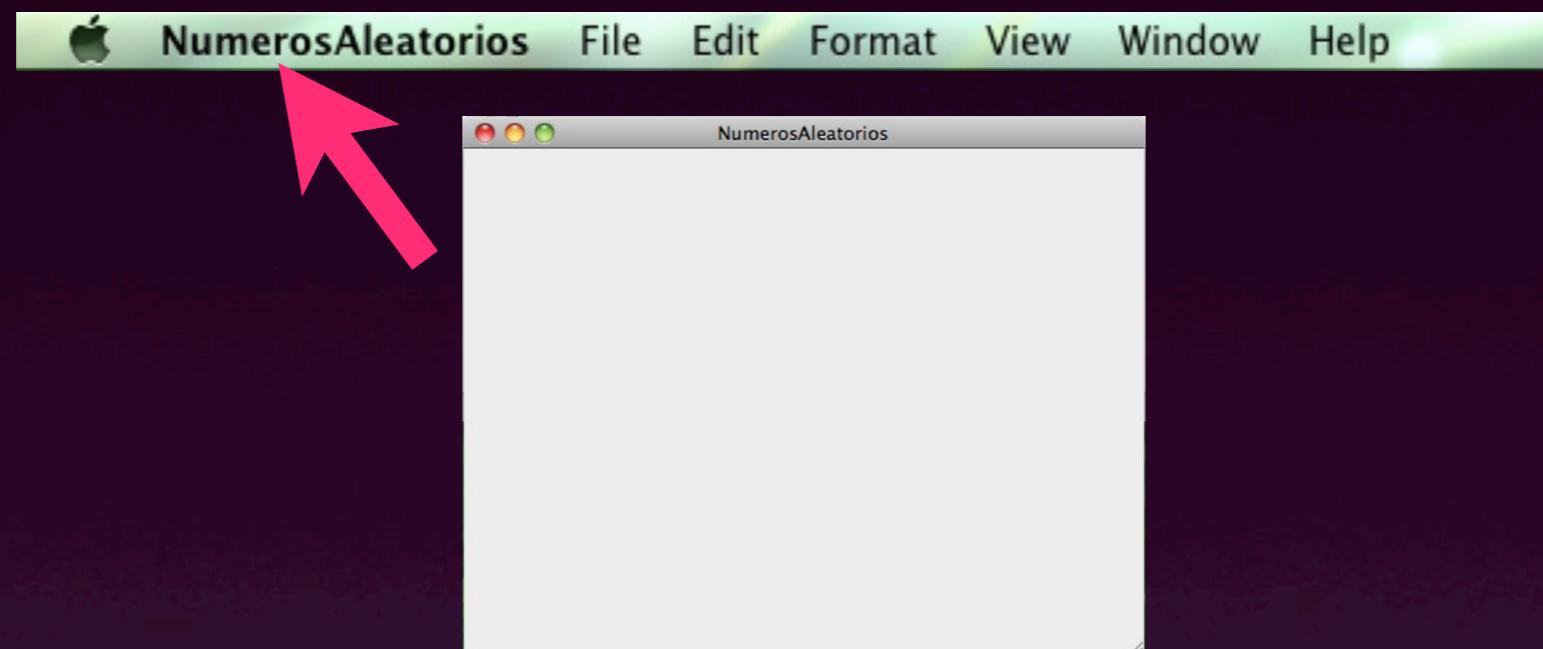
El esqueleto que vemos será compilado al terminar de programar nuestra aplicación y estará compuesto por al menos una ventana y un menú, como todas las aplicaciones en Mac OS X.

Haz click en el símbolo que tiene un martillo y un 'play' verde para construir la aplicación y hacerla funcionar (**Build and Run**, en inglés).



Mientras la aplicación se construye verás que el ícono de Xcode en el dock se cambia para indicarte que está trabajando. Una vez finalizado verás la ventana y el menú que habíamos dicho...

El nombre de tu nueva aplicación aparecerá en el menú; esto te indicará que tu aplicación está activa.



La ventana evidentemente estará vacía, puesto que no hemos desarrollado aún el esqueleto de nuestra aplicación, pero te darás cuenta que ya tenemos los elementos básicos de cualquier aplicación. Ahora tendremos que programar el nuevo código. ¡Vamos allá!

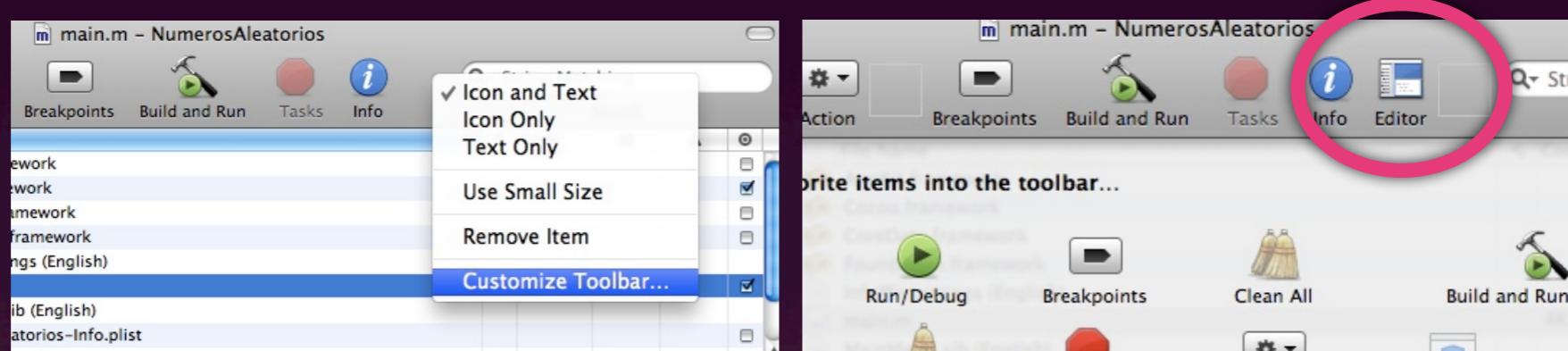
Cierra la aplicación 'NumerosAleatorios' (como cualquier aplicación que existe cmd+Q) y regresemos a Xcode.



La función principal (main Function)

Selecciona `main.m` haciendo click en él (ojo, si haces doble click en él lo abrirás en una nueva ventana). Cuando estés acostumbrado a hacer más códigos de programación te darás cuenta del tiempo que se ahorra usando el estilo de única-ventana.

Antes de seguir, vamos a añadir una nueva herramienta a la barra de herramientas del Xcode. Quedará añadida para siempre. Únicamente tendrás que hacer click con el botón derecho en la barra de herramientas como te indico en la imagen y seleccionar 'Customize Toolbar' (personalizar barra de herramientas). Ahí aparecerá un conjunto de herramientas y deberás arrastrar la herramienta 'Editor' sobre la barra de herramientas. Despues pulsa 'Done'.

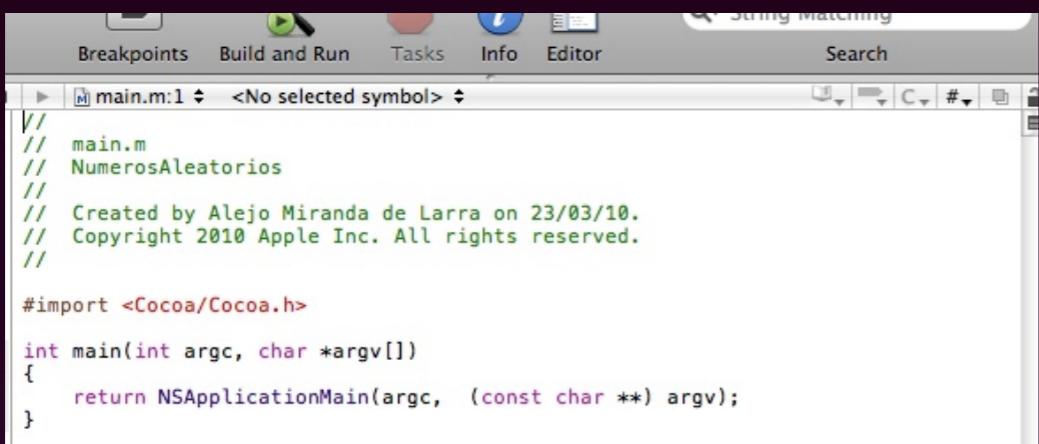


NOTA: Podrás personalizar tu espacio de trabajo las veces que quieras. Aquí te iré recomendando algunas cosas, pero siempre dependerá de lo que te sea más cómodo a la hora de trabajar.



