Uso básico de Xcode & Interface Builder en MAC OS X usando la librería Cocoa de Objective-C

Alcántara Cabrera Moisés
moycc94@gmail.com

Escuela Superior de Cómputo I.P.N.
XX Verano de la Investigación Científica
Universidad Autónoma de Puebla
Departamento de Aplicación de Microcomputadoras
Puebla, Pue. México.

26 de agosto de 2010

Resumen

En el documento se pretende presentar de una manera sencilla el uso básico de dos herramientas de programación, Xcode e Interface Builder, utilizando el sistema operativo MAC OS X. Empleando la librería Cocoa del lenguaje de programación Objective-C. El documento contiene una explicación del entorno de trabajo de estas aplicaciones, seguido del desarrollo de varios programas desde su creación hasta su ejecución.

1. Introducción

Mac OS X es un sistema operativo basado en UNIX, y desarrollado con tecnología de NeXT creada entre mediados de los 80's y finales de 1996 cuando Apple adquiere dicha compañía. La primera versión de este sistema operativo fue en 1999 llamada Mac OS X Server 1.0 ó "Hera".

Mac OS X está basado en el núcleo Mach. Ciertas partes de las implementaciones de UNIX por parte de FreeBSD y NetBSD fueron incorporadas en NEXTSTEP, en el que se basó Mac OS X. Nextstep fue el sistema operativo orientado a objetos desarrollado por Steve Jobs en NeXT después de dejar Apple en 1985. Mientras Jobs estaba afuera de Apple, la compañía intentó crear un sistema de "próxima generación" a través de los proyectos Taligent, Copland y Gershwin, con poco éxito [3].

Tiempo después NeXTSTEP fue seleccionado para ser la base del próximo sistema operativo de Apple. Steve Jobs regreso a Apple como CEO interino, y luego asumió el cargo total, acompañando la transformación de OPENSTEP en un sistema que sería adoptado para el mercado primario de Apple. El proyecto fue conocido inicialmente como Rhapsody y luego adoptó el nombre de Mac OS X [3].

2. Herramientas básicas de desarrollo

Para programar dentro de MAC OS X, éste sistema operativo contaba con las herramientas utilizadas por NeXT llamadas: Project Builder e Interface Builder, pero fue en Octubre de 2003 que se Apple modificó su entorno de desarrollo por el nuevo IDE llamado Xcode; el cual sustituyó a Project Builder únicamente, dependiendo de Interface Builder para el diseño de la interfaz de la aplicación.

Un proceso de desarrollo de un programa generalmente se divide en 3 partes: diseño, codificación y depuración. Cada uno de estas partes se conjuntan en herramintas importantes de desarrollo que utilizaremos. Será una breve descripción y más adelante observaremos el entorno de cada una de ellas.

En el diseño de una aplicación se debe tener en cuenta: la fucionalidad, la estructura y la interfaz de usuario, se deben esbozar ideas de la aplicación antes de escribir cualquier línea de código. Una de las herramientas que es muy eficaz para el diseño interfaces en MAC OS X es Interface Builder.

Interface Builder es una aplicación creada en 1988 por el desarrollador fránces Jean-Marie Hullot. Es una herramienta útil en el diseño de aplicaciones, lo que implica hacer interfaces más rápidas y gráficas. Permite unir componentes visuales, tales como son: ventanas, menus, botones, etc.; cada una de ellos tiene sus

propios atributos los cuales pueden modificarse, así como crear conexiones entre objetos. [1]. Guarda los archivos con extensión ".nib"que significa ("NeXTSTEP Interface Builder") pero con el cambio de sistema operativo y la creación de Xcode, la extensión cambió su nombre por "xib" que significa ("Xcode Interface Builder").

Xcode por su parte, es un IDE completo, con todas las características en torno a un flujo de trabajo sencillo que integra la edición de código fuente, con la construcción y compilación de los pasos, a través de una experiencia de depuración gráfica. [4].

Cocoa es un conjunto de frameworks orientados a objetos que permiten el desarrollo de aplicaciones nativas para Mac OS X, Principalmente el lenguaje con el que se programa con esta librería es Objective-C. Las aplicaciones automáticamente heredarán los comportamientos y las apariencias de Mac OS X, con pleno acceso a la potencia subyacente del sistema operativo UNIX. El uso de cocoa con el IDE de Xcode es simplemente la mejor manera de crear aplicaciones nativas de Mac [5].

3. Ambiente de trabajo de Xcode e Interface Builder

Comenzaré por explicar el ámbiente de trabajo de Xcode. Una vez ejecutado Xcode nos muestra una ventana de bienvenida con la posibilidad de elegir entre proyectos que hemos creado con anterioridad o crear uno nuevo, como lo vemos en la Figura 1.

Figura 1: Ventana de inicio de Xcode.

También nos permite elegir la opción de empezar con Xcode, la cual abre una ventana hacia un pequeño tutorial de como desarrollar aplicaciones en esta herramienta, éste contiene unos videos de ayuda básica, así como ligas hacía páginas oficiales de Apple en donde se encuentra información específica de la aplicación.

