Capítulo 3

Fecha Release: Agosto 13, 2006
Versión: 1.0
Ultima Actualización: Agosto 13, 2006

3. El diseñador de GUIs de Glade

En el capítulo dos tratamos el tema de Glade de forma somera. En este, nos adentraremos en el empleo de la herramienta Glade. Dedicamos un capítulo, debido a que es la utilidad que nos va a permitir diseñar de forma gráfica interfaces GUI que lucirán igual en Windows y en Linux.

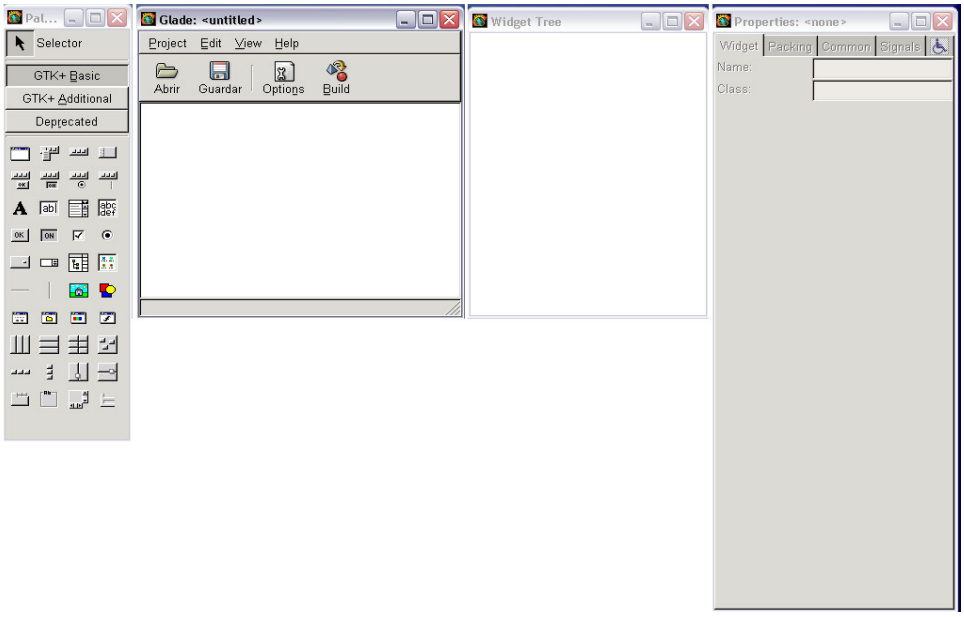
Glade es una herramienta Open Source que permite visualmente diseñar nuestra interfase de usuario utilizando el toolkit de Gtk, luego guarda su representación visual como un archivo XML que puede ser dinámicamente cargado y utilizado por una aplicación que utilice la librería adecuada.

Glade está disponible para el mundo Open Source en la plataforma Windows y Linux. La página de glade es http://glade.gnome.org. La dirección para conseguir la versión ("port") windows es http://gladewin32.sourceforge.net.

Las ventajas de utilizar un diseñador gráfico para nuestras GUI saltan a la vista. Una de las principales ventajas es la de poder definir mejor nuestra interfaz, lo cual es muy difícil de lograr a "puro código" o "a dedito" como lo llaman algunos. El contar con un lienzo (*canvas*) donde uno visualmente coloca y organiza objetos de forma visual, permiten que le demos nuestro toque (*look-and-feel*) similar en toda la aplicación.

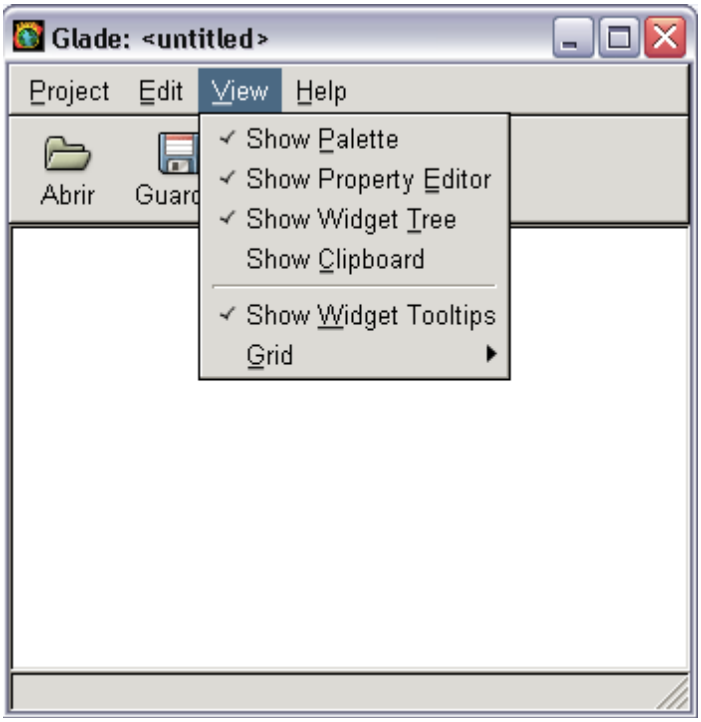
Para este capítulo, ejecutaremos la herramienta Glade por fuera del IDE de Microsoft .Net. Usted encontrará que en la instalación de mono y Gtk#, podremos cargar la utilidad. Carguemos la que aparece en el grupo de programas *Gtk# for Windows* opción *Applications*.

Cuando se carga, muestra la ventana de proyecto Glade sin widgets definidos.



Estas ventanas harán que pueda definir su interfaz de usuario de forma realmente sencilla. Para interactuar con Glade debemos hacerlo a través de su GUI, representada por sus tres ventanas principales (Proyecto o ventana principal, Paleta de Widgest y Propiedades). Estas tres ventanas interactúan unas con otras. En esta sección del documento, aprenderemos como interactuar con estas ventanas para construir nuestra GUI.

En la opción View del explorador de proyectos Glade, se puede seleccionar que ventanas del diseñador se muestran. Esto, por si desea desaparecer unas para que no interfieran con el nivel de detalle que queremos visualizar un diseño. Por ejemplo, suelo ocultar el árbol de widgets (Widget Tree). Lo utilizo solo cuando tengo formularios que incluyen muchos objetos dentro de contenedores y que necesito modificar uno en especial. Para ese caso en particular resulta muy útil el poder tomar control sobre el widget que necesitamos tomar de entre un grupo de controles (por ejemplo, barras de herramientas dentro cajas empaquetadoras.



Para este capítulo solo necesitamos el IDE de Visual Studio, el Glade 2.12.1 (o superior. Mono normalmente incluye una versión inferior a la disponible en el sitio oficial de Glade) y cinco cervezas. Destape la primera cerveza y demos un vistazo a los widgets GTK que se pueden programar en el Glade.

3.1. La ventana principal o proyecto.

Cuando iniciamos Glade, nos mostrará su ventana principal o visor de objetos del proyecto. Usted utilizará esta ventana para desarrollar su proyecto, la cual actuará como un contenedor de todos los ítems definidos para nuestra GUI.

En esta ventana podemos acceder a las opciones disponibles desde el propio menú de la ventana o desde su barra de herramientas. Las opciones de menú disponibles son Project (Proyecto), Edit (Edición), View (Ver) y Help (Ayuda??? solo incluye la opción acerca de...). Estas opciones pueden varias acorde a la versión de Glade que se ejecute.

Desde esta ventana podremos configurar al diseñador, guardar los cambios realizados al proyecto y construir en el archivo XML con las definiciones dadas en nuestro proyecto.

Una recomendación a considerar para no llevarse sustos, realice copias de seguridad periódicas de su proyecto para que pueda devolverse en versiones cuando tenga problemas con sus diseños. Una buena práctica es la ir guardando cualquier cambio que hagamos, debido a que esta versión de Glade no soporta la funcionalidad de deshacer. Si daña su diseño, debe salir de Glade sin guardar, volver a cargar su proyecto y continuar desde el último punto donde guardó. Pero eso son gajes del oficio, cuando vea los resultados de su aplicación corriendo en varias plataformas, le perdonara al Glade no poder cumplir esa funcionalidad.

3.2. La paleta de widgtes

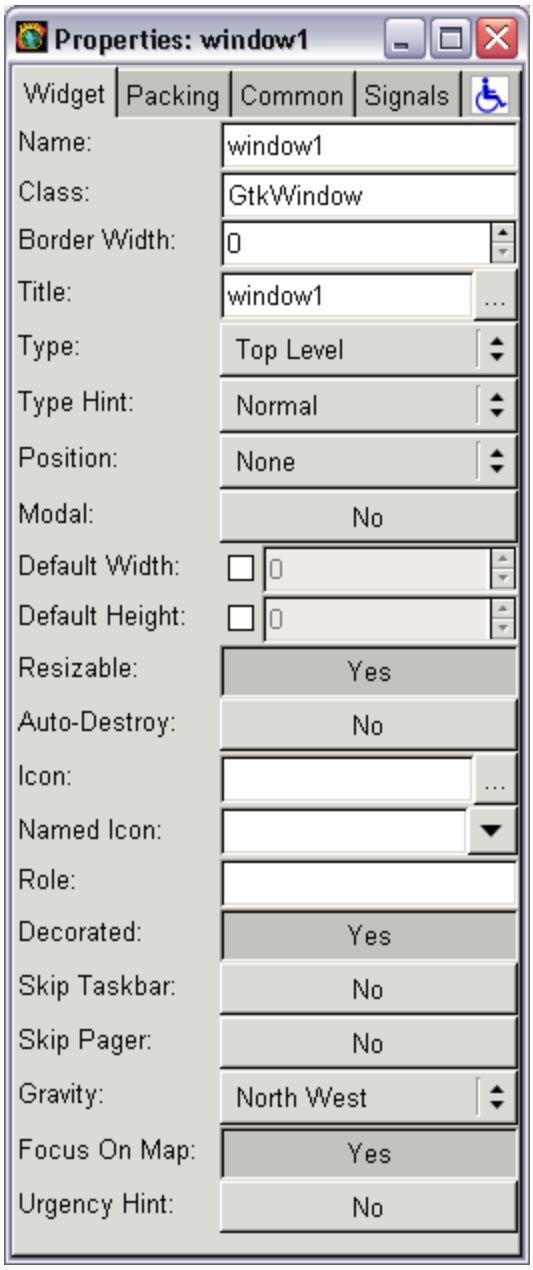
Esta paleta muestra los controles (llamados widgets en el mundo Gtk) disponibles para diseñar su interfaz de usuario. Usted puede ver el grupo de controles disponibles de forma agrupada. La paleta de widgets incluye tres secciones: GTK+ Basic, GTK+ Additional y Deprecated.

GTK+ Basic incluye los widgtes más comúnmente usados en el diseño de interfaces gráficas (por ejemplo ventanas, botones, etc. GTK+ Additional incluye widgets más avanzados y rara vez utilizados en interfaces gráficas (por ejemplo un control de calendario) y la sección Deprecated, incluye widgets que son candidatos a ser eliminados en futuras versiones debido a que han sido reemplazados o mejorados y que se mantienen por compatibilidad. Trate de no utilizar ningún widget en esta sección.

La buena noticia es que podemos definir nuestros propios controles y agregarlos a esta paleta. Para esta labor, estudie la documentación de Gtk en http://www.gtk.org/documentation.html y la documentación de la API de Glade en http://developer.gnome.org/doc/API/2.0/glib/index.html.

3.3. La ventana de propiedades (Properties)

Esta ventana permite ver y modificar una variedad de propiedades asociadas con el control u objeto que se tiene seleccionado. Incluye cinco categorías (tabs o pestañas) que permiten agrupar estas propiedades.



La categoría **Widget** contiene las propiedades asociadas al control para su representación visual y propias del tipo de objeto seleccionado.

La categoría **Packing** es para aquellos objetos que sirven como empaquetadores de otros, que tienen un arreglo de otros objetos o que permiten definir un orden.

El grupo **Common** incluye aquellas propiedades que son comunes a determinados controles.

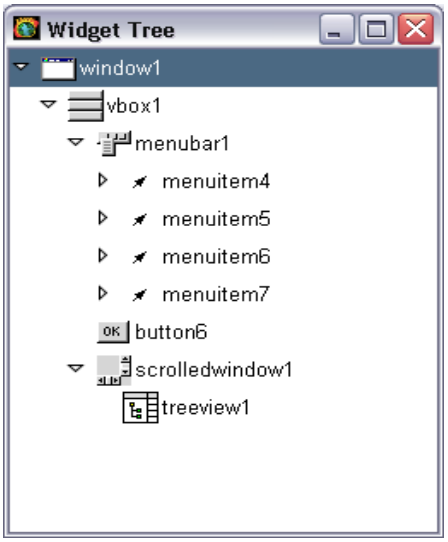
El grupo **Signals** permiten asociar capturadores de eventos con los eventos que es capaz de generar un control. Una vez defina un evento, podrá asociarlo a un método dentro de sus aplicaciones, el cual tomará control una vez que el evento suceda (fired).

La categoría Accessibility permite modificar los atributos para que nuestra aplicación sea más accesible a personas con discapacidades visuales (por ejemplo hacer zoom en algunos controles, aumentando al fuente de letras).

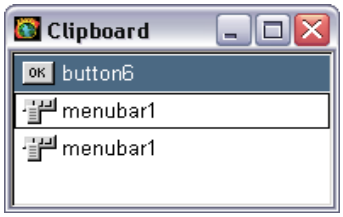
3.4. Otras ventanas

Existen otras ventanas disponibles en la opción Ver (View) de la ventana principal del proyecto Glade. Estas opciones se comportan como un interruptor, abriendo o cerrando estas ventanas. Estas ventanas son Widget tree windows (visor de widgets) y Clipboard window (Cortapapeles).

La ventana del árbol de widgets muestra una estructura jerárquica de los controles existentes en la ventana actualmente seleccionada.



La ventana del cortapapeles (Clipboard) hace un seguimiento de los controles que usted ha copiado a la memoria del cortapapeles. Se puede copiar más de un control de forma simultánea y en esta ventana se pueden observar todos esos controles. Se puede seleccionar un control a pegar en alguna ventana o contenedor, simplemente seleccionando el control desde esta ventana.



3.5. Los widgets de la Paleta

3.5.1. GTK+ Basic

La siguiente tabla muestra los widgets que están disponibles en la paleta grupo GTK+ Basic:

Widget	Descripción
 Window	Permite diseñar una ventana. Es el contenedor base de nuestra aplicación.
 Menu Bar	Para diseñar un sistema de Menú en la aplicación.
 Toolbar	Diseñar una barra de herramientas.
 Handle box	Contenedor de otros widgets que permite programar y capturar eventos que de forma nativa el widget no es capaz de generar. Por ejemplo, a un texto se le podría programar un evento de clic si lo asignamos dentro de este contenedor.
 Toolbar Button	Para asignar botones a la barra de herramientas.
 Toolbar Toggle Button	Para asignar botones de doble estado a una barra de herramientas.
 Toolbar Radio Button	Para asignar botones de radio a una barra de herramientas.
 Toolbar Separator Item	Separador de barras de herramientas
 Label	Asignar una etiqueta de texto
 Text Entry	Para diseñar campos de texto en los formularios.

Widget	Descripción
 ComboBox Entry	Diseñar listas desplegables que además permiten escribir texto que no aparezca en la lista.
 TextView	Diseñador de un visor de gran cantidad de texto
 Button	Colocar botones en el formulario
 Toggle Button	Botones de dos estados.
 Check Button	Botones de chequeo
 Radio Button	Botones de radio
 ComboBox	Listas desplegables
 Spin Button	Botón de incremento / decremento.
 List or Tree View	Para diseñar Vistas de Listas o Árboles. Muy útil, y quizás el widget con el más se pelean los desarrolladores de Gtk#.
 Icon View	Visor de iconos o imágenes.
Horizontal Separator	Línea separadora horizontal

Widget	Descripción
Vertical Separator	Línea separadora vertical
 Image	Para mostrar imágenes en un formulario
 Drawing Area	Un lienzo para dibujar en nuestros formularios.
 Dialog	Cajas de diálogo prediseñada para utilizar en un formulario.
 File Chooseer Dialog	Caja de diálogo para seleccionar un archivo desde el sistema de ficheros.
 Color Selection Dialog	Caja de diálogo selectora de colores.
 Font Selection Dialog	Caja de diálogo para seleccionar una fuente de letra.
 Horizontal Box	Empaquetador de otros widgets. Los distribuye de forma horizontal, uno detrás de otro.
 Vertical Box	Empaquetador de otros widgets. Los distribuye de forma vertical, uno debajo de otro.
 Table	Empaquetador de otros widgets en forma de tabla. Los distribuye mediante las celdas de la tabla.
 Fixed Position	Empaquetador de otros widgets de posicionamiento manual y fijo. Los widgets se quedan en el sitio en que se colocan y no se reorganizan de forma automática cuando se altera el tamaño del contenedor principal.