Ahora haciendo click en 'Editor' seleccionaras el modo de vista editor sobre la parte que hayas seleccionado del código de programación. En esta ocasión habíamos seleccionado `main.m` y tras hacer click en Editor accederemos una vista más sencilla de la función `main.m`

Verás algo así:



```
//  
// main.m  
// NumerosAleatorios  
//  
// Created by Alejo Miranda de Larra on 23/03/10.  
// Copyright 2010 Apple Inc. All rights reserved.  
  
#import <Cocoa/Cocoa.h>  
  
int main(int argc, char *argv[])  
{  
    return NSApplicationMain(argc, (const char **) argv);  
}
```

Casi nunca deberás modificar el contenido de `main.m` en tus aplicaciones. Como ves en el código, la función `main()` sencillamente llama a la función `NSApplicationMain()` que será la que cargará los objetos del interfaz de usuario sobre un archivo 'nib'.

No te asustes, te lo explico: Te familiarizarás rápido con los archivos 'nib'. Son los archivos que crea el 'Interface Builder'. Siendo más claros:

main ()

NSApplicationMain ()

Coloca los objetos en el
Interfaz Gráfico de Usuario

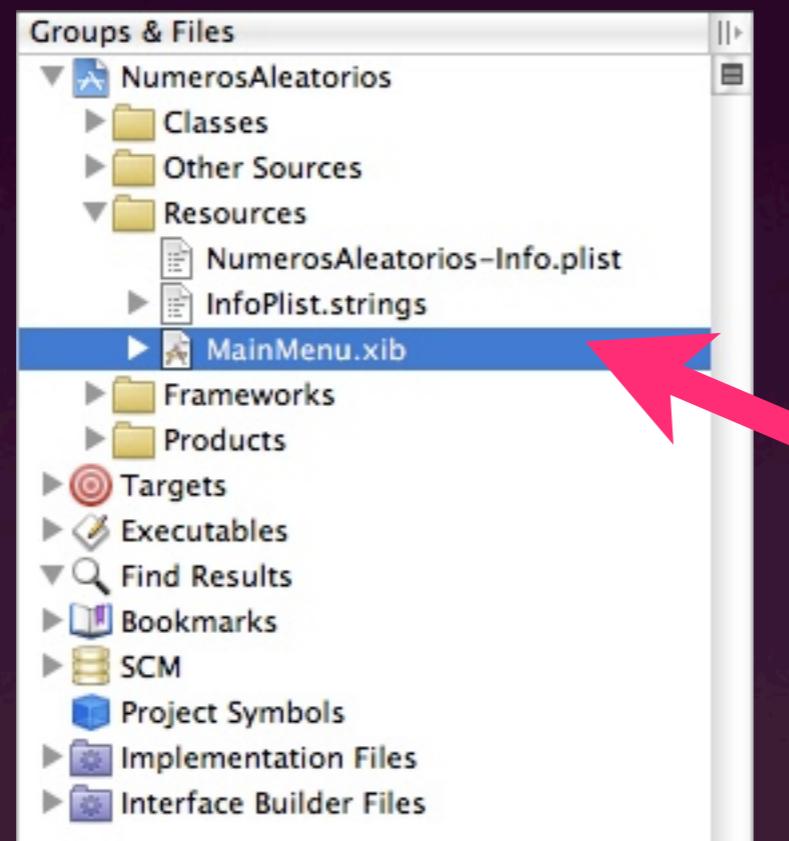
NOTA: NIB = NeXt Interfaz Builder (sí, como AppleNeXt, jeje)



Una vez que se ha cargado el archivo 'nib' tu aplicación simplemente esperará a que el usuario haga algo. Y cuando el usuario haga click o escriba, tu código será 'llamado' automáticamente. Aunque por ahora no sepamos nada más, el funcionamiento es fácil de comprender, ¿no? Vamos, que la función principal `main()` siempre te llevará a otra llamada `NSApplicationMain()` que será la que dirija el cotarro cuando el usuario haga algo. Todo pasará por la función `NSApplicationMain()` pero sólo reaccionará cuando el usuario haga algo.

En las últimas versiones de Xcode y iPhone SDK a los archivos NIB se les ha pasado a denominar **archivos XIB**, por lo que en tu Xcode probablemente tendrás algún archivo .xib ¡Vamos a verlo!

Busca en la carpeta 'Resources' que aparece en la ventana de la izquierda:



Haz doble click en el archivo .xib y se abrirá con la herramienta 'InterfaceBuilder'

Esto se pone más interesante, eh!

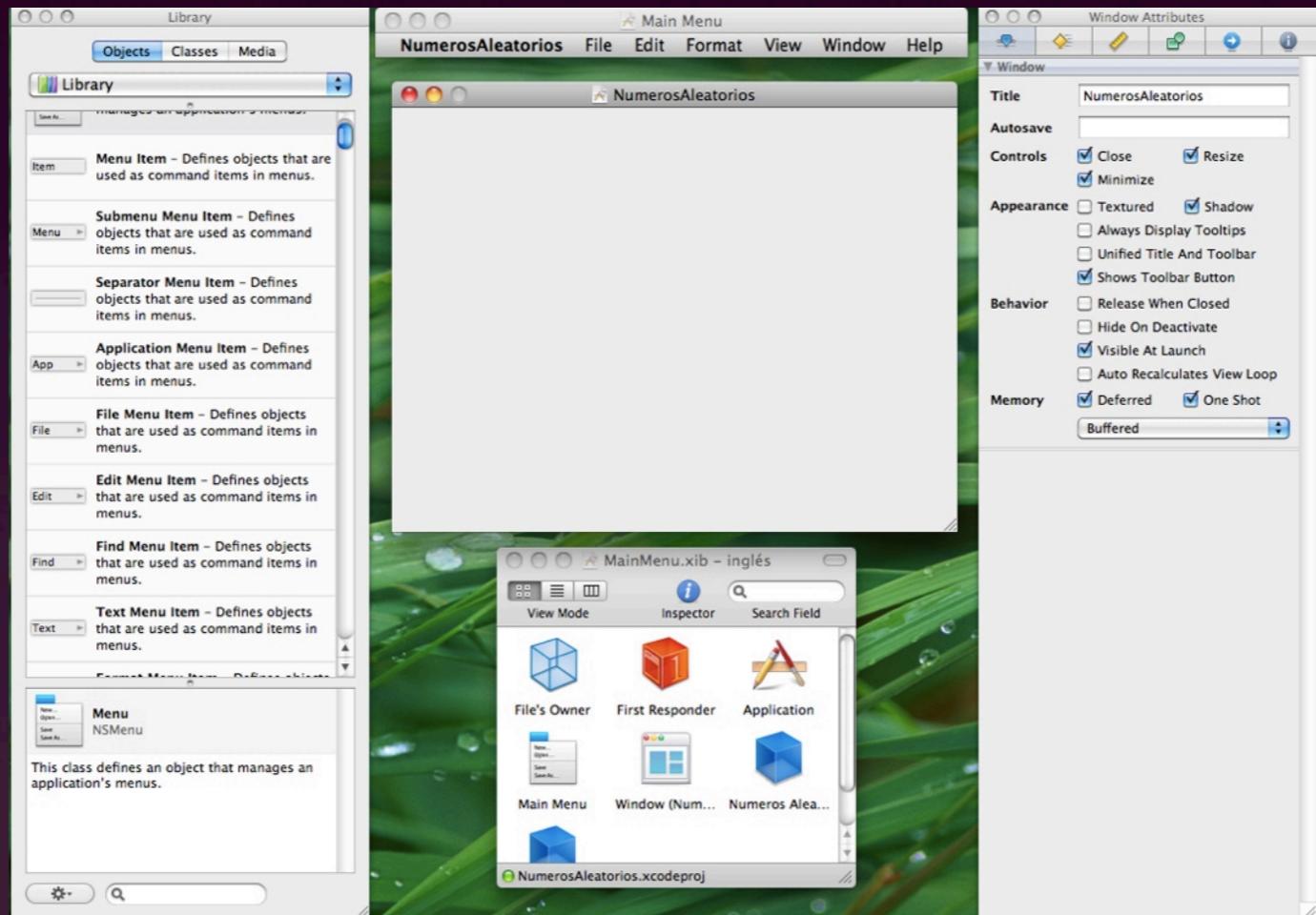


Interface Builder



Si has seguido los pasos anteriores correctamente se abrirán en tu escritorio muchísimas ventanas por lo que será el momento de minimizar (u ocultar) el Xcode para no liarte con tantas cosas. En el menú llamado 'Interface Builder' encontrarás la opción 'Hide Others' (ocultar otras, en inglés).

