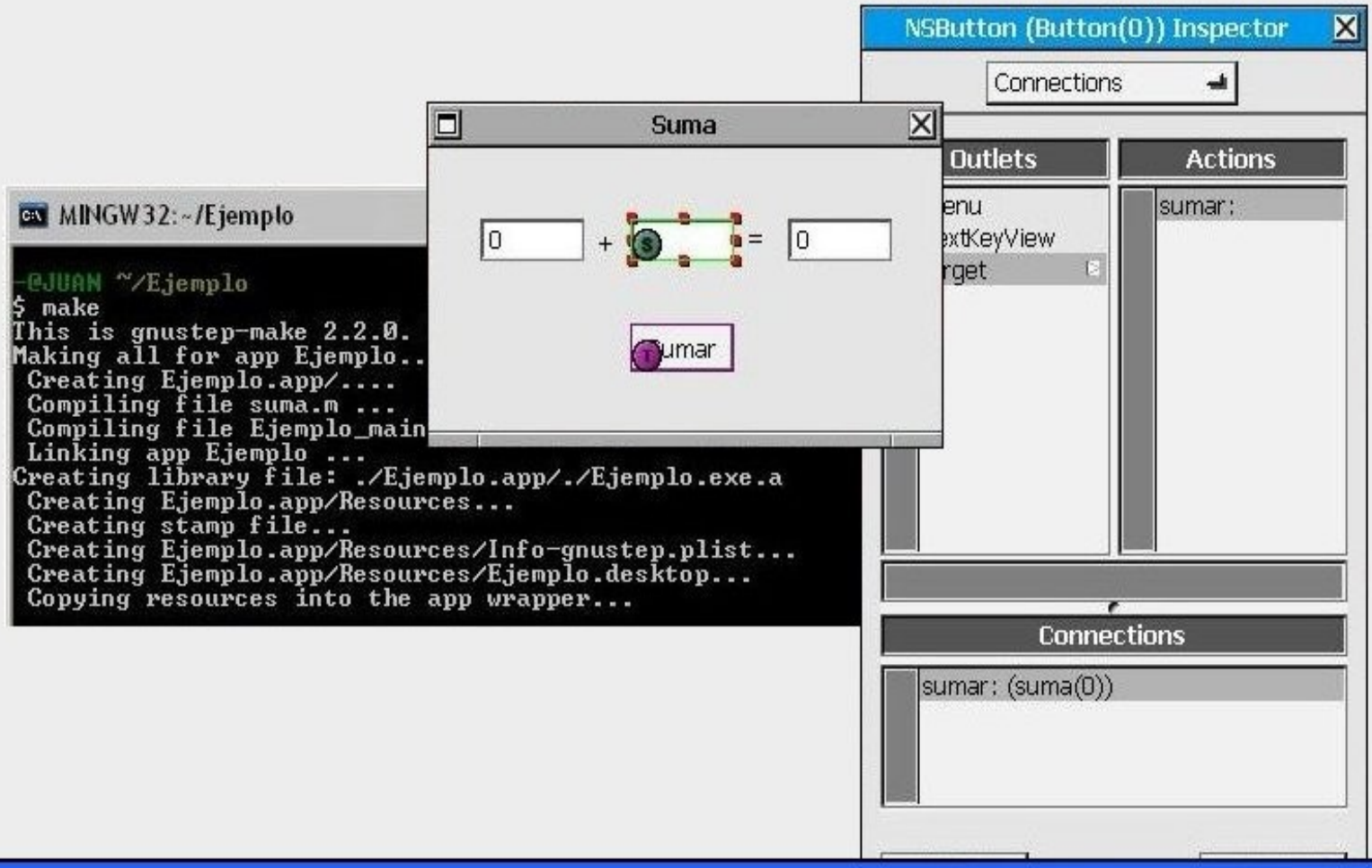
Un vistazo rápido a GNUstep en Windows

Germán Arias



Una guía paso a paso para crear una sencilla aplicación.

Documento publicado bajo licencia GFDLv1.3

Copyright (C) 2010 German Arias.

Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is available on http://www.gnu.org/licenses/gfdl.html.

Un vistazo rápido a GNUstep en Windows

Sobre este documento

El objetivo de este documento es mostrar la instalación, configuración y la realización de un sencillo ejemplo con GNUstep corriendo sobre Windows. No es su intención adentrar al lector en el lenguaje Objective-C o en el uso de los frameworks de GNUstep. Para ello puede consultarse otra documentación en http://gnustep.wordpress.com/about/manual-de-gnustep/ o la documentación oficial en inglés.

¿Que es GNUstep?

GNUstep es un entorno de desarrollo libre, de plataforma cruzada y orientado a objetos, para el desarrollo de aplicaciones de escritorio. Hace uso del lenguaje Objective-C, e implementa las especificaciones OpenStep. Consiste en un conjunto de Frameworks y herramientas de desarrollo. Para más información puedes visitar www.gnustep.org (en inglés) o www.gnustep.wordpress.com (en español).

Instalación

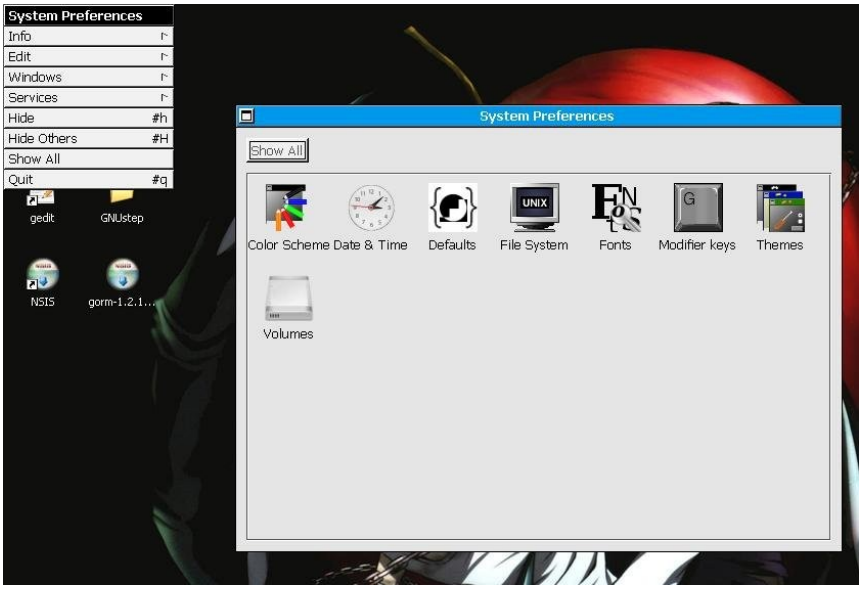
Los paquetes necesarios para poder utilizar el entorno de desarrollo GNUstep en Windows son los siguientes:

- GNUstep System-0.23.0
- GNUstep Core-0.23.0
- GNUstep Devel-1.0.0
- SystemPreferences-1.1.0
- Gorm-1.2.10
- Gedit
- NSIS

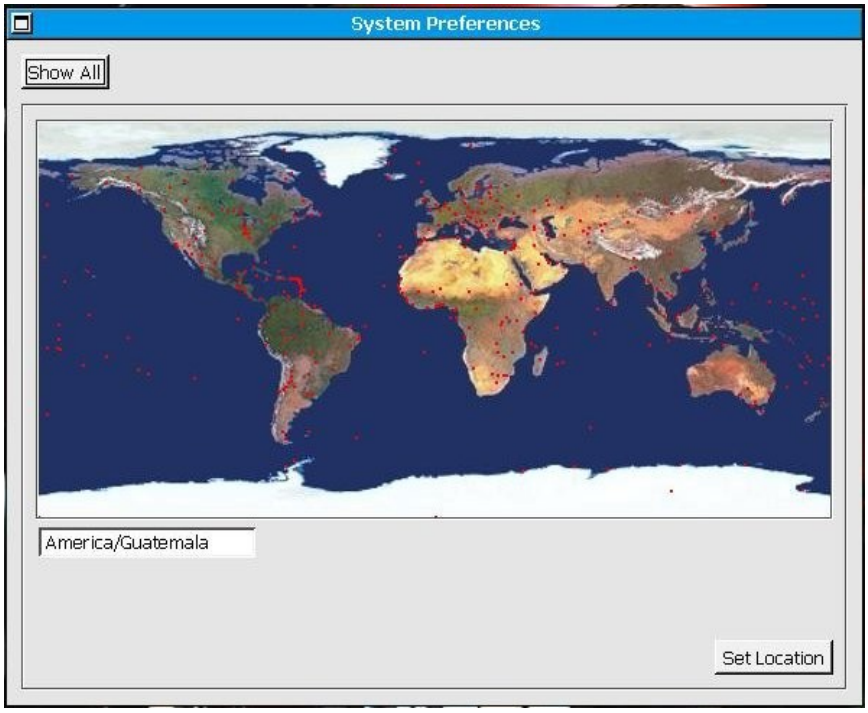
GNUstep System y GNUstep Core deben instalarse antes (y en este orden) que GNUstep Devel, SystemPreferences y Gorm. Gedit es necesario puesto que ProjectCenter no funciona apropiadamente en Windows y se necesita un editor que soporte codificación UTF-8 y tenga resaltado de sintaxis para Objective-C (aunque si se desea, puede utilizarse también Emacs, Xemacs, Vim, etc.). GNUstep Devel provee herramientas como SVN, CVS, etc. Mientras que NSIS es necesario para empaquetar las aplicaciones para Windows (crear instaladores).