Widget	Descripción
Horizontal Button Box	Crea un control contenedor horizontal con el número de botones que especifiquemos.
 Vertical Button Box	Crea un control contenedor vertical con el número de botones que especifiquemos.
 Horizontal Panes	Crea un panel con una barra divisoria que se desplaza en forma horizontal. Esta actúa como un separador de objetos que en la interfaz gráfica el usuario puede desplazar con el mouse para alterar el tamaño de las dos divisiones.
 Vertical Panes	Crea un panel con una barra divisoria que se desplaza en forma vertical.
 Notebook	Crea un contenedor con varios tabs o pestañas. Muy útil para organizar GUIs que incluyen muchos controles y que no caben en un formulario o que o harían lucir demasiado cargado en la pantalla.
 Frame	Crea un control agrupador (frame) que permite separar de forma lógica los controles de nuestra interfaz Automáticamente define un texto que actuará como título del grupo.
 Scrolled Window	Crea un control contenedor con barras de desplazamiento para ofrecerle movilidad a otros controles. Por ejemplo, un widget gráfico (image) o una vista de árbol (tree view) se puede acomodar dentro de este control y automáticamente contarán con barras de desplazamiento horizontal y vertical cuando su contenido no alcance a ser visualizado en el área actualmente visible.
 Status Bar	Permite definir una barra de estado a nuestra ventana. En este se puede visualizar mensajes informáticos de estado de nuestra aplicación al usuario, con el objetivo de no estar cargando de forma continua mensajes de diálogo con botones de aceptar.

3.5.2. GTK+ Additional

La siguiente tabla muestra los widgets que están disponibles en la paleta grupo GTK+ Additional:

Widget	Descripción
 About Dialog	Crea de forma automática una caja de diálogo "Acerca de" con todos los componentes típicos en la ventana de proyectos.
 Input Dialog	Crea de forma automática una caja de diálogo para dispositivos de entrada.
 ToolBar Button with Menu	Define un botón de barra de herramientas con un menu asignado dentro de una barra de herramientas.
 ToolBar Item	Define un item en una barra de herramienta.
 Horizontal Scale	Define una barra de escala tipo termómetro horizontal para controlar a otros widgets.
 Vertical Scale	Define una barra de escala tipo termómetro vertical para controlar a otros widgets.
 Horizontal Ruler	Define una regla horizontal en la ventana
 Vertical Ruler	Define una regla vertical en la ventana
 Alignment	Define un widget con capacidad de manejar alineamiento en la presentación

Widget	Descripción
 Event Box	Define un contenedor capaz de generar y recibir eventos. Útil para asociarle eventos a controles widgtes que no los poseen de forma nativa.
 Calendar	Agregar un calendario.
 Progress Bar	Agregar una barra de progreso.
 Layout	Permite definir un contenedor con barras de desplazamiento con la capacidad de acomodar a otros widgets de forma fija y desplazándolos en forma horizontal y vertical hasta donde se hayan diseñado. Muy útil para colocar objetos en formularios donde no sabemos hasta donde se deban colocar nuestros controles. En realidad define de forma automática un Scroll Window y un Fixed Layout.
 Aspect frame	Agrega un frame o agrupador de otros widgets pero toma un aspecto diferente al frame del GTK+ Basic.
 Arrow	Define de forma automática puntas de flecha permitiendo definir el sentido de la punta mediante sus propiedades.
 Expander	Permite definir un widget con la capacidad de expandirse cuando se hace clic sobre el texto asociado. Muy útil para definir listas de texto que se van abriendo cuando el usuario hace clic sobre cada item de texto. Especial para hacer efectos de tablas dinámicas por ejemplo.
 Curve	Agregar capacidad de crear gráficos de curva a nuestra GUI. Para gráficas estadísticas.
 Gamma Curve	Agregar capacidad de crear gráficos de curva a nuestra GUI pero además facilita el cambiar el tipo de gráfica a mostrar (curva, línea recta, etc). Para gráficas estadísticas.
 Horizontal Scrollbar	Definir una barra de desplazamiento horizontal dentro de un contenedor
 Vertical Scrollbar	Definir una barra de desplazamiento vertical dentro de un contenedor

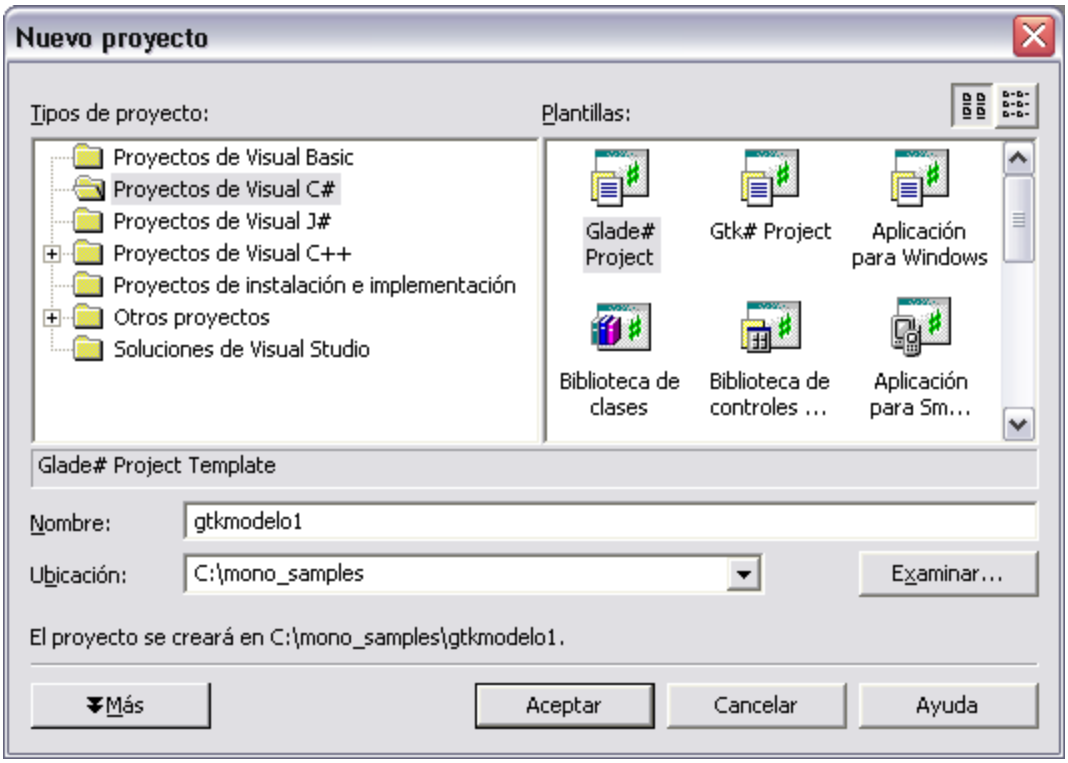
Widget	Descripción
 File Chooser	Define un Widget File Chooser (Selector de archivos) completamente funcional dentro de nuestra GUI.
 Color Selection	Define un Widget Selector de color completamente funcional dentro de nuestra GUI.
 Font Selection	Define un Widget selector de fuente de letras completamente funcional dentro de nuestra GUI.
 Cell View	Define un objeto Cell View (visor de celda).
 File Chooser Button	Crea un botón en la GUI que cuando el usuario hace clic sobre él, carga el selector de archivos. Sus propiedades permiten indicarle si abre archivos o carpetas de nuestro sistema de archivos.
 Color Chooser Button	Crea un botón en la GUI que cuando el usuario hace clic sobre él, carga el selector de colores.
 Font Chooser Button	Crea un botón en la GUI que cuando el usuario hace clic sobre él, carga el selector de fuentes.
 PopUp Menu	Define todo un sistema de menú Pop Up disponible en nuestro proyecto para que lo invoquemos y asignemos en la GUI que necesitemos de esa funcionalidad.
 View Port	Define un área con capacidad de mostrar a otro tipo de controles.
 Custom Widget	Define un área dentro de la GUI que permite cargar nuestros propios Widgets.

3.6. Creando nuestro proyecto Glade

En esta sección aprenderemos a crea un proyecto Glade. Para esta sección necesitamos cargar el IDE de Visual Studio. En primer lugar lo que haremos es definir un proyecto e ir incorporando los elementos necesarios para que nuestra interfaz sea funcional. Lo primero que haremos es diseñar la interfaz de forma básica (como luciría en la pantalla del usuario), luego la integraremos a nuestro código y por último agregaremos el código necesario para tener alguna interactividad con el usuario. Cargue el IDE del Visual Studio y traiga la segunda cerveza del refrigerador.

3.6.1. Creando el proyecto Glade

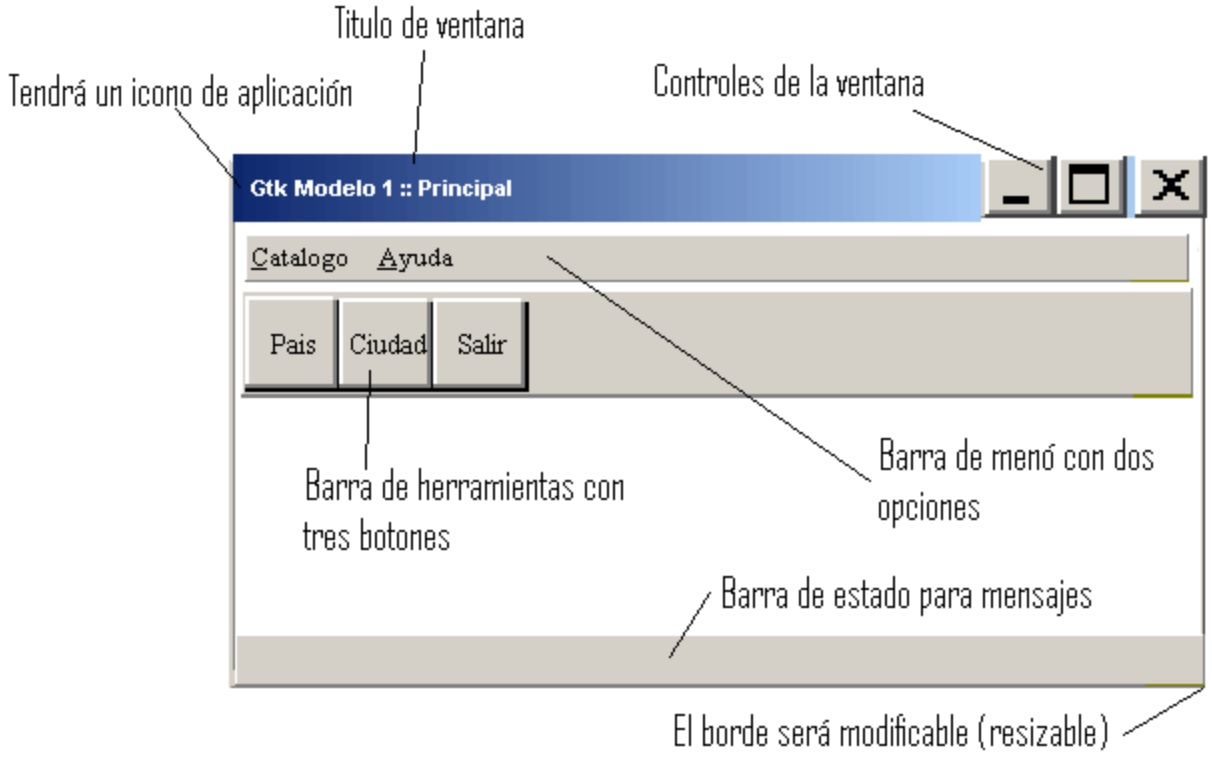
Haga clic en Archivo, Nuevo y Proyecto. Seleccione proyecto Glade# (Project Glade#) de las plantillas y especifique como nombre gtkmodelo1.



La plantilla incorporó todos los elementos básicos necesarios para que nuestra GUI funcione con el Gtk. El siguiente paso será diseñar nuestra GUI y ajustar el código que automáticamente generó la plantilla, esto, por que vamos a construir una solución muy diferente a la de la plantilla.

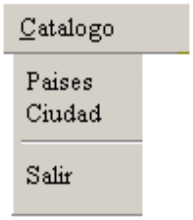
Por el momento dejaremos quieto al código y vamos a rediseñar la interfaz gráfica que la plantilla nos diseñó y lo ajustaremos de tal forma que incluya nuestro propio menú y nuestra propia barra de herramientas, pero reutilizando los controles que ya nos dejó hecho en Glade.

La GUI que vamos a diseñar tendrá la siguiente distribución:

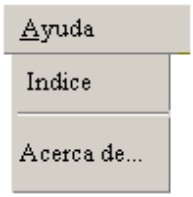


Claro que en Gtk no lucirá igual pero tendrá los mismos elementos de GUI que necesitamos.

El menú Catálogo tendrá tres opciones:



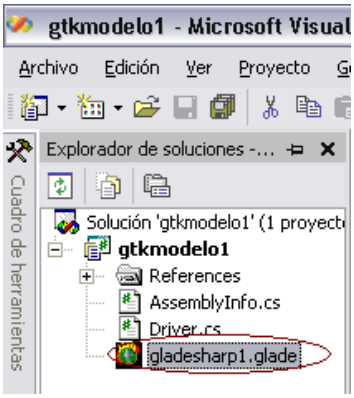
El menú Ayuda tendrá dos opciones:



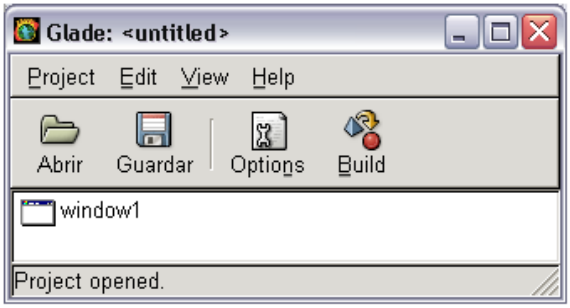
Aclarada esta parte, manos a la obra....

3.6.2. Creando la GUI principal

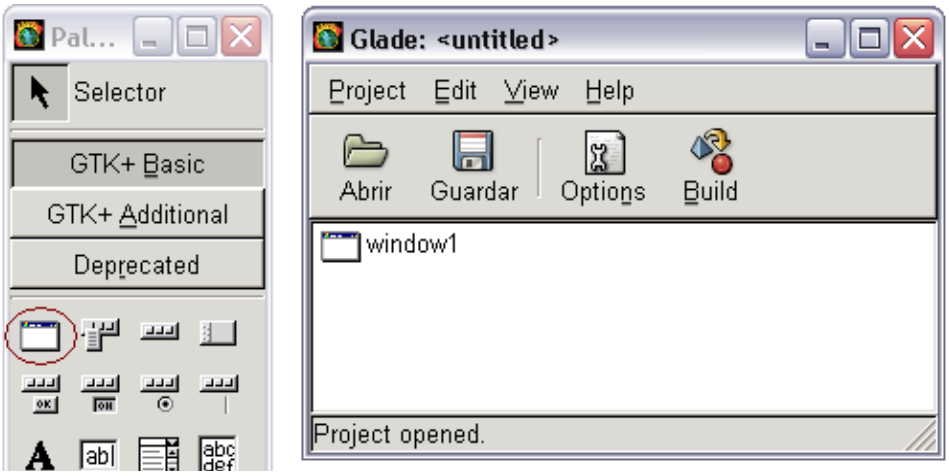
Haga doble clic en el explorador de proyecto sobre el objeto gladesharp1.glade.



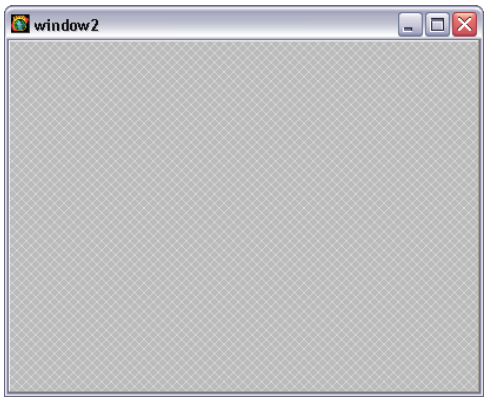
Ahora tenemos cargado al Glade. En la ventana de proyecto debe aparecer un objeto llamado window1 que es el objeto que la platilla nos dejó de forma automática. Dejemos quieto ese objeto y vamos a diseñar los nuestros.



Para construir nuestra interfase, seleccione el widget Window de la paleta. Debe abrirse automáticamente una nueva ventana donde empezaremos a diseñar nuestra GUI.