Al seleccionar un proyecto nuevo nos presenta una nueva ventana, la cual se observa en la Figura 2. Esta ventana nos da la posibilidad de elegir entre una aplicación de cocoa, un Script o una línea de comandos.

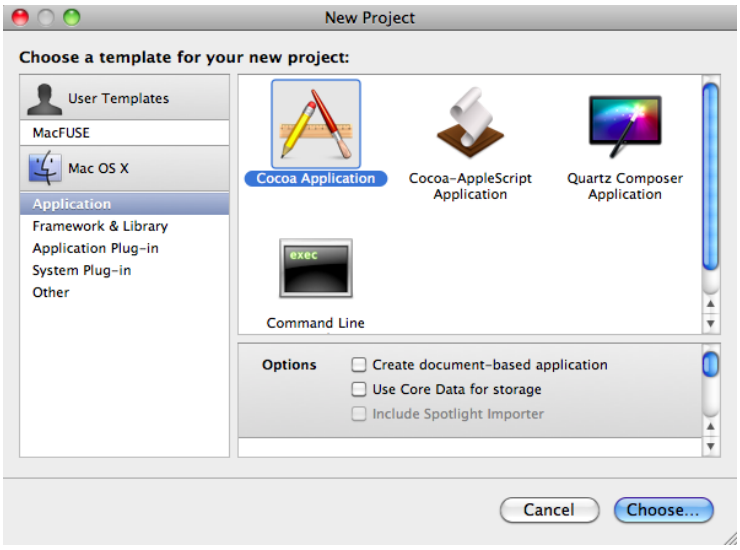

Figura 2: Elección de nuevo proyecto

Después de la eección del nuevo proyecto, nos pedirá un nombre y generará una ventana que contiene los archivos necesarios para la creación de nuestro proyecto, como se ve en la Figura 3.

Dentro de ésta imagen podemos ver que a la izquierda se encuentra nuestro grupo de archivos del proyecto que se divide los archivos del proyecto en carpetas, esto con el fin de organizarlos; en la parte de arriba esta una especificación de las versiones con las que se compila el programa, estas son modificables.

El martillo con la flecha representa nuestra herramienta de compilación y ejecución, en este programa esto lo realiza automáticamente al ser presionado.

En la parte central esta nuestro gestor de archivos, el cual contiene los archivos necesarios para la aplicación del programa, pero en este caso enlistados,

si alguno de ellos es modificable entonces aparece su contenido en el área de edición de archivos.

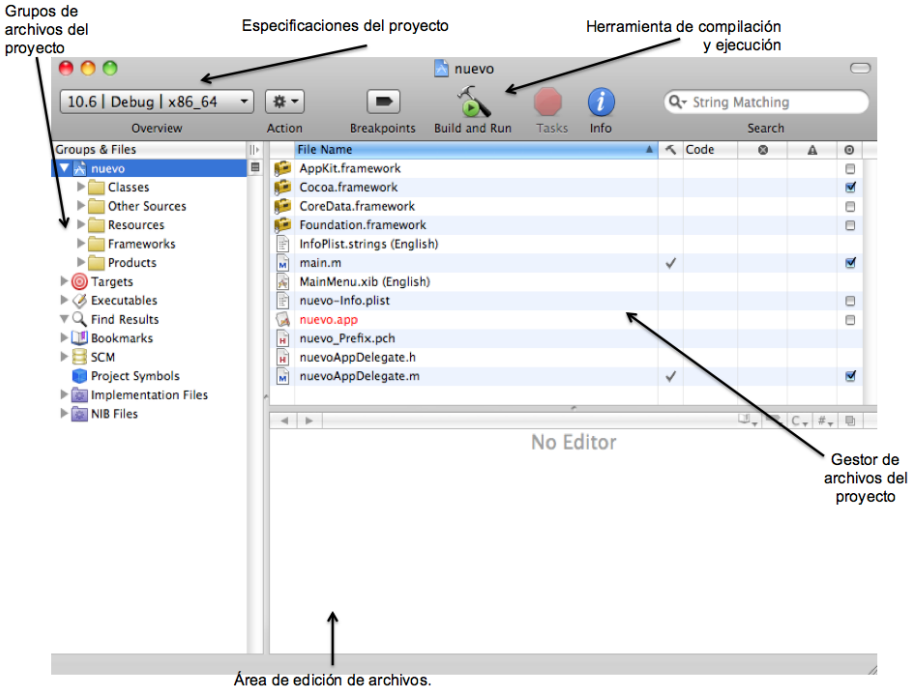

Figura 3: Entorno de trabajo de Xcode

Para el diseño de nuestra aplicación, Xcode hace uso de otra aplicación, Inteface Builder, la cual ayudará a la creación de intrfaces gráficas para nuestra aplicación.