Configuración

En SystemPreferences (*GNUstep -> Apps -> SystemPreferences*) es posible modificar los colores para las aplicaciones de GNUstep. En la imagen, en el módulo Color Schemes y en la sección Color Settings, he modificado los colores para: **Control Backing** (para el color de los controles), **Window Backing** (para el color del lienzo de las ventanas), y **Window Frame** (para el color de la barra de título). Aunque también puede resultar necesario modificar **3D Light Shadow** (el borde de los controles, para que se distingan del lienzo de la ventana), **Selected Menu Item** (el color del ítem seleccionado en el menú) y **Selected Control Backing** (el color del control seleccionado)

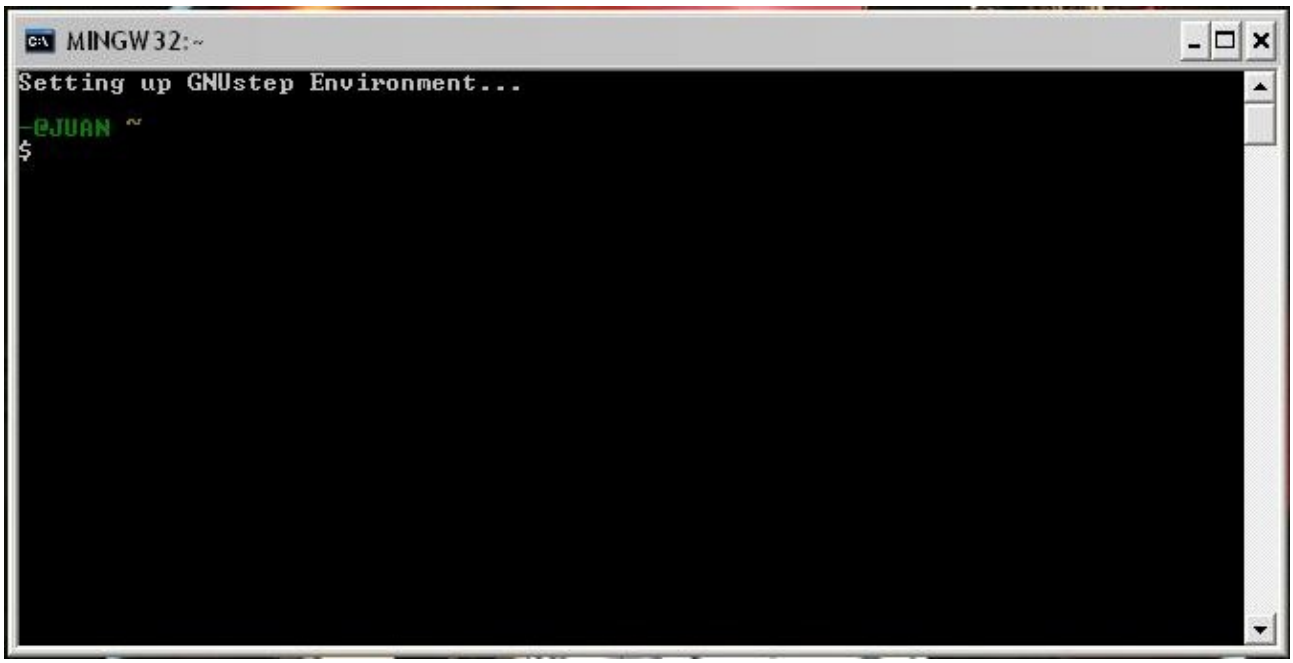


También puede configurarse la zona horaria en el módulo Date & Time seleccionando la ubicación en el mapamundi.



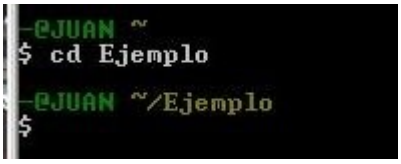
Un ejemplo

Rápidamente veremos como realizar una sencilla aplicación en Windows. Para ello, primero iniciemos el Shell instalado por GNUstep (*GNUstep -> Shell*). Esto creara una carpeta personal en la ruta *C:\GNUstep\home*, y nos ubicara en dicha carpeta:



Debemos crear una carpeta para almacenar los archivos de nuestra aplicación. Llamaremos a esta carpeta *Ejemplo*, y la creamos con el comando **mkdir**:

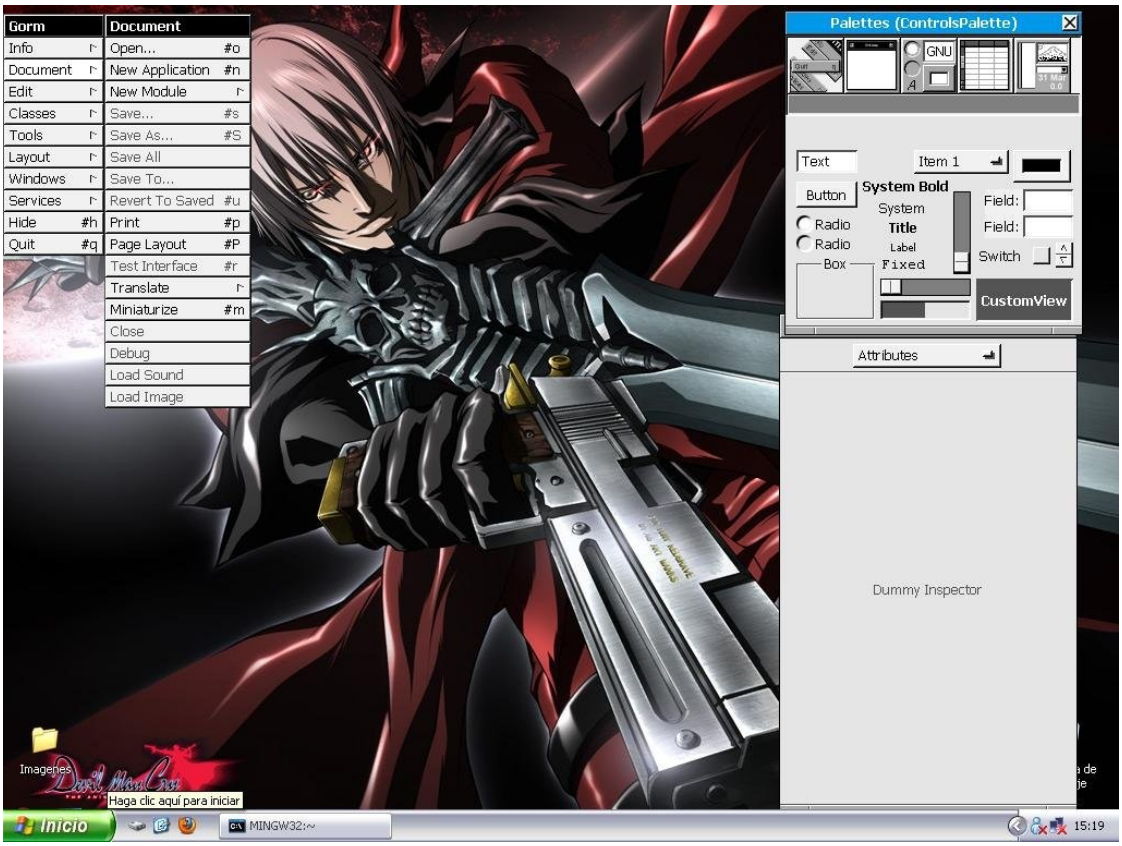
Ingresemos a dicha carpeta con el comando **cd**:



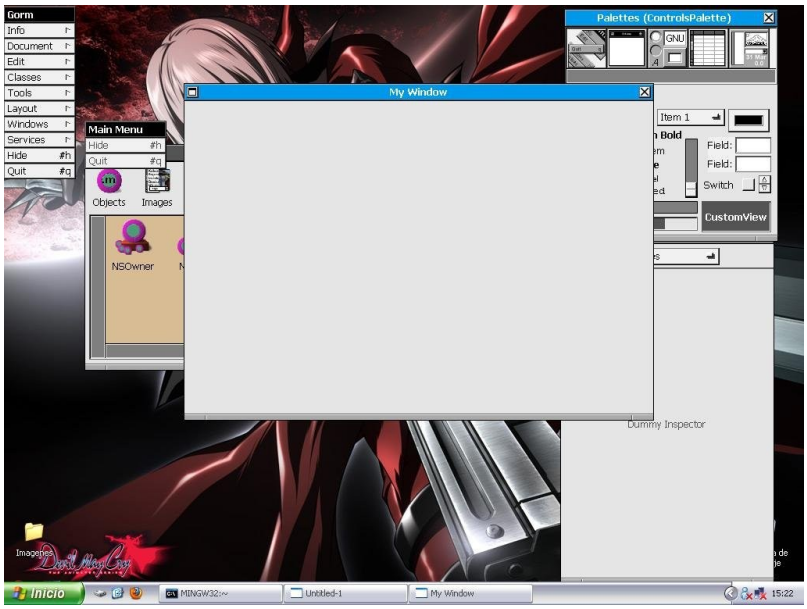
Dentro de esta carpeta debemos crear una carpeta llamada *Resources*, que es donde estará guardado el archivo **gorm** de la interfaz gráfica de nuestra aplicación.

Ahora pasemos a crear la interfaz gráfica de nuestra aplicación. Para ello iniciemos Gorm (*GNUstep -> Apps -> Gorm*).

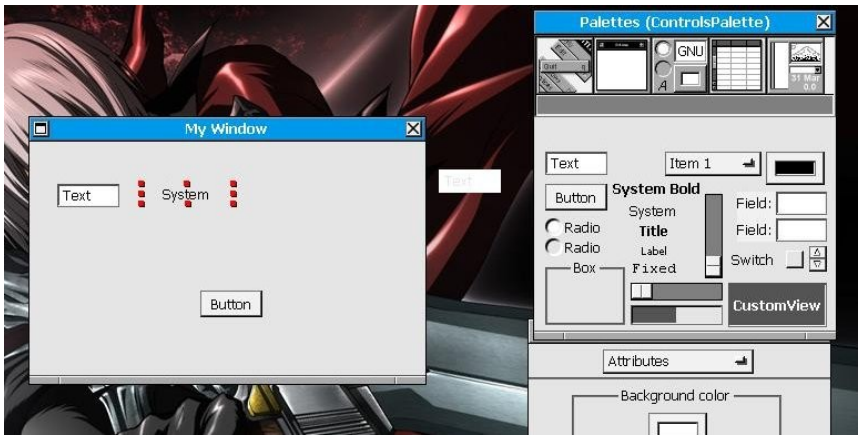
NOTA: Al realizar este ejemplo deben tenerse en cuenta algunos bugs de GORM en Windows. El primero es que el cursor no cambia apropiadamente, lo que significa que al arrastrar componentes desde la paleta a nuestra interfaz, veremos simplemente un rectángulo y no el componente correspondiente. El segundo bug sucede al querer cambiar el titulo del menú en el Inspector. Si se da un clic sobre el campo del nombre, no podremos cambiar el titulo. Lo que debe hacerse es darse un clic en los alrededores del campo, con esto el cursor se ubicara dentro del campo y podremos editarlo. Algo similar sucede al querer nombrar o renombrar una clase. Otra cosa a tener en mente, es que al querer abrir un archivo de interfaz con Gorm debe seleccionarse el archivo ***NombreAplicacion.gorm*** y no ***objects.gorm***. Aun cuando el panel de Gorm para abrir archivos muestra que ***NombreAplicacion.gorm*** es una carpeta y no un archivo.



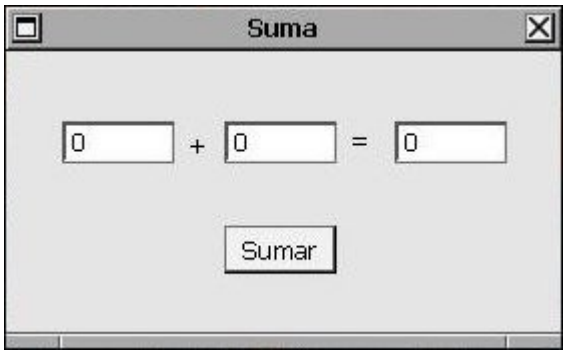
Y creamos una interfaz de aplicación desde el menú en *Document -> New Application*. Esto hará que aparezca un menú estándar (el de nuestra aplicación) y una ventana que estará configurada para aparecer al momento de lanzar nuestra aplicación.



Cambiemos el tamaño de la ventana (con la barra para dimensionar en la parte inferior de la ventana) y agreguemos controles desde la paleta arrastrándolos con un clic izquierdo. Debemos agregar dos componentes **Text**, dos etiquetas **System** y un componente **Button**. Dando un doble clic sobre cada uno de ellos podemos cambiar el texto que contienen. También con un doble clic puede cambiarse el texto en cada uno de las opciones del menú. El titulo del menú[1] y el de la ventana se cambian en el Inspector. Seleccionando el menú o la ventana y cambiando el nombre en el campo correspondiente (en la sección *Atributtes*).

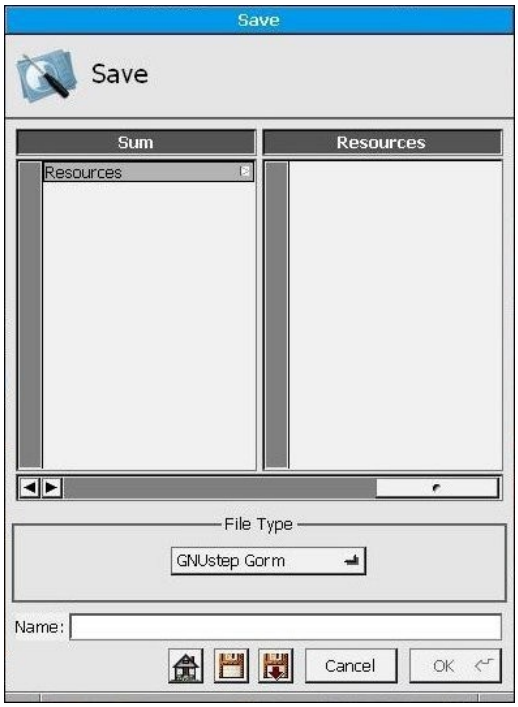


El resultado final debe ser el siguiente:



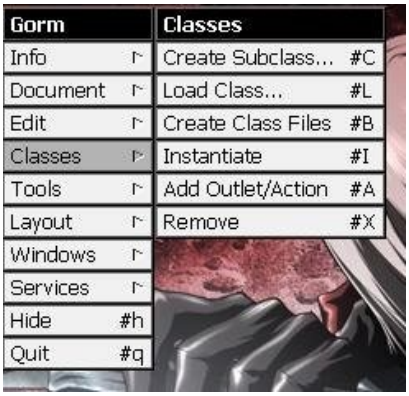
1 Recuérdese la nota sobre bugs.

Procedemos a guardar nuestra interfaz con el nombre **Ejemplo** en nuestra carpeta *Resources*. Para ello, en el menú, seleccionamos *Document -> Save*, y en el panel que aparece seleccionamos la carpeta *Resources* y escribimos el nombre **Ejemplo**. Por último un clic en OK, para guardar nuestra interfaz.