Esta fantástica herramienta te permitirá crear y editar los objetos del Interfaz de Usuario (el interfaz de usuario es "**lo que se ve**" de una aplicación): por ejemplo, botones o ventanas. Y además grabará esos objetos en un archivo. No sólo colocarás objetos sino que los relacionarás entre ellos mismos y entre cada objeto y las funciones de tu código para que **cuando el usuario haga algo con los objetos de la pantalla, tu código se ejecute.**

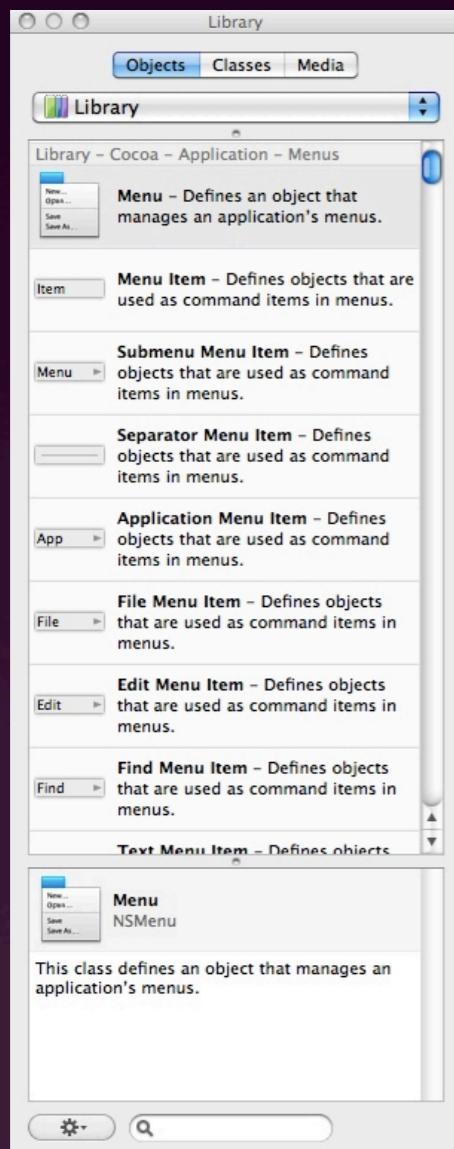




Quizá no te has dado cuenta, pero ya has aprendido muchísimo para Programar en Mac. Sabes usar el Xcode, has creado tu primer proyecto y estás investigando el Interface Builder. Vas por buen camino...

Library

La ventana 'Library' es el lugar donde podrás encontrar 'widgets' que podrás arrastrar sobre el interfaz que estás creando. Por ejemplo, si quieres un botón pues lo arrastras desde la ventana 'Library'.



Ventana en blanco

La ventana en blanco representa una ventana de la 'clase' `NSWindow` que está dentro de tu archivo `.xib` y a medida que arrastres objetos sobre tu ventana en blanco, éstos se añadirán al archivo `.xib`.

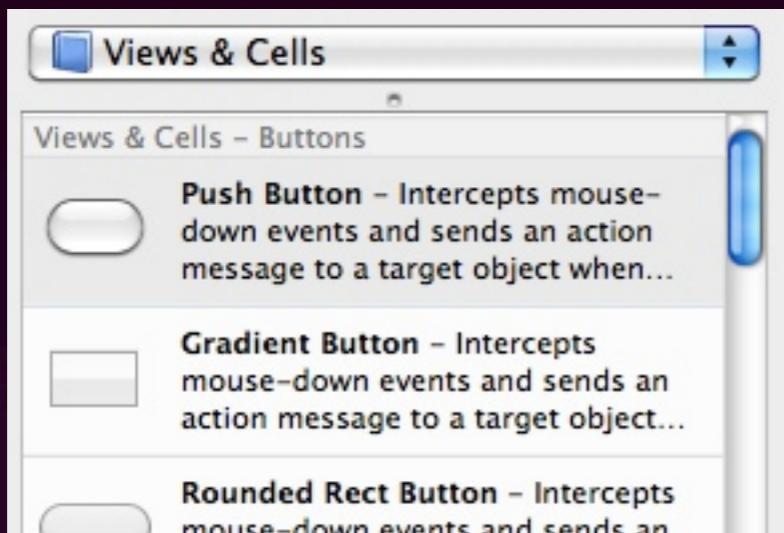
Una vez que hayas hecho algo con los objetos y hayas editado sus características al grabar el archivo `xib` se quedarán todos herméticamente dentro de él. Si ya te decíamos que Xcode y Interface Builder son herramientas fantásticas. Cuando la aplicación se ejecuta el archivo `xib` será leído y los objetos que contiene aparecerán también.

Los objetos son empaquetados por el Interface Builder y serán desempaquetados cuando la aplicación se ejecute.

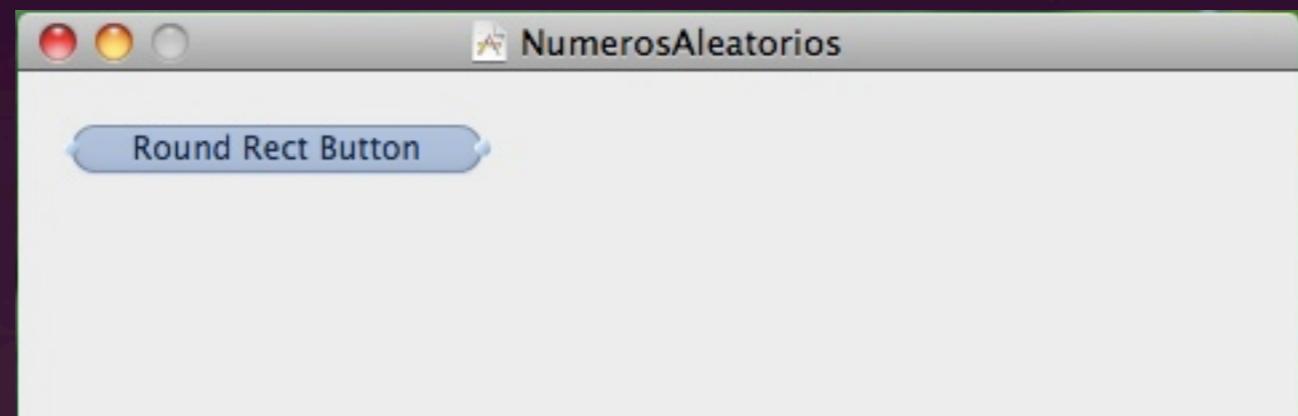


Ahora vamos a empezar a crear nuestro primer interfaz de usuario para nuestra aplicación 'NumerosAleatorios'.

- **Arrastra un botón** desde la Biblioteca de objetos (Library) sobre la Ventana en Blanco.

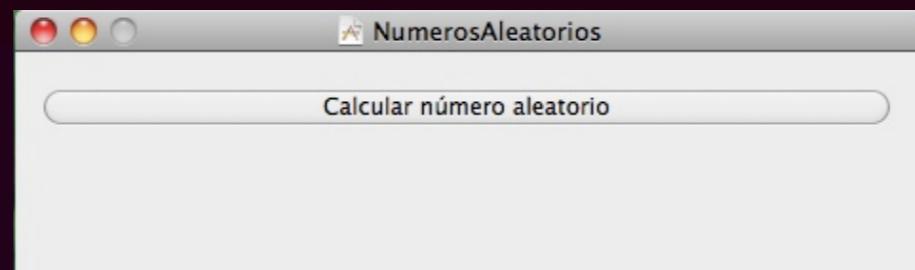


Truco: para que sea más fácil encontrarlo puedes usar el selector de la parte superior de la ventana y encontrarás los botones en la categoría 'Views & Cells'. Ahí selecciona el botón que quieras, por ejemplo 'Rounded Rect Button' y arrástralo a tu Ventana en Blanco. ¡Venga, que es fácil!