Esta aplicación puede ser ejecutada independientemente de Xcode, pero en este caso en nuestro caso basta con abrir el archivo con extensión ".xib" para que la aplicación abrá automáticamente; el entorno de desarrollo de Interface Builder se muestra en la Figura 4.

En esta figura podemos observar cuatro ventanas principales, la primera la ventana de librerías, en ella se localizan los objetos que podemos arrastrar a nuestra aplicación, las Clases que se pueden utilizar, teniendo en cuenta que se pueden crear nuevas clases y subclases, y en la pestaña final se encuentran las Imagenes disponibles para nuestra aplicación, también se pueden agregar nuevas al proyecto.

En la parte central se encuentra nuestra ventana de la aplicación que crearemos la cual tiene como nombre principal el nombre del proyecto y que a su vez tiene un menú específico para la aplicación; para modificar los atributos de la aplicación como de sus elementos existe la ventana de Inspector de atributos que además de la moficación de los mismos, muestra las conexiones entre objetos.

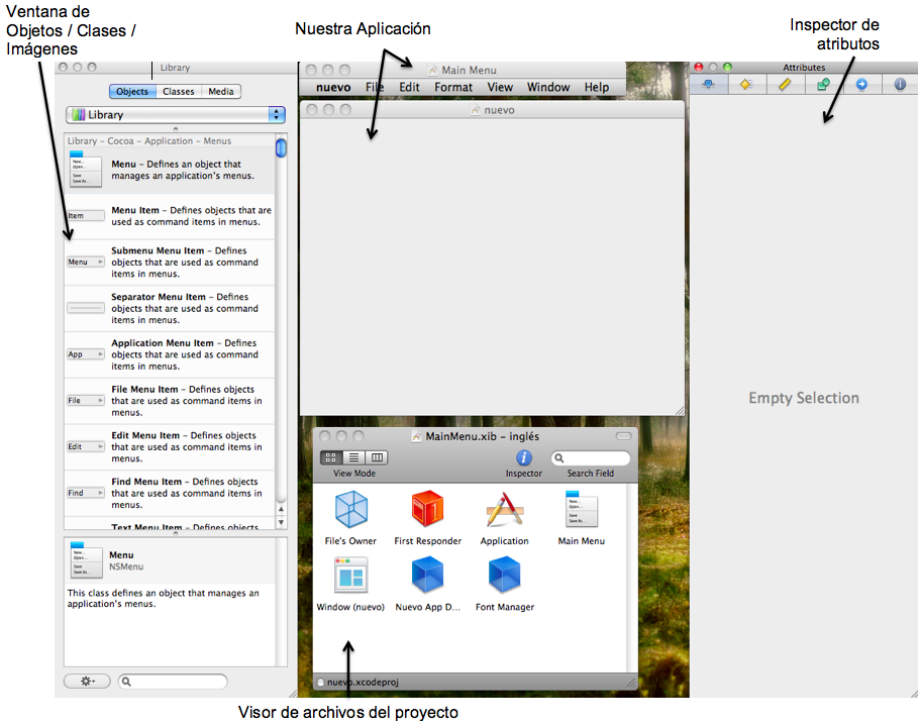

Figura 4: Entorno de trabajo de Interface Builder

Por último tenemos el visor de archivos del proyecto, en el cual se muestran los objetos inicializados, la ventana de nuestra aplicación, nuestro menú, etc. Esta hace que las conexiones entre objetos sea más fácil, así como la organización de los objetos de nuestra aplicación.

Conociendo el entorno de trabajo podrémos comenzar con el desarrollo de algunos ejemplos que se pueden desarrollar para conocer estas herramientas, con el objetivo de entender el uso de la interación de Interface Builder con Xcode, teniendo en cuenta que la programación de los ejemplos son con la librería cocoa del lenguaje de programación Objective-C.

4. "Desarrollo de aplicaciones"

4.1. "Ejemplo 1: Hola Mundo (paso a paso)"

Comenzaremos por ejemplo mas fácil, un hola mundo. Para esto se enumeran los pasos a seguir para el uso adecuado de las herramientas de programación.

1. Primero ejecutamos la aplicación, la cual tiene el siguiente icono de la Figura 5.

Figura 5: Icono Xcode

2. Una vez ejecutado elegimos la opción nuevo proyecto y en la siguiente ventana la opción de Aplicación de Cocoa, dandole el nombre de "holamundo" al proyecto. Esto nos genera una carpeta en la ruta específicada en donde queremos guardar el proyecto y la interfaz principal de Xcode sobre la que trabajaremos.

3. Para poder iniciar a trabajar sobre nuestra interfaz necesitamos buscar en nuestro gestor de archivos de la ventana de Xcode el archivo con extensión ".xib" el cual se encuetra en la carpeta de recursos, y presionar doble click sobre él. Esto nos abre Interface Builder en el cual crearemos nuestra interfaz.

4. Nos muestra una ventana con el título de nuestro proyecto, al igual que su menú. Podemos modificar el nombre de la ventana desde nuestra ventana de Atributos en la opción que dice Titulo, y de igual forma para su menú correspondiente.