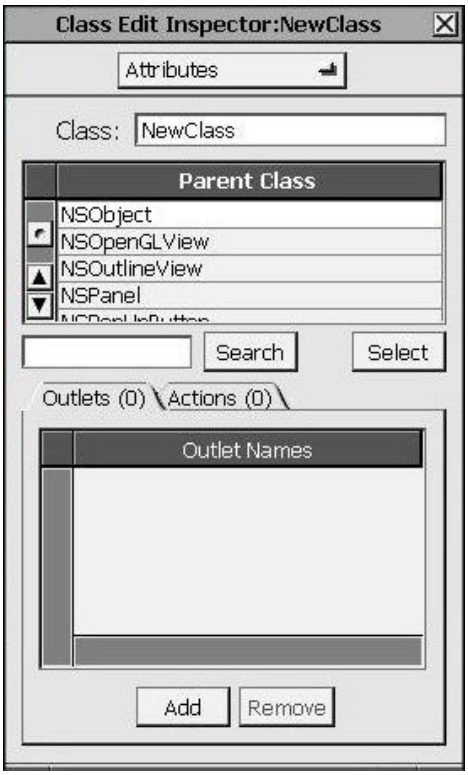


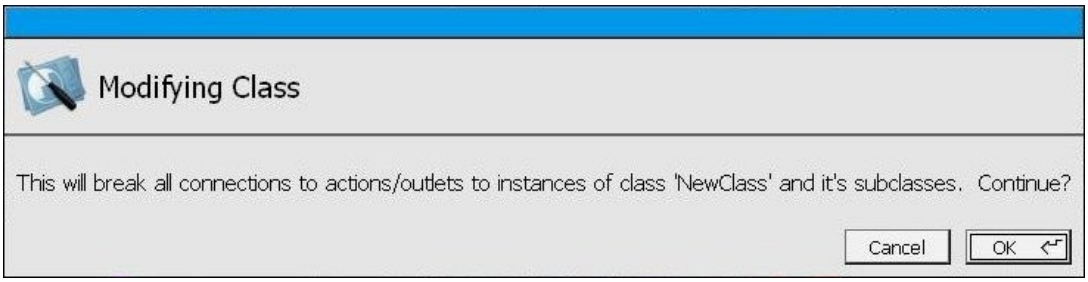
Ahora vamos a crear una clase que controle el funcionamiento de nuestra aplicación. Para ello en la ventana de nuestro documento, seleccionamos la pestaña **Classes**, y seleccionamos la clase **NSObject**.

Ahora en el menú seleccionamos *Classes -> Create Sublcass...*



Esto creara una subclase de la clase NSObject. En el Inspector, le cambiamos el nombre[2] a esta nueva clase. En este caso la llamaremos **suma**. Escribiendo *suma* presionamos ENTER, lo que presentara un panel de alerta.





Este panel nos advierte de que las conexiones de nuestro objeto se perderán al renombrarlo. Sin embargo, esta es una clase nueva, por lo que no debemos preocuparnos de esto, y damos un clic en OK.

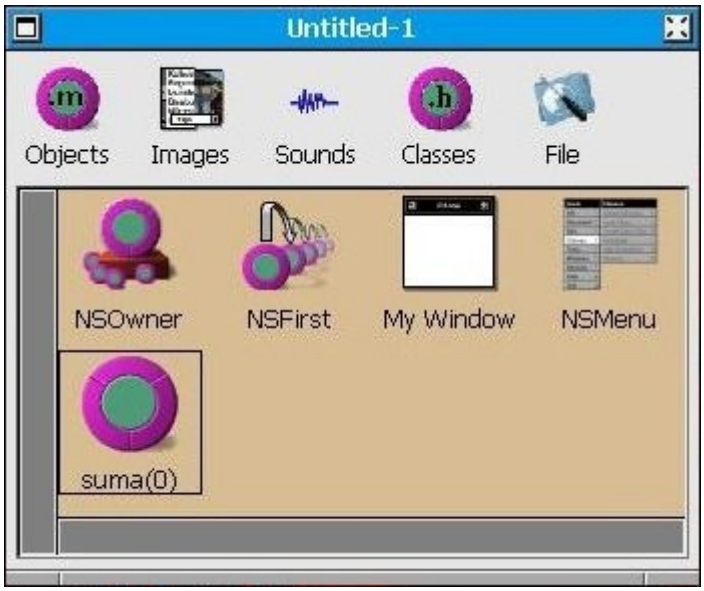
Ahora debemos agregarle tres Outlets a nuestra clase que conectaran con los tres componentes **Text** en nuestra interfaz. Para ello en la sección Outlets, mediante el botón *Add*, agregamos tres Outlets y les cambiamos el nombre a *sumando1*, *sumando2* y *resultado*.

2 Recuérdese la nota sobre bugs.

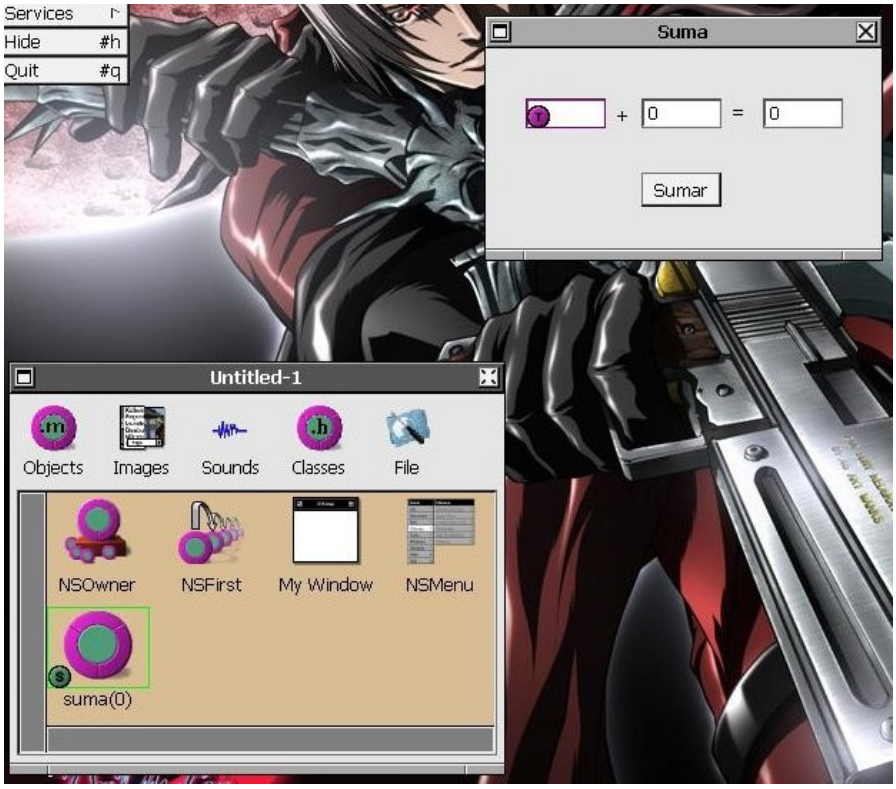
Debemos agregarle también un Action al cual estará conectado el botón de nuestra interfaz. Esto se hace en la sección Actions mediante el botón *Add*. Llamaremos a este Action *sumar:* (los dos puntos al final son importantes).

Ahora, seleccionamos nuestro clase **suma**,

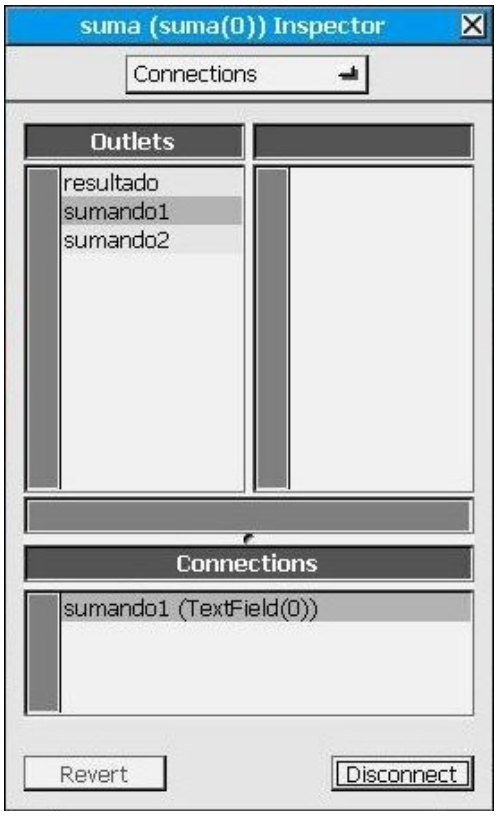
Y en el menú seleccionamos *Classes -> Instantiate*, esto creara un objeto en la sección Objects de nuestro documento. Y dicho objeto estará creado a partir de nuestra clase **suma** (es decir, poseerá los Outlets y Actions que definimos en la clase).