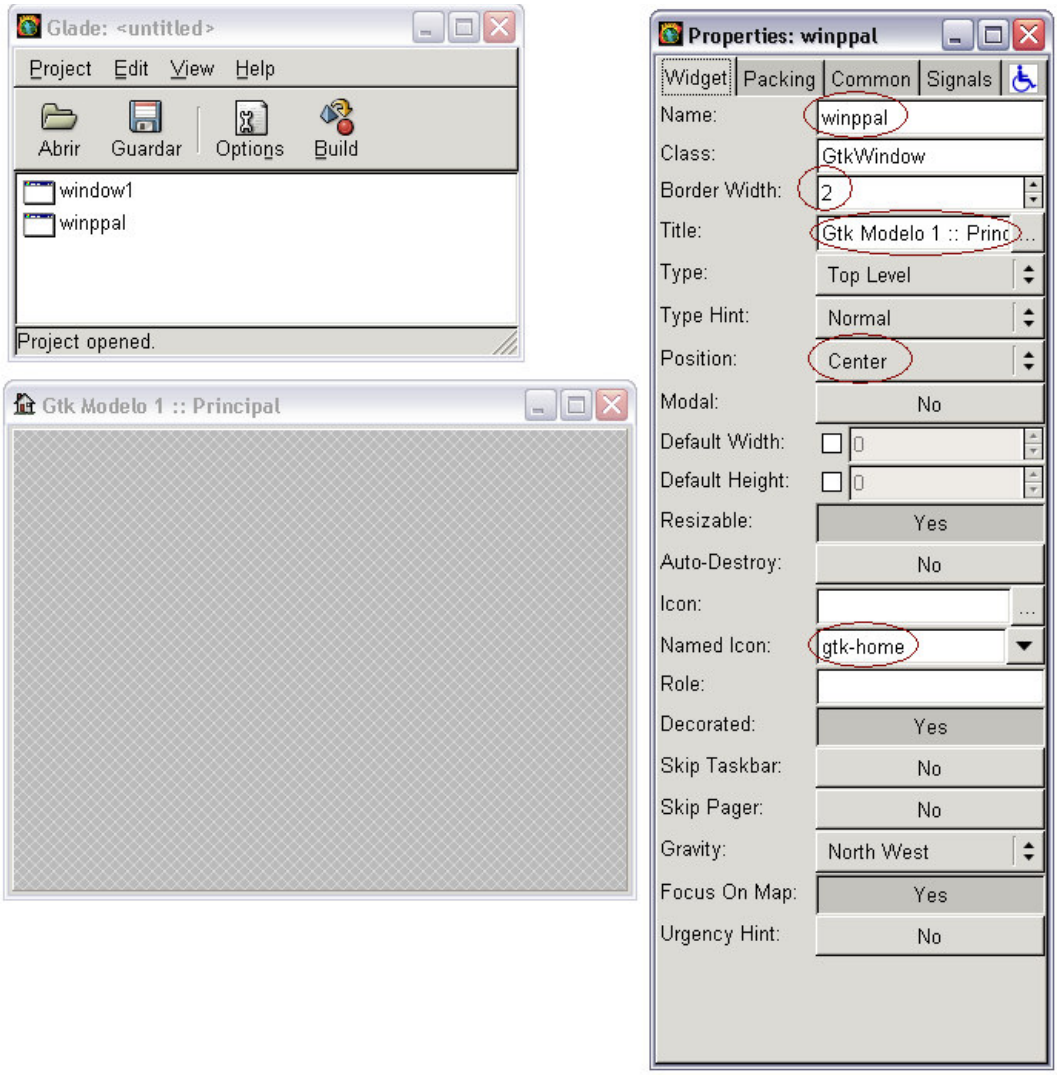
Ahora debemos tener una ventana vacía con el lienzo donde empezaremos a trabajar nuestra GUI.



Sin hacer clic en otra parte, en estos momentos en la ventana de propiedades debe estar mostrando las propiedades disponibles para nuestra ventana. En la figura se señala con un circulo cuales propiedades cambiaremos.

Modifique las siguientes propiedades según se muestra en esta tabla:

Propiedad	Valor	Descripción
Name (Nombre)	Winppal	Nombre del objeto. Este nombre sirve para identificar nuestra ventana desde el archivo XML Glade.
Border Width(ancho del borde)	2	Este atributo indica que a que distancia se colocarán los objetos que se integren a la ventana.
Title (título de la ventana)	Gtk Modelo 1 :: Principal	El título que tendrá la ventana.
Position (Posicion)	Center	La posición que tendrá la ventana cuando se muestre en pantalla. Con este parámetro la haremos centrar cuando arranque la aplicación.
Named Icon (Icon nombrado)	Gtk-home	El icono de la ventana. En este caso, llamaremos uno de los iconos que tiene en el inventario el Gtk. Revise la cantidad de iconos que podemos utilizar en nuestra GUI de forma automática.



Observe como de forma automática Glade nos va mostrando como está quedando nuestra GUI. Esto es algo realmente ventajoso por que no tenemos que esperar a ejecutar nuestra aplicación para ver los efectos de diseño logrado.

Ahora diseñaremos el contenido de nuestra GUI. Antes de colocar objetos en una ventana, debe señalar el widget de la paleta y hacer un clic sobre el punto exacto donde queremos colocarlo. Si no se puede alojar un objeto determinado sobre un contenedor, Glade lo anunciará con un mensaje en una caja de diálogo.

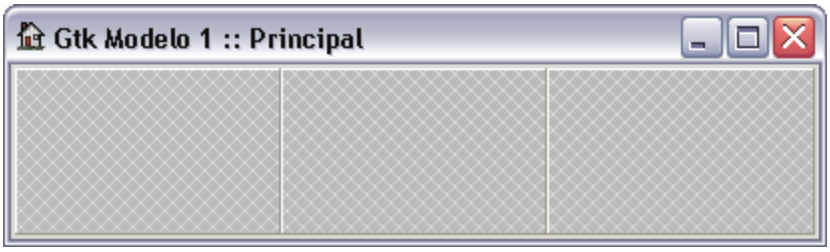
Antes de agregar elementos a una ventana se debe tener clara la distribución que haremos de la GUI. Ahí es cuando entran al escenario los widgets empaquetadores de otros. Estos nos permiten distribuir los demás objetos en la interfaz que estamos diseñando. Por eso es tan importante tener una imagen trazada de como queremos nuestra interfaz.

Estos son algunos de los contenedores organizadores que permiten agrupar a otros elementos y que comúnmente utilizaremos en el diseño de una GUI:

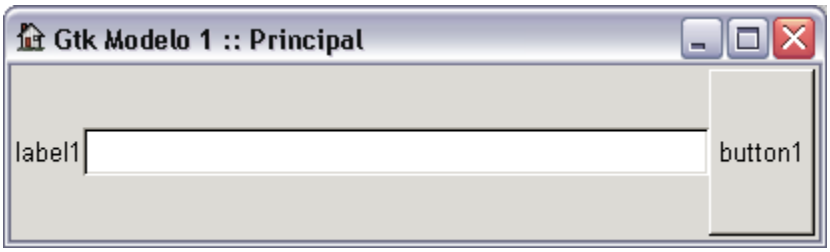
Horizontal Box

Define zonas para acomodar a los objetos. Los distribuye uno detrás del otro. Cuando se asigna a la ventana solicita la cantidad de columnas a distribuir. En cada columna debemos colocar un widget.

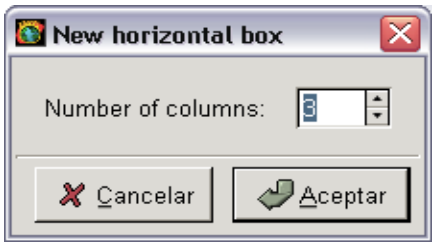
Esta es una muestra de la ventana con tres columnas definidas en el organizador:



Así queda cuando le agregamos tres widgets dentro de cada una de las columnas:



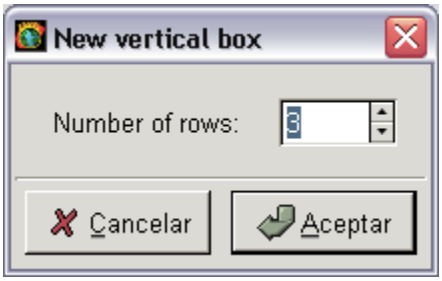
Una ventaja del diseñador es que nos solicita la cantidad de columnas que deseamos en el contenedor. Debemos saber de antemano cuantos objetos se van a alojar en este contenedor. Ahí es donde reside la importancia de tener un esbozo en alguna parte de nuestra interfase a diseñar.



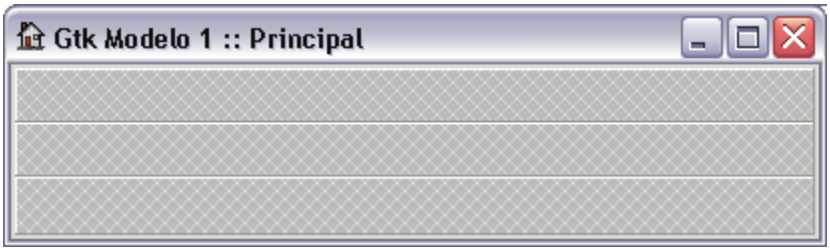
Claro que el diseñado permite que esta definición de columnas inicial sea alterada para aumentarlas o disminuirlas.

Vertical Box

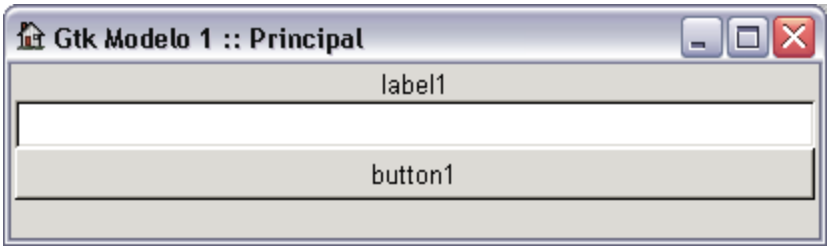
Define zonas para acomodar a los objetos. Los distribuye uno debajo del otro. Cuando se asigna a la ventana solicita la cantidad de líneas a distribuir. En cada línea debemos colocar un widget.



Esta es una muestra de la ventana con tres líneas definidas en el organizador:



Así queda cuando le agregamos tres widgets dentro de cada una de las columnas:

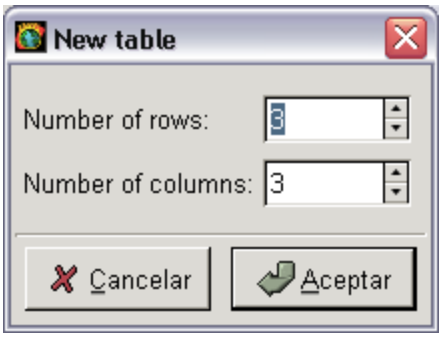


Los contenedores Vertical Box y Horizontal Box pueden contener a su vez a otros contenedores. Estos contenedores automáticamente "estiran" a los otros widgets para que ocupen el tamaño que se tenga definido dentro de la ventana que los contiene. Este tipo de comportamiento será útil dependiendo del tipo de ventana a diseñar. Se puede ver feo si el diseño y el tipo de ventana no es el adecuado para llevar uno de estos tipos.

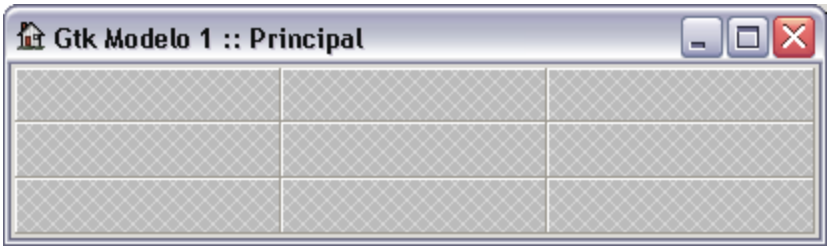
Se utilizan muchos estos dos contenedores para distribuir los objetos de una ventana.

Table

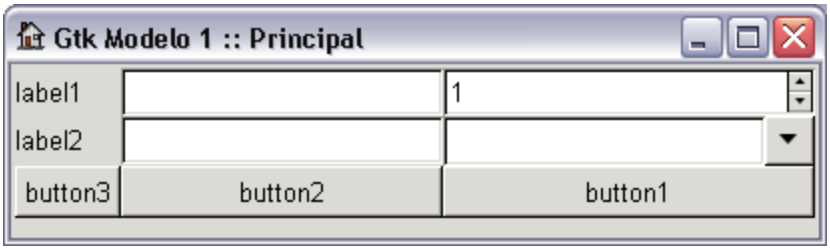
Define zonas para acomodar a los objetos en la forma de una tabla. Cuando se asigna a la ventana solicita la cantidad de columnas y líneas a distribuir. En cada celda debemos colocar un widget.



Esta es una muestra de la ventana con una tabla de 3 x 3 celdas definidas en el organizador:



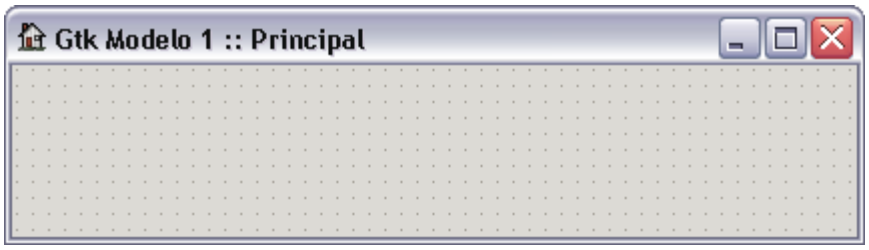
Así queda cuando le agregamos nueve widgets dentro de cada una de las columnas:



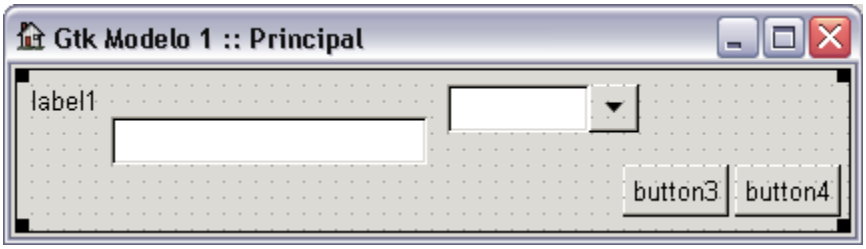
Fixed Position

Define zonas fijas para acomodar a los objetos. Los widgets que contenga se quedaran quietos cuando desplacemos los límites de la ventana. Muy útil para diseñar formularios que no necesitan que sus campos se desplacen típico de ventanas que no permiten alterar su tamaño.

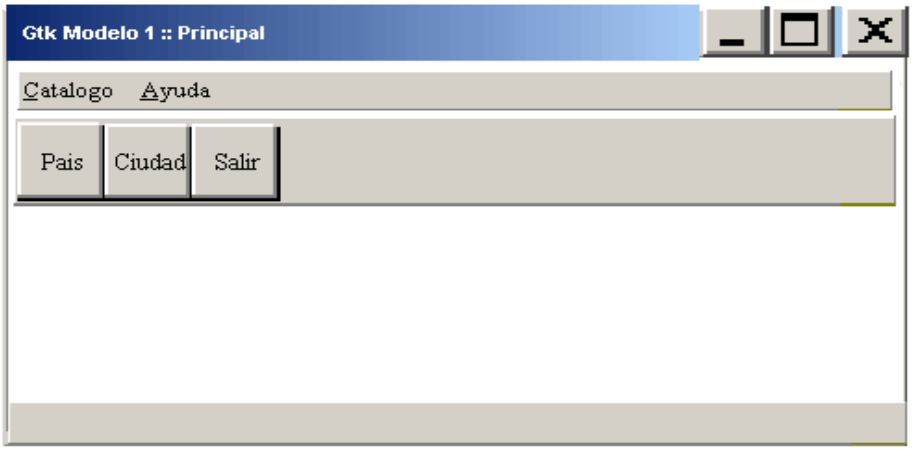
Cunado lo agregamos al formulario este aparece como un lienzo vacío, esperando a que empecemos a acomodar los widgets que contendrá.