5. Ahora nos dirigimos a la librería de objetos y elegimos una etiqueta la cual arrastraremos a nuestra ventana y para editar su contenido dando doble click sobre el texto o en el inspector de atributos. Para modificar la letra, color y tamaño nos dirigimos hacia el menú de Interface Builder y elegir Font —>Show Fonts y seleccionar las características que se desee para el texto.

6. Ahora guardamos los cambios en nuestra interfaz con File —>Save.

7. Procedemos a compilar y ejecutar nuestro proyecto, para esto en nuestra ventana de Xcode ubicamos un icono de un martilo el cual dice compilar y ejecutar, presionamos el icono y comenzará a compilarlo, si hay algún error

este muestra en el área de edición el archivo que lo contiene, y en la esquina inferior derecha el número de errores y advertencias, de lo contrario corre automáticamente el programa.

8. Si el programa compiló con éxito se muestra la siguiente Figura 6.

Figura 6: Ejecución hola mundo

9. Por último nos damos cuenta que dentro de nuestro gestor de archivos el archivo ejecutable de nuestra aplicación "holamundo.app" que al inicio estaba en color rojo, ahora cambia a color negro y que al presionar doble click podremos ejecutarlo; este archivo se encuentra ubicado en la carpeta de Productos.

4.2. "Ejemplo 2: Conversor de temperatura (Empezado con código)"

La siguiente aplicación es un conversor de grados Celsius a Fahrenheit, también muy simple, pero con esto definiremos como se realian las conexiones entre objetos, la creación de nuestros archivos .m y .h, así como la creación e inicialización de nuevas subclases y comenzaremos con el uso de código dentro de nuestra aplicación.

Primero creamos un nuevo proyecto con el nombre de "conversor", abrimos nuestro archivo ".xib" y creamos una interfaz como se muestra en la Figura 7. Para ello arrastraremos los objetos necesarios desde la librería de objetos y tomando en cuenta las siguientes recomendaciones:

Figura 7: Interfaz conversor de temperatura

- Para poner las letras de estilo y tamaño deseado utilizar el panel Show Fonts que se encuentra en la opción Font del menú principal de Interface Builder.
- En las cajas de texto debemos modificar sus atributos de tal forma que la primera de ellas que es la de entrada, sea editable y su alineación de texto sea hacia la derecha y en la segunda será alineación a la derecha también pero esta no será editable.
- Agreamos el botón y le cambiamos el texto; pero para meter la imagen a un lado del botón basta con ir a la ventana de librería y en la pestaña de Imagenes elegir una y arrastarla hasta el botón al cual se le quiere agregar. Se puede modificar la posición de el texto con respecto a la imagen en la opción posición de la ventana de atributos, siempre y cuando el botón este seleccionado.
- Por último debemos compactar nuestra ventana desde la esquina inferior derecha, para que se vea con la Figura 7.

Una vez creada la interface debemos definir las salidas y las acciones que se utilizarán en el programa, para ello en la pestaña de Clases de nuestra ventana de librerías debemos elegir la clase NSObject y con un click secundario elegir crear una subclase, a la cual se le dará el nombre de Calcular.

Debajo de la mitad de esta ventana podemos ver una sección nueva como se muestra en la Figura 8 en la cual agregaremos nuestras salidas y acciones, en las diferentes pestañas. Abrimos la pestaña de Outlets y agregamos dos salidas, la primera se llamará entrada y la segunda salida; nos pasamos a la pesaña de acciones y agregamos la acción calcular. Recordemos que para agregarlas se encuentra un signo como este: "+" en la parte inferior izquierda.

Una vez hecho esto debemos crear nuestros archivos .m y .h; dando doble click sobre nuestra subclase que creamos, elegimos la segunda opción que es crear archivos, nos saldrá una ventana que advierte que se añadirán nuevos archivos al proyecto, aceptamos la adición. Ahora podemos ver en nuestro gestor de archivos que existen dos archivos nuevos, Calcular.h y Calcular.m los cuales

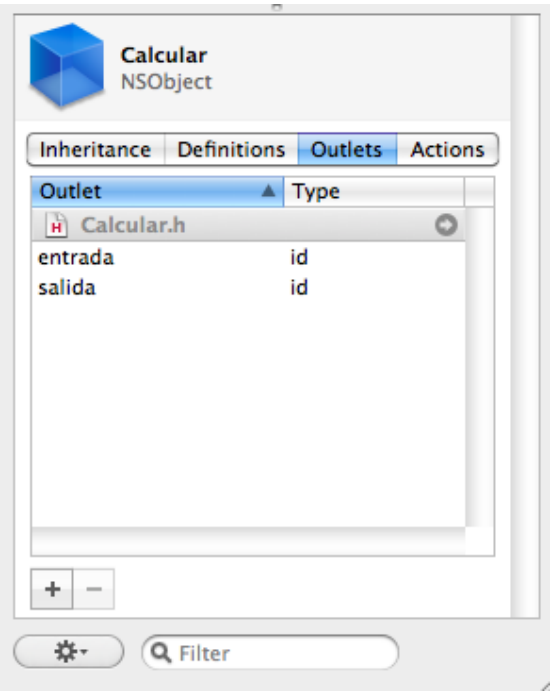

Figura 8: Adición de Salidas y Acciones

contienen un esqueleto de nuestro código. Para inicializar nuestro Objeto de la subclase basta con arrastrar la subclase hasta el visor de archivos.

