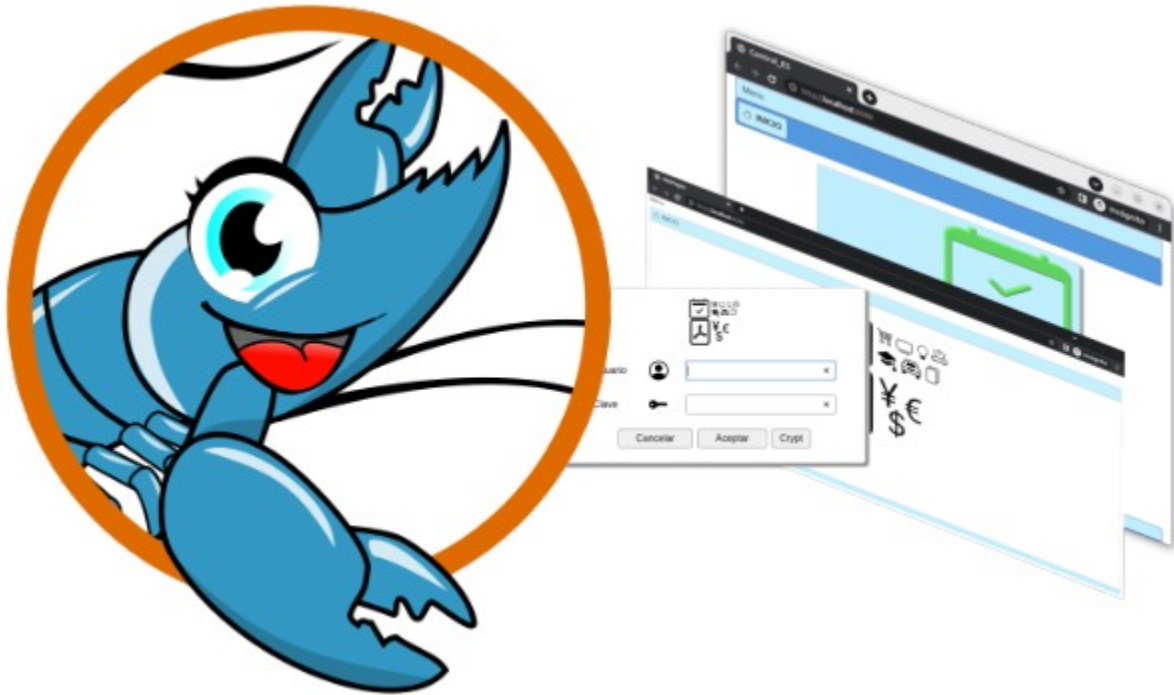


# Crear WebApp con Gambas3



Por Omar Moreno  
omarmorenoc21@gmail.com

## INDICE

Qué es Gambas .....	3
Requisitos para este Manual .....	3
Un ejemplo sencillo en cuatro pasos .....	4
Presentación (Maquetado) .....	10
Crear WebApp de varios WebForm .....	13
WFrmPrincipal .....	15
WfrmProcesos (X,Y, Z) .....	20
WFrmLogin .....	22
WFrmLogin con ShowModal() .....	24
Tipos de Llamados .....	26
Los Códigos .....	27
Pendientes .....	32

## **¿Que es Gambas3?**

Gambas es un entorno de desarrollo gratuito y una plataforma de desarrollo completamente poderosa.

Esta basado en un intérprete de Basic con extensiones de objetos, tan fácil como Visual Basic™.

Si alguna vez programaste con Visual Basic™ de Microsoft entonces estarás muy familiarizado con este lenguaje, solo que este funciona principalmente para el sistema operativo Linux, pero si estas iniciando en el mundo de la programación entonces bienvenido a un excelente y poderosa herramienta de desarrollo.

Gambas te permite realizar programas para consola, escritorio y web, en este pequeño manual veremos solamente el desarrollo de aplicaciones tipo WebForm con el componente gb.web.gui.

Para saber mas te dejo la url de su pagina oficial en donde encontraras todo lo relacionado al lenguaje: <https://gambas.sourceforge.net>

## **Requisitos para este Manual:**

1. Tener instalado Gambas versión mínima: 3.18.2.
2. Conocimientos básicos de HTML, CSS y opcionalmente javascript.

Si no tienes instalado Gambas3, en su pagina oficial están los pasos para la instalación de la versiona estable y la versión de desarrollo.