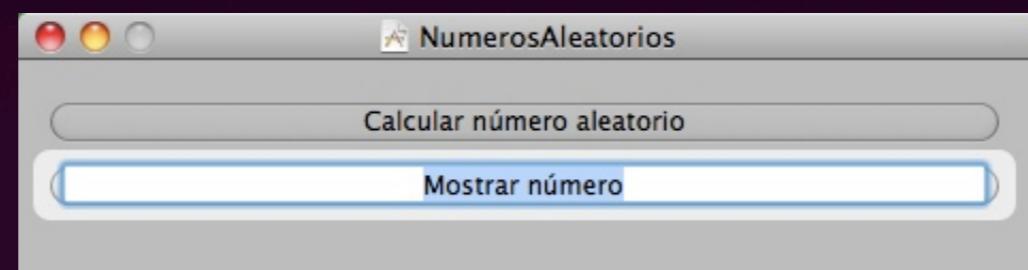




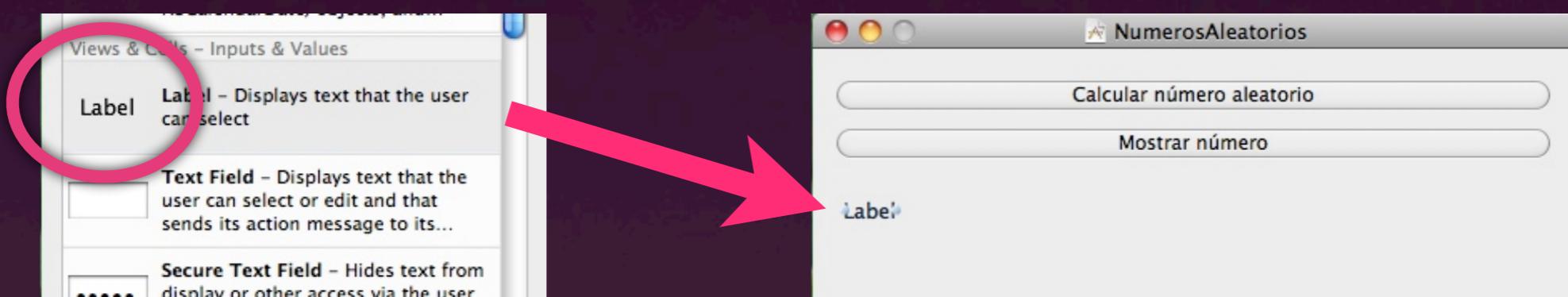
Ahora haz doble click en el botón que acabas de arrastrar para cambiarle el nombre. Lo podemos llamar 'calcular número aleatorio'. Podrás ampliar el botón todo lo que quieras fácilmente y te quedará algo así:



Ahora vamos a seleccionar el botón creado y hacemos 'CopyPaste' para crear uno nuevo que colocaremos debajo del primero. Al nuevo botón lo podemos llamar 'Mostrar número'.



Sencillo, ¿verdad? Ahora seleccionaremos una etiqueta (**Label**) de campo para texto ('Text Field') de la Biblioteca y la arrastramos sobre nuestra ventana una vez más. ¡¡Ojo, no hablo del campo de texto en sí, sino de la etiqueta Label!!





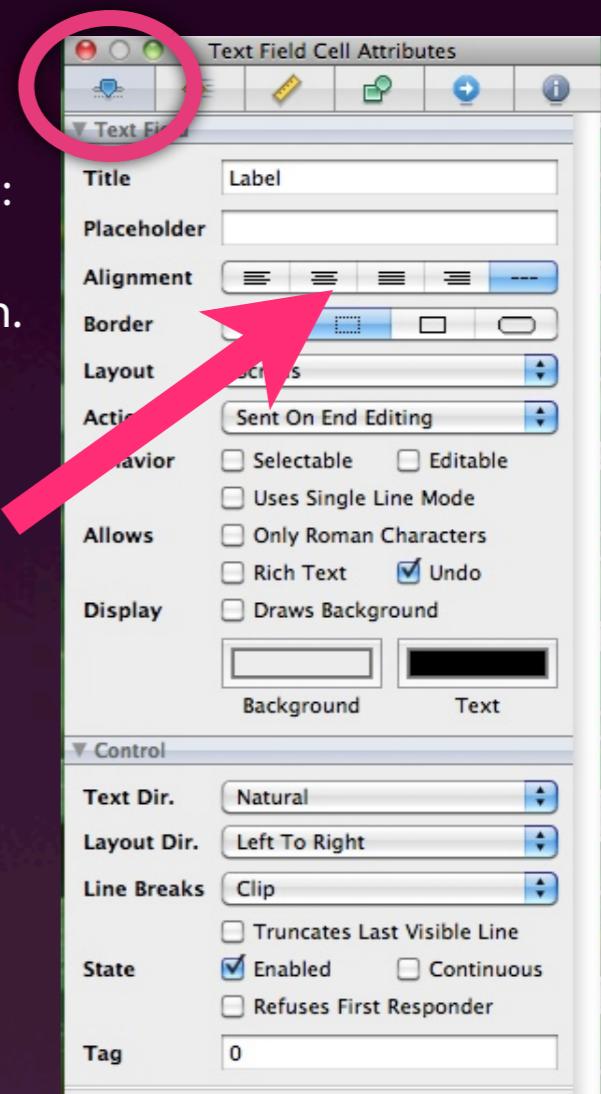
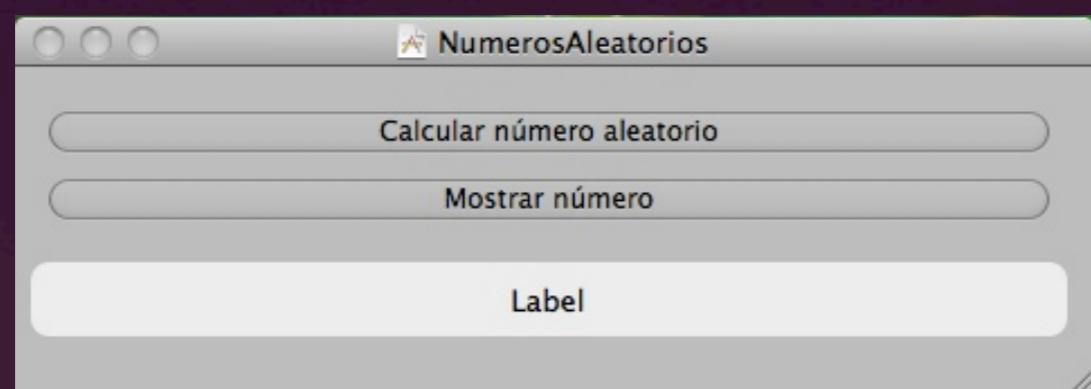
Para hacer la etiqueta de texto tan grande como los botones estírala hasta los límites de la ventana. Verás que aparecen unas líneas azules para ayudarte con las dimensiones y los tamaños. Estas líneas son las guías que Apple pone a tu disposición para que tu interfaz se parezca a los parámetros de Apple GUI Guidelines (es decir, que se parezca a la imagen típica de las aplicaciones del Mac OS X).

Cambia también el tamaño de tu ventana para ajustarla a la imagen que tú deseas y que no quede demasiado espacio en blanco.

Ahora vamos a centrarnos en los contenidos del interfaz y el manual se complicará un poco. Pero no te preocupes, como ves lo vamos a ir tratando poco a poco.

Nos hará falta utilizar el **inspector**, que es la ventana que tenemos a la derecha:

- Selecciona la Etiqueta que acabamos de incorporar a nuestra Aplicación.
- En el Inspector seleccionamos el primer ícono ('Attributes')
- Seleccionamos justificación al centro.