El siguiente paso es crear las conexiones. Comenzaremos con nuestra caja de texto de entrada y presionando el botón Ctrl del teclado arrastraños la línea de conexión hasta el botón de Calcular y dentro de las opciones que se despliegan automáticamente elegimos la acción perform click y se conecta inmediatamente esto lo podemos ver en la ventana de Inspector de Atributos como se muestra en la Figura 9

De la misma forma hacemos la conexión desde el botón Calcular hasta el icono Calcular ubicado en el visor de archivos, conectandolo con la acción calcular. Después tomamos nuestr objeto calcular del visor de archivos y lo conectamos con la caja de entrada y la acción entrada, y de la misma forma con la caja de salida y la acción salida.

Una vez terminadas las conexiones modificamos nuestro archivo Calcular.h, en este archivo definiremos las clases, archivos de cabecera y varaibles globales. Su contenido define las cajas de texto como salidas, nuestra acción calcular la cual se implementa en el archivo Calcular.m, el código debe tener las siguientes

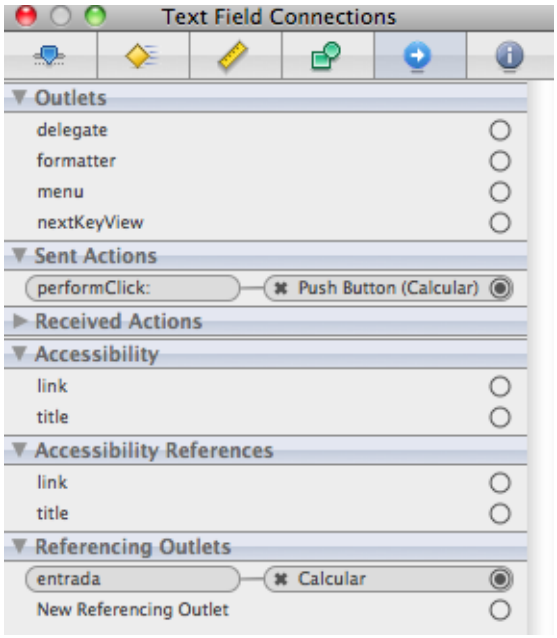

Figura 9: Conexión de objetos

líneas de código:

```
//
//  Calcular.h
//
//  Created by MacBook-MAC on 19/08/10.
//  Copyright 2010 IPN. All rights reserved.
//

#import <Cocoa/Cocoa.h>

@interface Calcular : NSObject {
    IBOutlet id entrada;
    IBOutlet id salida;
}
- (IBAction)calcular:(id)sender;
@end
```

Al término de la edición de Calcular.h guardamos los cambios y abrimos el archivo Calcular.m, en este archivo implementamos la acción calcular, obtenemos el contenido de la caja de texto, hacemos la conevrsión y presentamos en la caja de texto de salida el resultado. Además se improta nuestro archivo

Calcular.h. El código debe ser:

```
//
//  Calcular.m
//
//  Created by MacBook-MAC on 16/08/10.
//  Copyright 2010 IPN. All rights reserved.
//

#import "Calcular.h"

@implementation Calcular
- (IBAction)calcular:(id)sender {
    float gradosF;
[entrada selectText:self];
gradosF= ((9.0*[entrada floatValue])/5.0)+32.0;
[salida setFloatValue:gradosF];

}
@end
```

Guardamos los cambios y compilamos nuestro programa. Se debe ver como la Figura 10