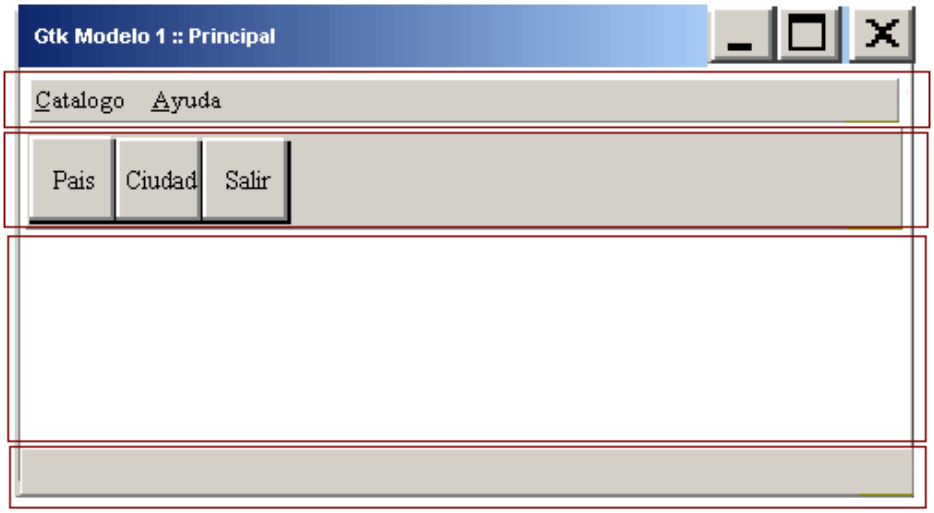
Cuando se agrega este tipo de contenedor, tenemos que posicionar cada objeto dentro del contenedor. Si desplazamos el borde la ventana veremos que estos se quedan fijos dentro del formulario.



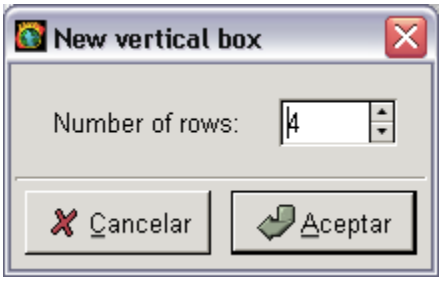
Ahora es tiempo de empezar a colocar los widgets que necesitamos para nuestra GUI de ventana principal de la aplicación. Revisemos que tipo de contenedor necesitamos para nuestra ventana. Habíamos diseñado el siguiente esquema para la interfaz:



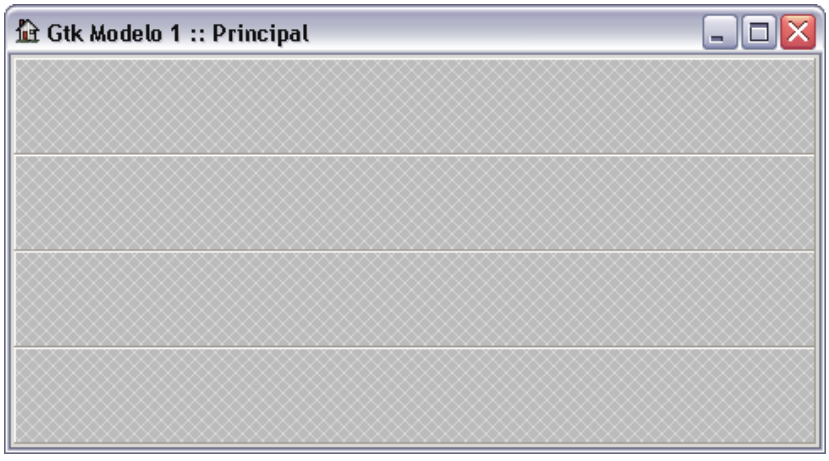
Revisando la GUI notamos que tiene cuatro áreas: menú, barra de herramientas, un área sin nada dentro y la barra de estado. Esto nos lleva a concluir que necesitamos un widget que nos acomode todo en cuatro líneas hacia abajo: Vertical Box.



Coloquemos entonces un vertical Box desde la paleta de Widgets y definamos 4 líneas.

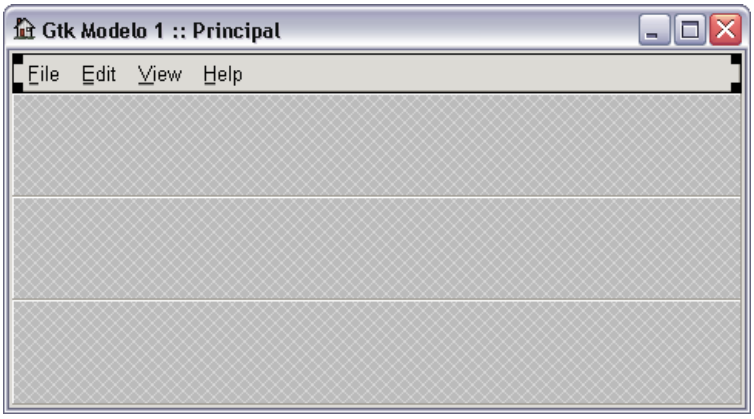


Ahora su formulario debe lucir así:

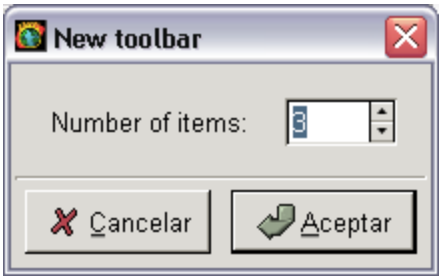


Ahora coloquemos un widget de barra de menú en la primer línea.

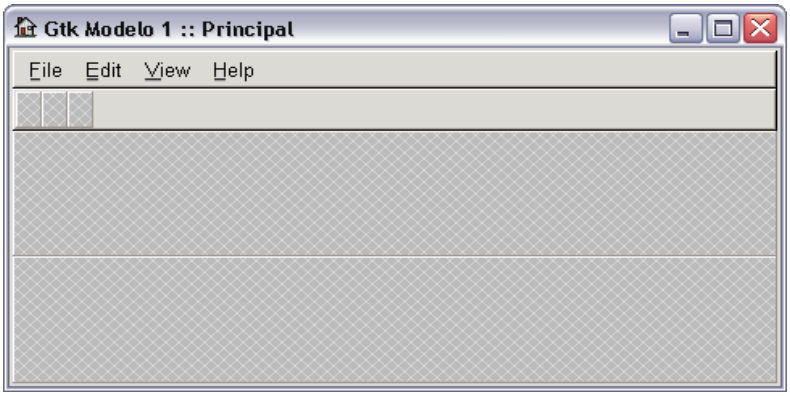
Debe lucir de la siguiente forma (el widget automáticamente nos define cuatro opciones, por el momento dejémoslo así, luego lo personalizaremos para que se acomode a nuestro diseño):



Vamos a diseñar nuestra barra de herramientas. Seleccione el widget ToolBar y haga clic sobre la segunda línea del organizador vertical; especifique como dimensión 3 botones.

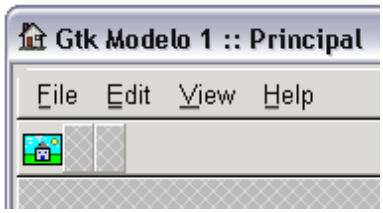


Ahora nuestro formulario debe lucir así:

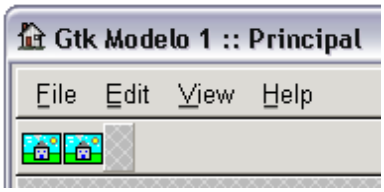


Agreguemos los tres botones a la barra de herramienta haciendo clic en la paleta en el ToolBar Button (botón de barra de herramientas) y señalando la primer posición reservada dentro de la barra de herramientas en el diseño.

Si hizo las cosas bien, su formulario debe lucir así:



Coloque el siguiente botón y debe tener su formulario luciendo de la siguiente manera:



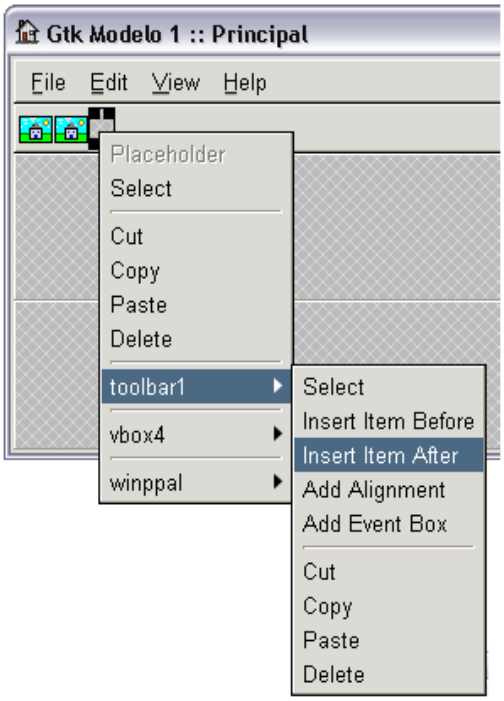
Antes de insertar el tercer botón a la barra, coloquemos un típico separador de botones para indicar que nuestro último botón trata de un tema muy diferente a los dos botones anteriores (estamos suponiendo). Como se le indicó a la barra de herramientas que tendría tres posiciones, debemos insertar una nueva posición para acomodar al separador.

El diseñador tiene la capacidad de eliminar, insertar, copiar y pegar objetos dentro de contenedores que son como una especie de arreglo.

Esta funcionalidad se activa al hacer clic derecho sobre el objeto contenedor. Para el caso de nuestra barra de herramientas, haremos clic derecho sobre la barra de herramientas exactamente sobre la última casilla vacía.

Glade debe mostrar un menú Pop Up con opciones que muestran los objetos contendores que tenemos ubicados en el diseño.

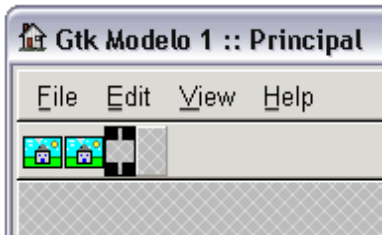
Seleccione el contenedor de toolbar1 de la lista y note que posee varias opciones para controlar las posiciones en la barra de herramientas.



Las opciones *Insert item Before* e *Insert item after* (antes y después) permiten insertar posiciones antes y después de la posición seleccionada. En nuestro caso le indicaremos que inserte una posición después (*Insert item After*).

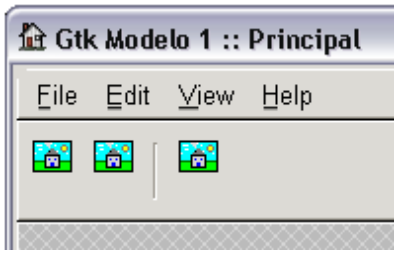
Note además que desde este menú flotante (Pop Up) podemos cortar, copiar, pegar y eliminar objetos dentro de la barra de herramientas.

Su formulario debe lucir de la siguiente manera luego de insertar una posición:



Ahora inserte un widget separador (*ToolBar Separator Item*) de barra de herramientas.

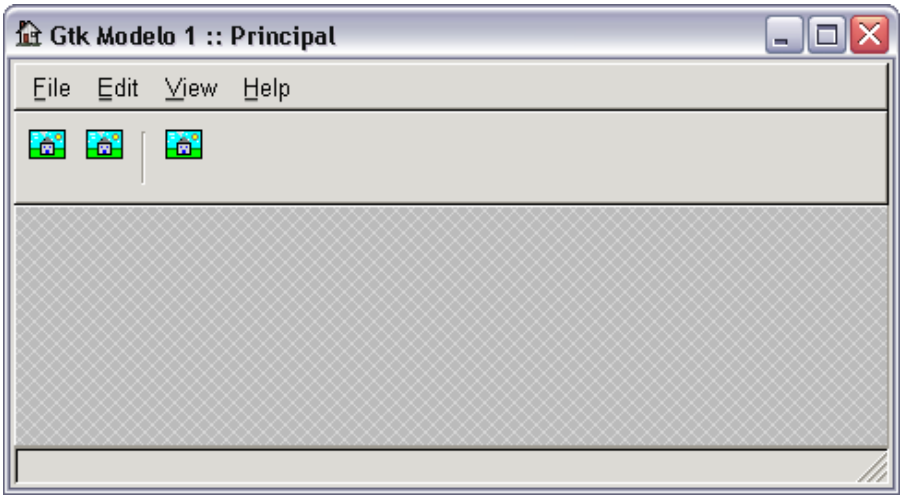
Por último, inserte un botón de barra de herramientas en la última posición disponible.



Ya tenemos lista en nuestra interfase la representación del menú y la barra de herramientas. Ahora solo nos falta agregar en la última línea del contenedor Vertical Box, la barra de estado. Pero antes de agregarla hagamos un cambio a la forma en que el contenedor de la barra deberá presentar

Seleccione de la paleta de widgets el control barra de estado (Status bar) y haga clic sobre la última línea del diseño.

Ahora, nuestra ventana tomó la forma que inicialmente teníamos el diseño.

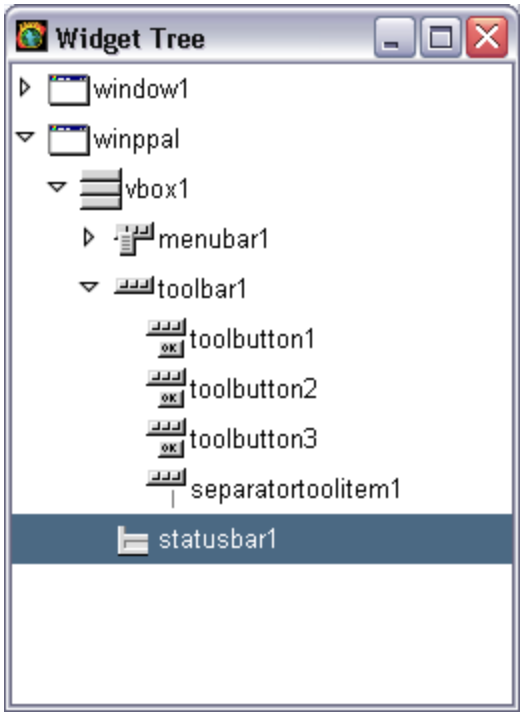


Fue una labor ardua la de construir nuestro primer diseño desde cero, pero no hemos terminado aún con la GUI que necesitamos. Antes de continuar debes ir al refrigerador y destapar la tercera cerveza.

3.6.2. Completando la GUI principal

Ya contamos con un diseño de objetos (widgets) que se acercan al diseño de la GUI que deseábamos. Vamos a terminar ese diseño colocando los elementos que darán el toque que esperamos del diseño.

Damos una revisada a los objetos que Glade tiene en el árbol de widgets. Seleccione *View* (ver) y luego *Show Widget Tree* (Mostrar el árbol de widgets).

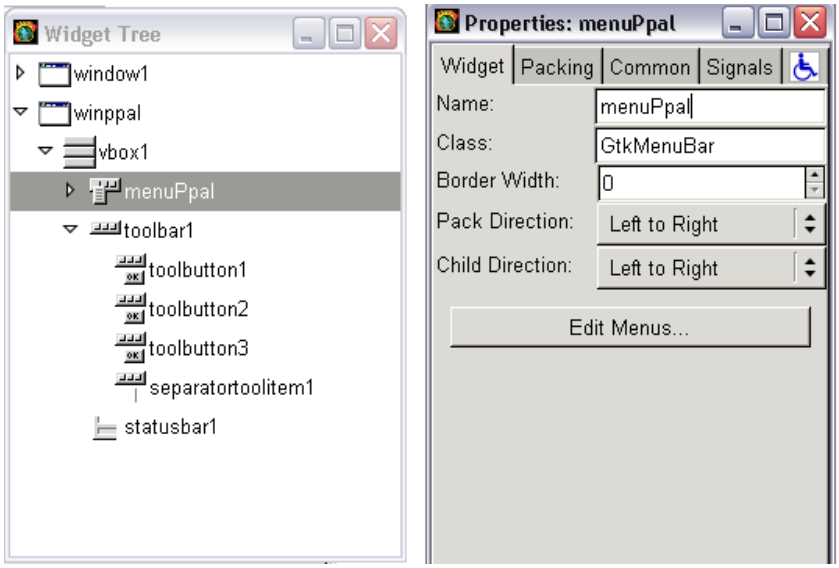


No importa que en estos momentos no se llamen igual al que les presento en la imagen (Glade empieza a numerar los objetos que vamos creando en cada instancia. Como posiblemente mientras de colocan los objetos podemos ir eliminándolos, entonces la numeración puede llegar a cambiar). Por eso, le daremos un nombre a cada uno de los objetos de la interfaz. Este árbol es muy útil para ejecutar esa tarea. Solo nombraremos aquellos objetos sobre los cuales necesitamos hacer referencia dentro de nuestro código.