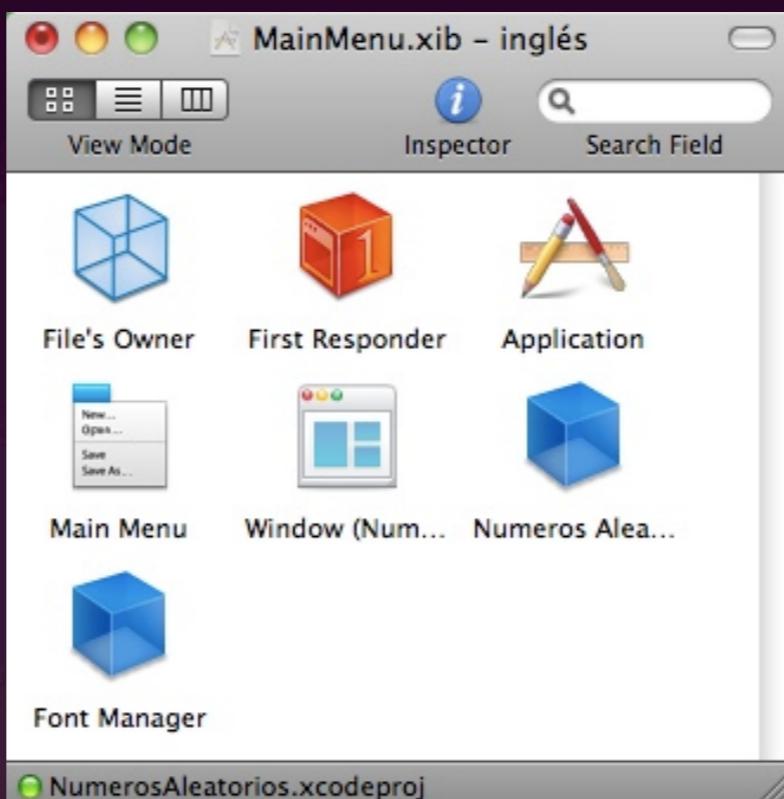


La ventana Doc (the 'Doc Window')

Te habrás dado cuenta de que hay otra ventana más en Interface Builder: se trata de la ventana 'Doc', para que nos entendamos es la ventana de documento, pero la llamaré ventana 'Doc'.

Como en casi todo en el mundo de la informática, no todos los elementos de programación son visibles. Los dos botones que hemos arrastrado sí lo son y también la etiqueta Label. Como ya sabes, los elementos visibles aparecerán en la ventana en blanco (que ya no está en blanco) mientras que los elementos invisibles los controlaremos desde la ventana Doc.

Por tanto, **la ventana Doc es la ventana de los objetos invisibles**. Los elementos de dicha ventana representan a los objetos invisibles.





La ventana Doc es una de las más atractivas en la Programación en Mac (por lo menos a mí siempre me lo ha resultado).

En ella podemos ver elementos que representan al menú principal (main menu) y a la ventana de nuestra aplicación.

- First Responder es un elemento ficticio pero que realiza una ficción muy útil. Será explicada en capítulos posteriores.
- File's Owner en este archivo xib es el objeto `NSApplication` para nuestra aplicación. Este objeto dirige los eventos de la 'cola de eventos' y los coloca en la ventana apropiada. También trataremos este objeto más adelante.

Vamos a crear una 'clase' (Class)

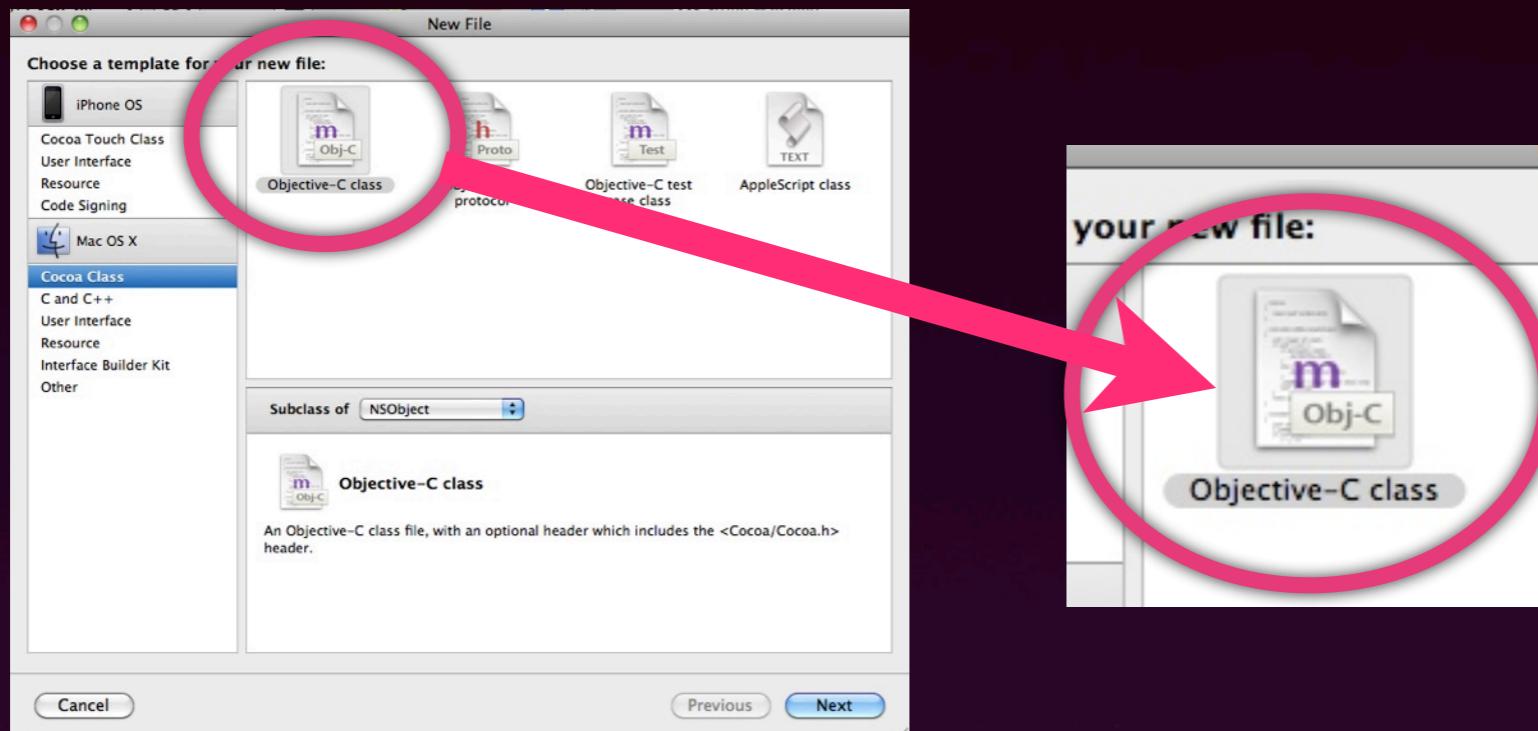
En el lenguaje de programación Objective-C cada clase se define a partir de dos archivos: un archivo de cabecera (header) y un archivo de implementación.

Al archivo Header se le conoce también como el archivo de la Aplicación (el archivo de programa). En él se declaran las variables y métodos que contiene la clase que hayamos creado. Sin embargo, el archivo de implementación lo que hace es desarrollar qué es lo que esos "métodos" harán.

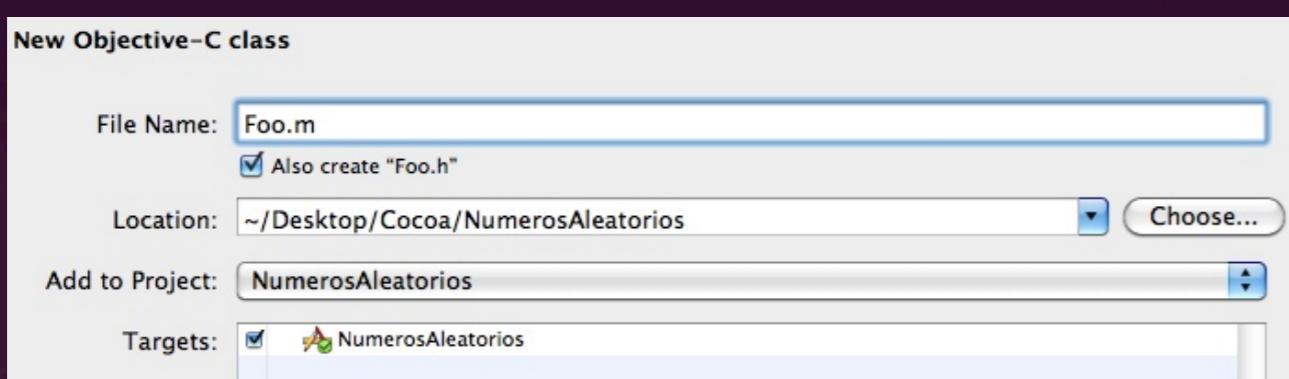
Importante: Como ves, llegados a este punto es importante que recuerdes lo que sabes del lenguaje de programación C: *clases, métodos y variables*. Si nunca has oído hablar de esto no te preocupes, porque en AppleNeXt.com también te los explicamos.