En el menú del sitio dar clic en **Ayuda** luego en el submenu **Instalación y Compilación:**

### **Desarrollo:**

Esta versión tiene la reparación de los últimos Bug que puedan aparecer.

Se deben instalar todos los requerimientos, compilar e instalar.

Recomiendo hacerlo en una maquina virtual ejemplo (Virtual Box).

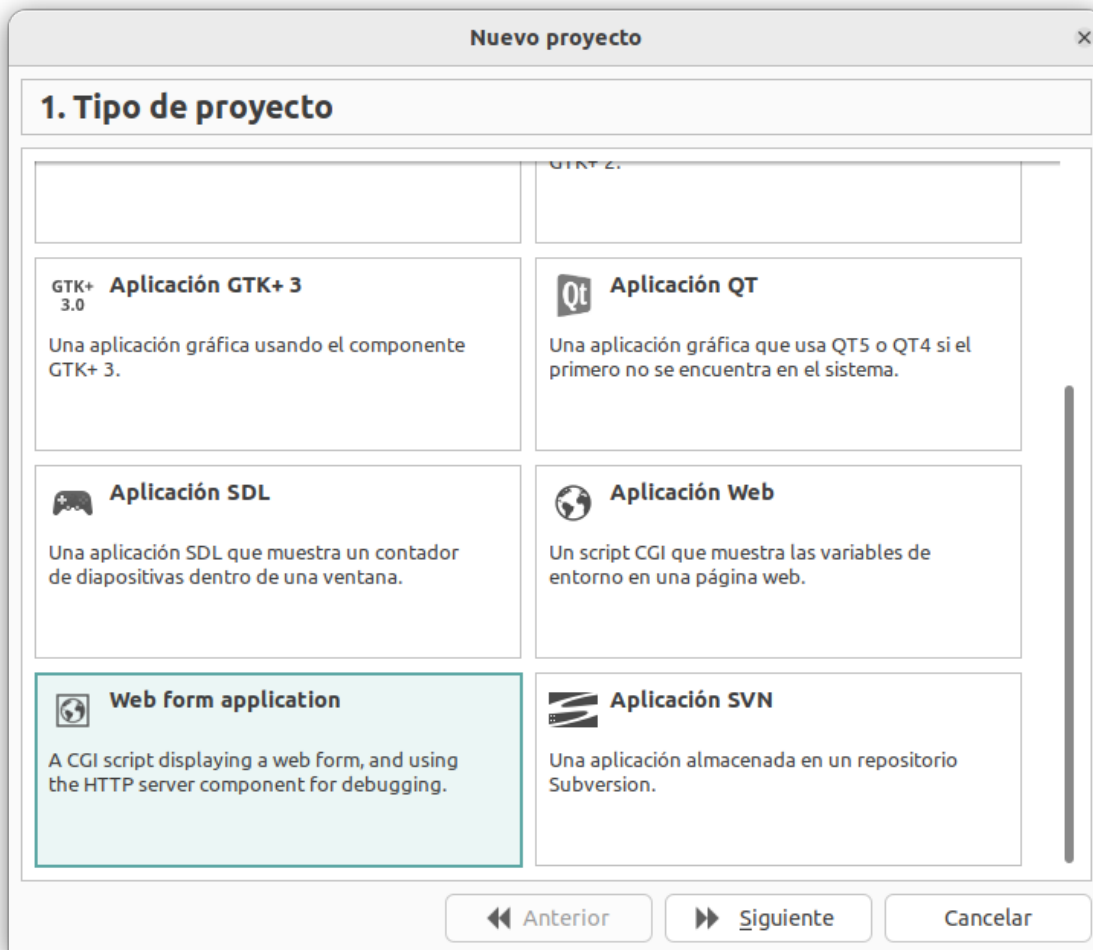
### **Estable:**

Esta versión es la mas segura para producción en cada distribución de Linux.

En el link: [Instrucciones especificas para cada distribución](#) podrás encontrar los pasos de instalación.