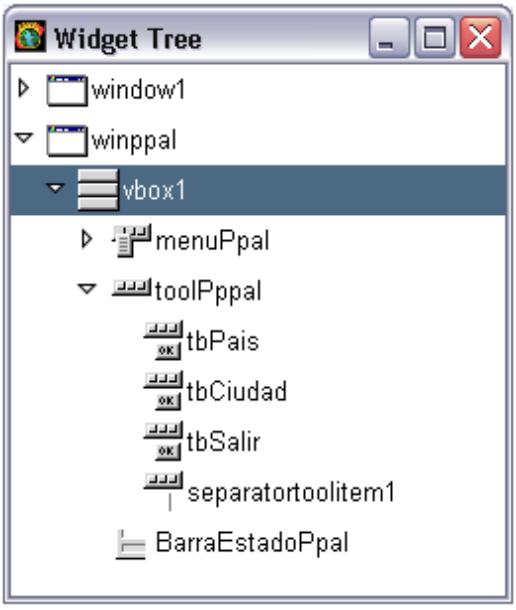
Para nombrarlos, seleccione el objeto (widget) dentro del árbol de widgets (Widget Tree) y de la ventana de propiedades, seleccione la propiedad nombre (Name). Escriba el nombre y repita la operación para cada objeto del árbol requerido.

Al objeto menú déle el nombre de menuPpal (menubar1), al objeto barra de herramientas (toolbar1) déle el nombre de toolPppal; al primer botón de la barra de herramientas (toolbutton1) déle el nombre de tbPais, al segundo (toolbutton2) déle el nombre de tbCiudad y al cuarto botón (toolbutton3) déle el nombre de tbSalir. A la barra de estado (statusbar1) déle el nombre de BarraEstadoPpal.

Con estos nombres, será posible referenciar los objetos de Glade dentro de nuestro código. Esto nos dará una mejor ubicación a la hora de querer invocar métodos y propiedades de los objetos diseñados en Glade.



Una vez se hayan definido todos los nombres, su ventana de widgets tree debe lucir así:



Ahora, personalicemos la barra de herramientas para que luzca como una verdadera barra de herramientas en nuestra GUI.

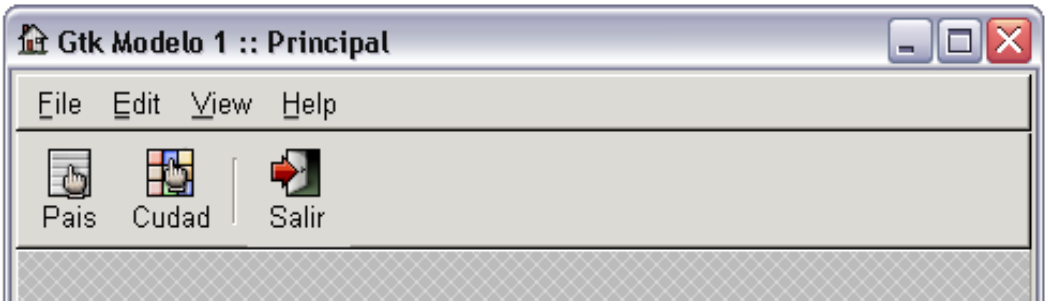
Para lograr esto, seleccione el primer botón de la barra de herramientas y seleccione en la ventana de propiedades, la propiedad Named Icon.

Como icono de este botón seleccione de la lista el llamado *gtk-index*. Claro está que puede definir sus propios iconos con extensión png desde el botón que aparece al lado de la lista de iconos en el inventario de Gtk en la propiedad icono, tomaremos prestado uno del inventario Gtk para que podamos mantener la misma presentación en la GUI. En la propiedad Label (Etiqueta) escriba *País*.

Ahora repita la misma operación con el segundo botón de la barra de herramientas *tbCiudad*. Especifique como Label el texto *Ciudad* y como icono, seleccione Gtk-select-color.

Por último repita la misma operación con el cuarto botón de la barra de herramientas *tbSalir*. Especifique como Label el texto Salir y como icono, seleccione Gtk-quit.

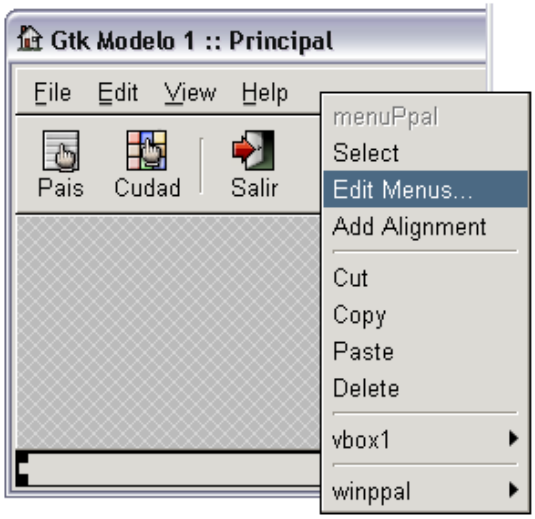
Ahora su barra de herramientas deberá lucir así:



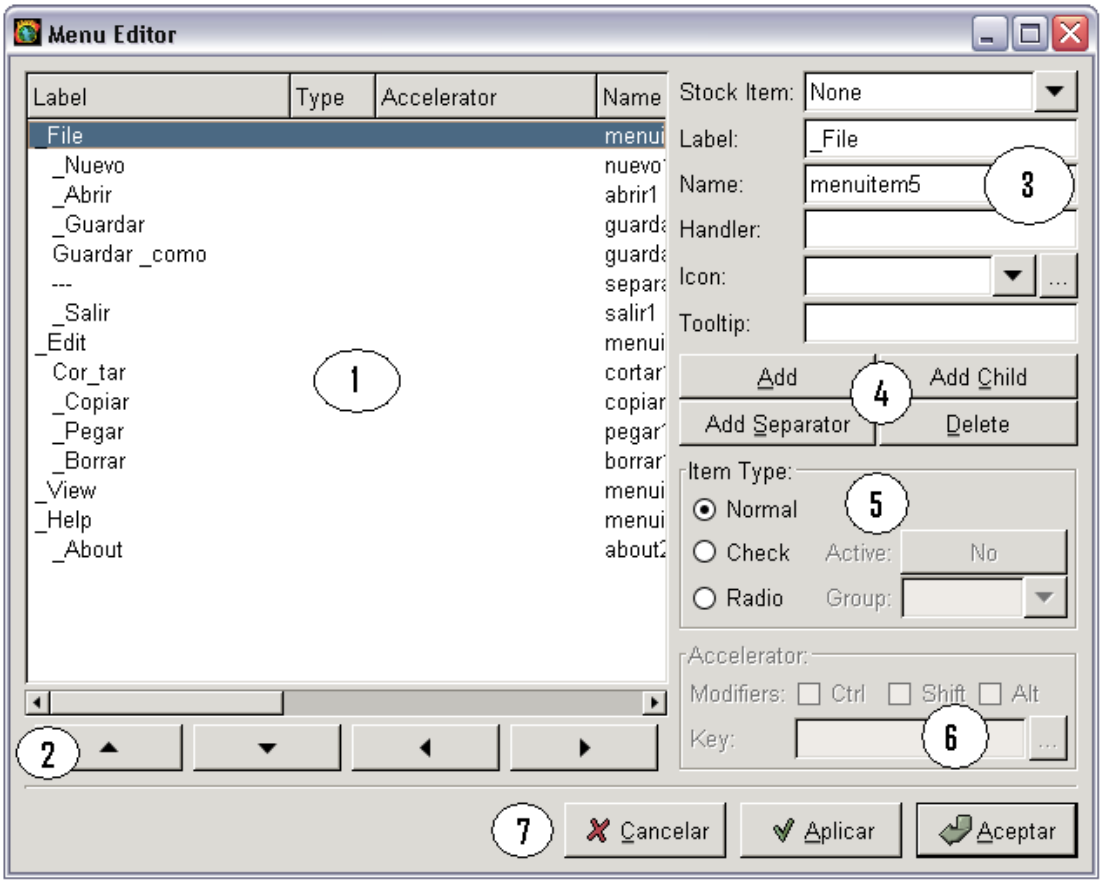
Ahora solo nos falta modificar el sistema de menú para que nuestro sistema luzca igual al planeado.

3.6.3. Completando el menú principal

Para modificar el menú haga clic derecho sobre el objeto de barra de menu (menuPpal) y del menú emergente (Pop Up) seleccione la opción Edit Menus (Editar).



Cuando carga el editor de menu nos muestra la siguiente caja de diálogo:



Se han colocado unos círculos denotando las áreas que componente al cuadro de diálogo Edito de Menu.

El área 1 es el área de opciones programadas en el menú de barra. Observe que se intentan las opciones que pertenecen a cada menú.

El área 2 permiten organizar las opciones del menú: moviéndolos hacia arriba, abajo, indentándolos hacia dentro y afuera.

El área 3 incluye las propiedades de la opción seleccionada con el mouse en el área 1.

El área 4 permite controlar las tareas de insertar, eliminar, agregar separadores entre las opciones del menú.

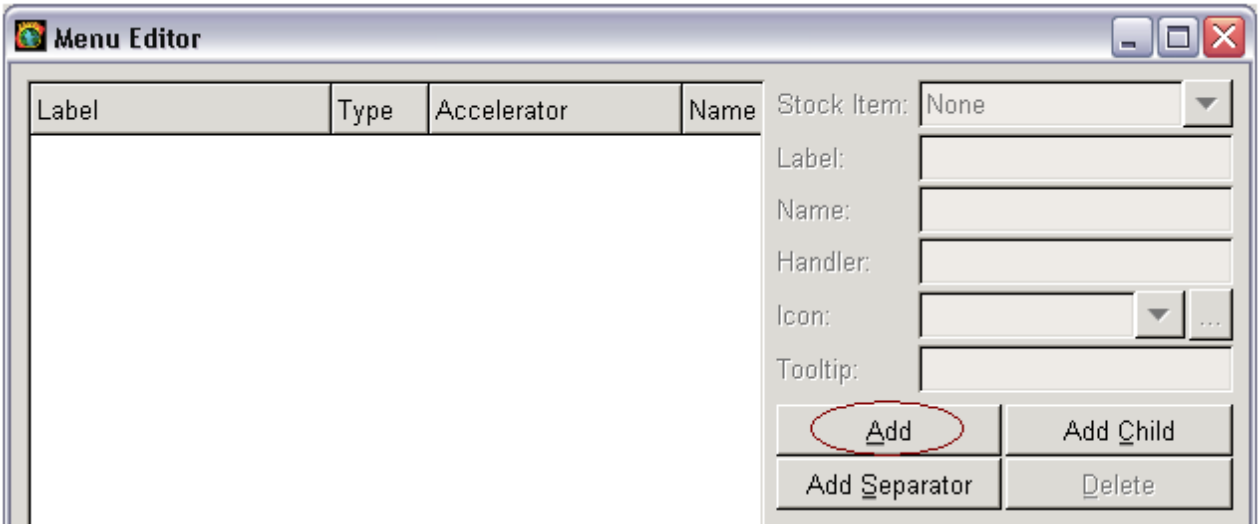
El área 5 permite programar el tipo de opción que deseamos aparezca en el menú.

El área 6 es para especificar que tipo de atajos (Hot keys o short cuts) deseamos en el menú.

Y el área 7 es para indicar si guardamos o no los cambios hechos en el editor de menú.

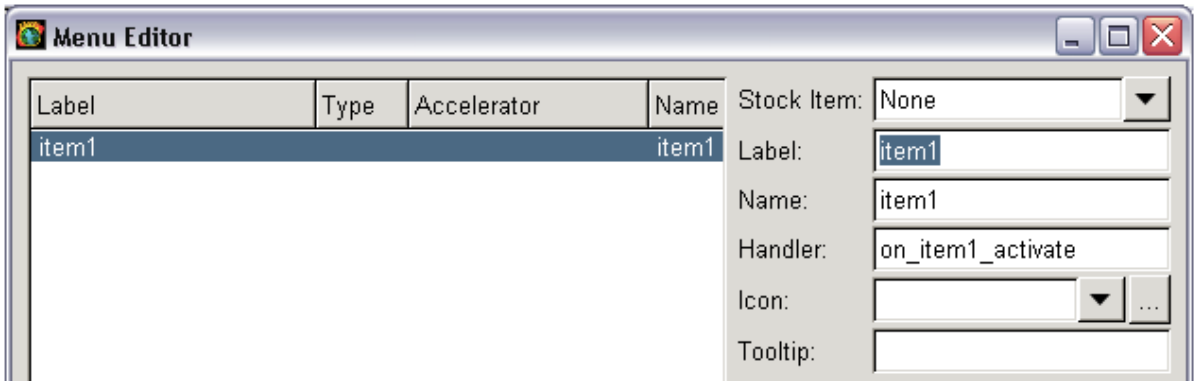
Vamos a personalizar nuestro sistema de menú según lo tenemos planeado.

Haga clic el item del menú (File) y pulse la tecla eliminar (delete o suprimir) hasta que no quede ninguna opción en el editor de menú.



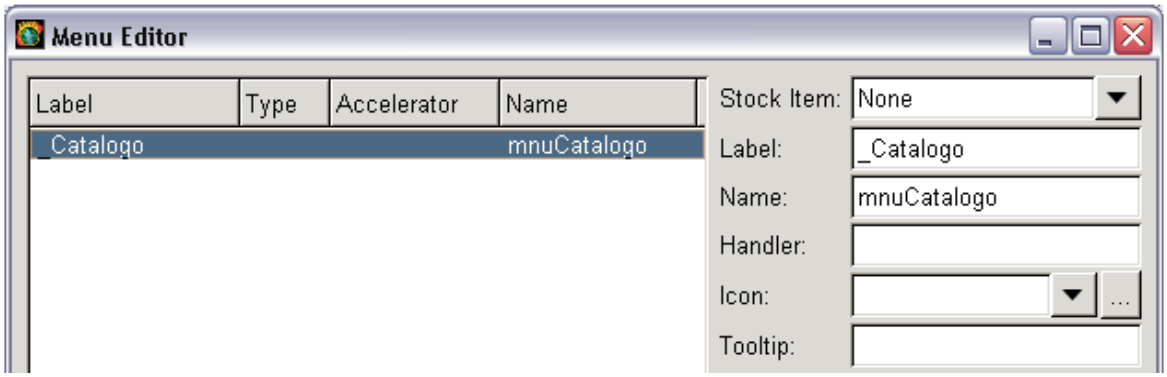
Ahora diseñemos las dos opciones del menú: Catalogo y Ayuda.

Haga clic en el botón Add (Agregar) y ajuste las siguientes propiedades para este nuevo item: como Label especifique _Catalogo, Name (nombre) escriba mnuCatalogo, en la propiedad Handler (rutina capturadora del evento clic) elimine lo que escribió por defecto el editor.



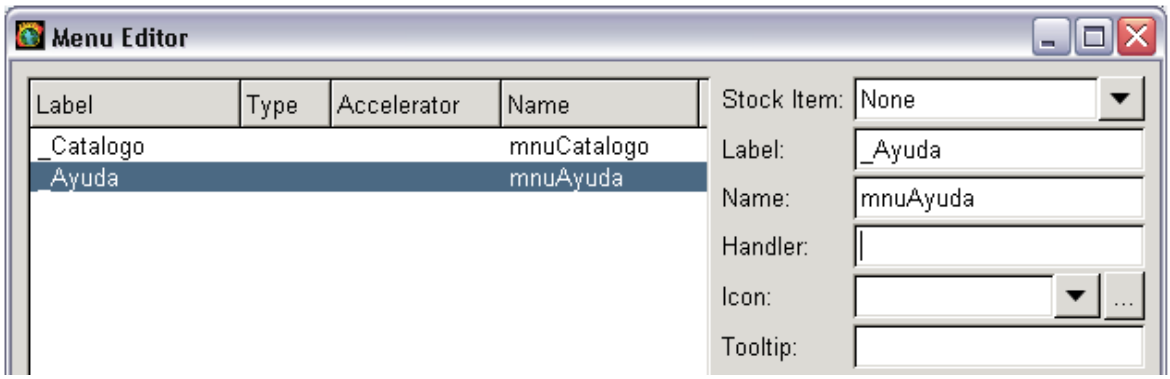
Observe que mientras escribe el Label del item, el editor va definiendo de forma automática el nombre del item y la rutina que va a capturar el evento de clic en la opción (Handler). En este caso no programamos ninguna rutina por tratarse de una opción que simplemente debe desplegar a un submenú.