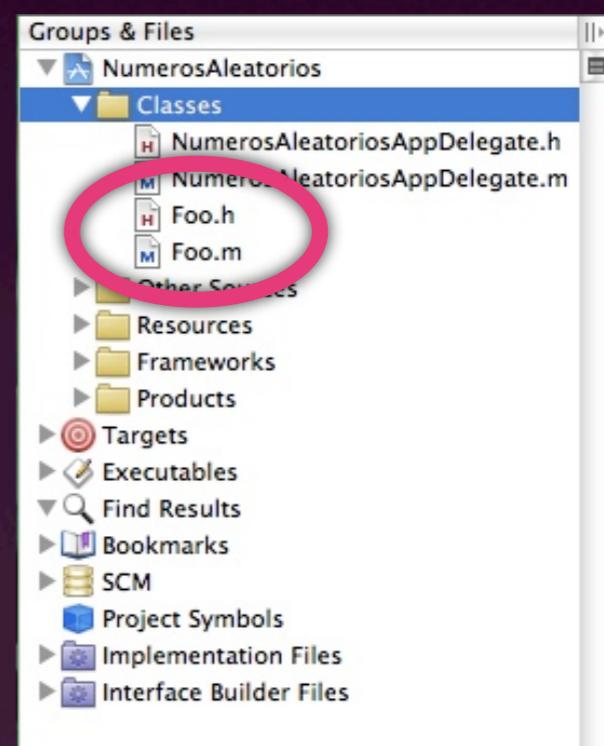
Para crear la 'Clase' deberás regresar a Xcode. En el menú archivo selecciona 'New File' y se abrirá la siguiente ventana:



Deberás seleccionar el archivo 'Objective-C class'. Ponle de nombre `Foo.m`



Los archivos `Foo.h` y `Foo.m` aparecerán en tu proyecto dentro de 'Classes':
Si aparecen en otra parte puedes arrastrarlos a la carpeta 'Classes'

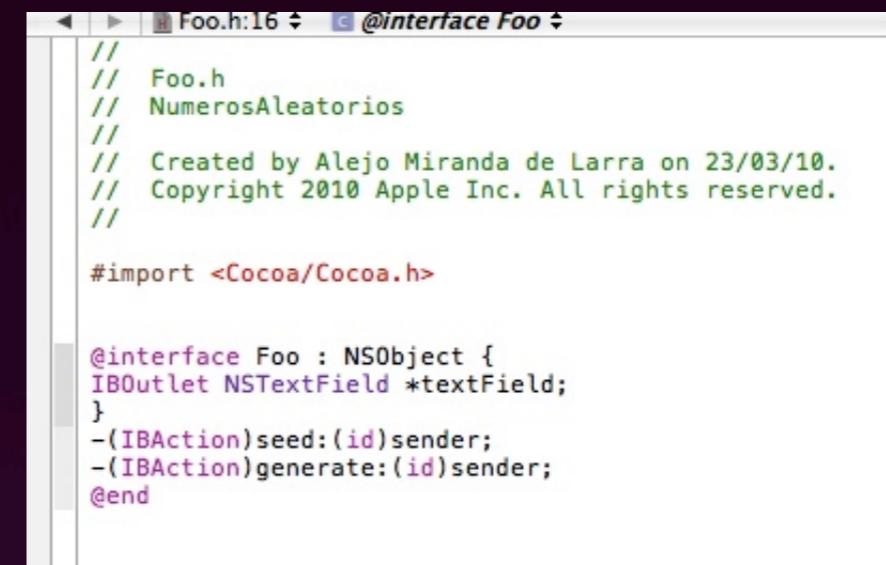




En `Foo.h` podrás añadir *variables* y *métodos* a la *clase* que has creado. A las variables que apuntan hacia otros objetos se les llama *outlets* y a los métodos que se pueden accionar por parte del usuario se les llama *acciones* ('actions').

Vamos a editar `Foo.h` para que quede algo así:

```
#import <Cocoa/Cocoa.h>
@interface Foo : NSObject {
    IBOutlet NSTextField *textField;
}
-(IBAction)seed:(id)sender;
-(IBAction)generate:(id)sender;
@end
```



Leyendo esto, alguien que sepa de Objective-C podría decir tres cosas de estas líneas de código:

1. `Foo` es una subclase de `NSObject`.
2. `Foo` tiene una variable: `textField` es un puntero hacia una parte de `NSTextField`
3. `Foo` tiene 2 métodos: `seed:` y `generate:` (que son métodos *acciones*).

NOTA: No te preocupes si aún no entiendes ni palabra de todo esto. Lo veremos dentro de muy poco. Recuerda que simplemente estamos creando una aplicación desde cero y más adelante entenderás todo eso.

Date cuenta que los nombres de los métodos y las variables comienzan por minúscula pero al juntar varias palabras las siguientes se ponen en mayúscula para diferenciarlas. Esto es así por un acuerdo no escrito global entre programadores. Sin embargo, también se ha convenido que los nombres de las Clases comiencen en mayúscula (`Foo.h`)

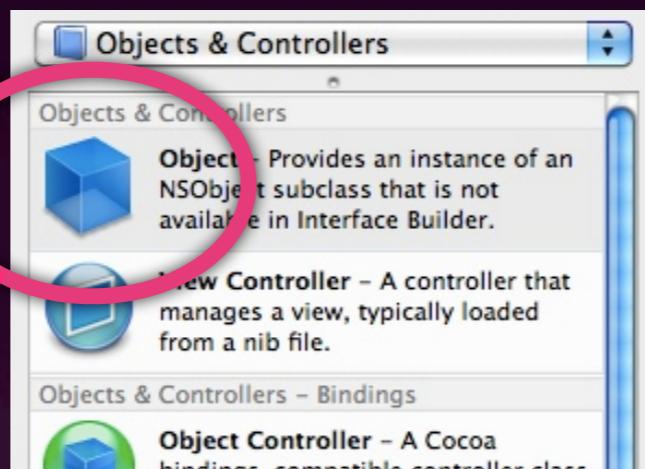


Vamos a dar vida a esto:

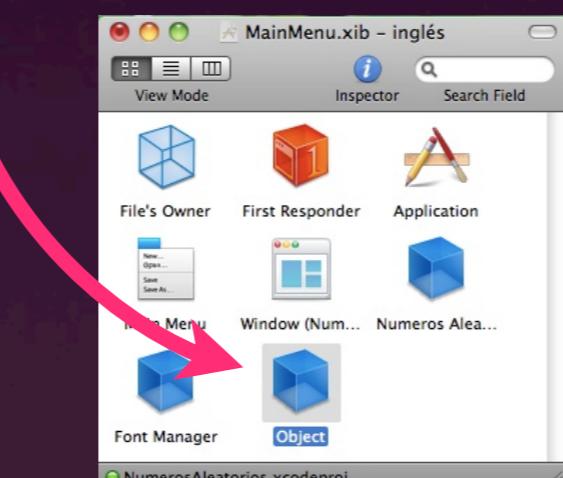
Lo siguiente que harás es crear un flujo de actividad para la Clase Foo en tu archivo xib (en Interface Builder).

Así que volvemos a Interface Builder.

En la Biblioteca vamos a seleccionar un 'Object' (el icono es un cubo azul) que encontrarás en el selector en el menú 'Objects & Controllers':

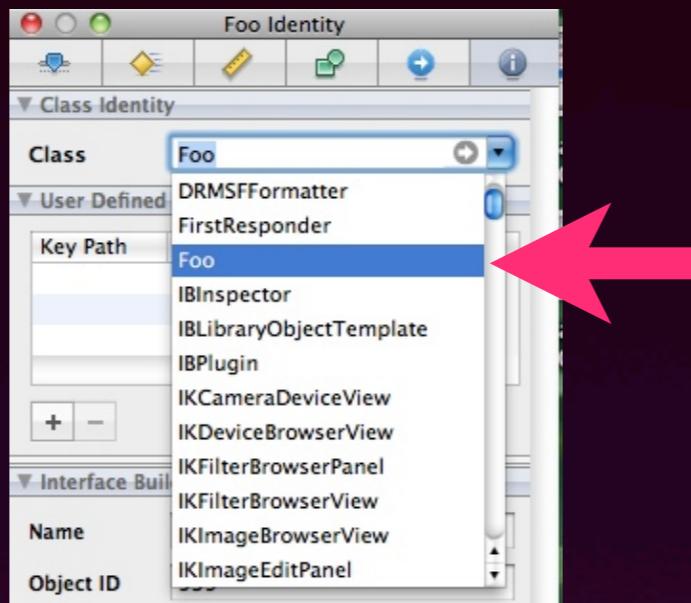


Lo arrastraremos a la ventana Doc (recuerda que es la ventana de los elementos invisibles)





En el Inspector seleccionamos la opción 'Identity' (cuyo icono es una "i" azul) y le asignamos la 'Class' que hemos creado, es decir, Foo.