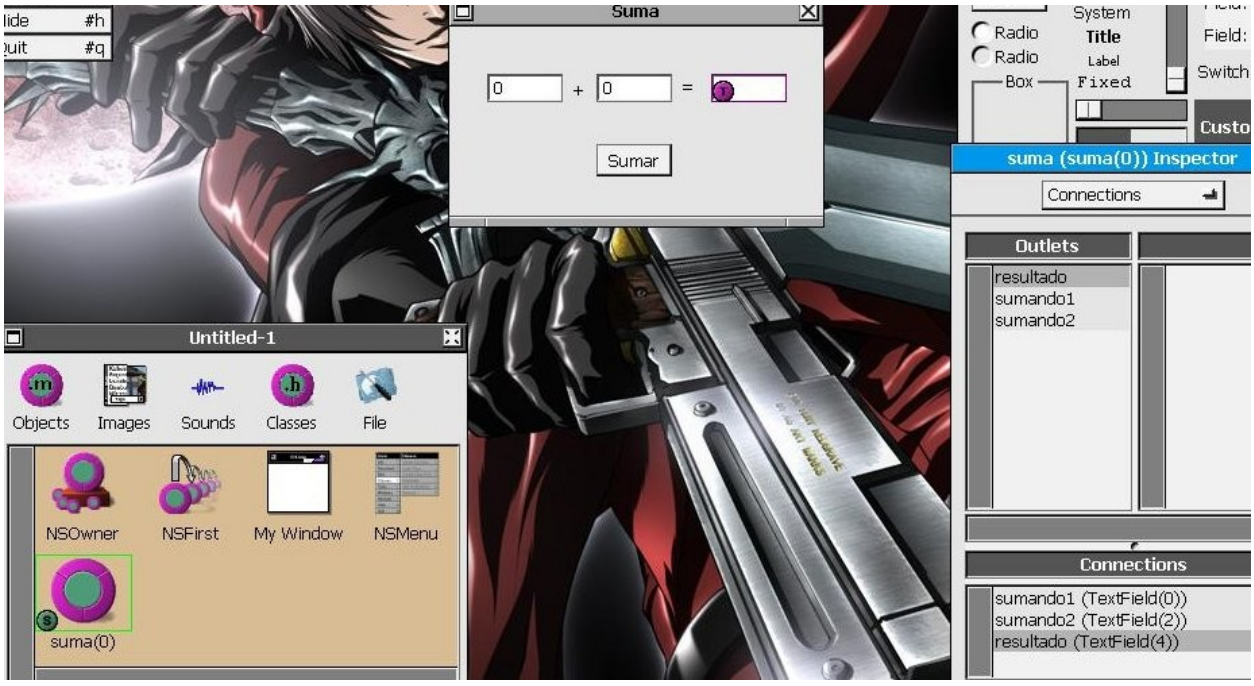
Ahora solo es cuestión de realizar las conexiones. Primero conectemos los tres Outlets. Para ello, manteniendo la tecla *Ctrl* presionada, arrastramos nuestro objeto **suma** al primer componente **Text** de nuestra interfaz. Debe aparecer un pequeño circulo verde con la letra **s** de *source* (fuente) en nuestro objeto **suma**, y un pequeño circulo rojo con la letra **t** de *target* (objetivo) en el componente **Text**.



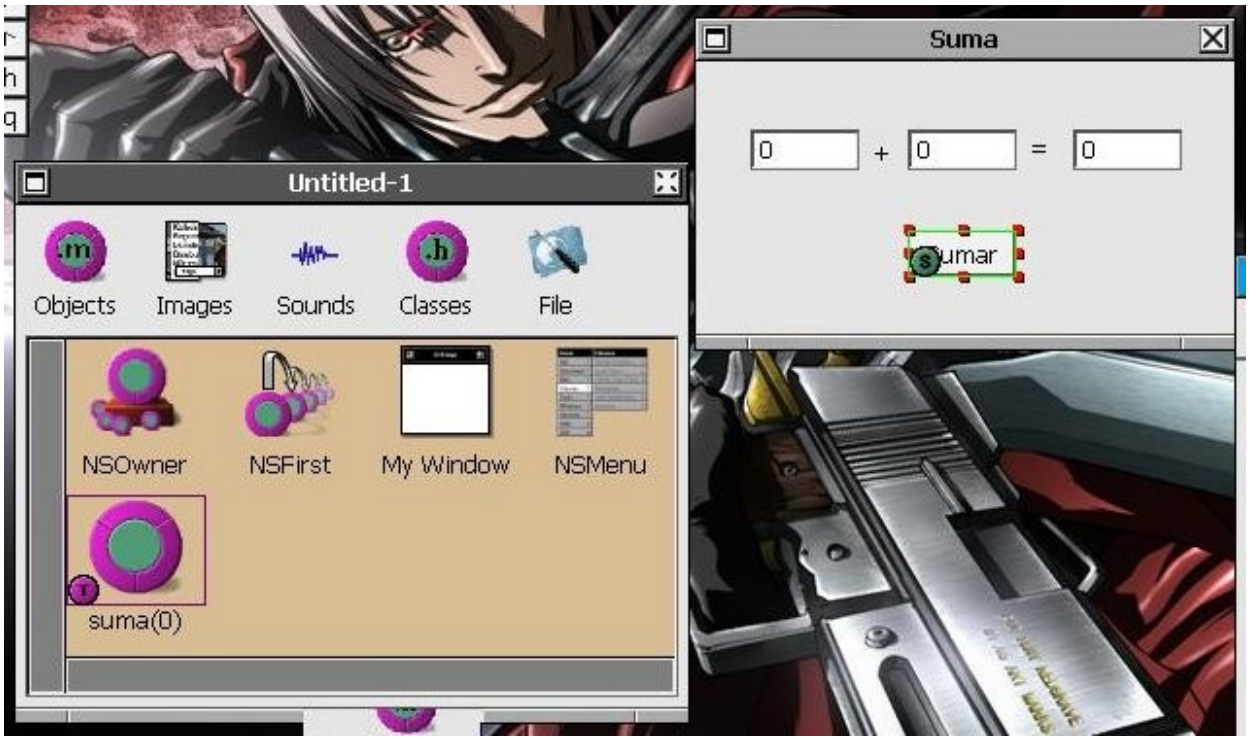
Ahora en el Inspector seleccionamos el Outlet que deseamos conectar, en este caso es el llamado *sumando1*. Seleccionado el Outlet damos un clic en el botón *Connect* para establecer la conexión, que debe aparecer ahora en la sección *Connections*.



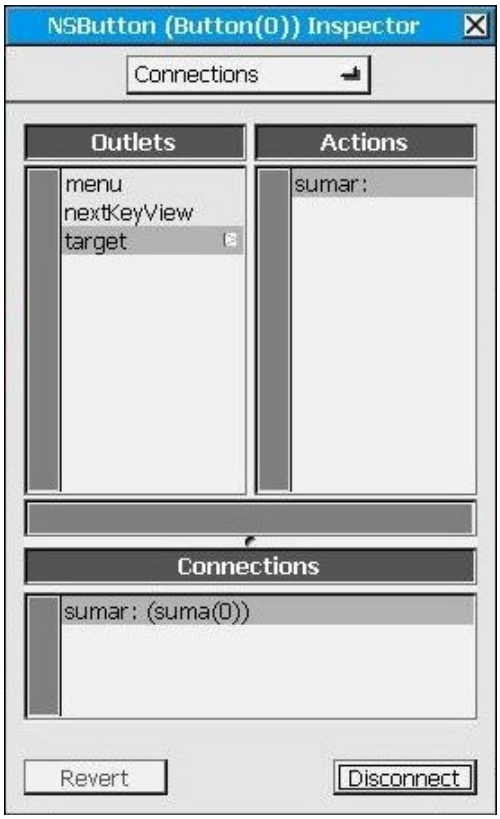
De igual forma conectamos nuestro objeto **suma** con el segundo componente **Text** mediante el Outlet *sumando2*. Y con el tercer componente **Text** mediante el Outlet *resultado*. Estos Outlets le permitirán a nuestro objeto **suma** obtener datos (en el caso de los sumandos) y escribir datos (en el caso del resultado) en los componentes **Text**.



Ahora debemos conectar el botón Sumar con el Action *sumar:*. Manteniendo la tecla *Ctrl* presionada, arrastramos el botón **Sumar** hacia nuestro objeto **suma** (observese que ahora la conexión es de un componente, el botón, hacia nuestro objeto **suma**). Esto hará que el botón **Sumar** ejecute el Action (método) *sumar:* al ser presionado.