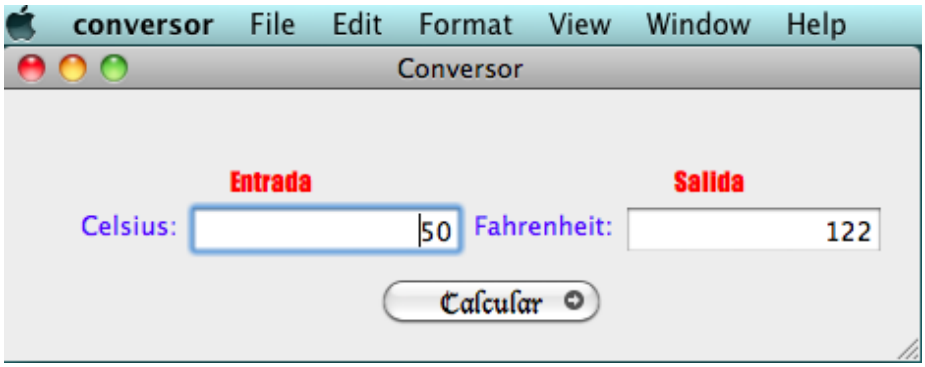

Figura 10: Ejecución Conversor de temperatura

Si se presentan errores de compilación se debe revisar la sintaxis del código, las llaves, etc. alguna de las más comúnes es que no se haya guardado la intefaz gráfica de la aplicación y que por tanto el archivo ".xib" no fue actualizado y no muestra nada. Si no genera el resultado volver a revisar que las conexiones esten hechas correctamente. Otro de los errores más comunes es que la imagen en el botón no se despliege, para ello hay que tener en cuenta que la imgen debe ir fuera del botón no dentro de él al momento de arrastrarlo.

4.3. "Ejemplo 3: Calculadora"

En este ejemplo observaremos a usar algunas matrices de objetos identificando cada uno por separado, ha realizar conexiones con la función *delegate*, también veremos como se crean nuevas ventanas o paneles.

Empezaremos por abrir un nuevo proyecto y en esta ocación le pondrémos el nombre de "calc" para refeirnos a la calculadora. Abrimos nuestro archivo de interfaz y creamos una interfaz parecida a la que se encuentra en la Figura 11

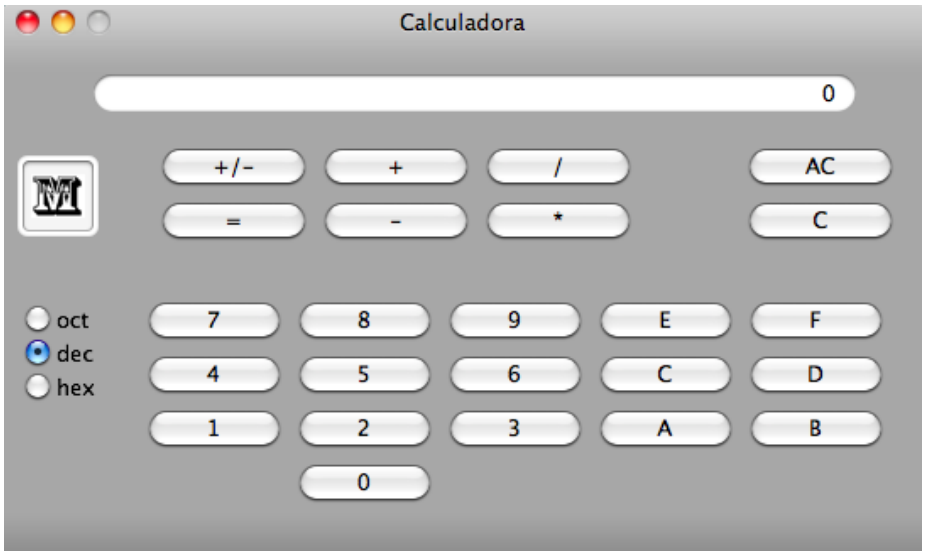

Figura 11: Interfaz de la calculadora

Para crear esta interfaz debemos tomar en cuenta algunos puntos importantes de los objetos que agregaremos, para ello debemos seguir los siguientes pasos:

- Primeramente la caja de texto debe tener una alineación del texto hacía la derecha, y que el objeto no pueda ser editable.
- Para crear las matrices debemos agregar un solo botón, ir al menú de Interface Builder elegir la opción Layout —>Embed Objects In —>Matrix y dentro del Inspector de Atributos modificar el número de columnas y filas que se requieran. Debemos observar que para las dos teclas que faltan a los costados del número cero sus atributos fueron modificados para ser invidibles, pero también inactivos. El proceso es el mismo para la matriz de operadores y botones de limpieza, pero en el caso de los botones de radio,

basta con mantener presionado el botón de Alt del teclado y arrastrar hacia abajo una de las esquinas para que aumente el número de botones.

- Ahora debemos ponerle una etiqueta a cada botón para distinguirlos dentro de cada matrix. Este cambio se hace presionando doble click al botón, y en la ventana de atributos podemos cambiar el título y cambiar su etiqueta o *tag*, el cual para los botones será el mismo número y para el teclado alfabético la letra A será el 10, la B el 11 y así sucesivamente. Para los botones de limpieza no tienen etiqueta dado que tienen una función única. Mientras que para los de los operadores y los de las bases, se muestran en el siguiente Cuadro 1.

Título	Etiqueta (*Tag*)	Título	Etiqueta (*Tag*)
+	1001	Octal	8
-	1002	Decimal	10
*	1003	Hex	16
/	1004		
=	1005		
+/-	1000		

Cuadro 1: Etiquetas de matrices.

Una vez terminada la interfaz, vamos a crear una nueva subclase de la clase NSObject a la cual llamaremos Control y le agregaremos las siguientes salidas y acciones como lo explicamos en el ejemplo anterior; estas se muestran en el Cuadro 2.