Nota: si no apareciese Foo en la lista de Classes deberás revisar los pasos anteriores. Probablemente has olvidado grabar después de modificar el código de `Foo.h`

Conexiones

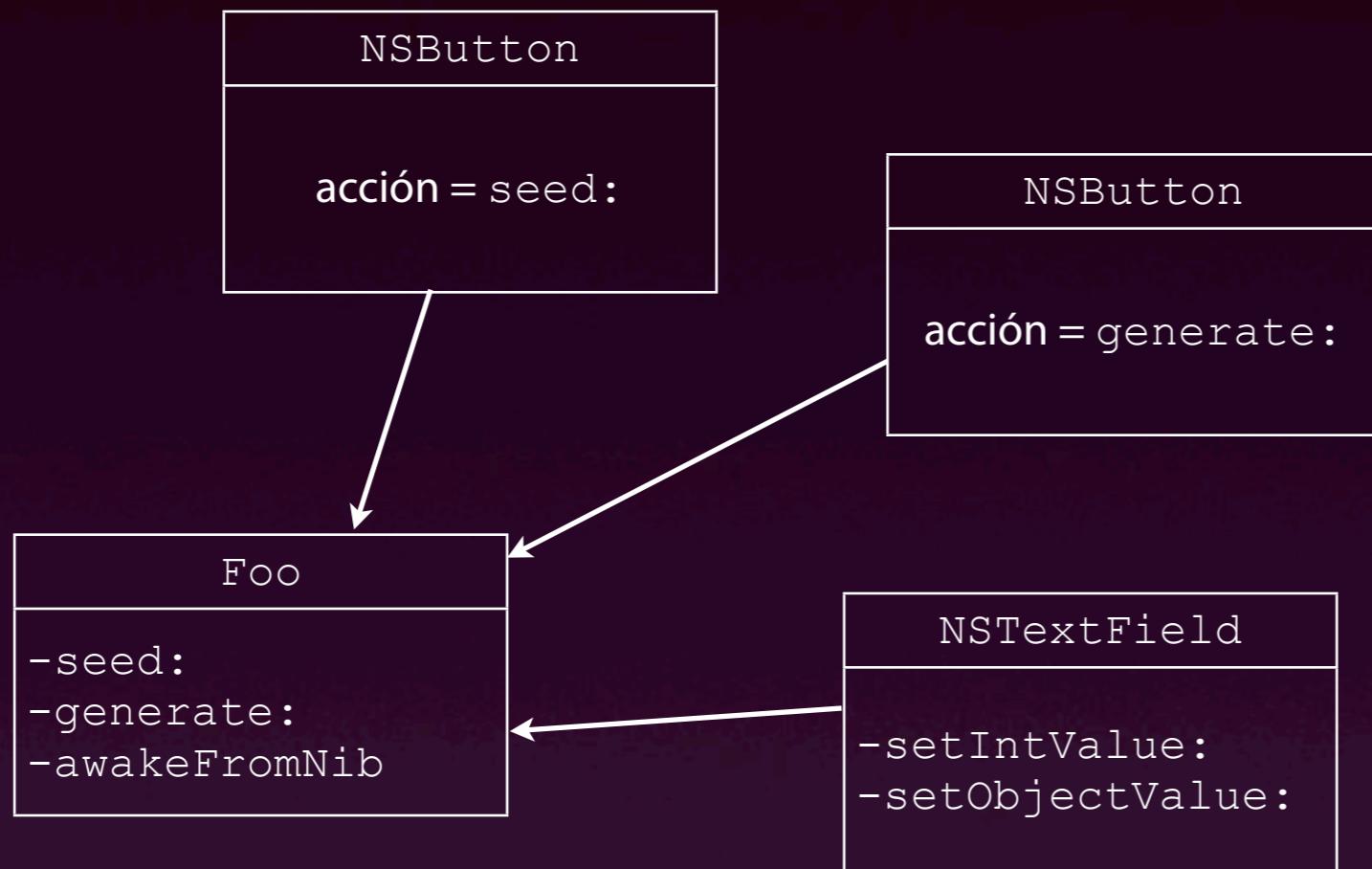
Gran parte de las programación orientada a objetos (recuerda que de eso se trata Objective-C, y por tanto, Cocoa y la Programación en Mac).

Por eso, ahora lo que haremos es comenzar a conectar unos objetos con otros. Técnicamente vamos a establecer los *outlets* de nuestros objetos.

ATENCIÓN: Para conectar unos objetos con otros, usaremos control+arrastrar desde un objeto sobre otro.



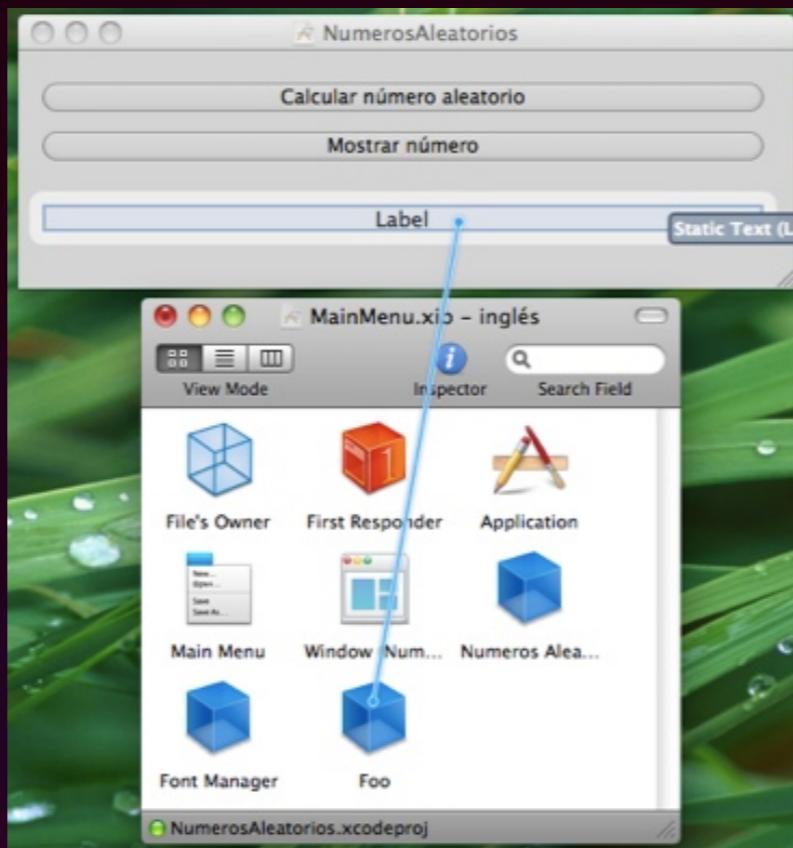
Como aún no tenemos mucha idea, vamos a fiarnos del siguiente diagrama sobre qué objetos tenemos que conectar:



Sé que esto puede asustarte un poco. Pero pronto lo entenderás todo. Recuerda que es tu primera aplicación y **todo esto sólo debe servirte para que cuando lo expliquemos ya lo hayas visto con anterioridad**.

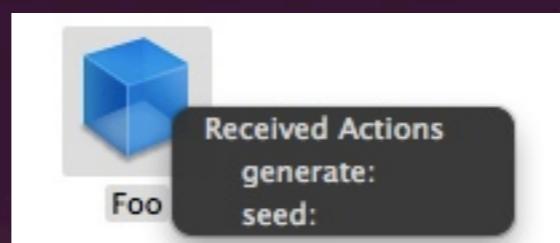
Según este diagrama lo que vamos a hacer es establecer que la variable `textField` apunte al objeto `NSTextField` en el hueco donde ahora mismo pone 'Label'.

Vamos allá: **Haz control+arrastrar** (o arrastrar con botón derecho) desde `foo` (en la ventana Doc) hasta la etiqueta que hemos llamado `Label`.