Si hizo las cosas bien, entonces su editor debe lucir de la siguiente forma:

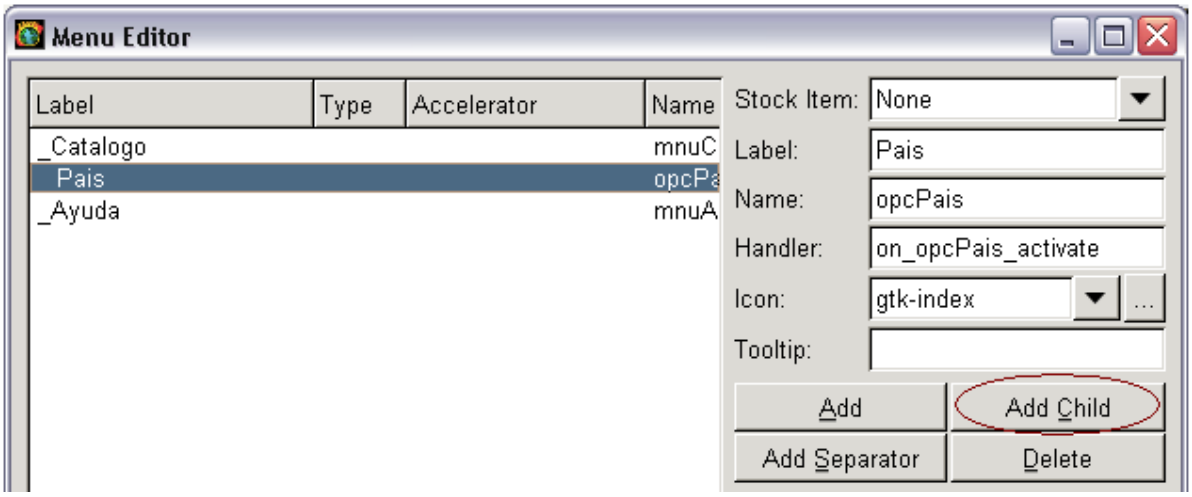


Ahora definamos la segunda opción del menú: Ayuda.

Pulse nuevamente el botón Add (agregar) y especifique los siguientes atributos: como Label escriba _Ayuda, nombre (Name) escriba mnuAyuda y como rutina capturadora del evento clic elimine lo que propuso el editor.

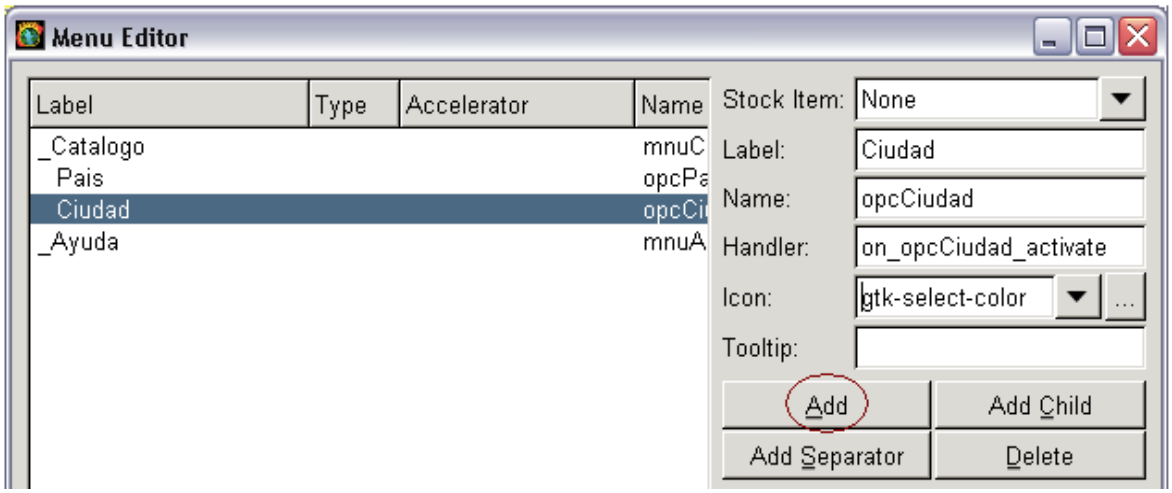


Ahora nos falta programar las subopciones en cada menú. Empecemos por la opción Catalogo. Haga clic en la lista de opciones sobre Catalogo y pulse el botón *Add Child* (Agregar hijo). Suministre como propiedades lo siguiente: *Label* (etiqueta) el texto País, en *Name* (nombre) especifique *opcPais*, como handler especifique *on_opcPais_activate* y como icono seleccione *gtk-index*.



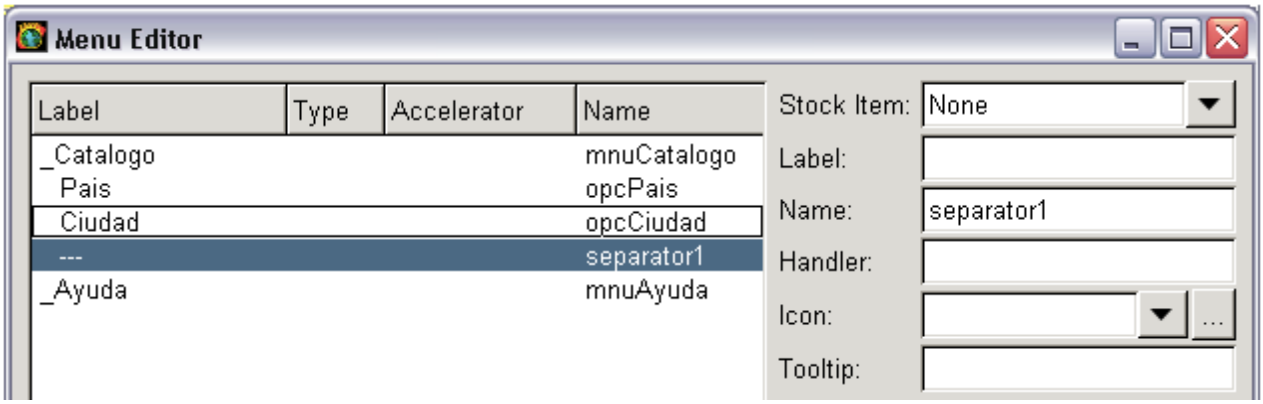
Se ha especificado un handler debido a que esta opción debe ejecutar algo una vez el usuario la seleccione con el mouse. Esa rutina la debemos definir dentro de nuestro código. Luego veremos como la definimos. Note además que a esta opción se le agregó el mismo icono de su equivalente en la barra de herramienta. Recuerde que puede especificar su propio icono que sea más consecuente con lo que dice el texto.

Ahora, definamos la opción Ciudad. Para esto, ubique el cursor sobre la opción *País* y haga clic en el botón *Add* (*Agregar*). Este botón realmente, agrega ítems bajo el actualmente seleccionado, con atributos similares al seleccionado. En este caso, creará una sub opción del menú *Catalogo*. Especifique las siguientes propiedades: *Label* igual a *Ciudad*, name igual a *opcCiudad*, handler igual a on_*opcCiudad_activate* y como icono seleccione *gtk-select-color*.



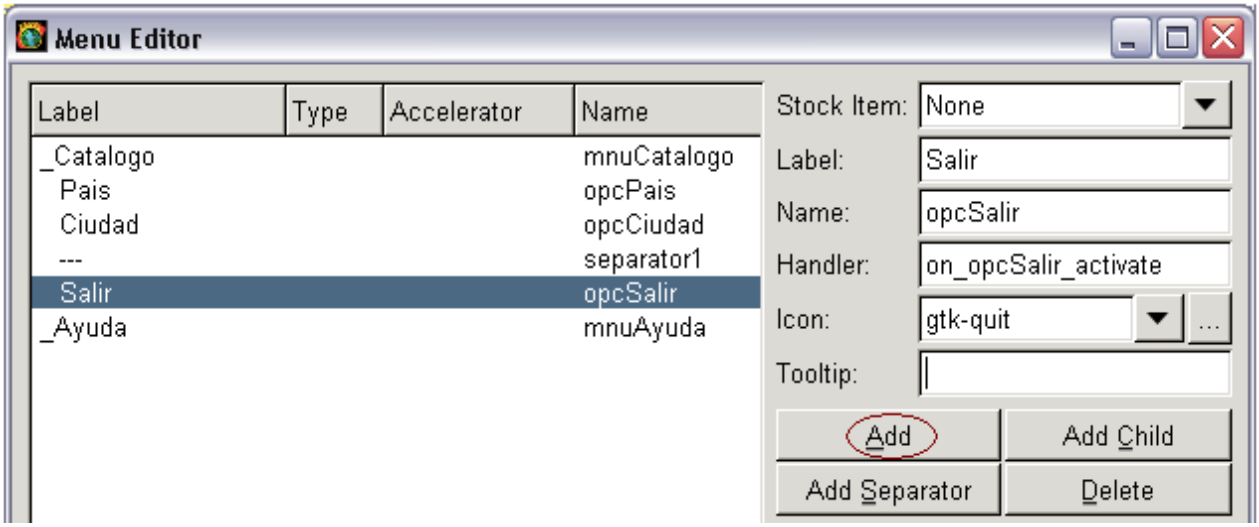
Note que si en el momento de escribir el Label escribe el carácter de subrayado en cualquier parte del texto, se estará definiendo un atajo de forma automática a esa tecla.

Ahora definamos el separador del submenú catálogo. Seleccione la opción *Ciudad* y haga clic *Add Separator* (Agregar Separador). De forma automática se inserta el separador debajo de la opción *Ciudad*. Para un separador no necesitamos definir ningún atributo.

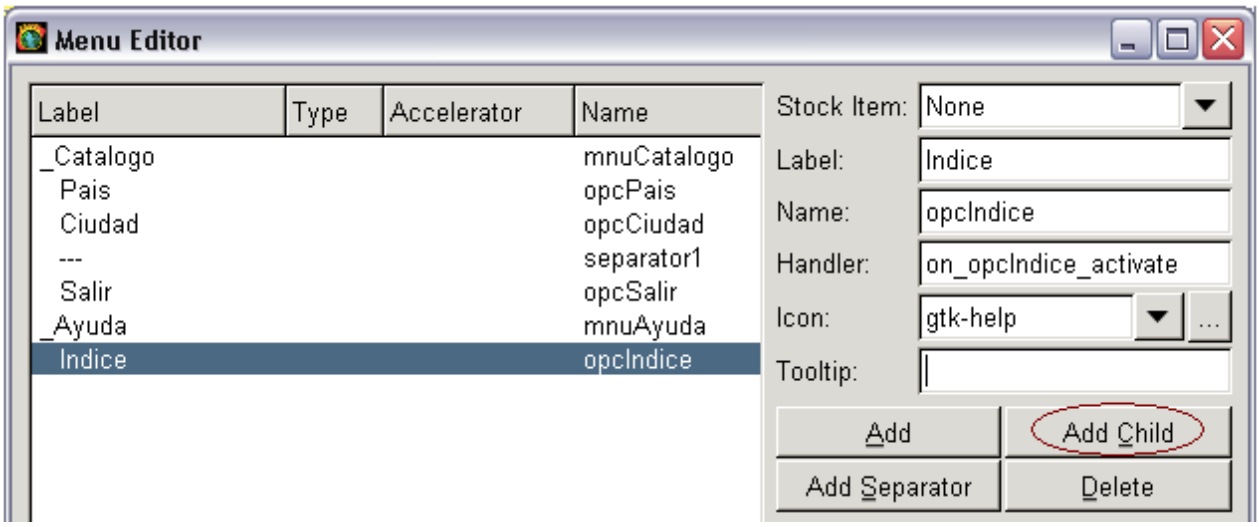


Ahora definamos la última opción de este submenú: Salir.

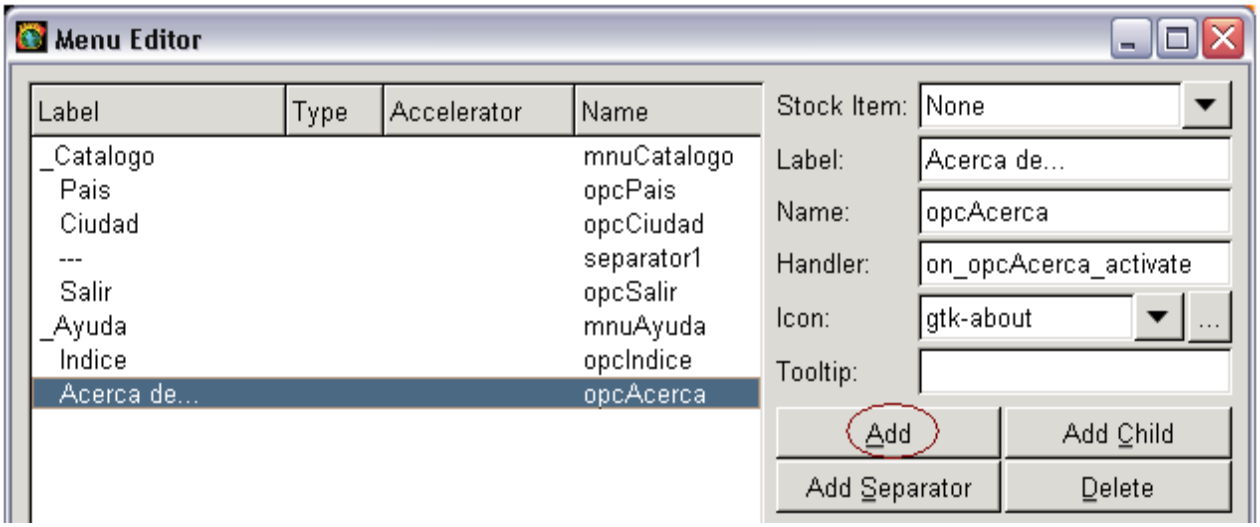
Seleccione el separador recién agregado y haga clic en el botón *Add*. Como atributos de la nueva opción especifique lo siguiente: *Label* igual a *Salir*, Name igual a *opcSalir*, *Handler* igual a *on_opcSalir_activate* y como icono seleccione *gtk-quit*.



Solo nos falta definir el submenú de la opción Ayuda del menú principal. Para hacer esto, haga clic en la lista de opciones sobre la opción Ayuda; luego haga clic en el botón Add Child (Agregar hijo) y especifique los siguientes atributos: en *Label* escriba *Índice*, en *Name* escriba *opcIndice*, en *Handler* especifique *on_opcIndice_activate* y como icono seleccione de la lista *gtk-help*.

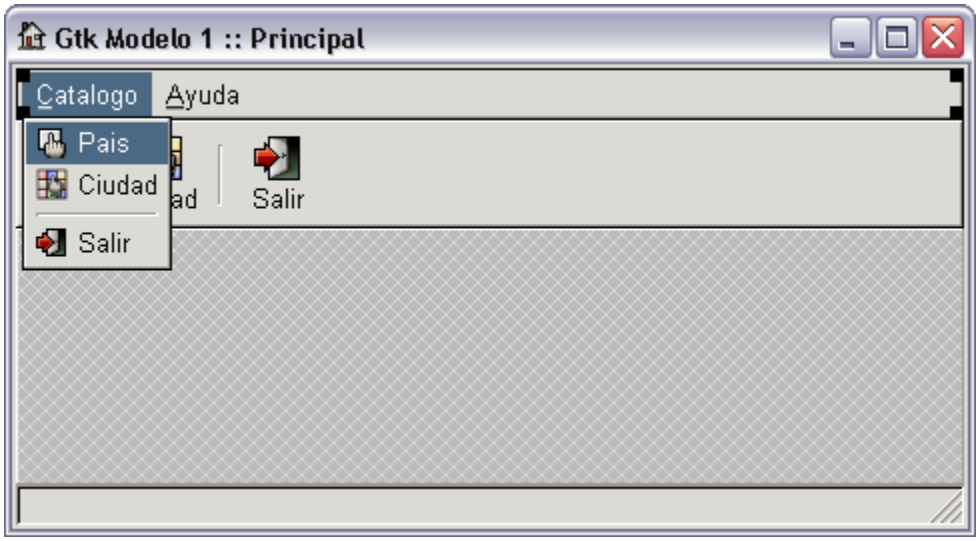


Solo nos falta definir la última opción del submenu Ayuda. Para definirla, seleccione de la lista la opción Índice y haga clic en el botón Add para agregar un item similar en el submenú. Especifique los siguientes atributos: como *Label* escriba *Acerca de...,* como *Name* especifique *opcAcerca*, como *Handler* escriba on_opcAcerca_activate y como icono seleccione *gtk-about*.



Ahora ya tenemos el sistema de menú definido en la interfaz GUI tal cual lo habíamos planeado. Ahora falta darle los últimos toques a la interfaz antes de volver al IDE de Visual Studio y continuar programando la interfaz. Para salir del Editor de Menú haga clic en el botón [*Aceptar*].