Salidas	**Acciones**
lectura	limpiar
baseMatriz	limTodo
	metedDig
	meterOp
	ponerBase

Cuadro 2: Etiquetas de matrices.

Una vez terminadas creamos nuestros archivos .m y .h e inicializamos nuestro instancia de la clase Control arrastrandola a nuestro visor de archivos para hacer las conexiones correspondientes. Enseguida se enlistan las conexiones a realizar:

1. Conectar la matriz del teclado alfanumérico con la instancia Control y elegir la acción meterDig.

2. Dando doble click sobre el botón CA conectarlo con la instancia Control y la acción limTodo.

3. Igualmente con doble click pero ahora sobre el botón C y conectarlo con la instancia Control y la acción limpiar.

4. Ahora de la instancia Control hacia la caja de texto en donde obtendremos el resultado de nuestras operaciones, eligiendo la salida llamada lectura.

5. Desde la matriz de operadores creamos la conexión con la instancia Control y la acción meterOp.

6. Nuevamente de la instancia de la clase Control pero ahora hacia la matriz de botones de elección de base, eligiendo con en las opciones de salida la llamada baseMatriz.

7. Y ahora de manera contraria a la anteior de la matriz de obtención de base hacia la instancia Control tomando la acción ponerBase.

Ahora podemos modificar nuestro archivo Control.h en el cual definiremos las variables de nuestros botones de operación, las variables globales y la función que utilizaremos para cada acción. El código debe quedar de la siguiente forma:

```
//
//  Control.h
//
//  Created by MacBook-MAC on 19/08/10.
//  Copyright 2010 IPN. All rights reserved.
//

#import <Cocoa/Cocoa.h>
#define SIGNO 1000
#define SUMA 1001
#define RESTA 1002
#define MULTIPLICACION 1003
#define DIVISION 1004
#define IGUAL 1005

@interface Control : NSObject {
    IBOutlet id lectura;
    IBOutlet id baseMatriz;
double X;
double Y;
    BOOL band1;
    BOOL band2;
int op;
int base;
}
```

```
- (IBAction)limTodo:(id)sender;
- (IBAction)limpiar:(id)sender;
- (IBAction)meterDig:(id)sender;
- (IBAction)meterOp:(id)sender;
- (IBAction)ponerBase:(id)sender;
@end
```

Mientras que para nuestro archivo Control.m la implementación queda de la siguiente forma:

```
//
//  Control.m
//
//  Created by MacBook-MAC on 17/08/10.
//  Copyright 2010 IPN. All rights reserved.
//

#import "Control.h"

@implementation Control

- (id)init
{
self = [super init];
X = 0.0;
return self;
}

- (IBAction)limTodo:(id)sender {
X = 0.0;
Y = 0.0;
band2 = NO;
band1 = NO;
    [lectura setDoubleValue:0.0];
}

- (IBAction)limpiar:(id)sender {
X=0.0;
    [lectura setDoubleValue:0.0];
}

- (IBAction)meterDig:(id)sender {
    if(band1){
Y=X;
X=0.0;
band1 = NO;
}
```

```
X=(X*10)+[[sender selectedCell] tag];
[lectura setDoubleValue:X];
}

- (IBAction)meterOp:(id)sender {
    if(band2){
switch(op){
case SIGNO:
X=-X;
break;
case SUMA:
X=Y+X;
break;
case RESTA:
X=Y-X;
break;
case MULTIPLICACION:
X=Y*X;
break;
case DIVISION:
X=Y/X;
break;
}
}
Y =X;
band2 = YES;
op = [[sender selectedCell]tag];
band1 = YES;
[lectura setDoubleValue:X];
}

- (IBAction)ponerBase:(id)sender {
NSString *myString;
if(band2){
switch(base){
case 8:
myString = [NSString stringWithFormat:@"%o", (unsigned int)X];
break;
case 10:
myString = [NSString stringWithFormat:@"%15.10g", X];
break;
case 16:
myString = [NSString stringWithFormat:@"%x", (unsigned int)X];
break;
}
}
```

```
Y =X;
band2 = YES;
base = [[sender selectedCell]tag];
band1 = YES;
[lectura setStringValue:myString];
}
@end
```

Al final guardamos los dos archivos y procedemos a realizar la conexión entre Files Owner y el objeto creado inmediatamente al crear el proyecto en el visor de archivos llamado Calc App Delegate, este nos ayudará a que automáticamente las acciones se realicen en respuesta a una acción del usuario, esta función proporciona un sistema para modificar el comportamiento de un objeto y en este caso para controlar con un objeto otro objeto.

Para agregar un panel de información a nuestro menú basta con arrastrar un objeto Panel a nuestra área de trabajo, en el podemos agregar imagenes y texto como cualquier ventana, este puede quedar como se ve en la Figura 12.