Ahora vamos a hacer que el outlet del botón Calcular número aleatorio apunte a `Foo`. Y además queremos que el botón siga el método `seed:`

Haremos lo mismo. Control+Arrastrar desde el botón 'Calcular..' hasta la instancia `Foo`. Y cuando el panel aparezca deberemos seleccionar el método `seed:` Haremos lo mismo con el botón "Mostrar..." pero seleccionando en el panel `generate:`



Recuerda seleccionar cada botón antes de hacer la conexión. De lo contrario quizá no aparezca el panel con las opciones `seed:` y `generate:`



Muy bien! Ya lo has hecho!

Como has trabajado con Interface Builder sólo tendrás que grabar y quedará todo guardado en el archivo xib.

Vuelta a Xcode

Probablemente seas nuevo en esto de la Programación así que **cuando terminemos este ejemplo haremos una breve explicación sobre todos estos términos (Clases, métodos, variables, instancias...)**. Todo esto lo conocerá bien alguien que ya sepa algo del Lenguaje C y del Objective-C, pero no te preocupes: tú también acabarás sabiéndolo con mucha facilidad.

Por el momento simplemente haremos Copy&Paste de este código (que corresponde a los métodos) justo antes de @end

```
- (IBAction)generate:(id)sender
{
    // Generar un número del 1 al 100 ambos incluidos
    int generated;
    generated = (random() % 100) + 1;
    NSLog(@"generated = %d", generated);
    // Le decimos al textField que cambie lo que se está mostrando
    [textField setIntValue:generated];
}
- (IBAction)seed:(id)sender
{
    // Determinar el número aleatorio con la hora
    srand(time(NULL));
    [textField setStringValue:@"Generator seeded"];
}
```

No te olvides de Guardar.

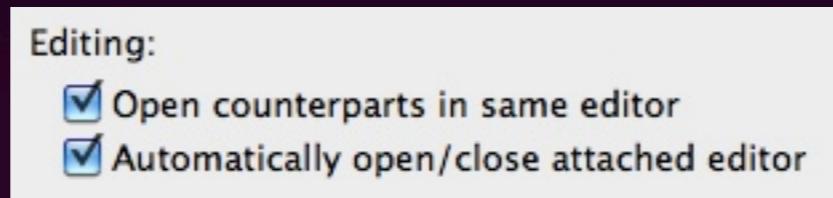


Ya hemos terminado.

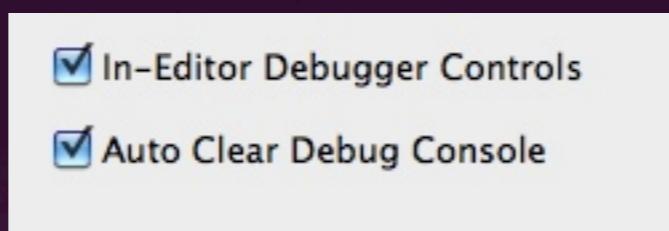
CONSEJOS: Pero antes de pulsar ‘Build & Run’ te voy a recomendar que hagas algunos cambios en tus preferencias que te serán de utilidad:

Entra en ‘Preferences’ (Menú Xcode > Preferences...):

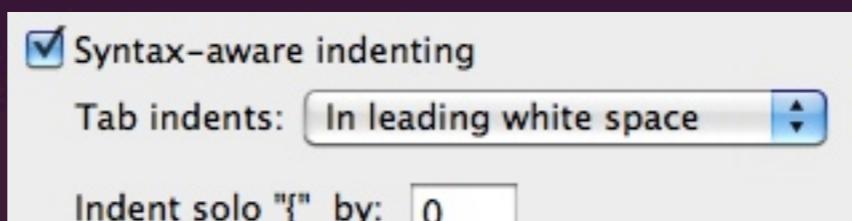
- En ‘General’, asegúrate de que tienes marcado “Open counterparts in same editor” (para pasar de `.h` a `.m` con agilidad en un mismo editor).



- En ‘Debugging’, deberás tener marcado ‘Auto Clear Debug Console’ (para que se borre el registro de errores tras cada proceso de ‘debug’). ‘Debug’, como ya dijimos antes, es eliminar errores de código.



- Mi último consejo es que en ‘Indentation’ tengas marcada la opción ‘Syntax-aware indenting’:

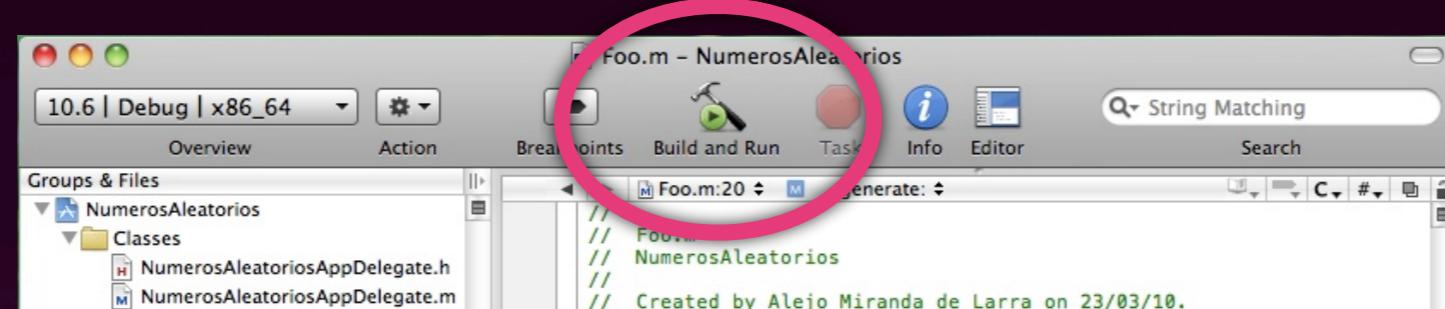




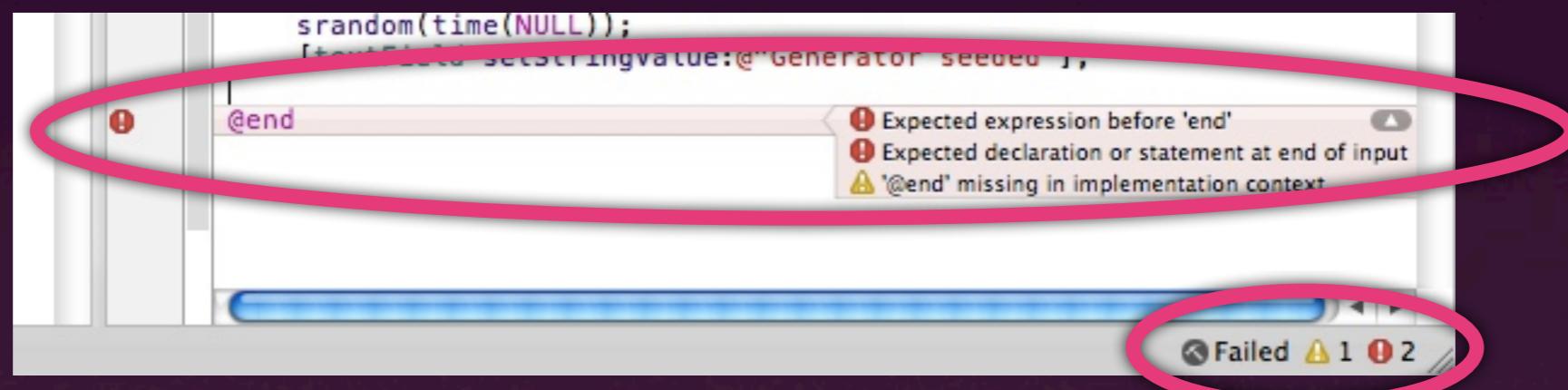
'Build & Run' tu primera Aplicación de Mac OS X

Ha llegado el momento de la verdad. Pese a la literatura que hemos usado para explicarte cada cosa, los pasos no han sido tantos ni tan complicados.

Tu Aplicación 'NumerosAleatorios' ya está terminada. Haz click en 'Built & Run'.



Si tu código tiene algún error, aparecerá un mensaje del compilador en la parte izquierda del editor y en la línea errónea. Además, abajo a la derecha aparecerá 'Failed' (fracasado) con la indicación de dos mensajes de error y una alerta (en este caso):



NOTA: A ti no debería aparecerte ningún error si has seguido todos los pasos. Yo he quitado la llave de la línea anterior a @end para ponerte el ejemplo del error, pero ahora mismo lo corrojo y guardo.



¡¡Perfecto!! Ya funciona correctamente :)

Enhorabuena, porque acabas de programar tu primera Aplicación usando Cocoa.