Ahora su ventana debe lucir de la siguiente forma:



Note que su sistema de menú es completamente navegable en el diseñador de Glade, lo cual es una ventaja por que nos permite revisar como lucirá cuando ejecutemos nuestra aplicación.

3.6.5. Dando los últimos toques a nuestra GUI

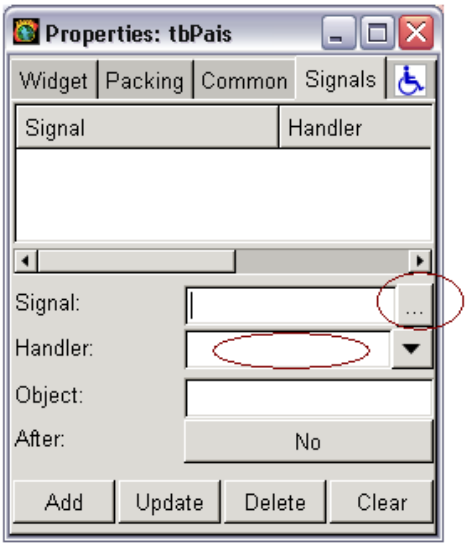
Cuando se definieron las opciones del menú, decidimos incluir una barra de herramientas que incluye tres botones, ejecutando las mismas operaciones existentes en el submenú Catálogo. Solo nos falta programar capturadores de eventos iguales a los empleados en las opciones del submenú.

Las rutinas definidas en las subopciones son las siguientes:

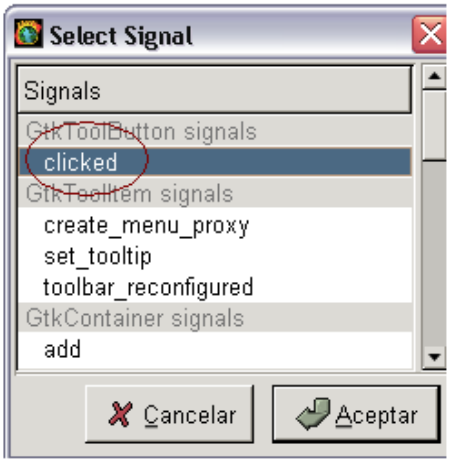
Sub Opción en Catalogo	*Nombre rutina*	*Barra de herramientas similar*
País (opcPais)	on_opcPais_activate	BtPais
Ciudad (opcCiudad)	on_opcCiudad_activate	BtCiudad
Salir (opcSalir)	on_opcSalir_activate	BtSalir

Vamos entonces a finalizar nuestro diseño, invocando estos mismos capturadores de eventos en los botones de la barra de herramienta.

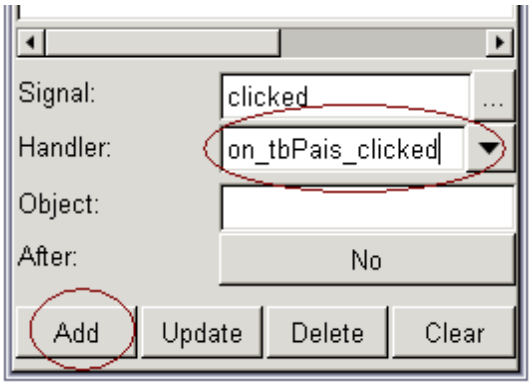
Para hacer esto, seleccione la primer opción en la barra de herramientas (btPais) y en la ventana de propiedades, seleccione la pestaña Signals (señales o eventos) ejecutemos la siguiente operación. En la parte inferior de la pestaña Signals aparece un atributo llamado Signal y otro denominado Handler. En estos dos atributos es que haremos la programación del evento clic en la barra de herramientas.



Haga clic en la caja de selección de Signal y seleccione de la lista que aparece, el evento click.

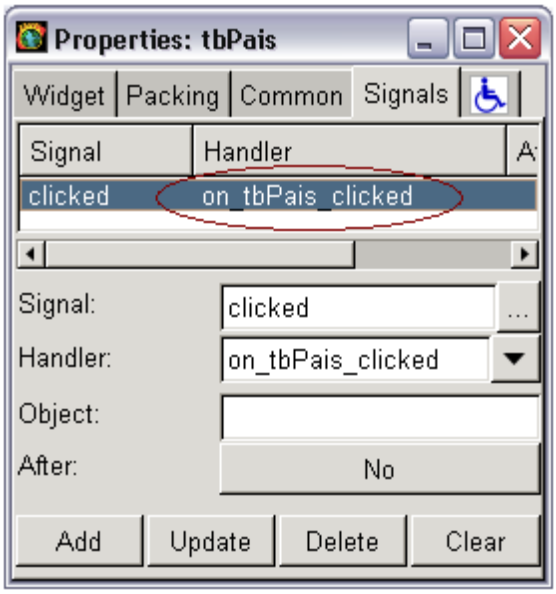


Cuando haga clic en el botón Aceptar, volverá a la ventana de propiedades y observará que Glade ha llenado de forma automática un nombre de Handler llamado *on_tbPais_clicked*.

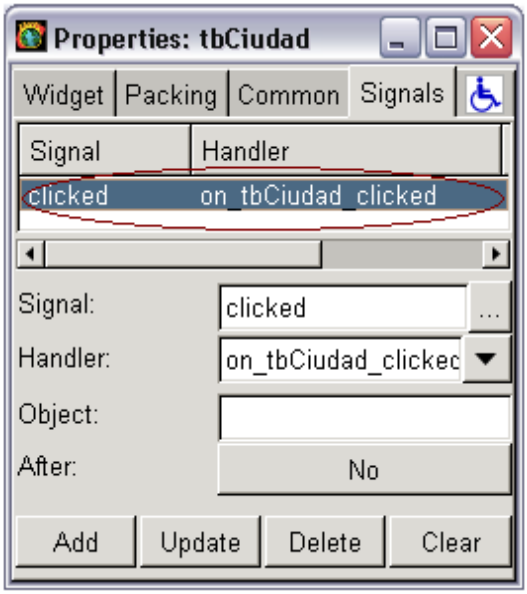


Ahora solo nos falta agregar este handler a la lista de Signals. Para hacer esto, solo basta hacer clic en el botón *Add* (Agregar). Note que en esta misma área existe Update (Modificar), Delete (Eliminar) y Clear (borrar toda la lista). Esto será muy útil cuando estemos alterando señales y capturadores de eventos en un desarrollo que exige más manejo de eventos que el de este ejemplo.

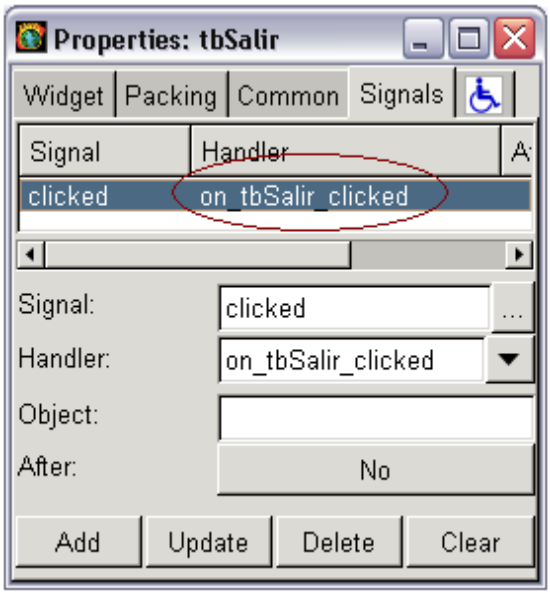
Ahora su lista debe lucir como la siguiente imagen:



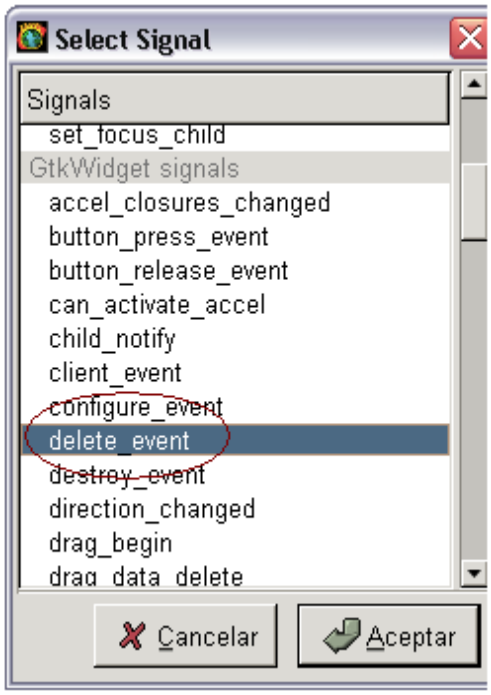
Para programar el botón tbCiudad selecciónelo del diseño de la ventana y repita los pasos anteriores. Note que Glade nombro la rutina capturadora del evento de forma automática. Al final, la pestaña de Signals del botón tbCiudad debe lucir similar a la siguiente figura:



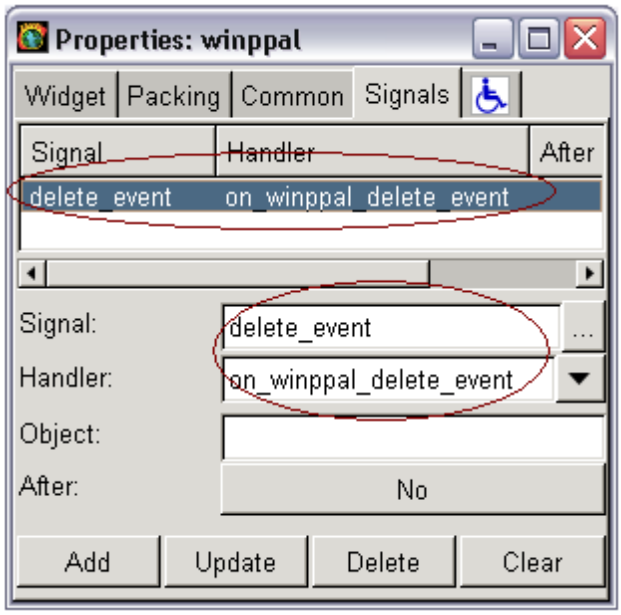
Para programar el botón tbSalir selecciónelo del diseño de la ventana y repita los pasos anteriores. Note que Glade nombro la rutina capturadora del evento de forma automática. Al final, la pestaña de Signals del botón tbSalir debe lucir similar a la siguiente figura:



El último evento que programaremos será el de capturar el cierre de la ventana. Para hacer esto cierre la ventana que estamos diseñando (winppal) y por seguridad, haga clic en el objeto winppal de la ventana de Glade (objetos del proyecto). Una vez seleccionada, haga clic en la ventana propiedades, seleccione la pestaña Signals y programe el evento *delete_event*.



Si hizo las cosas bien, debe tener las propiedades definidas de la siguiente forma:



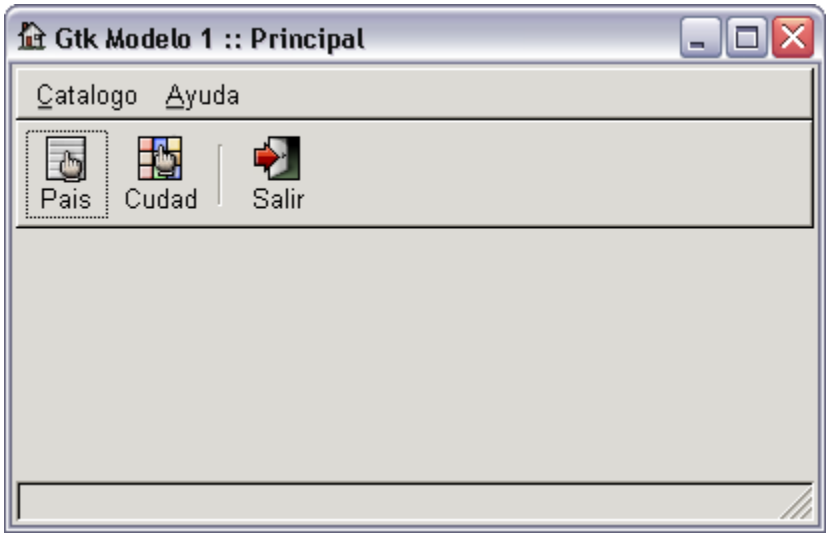
Con este evento, nuestro código podrá atrapar el momento en que se intente cerrar la ventana.

3.6.4. Finalizando el diseño de nuestra GUI

Solo nos falta agregar un pequeño truco para que la interfaz en el momento de ejecutar se vea bien. Con lo que tenemos hasta el momento, cuando se ejecute nuestra aplicación obtendremos la siguiente representación en pantalla:

La barra de estado se encuentra ubicada después de la barra de herramientas. Por supuesto que no debe de ir en ese punto. Lo que haremos es aplicar un truco que le aprendimos al asistente de Gtk# en el objeto window1. Revisando ese objeto, solucionaron este inconveniente insertando en la tercer línea del Vertical Box un widget Drawing Area (área de dibujo).

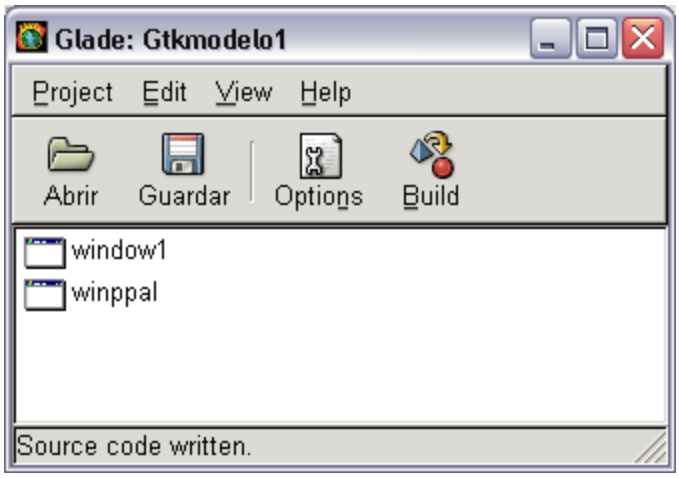
Ahora, el diseño de la ventana debe lucir de la siguiente forma:



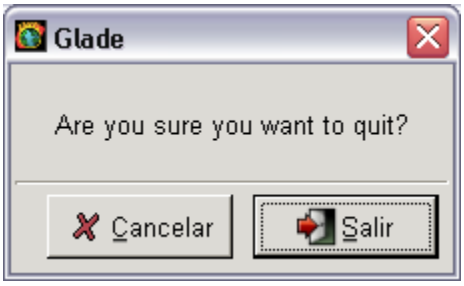
El widget drawing area se estirará hasta llenar de forma uniforme el rectángulo central de nuestra interfaz, logrando el efecto que deseamos para nuestro diseño. Recursivos no?

Ahora ya tenemos todo un diseño de GUI finalizado. Solo nos falta indicarle a Glade que deseamos construir el código XML que necesita el wrapper para integrarlo a nuestro código. Cierre la ventana winppal si aún la tiene abierta. Guarde los cambios que hemos hecho hasta el momento. No lo había guardado nunca? Es una muy mala idea no hacerlo con Glade de por medio.

Para construir nuestro código XML, basta con hacer clic en el botón Build (Construir). Glade generará de forma automática el archivo que incluye nuestro diseño GUI. Cuando termina de escribir el código notifica en la barra de tareas que el código ha sido escrito.



Ahora cierre el Glade y volvamos al IDE de Microsoft Visual Studio. Responsa haciendo clic en el botón [*Salir*] y ya estamos de vuelta en el IDE.



Nota: para copiar elementos en la memoria, lo mejor es hacerlo a través del menú emergente (pop Up) que Glade muestra en los sitios donde se puede copiar. Si no se hace de tal forma, el simple Ctrl + C puede no llegar a traer nada desde la interfase Glade hacia por ejemplo, el IDE Visual Studio.

Ahora solo nos falta agregar el código necesario para que nuestra aplicación funcione. Solo le hace falta ir al refrigerador y destapar la cuarta cerveza!!!.

3.6.5. Agregando el código para integrar la GUI a nuestra aplicación

Ahora ya tenemos una interfaz gráfica terminada y con eventos programados. Lo que haremos en esta sección será agregar el código necesario para integrar la GUI y atrapar mediante código los eventos programados.

Vamos a modificar el código para que haga exactamente lo que queremos que haga.