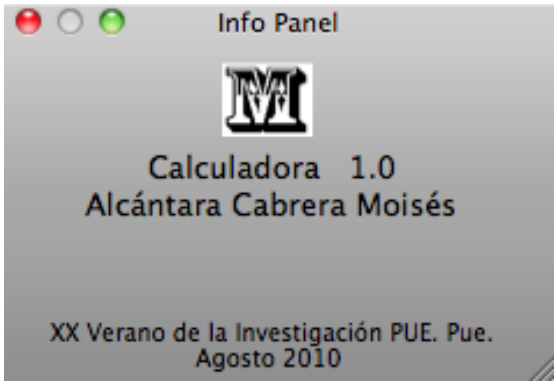

Figura 12: Interfaz de la calculadora

Una vez que lo tenemos listo conectamos del menú de nuestra aplicación la opción About Calc que se encuentra en calculadora con nuestro panel de información con la acción makeKeyAndOrderFront.

Guardamos las interfaces, y compilamos el proyecto. La ejecución debe verse como en la Figura 13 la cual muestra también el panel de información ejecutado.

Debemos probar cada una de las acciones que se encuentran en la calculadora, las cuales deben funcionar correctamente, de lo contrario se deben checar las advertencias o errores, la sintaxis, las conexiones, etc.

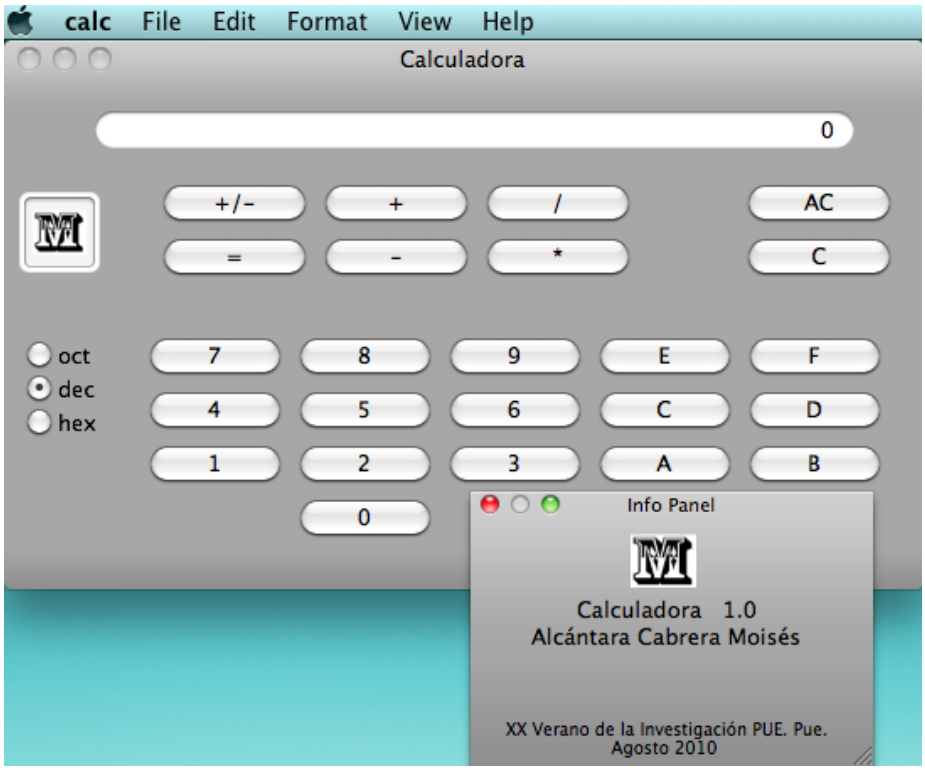

Figura 13: Interfaz de la calculadora

5. Conclusión

Notamos el uso básico de las herramietnas de programación Xcode e Interface Builder en MAC OS X, utilizando la librería cocoa de el lenguaje Objective-C. Debemos tener en cuenta que la forma de programación en estas herramientas proviene del desaparecido sistema operativo NeXTSTEP con Project Builder e Interface Builder pero con grandes cambios que ayudan a el usuario a fácilitar el desarrollo de una aplicación.

Los programas aquí presentados fueron compilados y ejecutados con el siguiente software:

- Versión del sistema MAC OS X Versión 10.6.4
- Xcode Versión 3.2.2 64 bit.
- Interface Builder Versión 3.2.2 (762)

6. Referencias

[1] NeXT Publications, *NeXTSTEP DEVELOPMENT TOOLS AND TECHNIQUES*, Ed. Addison-Wesley, USA, 1992.

[2] Aaron Hillegass, *Cocoa programming for MAC OS X*, Ed. Addison-Wesley, USA, May 2008.

[3] Wikipedia, MAC OS X, *http:es.wikipedia.org/wiki/Mac_OS_X* Fecha de modificación: Agosto 2010.

[4] Developer Tools Xcode, *http://developer.apple.com/tools/xcode/*, Fecha de modificación: 2010.

[5] MAC OS X, Cocoa. *http://developer.apple.com/cocoa/* Fecha de modificación: 2010.