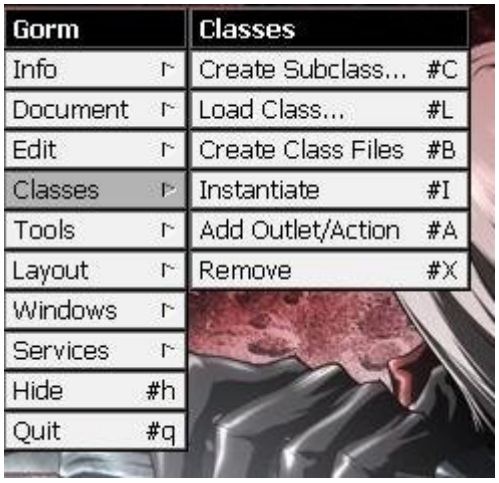


Y en el Inspector, en la sección *target*, seleccionamos el Action *sumar:* y con un clic en el botón *Connect* creamos la conexión.



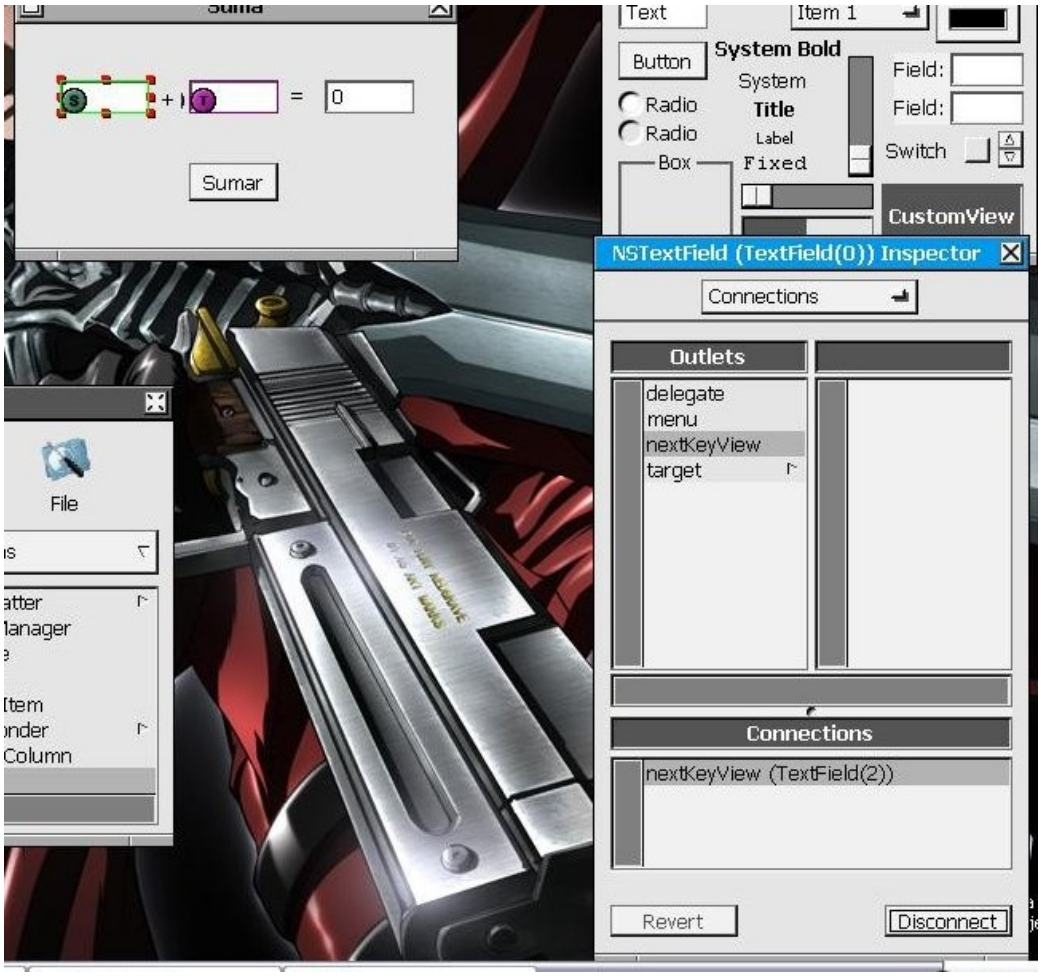
Establecidas las conexiones con nuestro objeto **suma**, podemos crear los correspondientes archivos de este. Primero seleccionemos nuestra clase en la sección **Classes** de nuestro documento gorm.

Vamos al menú y seleccionamos *Classes* -> *Create Class Files*

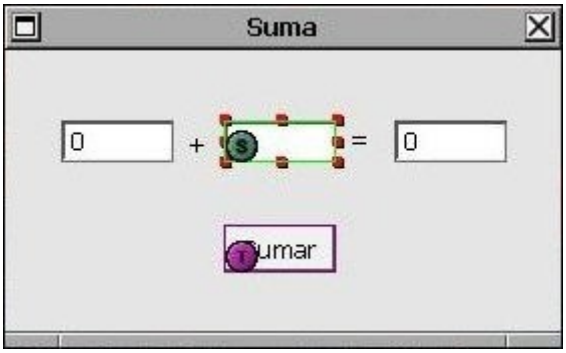


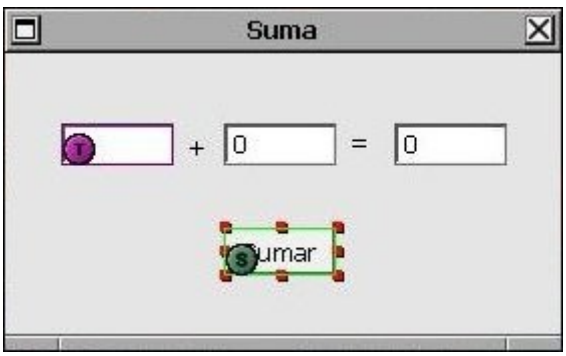
En el panel que se abre seleccionamos nuestra carpeta **Ejemplo** para guardar los dos archivos de nuestro objeto, *suma.h* y *suma.m* (no debemos cambiarles el nombre).

Por último, vamos a establecer la forma en que el usuario podrá moverse a través de los componentes de nuestra interfaz mediante la tecla *Tab*. Manteniendo la tecla *Ctrl* presionada, arrastramos el primer componente **Text** hacia el segundo. Y en el Inspector seleccionamos el Outlet *nextKeyView* y con un clic en el botón *Connect* establecemos la conexión.



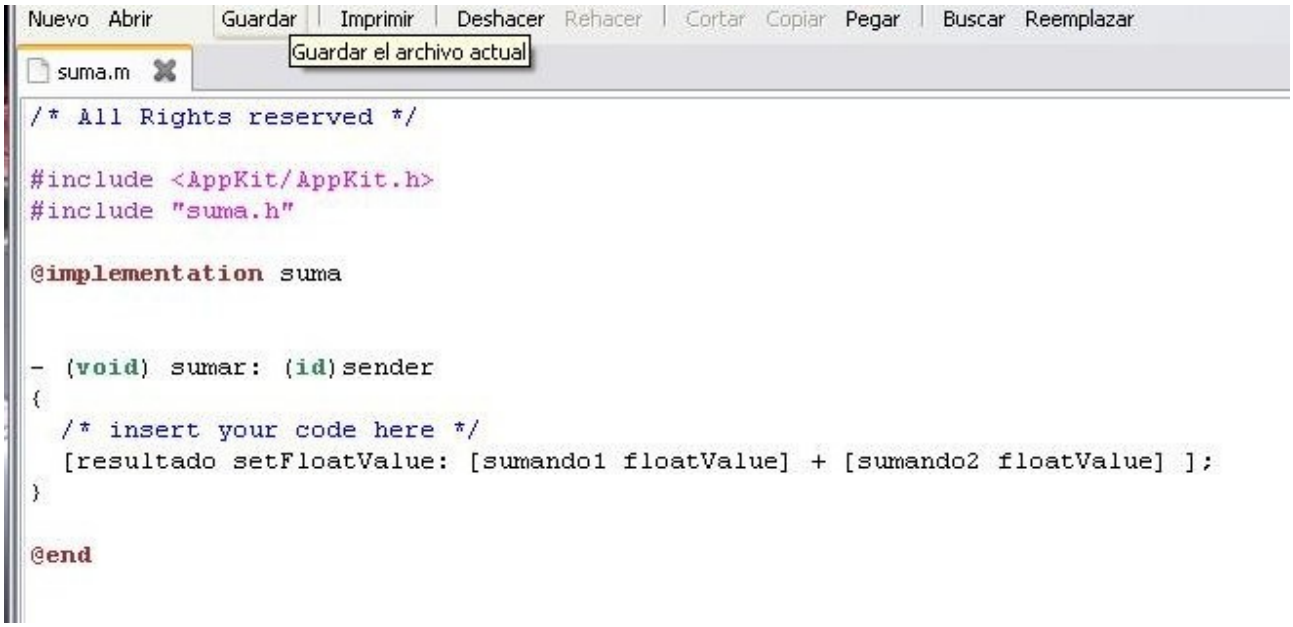
De igual forma conectamos el segundo componente **Text** con el botón **Sumar**. Y el botón **Sumar** con el primer componente **Text**. En ambos casos mediante el Outlet *nextKeyView*.