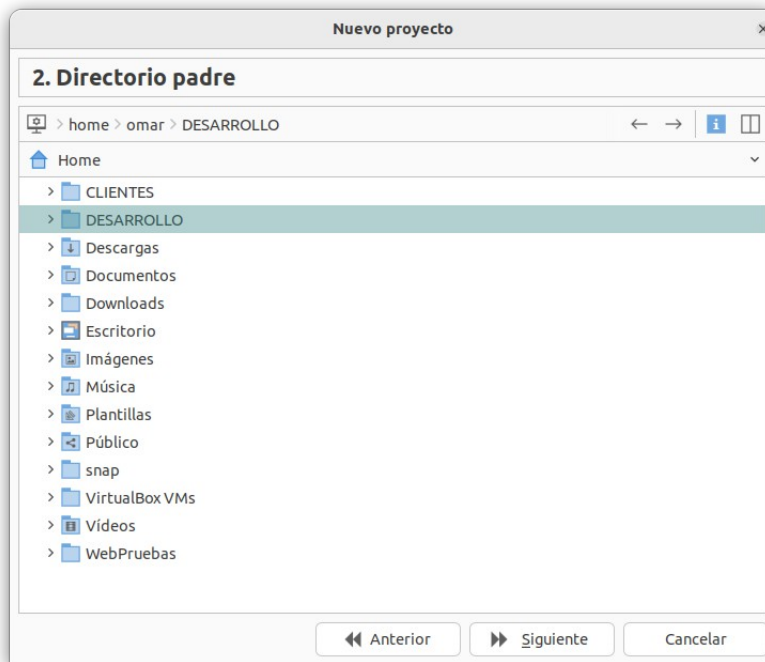
### Un ejemplo sencillo en cuatro pasos:

1. Ejecuta Gambas, el IDE te mostrara una ventana con el consejo del día, leelo para aprender un poco mas y luego cierras esa ventana.
2. Dar clic en la opción Nuevo Proyecto, luego te aparecerá la ventana de los tipos de proyectos, selecciona **Web Form Application** y pulsas en botón siguiente:

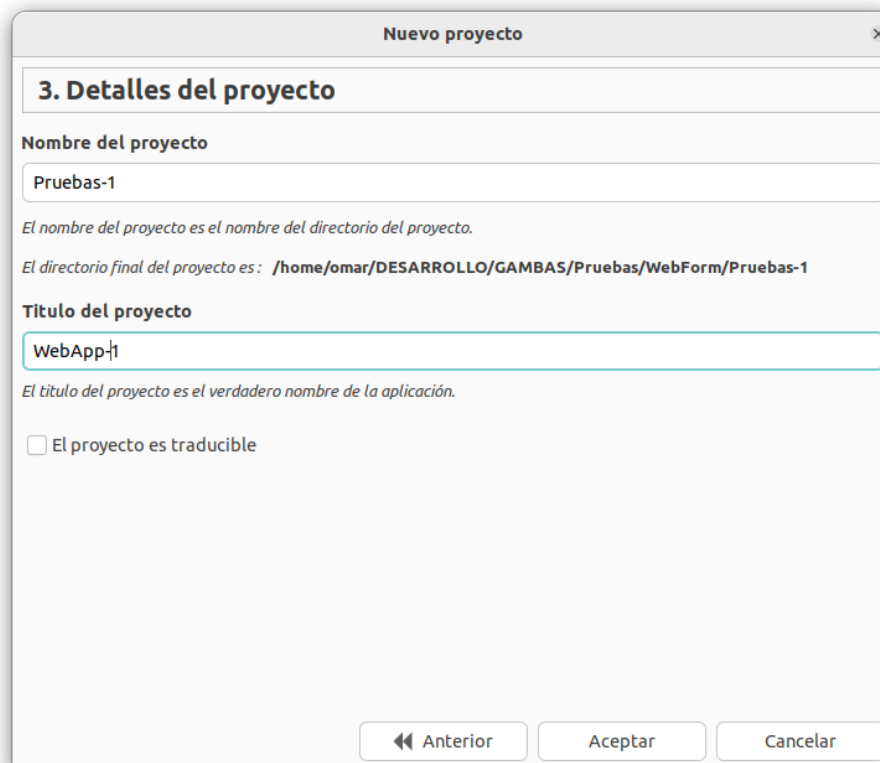


Si no ves Web Form Application, es por que debes desplazar la barra lateral hasta que aparezca.