Identifique la región de código #region Glade Widgets y cambie el texto en negrita. Esta línea crea una referencia interna con el objeto ventana definido en el proyecto Glade.

```
#region Glade Widgets

[Widget] Gtk.Window winppal;

#endregion
```

Busque la rutina Driver y modifique el texto que aparece en **negrita**. Con este texto se le informa al IDE que busque dentro del archivo XML de glade, la definición de la ventana *winppal*, el cual es el nombre que le definimos a nuestro objeto window en el proyecto Glade.

```
public Driver(string[] args)
{
   Application.Init();

   Glade.XML gxml = new Glade.XML (null, "gladesharp1.glade", "winppal", null);
   gxml.Autoconnect (this);
   Application.Run();
}
```

Ahora busque la rutina llamada *on_window1_delete_event* y cámbiele el nombre a *on_winppal_delete_event.* Esta sección de código se ejecutará cuando se intente cerrar la ventana de la aplicación. Podría agregar por ejemplo, una pregunta al usuario consultando sobre el cierre o no de la aplicación.

```
public void on_window1_delete_event (object o, DeleteEventArgs args)
{
   Application.Quit();
   args.RetVal = true;
}
```

Ahora busque la región o sección de código llamada **#region Button Click Event handlers** y elimínela con todo y rutinas. Haga lo mismo con la sección de código **#region Menu item handlers**.

Lo que haremos ahora es crear nuestra propias secciones de código más acorde a lo que necesitamos hacer con nuestra GUI.

Hagamos un inventario de las rutinas que asociamos a eventos en las opciones del sistema de menú principal y barras de herramienta (para que no se nos quede sin programar ninguna, además por que se generaría un error en tiempo de ejecución cuando el runtime de .Net no encuentre la rutina que debe ejecutar cuando suceda el evento programado.

Rutinas asociadas a las subopciones de menú

Menu	***Subopcion (Evento Clic)***	***Rutina capturadora del evento***
Catalogo	País	on_opcPais_activate
	Ciudad	on_opcCiudad_activate
	Salir	on_opcSalir_activate
Ayuda	Índice	on_opcIndice_activate
	Ayuda	on_ocpAcerca_activate

Rutinas asociadas a la barra de herramienta

Botón (Evento Clic)	***Rutina capturadora del evento***
btPais	on_tbPais_clicked
btCiudad	on_tbCiudad_clicked
btSalir	on_tbSalir_clicked

Rutinas asociadas a la ventana

Evento	*Rutina capturadora del evento*
delete_event	on_winppal_delete_event

Como plantilla de ejemplo, tenga presente que la rutina capturadora de evento clic para una opción de menú es la siguiente:

```
protected void on_OpcionMenu_activate(object o, EventArgs args)
{
        return;
}
```

La rutina capturadora de evento clic para una barra de herramienta es la siguiente:

```
protected void on_BotonHerramienta_clicked(object o, EventArgs args)
{
        return;
}
```

Y para capturar el cierre de una ventana es la siguiente:

```
public void on_NombreVentana_delete_event (object o, DeleteEventArgs args)
{
   Application.Quit();
   args.RetVal = true;
}
```

Volviendo al IDE, en el código eliminamos todas las rutinas que estaban predefinidas por el asistente Gtk#.

Después de la rutina on_winppal:delete_event defina dos regiones de código de la siguiente forma:

```
// conectar las señales definidas en Glade
public void on_winppal_delete_event (object o, DeleteEventArgs args)
{
   Application.Quit();
   args.RetVal = true;
}

#region Capturar Evento clic en botones de barra de herramienta

#endregion

#region Capturar Evento clic en opciones de menu

#endregion
```

En cada una de estas secciones agruparemos las rutinas de tal forma que se vea muy organizado el código, algo muy útil cuando el proyecto empiece a crecer en código.

La región de código *#region Capturar Evento clic en botones de barra de herramienta* debe incluir las siguientes rutinas:

```
#region Capturar Evento clic en botones de barra de herramienta

protected void on_tbPais_clicked(object o, EventArgs args)
{
        return;
}
protected void on_tbCiudad_clicked(object o, EventArgs args)
{
        return;
}
protected void on_tbSalir_clicked(object o, EventArgs args)
{
        Application.Quit();
        return;
}

#endregion
```

La región de código *#region Capturar Evento clic en opciones de menu* debe incluir las siguientes rutinas:

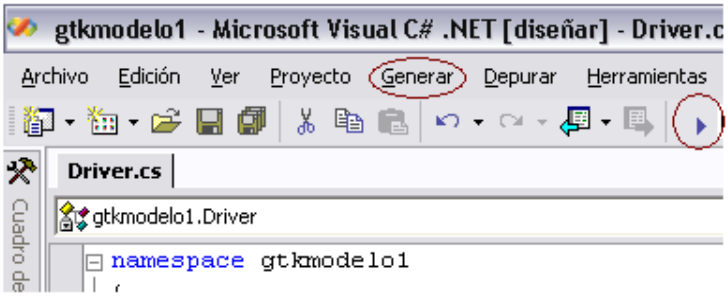
```
#region Capturar Evento clic en opciones de menu

protected void on_opcPais_activate(object o, EventArgs args)
{
        return;
}
protected void on_opcCiudad_activate(object o, EventArgs args)
{
        return;
}
protected void on_opcSalir_activate(object o, EventArgs args)
{
        Application.Quit();
        return;
}
protected void on_opcIndice_activate(object o, EventArgs args)
{
        return;
}
protected void on_ocpAcerca_activate(object o, EventArgs args)
{
        return;
}
#endregion
```

Aquí está el código completo de la aplicación:

```
using System;
using Gtk;
using Glade;

namespace gtkmodelo1
{
	/// <summary>
	/// Gtk Modelo1. Un modelo para el desarrollo con Gtk y mono
	/// </summary>
	class Driver
	{
		#region Glade Widgets

		[Widget] Gtk.Window winppal;

		#endregion


		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
			new Driver(args);
		}

		public Driver(string[] args)
		{
		   Application.Init();

		   Glade.XML gxml = new Glade.XML (null, "gladesharp1.glade", "winppal", null);
		   gxml.Autoconnect (this);
		   Application.Run();
		}
		// conectar las señales definidas en Glade
		public void on_winppal_delete_event (object o, DeleteEventArgs args)
		{
		   Application.Quit();
		   args.RetVal = true;
		}

		#region Capturar Evento clic en botones de barra de herramienta

		protected void on_tbPais_clicked(object o, EventArgs args)
		{
			return;
		}
		protected void on_tbCiudad_clicked(object o, EventArgs args)
		{
			return;
		}
```

```
		protected void on_tbSalir_clicked(object o, EventArgs args)
		{
			Application.Quit();
			return;
		}

		#endregion

		#region Capturar Evento clic en opciones de menu

		protected void on_opcPais_activate(object o, EventArgs args)
		{
			return;
		}
		protected void on_opcCiudad_activate(object o, EventArgs args)
		{
			return;
		}
		protected void on_opcSalir_activate(object o, EventArgs args)
		{
			Application.Quit();
			return;
		}
		protected void on_opcIndice_activate(object o, EventArgs args)
		{
			return;
		}
		protected void on_ocpAcerca_activate(object o, EventArgs args)
		{
			return;
		}
		#endregion

	}
}
```

Ahora solo falta generar nuestro código ejecutable. Haga clic en la opción *Generar* del IDE Visual Studio .Net y seleccione la opción *Generar Solución*. Ahora ya puede ejecutar su aplicación desde el botón *Iniciar* del IDE.



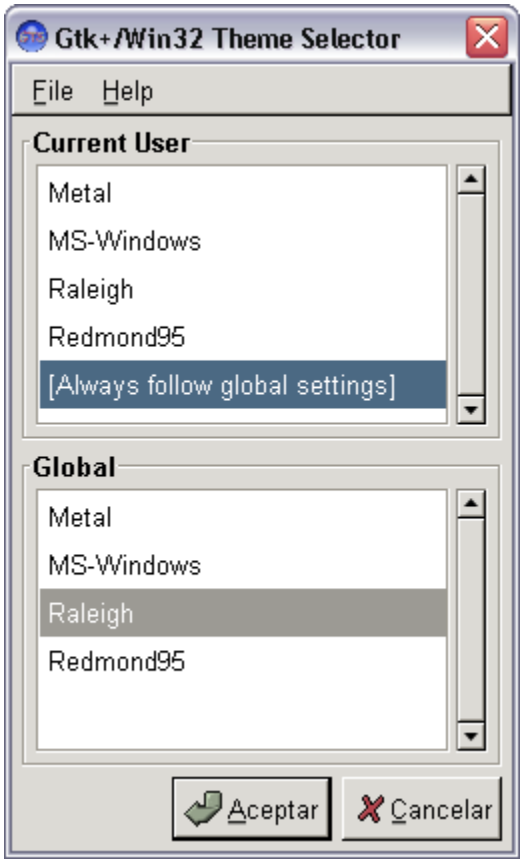
Cuando se ejecute la aplicación se obtendrá una imagen similar a esta:

Pruebe su funcionalidad y notará que tiene todos los elementos y comportamientos típicos de una ventana principal de aplicación.

3.6.6. Cambiando el tema de Glade

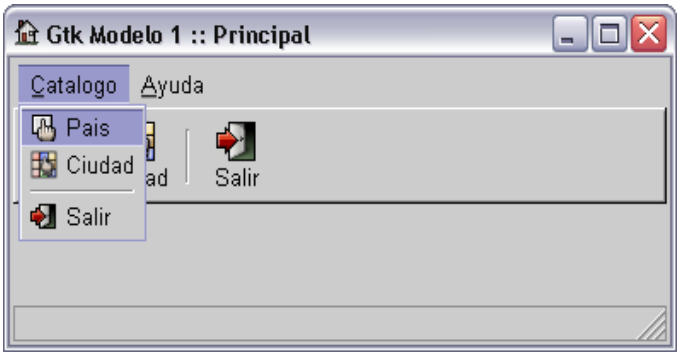
El Glade cuando se instala, incorpora una utilidad para cambiar de forma sencilla el tema en que debe mostrar las gráficas GTK. Esta es una forma muy sencilla de probar lo potente que es el GTK para adquirir un diseño de presentación (look and feel) diferente con solo cambiar un parámetro.

La utilidad se carga desde el grupo de *Gtk# for Windows*, opción *Applications;* haga clic sobre *ThemeSelector* (Selector de temas). Se cargará la siguiente caja de diálogo:

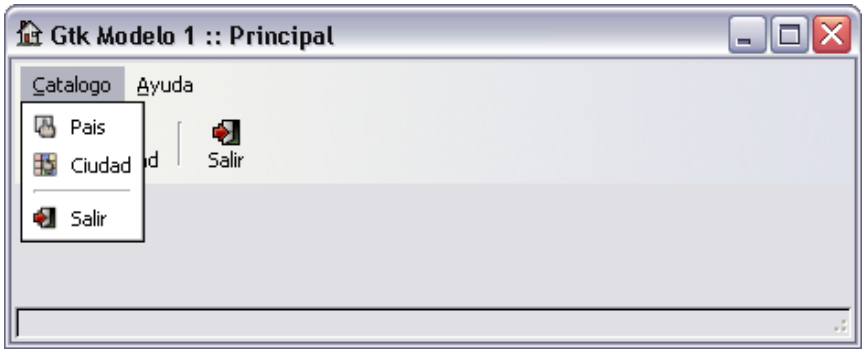


En la parte que dice *Current User*, deje seleccionada la opción [Always follow global settins] (siempre siga la configuración global). En la sección Global, seleccione el tipo de look-and-feel deseado.

Por ejemplo si seleccionamos *Metal*, y hacemos clic en el botón [*Aceptar*], se obtiene la siguiente presentación de nuestra aplicación al volverla a ejecutar.

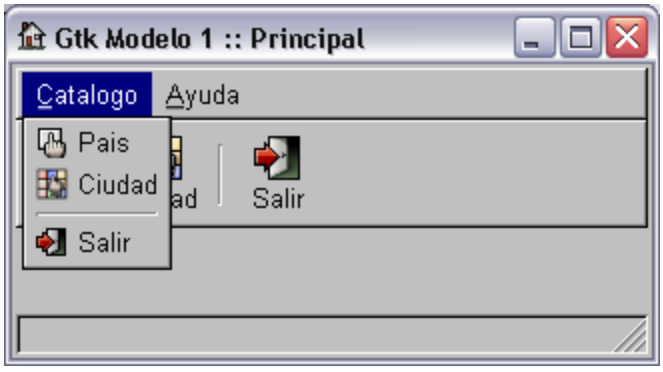


La misma aplicación con la opción *MS-Windows* seleccionada en el ThemeSelector:



La misma aplicación con la opción *Raleigh* seleccionada en el ThemeSelector:

La misma aplicación con la opción *Redmon95* seleccionada en el ThemeSelector:



Con esta utilidad, tenemos cuatro sencillas formas de ver nuestra misma aplicación y sin tener que escribir nada de código para esto.

3.7. Probando al mono

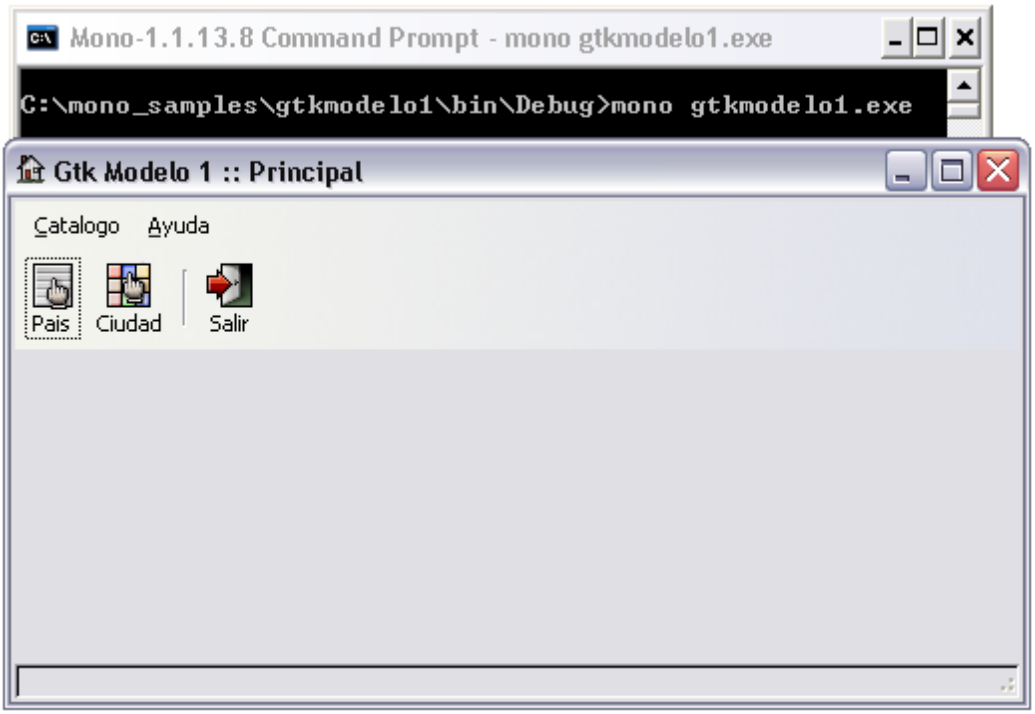
Hasta el momento hemos desarrollado nuestra aplicación con el IDE de Microsoft Visual Studio. Pero, será que nuestro compilado será compatible con el mono? Vamos a hacer la prueba.

Primero, abra una ventana de comando desde la opción que mono trae para esto. Haga clic en *Inicio*, *Todos los programas*, *Mono 1.1.13.8 for Windows* y seleccione la opción *Mono 1.1.13.8 Command Prompt*. Esto iniciará una sesión de comandos con todos los recursos de mono listo.

Vamos hasta la carpeta donde reside el compilado de nuestro proyecto gtkmodelo1 en la ruta:

c:\mono_samples\gtkmodelo1\bin\debug\

Luego escriba la orden: mono gtkmodelo1.exe



Observe que la aplicación ejecuta, lo cual indica que hasta el momento se ha mantenido la compatibilidad multiplataforma. Es muy buena práctica ir probando la compatibilidad entre .net para evitar el tener que volver a reescribir código cuando intentemos ejecutar nuestra rutina en cualquiera de las dos plataformas.

En otros capítulos, probaremos el desarrollo desde otros IDES diferentes al de Microsoft .Net.

3.8. Resumen

En este capitulo diseñamos nuestra primer interfaz en glade y la vinculamos a nuestro código. La interfaz diseñada es la típica ventana principal de una aplicación. Se dieron además los modelos o plantillas de código para los eventos que se pueden programar en este tipo de interfaces.

Aprendimos además a interactuar con el Glade en las rutinas más básicas de diseño de una GUI.

En el Capítulo 4 aprenderemos a crear utilidades GTK y a integrarlas de la forma correcta a la interfaz GUI principal.

Hasta la próxima entrega.

Cordialmente,

Mauricio Cano Ossa

Un discípulo más del dios de los monos...