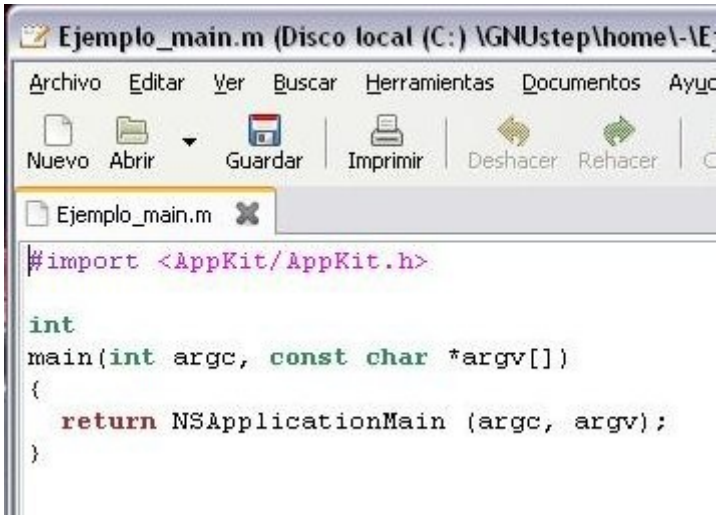


Por último guardamos los cambios realizamos en nuestra interfaz (**Document -> Save**) y con esto hemos terminado la parte gráfica de nuestra aplicación.

Mediante la aplicación Gedit[3] vamos ahora a modificar nuestro archivo **suma.m**. Agregaremos la línea justo debajo del comentario que dice “insert your code here”. Esta línea es la que lleva a cabo la suma, y pone el resultado en el componente **Text** correspondiente. Observese el uso de los Outlets en la ejecución de los mensajes (el método *floatValue* devuelve el valor ingresado en el componente **Text** correspondiente, mientra que el método *setFloatValue:* escribe un valor en el mismo).



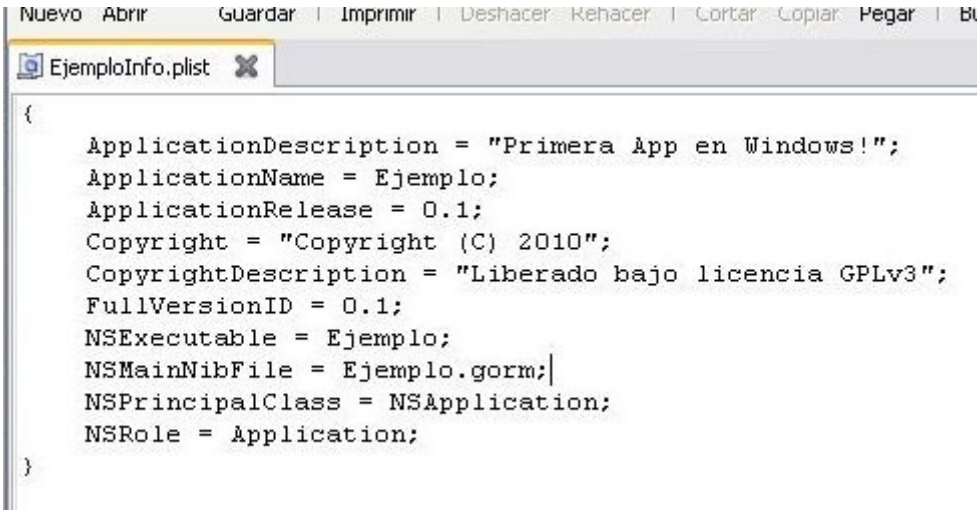
Debemos ahora crear tres archivos para que nuestra aplicación quede terminada. Al primero de ellos lo llamaremos **Ejemplo_main.m** y debe quedar de la siguiente forma:



Este archivo, comos los dos siguientes, debemos guardarlos en nuestra carpeta **Ejemplo**. El código en este archivo es el primero que se ejecuta al iniciar nuestra aplicación. Y es prácticamente el mismo para cualquier aplicación.

3 Para activar el resaltado de sintaxis en Gedit, vamos a *Ver* -> *Modo resaltado* -> *Fuentes* -> *Objective-C*.

Necesitamos ahora un archivo al que llamaremos **EjemploInfo.plist**, que debe tener la siguiente forma:



Este archivo contiene información básica sobre nuestra aplicación que aparecerá en el panel de información: descripción de la aplicación, licencia, autor, ... (nuestro ejemplo por el momento no tiene un panel de información[4]). También contiene la información sobre el nombre del ejecutable, el archivo **gorm** que debe mostrarse al inicio de nuestra aplicación (la linea que dice NSMainNibFile). Así como otra información importante. Observese que todas las lineás llevan un punto y coma al final. Otros datos que pueden agregarse son los autores y la dirección de Internet:

```
Authors = (
 "German Arias"
);
URL = www.Ejemplo.com;
```

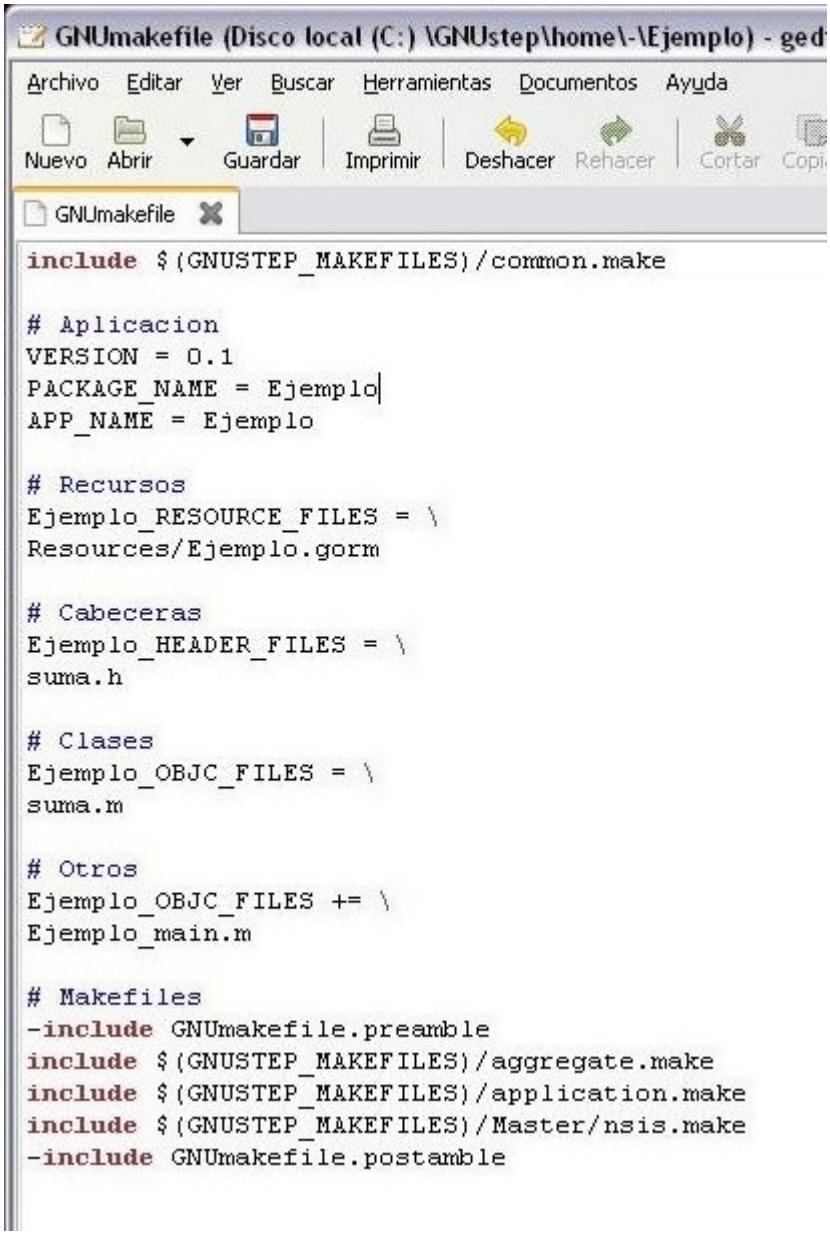
El autor va entre paréntesis, en este caso mi nombre esta entre comillas debido a que contiene un espacio entre mi nombre y mi apellido. Pero si no hay espacio las comillas no son necesarias. Si hay más de un autor, estos deben separarse por comas. Por ejemplo:

```
Authors = (
 "German Arias",
 Fulano
);
```

El punto y coma sólo va después del cierre de los paréntesis.