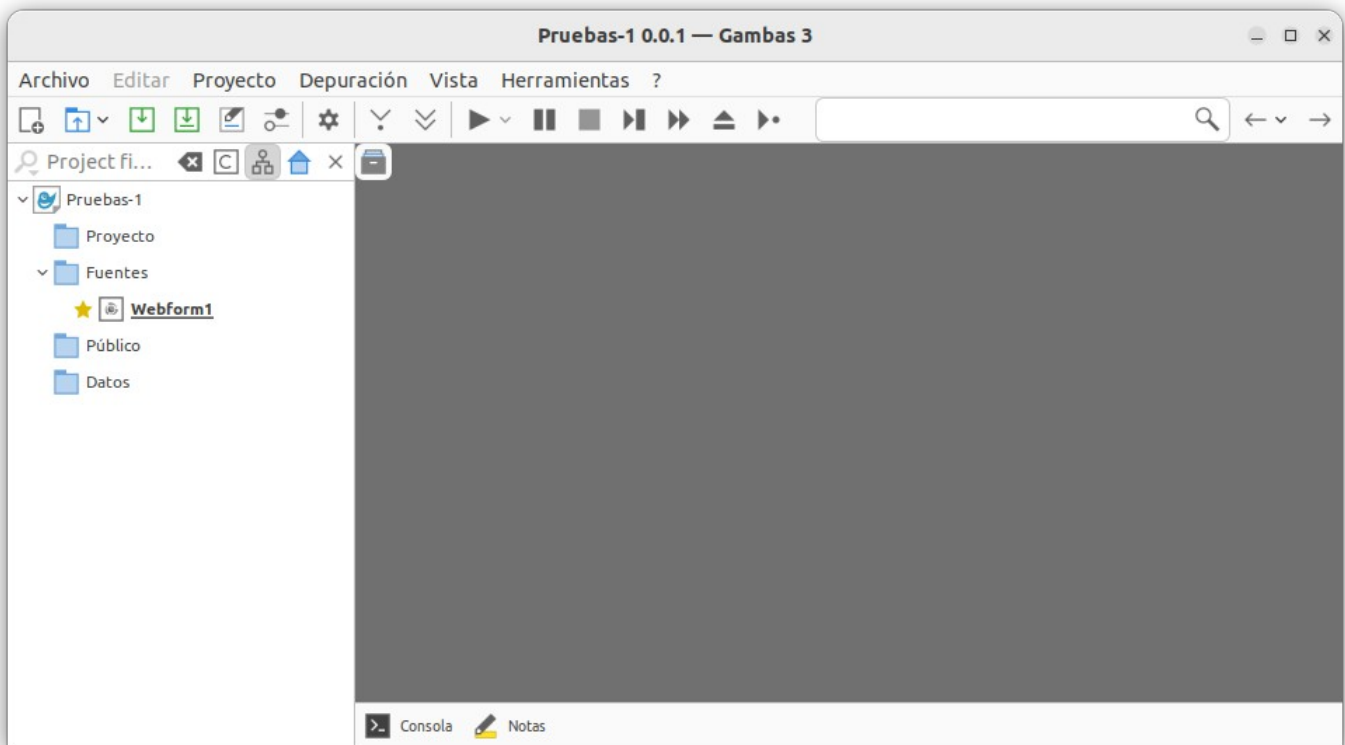
3. En la siguiente ventana deberás elegir la carpeta en donde guardaras tu proyecto:



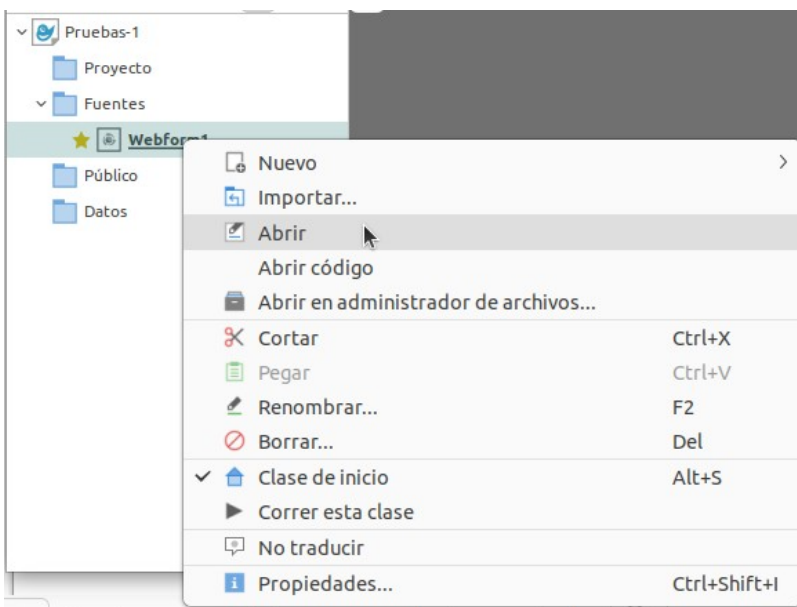
4. Finalmente escribes los detalles del proyecto, (Nombre,Titulo) luego deberás dar clic en Aceptar:



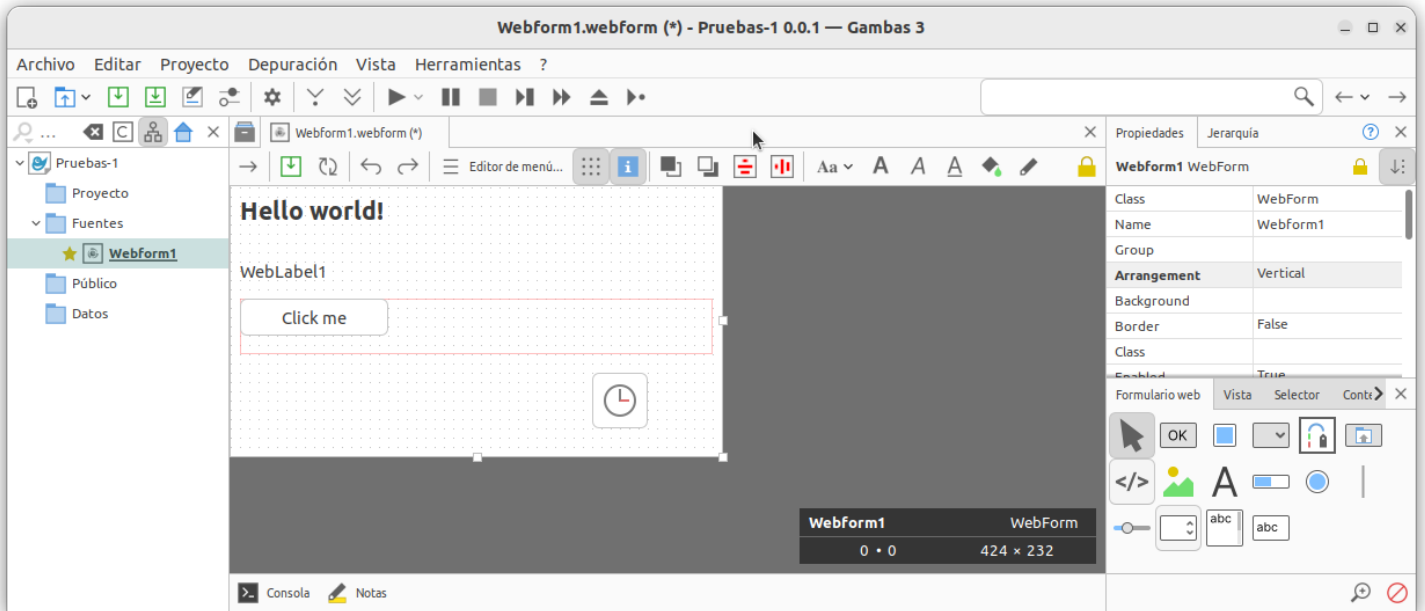
Una vez que has finalizado los 4 pasos anteriores Gambas te mostrara el entorno de desarrollo integrado (IDE):



En el directorio Fuentes se crearan todas las clases WebForm, observa que Gambas ha creado un WebForm **Webform1** el cual puedes abrir dándole doble clic o pulsando el botón derecho del mouse y luego seleccionado la opción: Abrir.