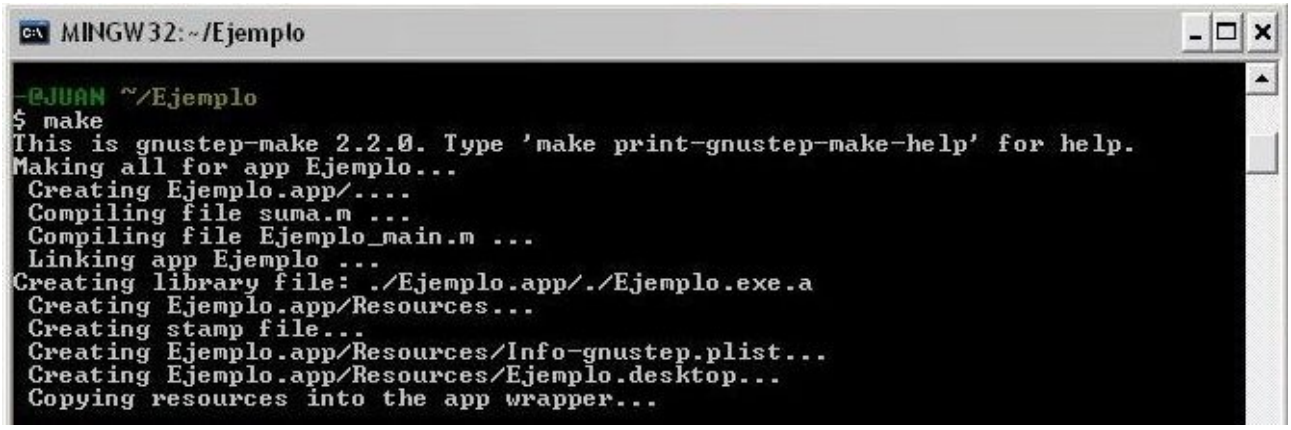
4 En Gorm puede agregarse un panel de información arrastrando un ítem *Info* de la paleta (en la sección *MenusPalette*) al menú de nuestra aplicación.

Finalmente necesitamos un archivo GNUmakefile que le diga a la herramienta GNUstep make como construir nuestra aplicación. Dicho archivo debe tener la siguiente estructura:



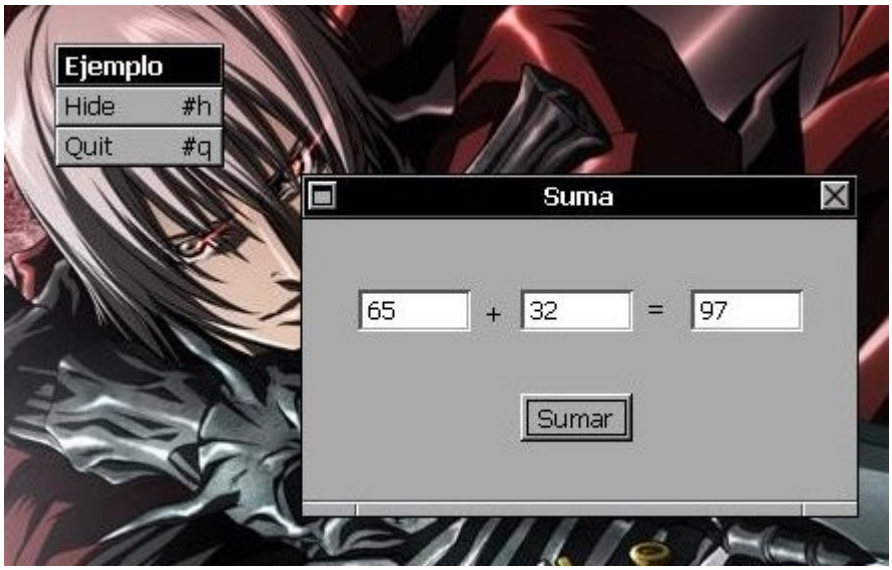
La estructura de este archivo se explica con más detalle en otros documentos.

Guardados estos archivos, regresamos al Shell para compilar nuestra aplicación. Estando en la carpeta **Ejemplo** ejecutamos el comando **make**, lo que inicia la compilación:



Si no hay ningún error todo se compilara correctamente. Y podremos probar nuestra aplicación con el comando **openapp** como se muestra:

Esto lanza nuestra aplicación (no nos preocupemos por los colores ahora, al empaquetar la aplicación se corrige esto):



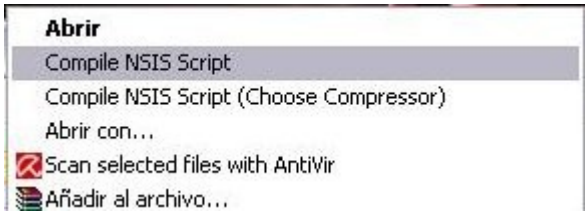
Seleccionando el primer componente **Text** ingresamos un valor, y con la tecla *Tab* nos movemos al siguiente componete para ingresar el segundo sumando. Nuevamente con *Tab* nos movemos al botón **Sumar** el cual activamos con la tecla espaciadora para llevar a acabo la operación de suma. La tecla *Tab* nos lleva nuevamente al primer componente **Text** para realizar otra operación.

Si todo funciona bien, cerremos nuestra aplicación (seleccionando *Quit* en el menú) y volvamos al Shell para escribir el comando **make nsis**. Lo que creara un script para empaquetar nuestra aplicación:

```
-@JUAN ~/Ejemplo
$ make nsis
This is gnustep-make 2.2.0. Type 'make print-gnustep-make-help' for help.
make[1]: Entering directory `/home/-/Ejemplo'
Making install for app Ejemplo...
 Creating /home/-/Ejemplo/obj/package//GNUstep/Local/Applications...
 Installing bundle directory...
 Creating /home/-/Ejemplo/obj/package//GNUstep/Local/Tools/....
make[1]: Leaving directory `/home/-/Ejemplo'
Generating the nsi script...
```

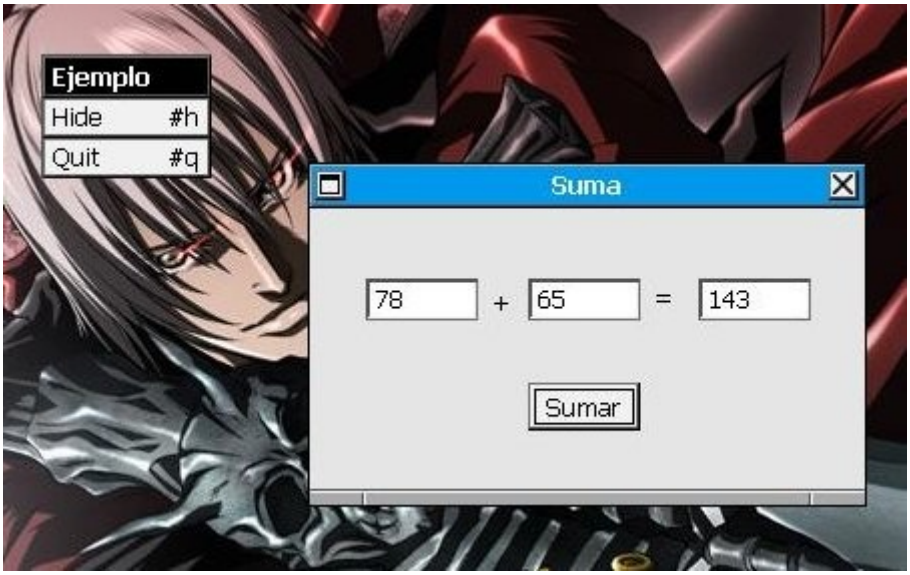
Terminado el proceso, vamos en nuestro navegador de archivos a nuestra carpeta **Ejemplo**. En la cual debe haber un archivo llamado **Ejemplo** de tipo **NSIS Script File**.

Con un clic derecho sobre dicho archivo, seleccionamos la opción **Compile NSIS Script**.



Finalizada la compilación encontraremos en la misma carpeta el instalador de nuestra aplicación.

Instalada nuestra aplicación, podemos iniciar esta a través del menú principal, *GNUstep -> Apps -> Ejemplo*.



Y esto es todo. Nuestra primera aplicación esta terminada.