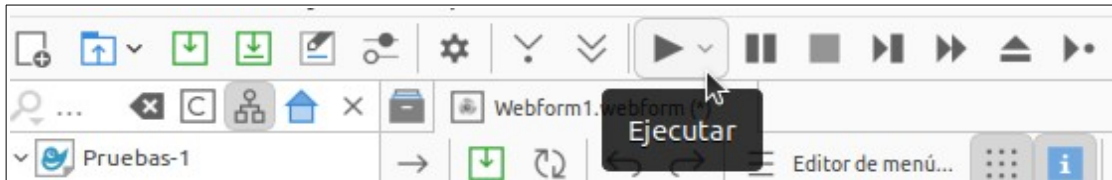


Cuando abras el webform1 el IDE te mostrara los objetos creados dentro del webform1 y ademas a la derecha te mostrara las ventas de objetos que puedes añadir al webform con sus respectivas propiedades y jerarquía que ocupan dentro del webform:

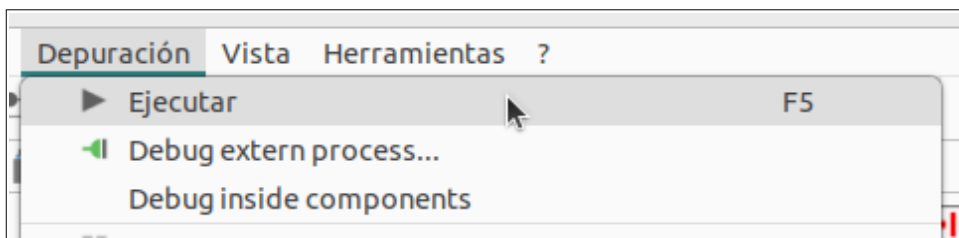


Este ejemplo ya es una WebApp funcional y para probarlo (Ejecutar) tienes varias opciones:

- Pulsando la tecla F5 de tu teclado.
- Dando Clic en el Botón Ejecutar de la barra de botones



- Seleccionando del menú Depuración la opción Ejecutar

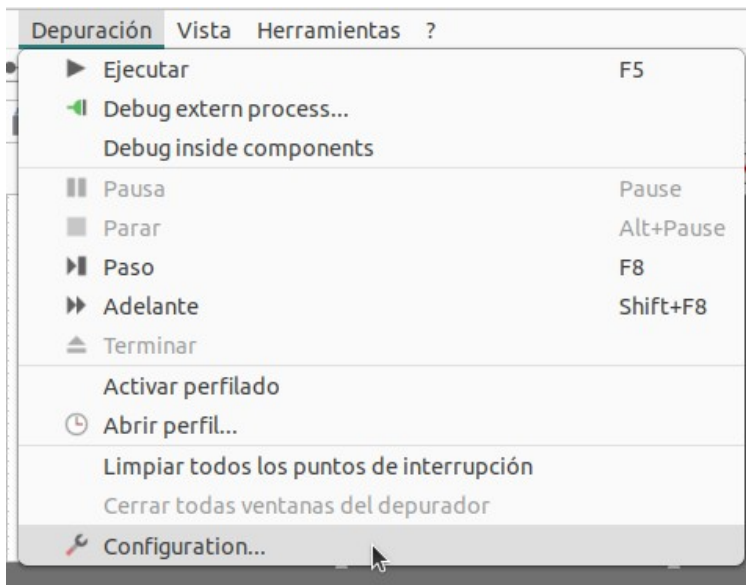


Cualquiera de las opciones que elijas abrirán el navegador web de tu PC y te mostrara la WebApp.

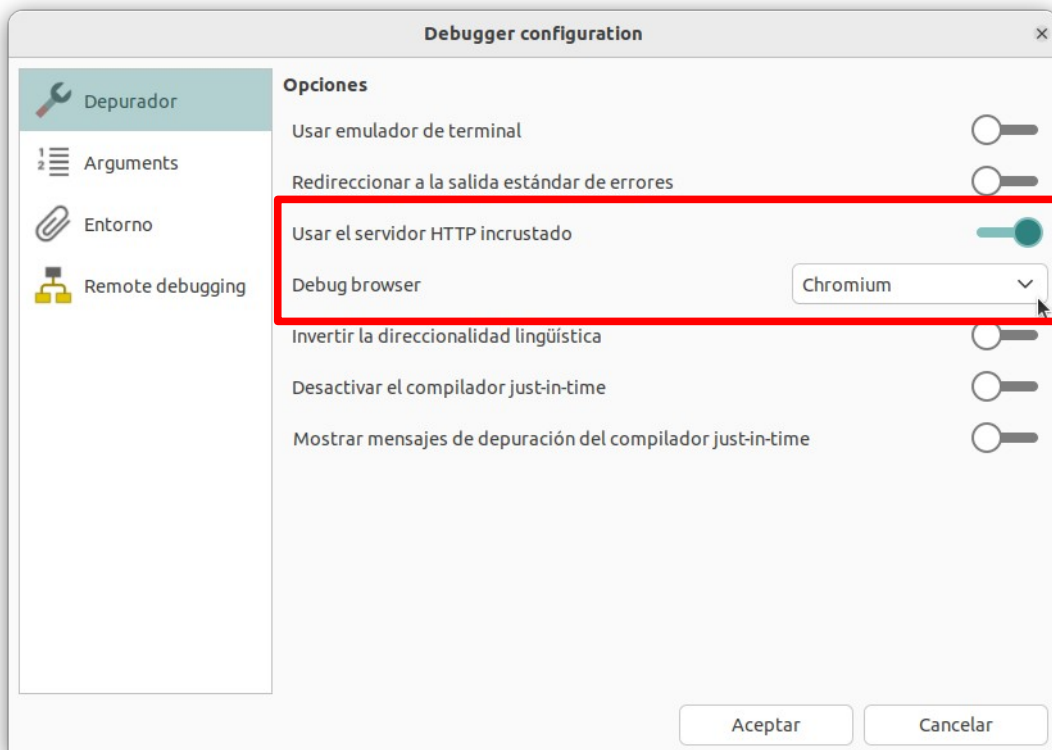
Si el navegador Web de tu PC no se abre con la WebApp, sigue estos pasos de configuración en el IDE de Gambas:

Dos pasos para ejecutar una WebApp con el Servidor Integrado de Gambas:

1. Selecciona el menú Depuración y luego el submenu Configuración:



2. En la ventana de configuración, opciones del Depurador, deberás activar: **Usar el servidor HTTP incrustado** y en la opción **Debug browser** deberás seleccionar el navegador que tenga su PC, en este caso tengo instalado Chrome, pero puedes utilizar Firefox.

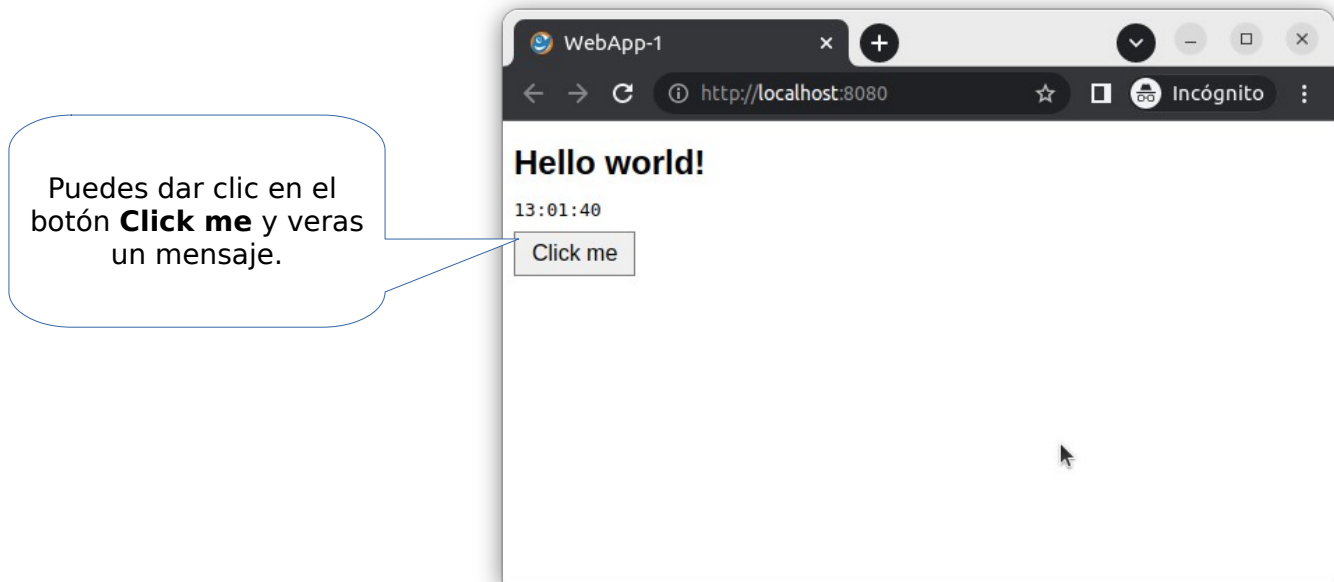




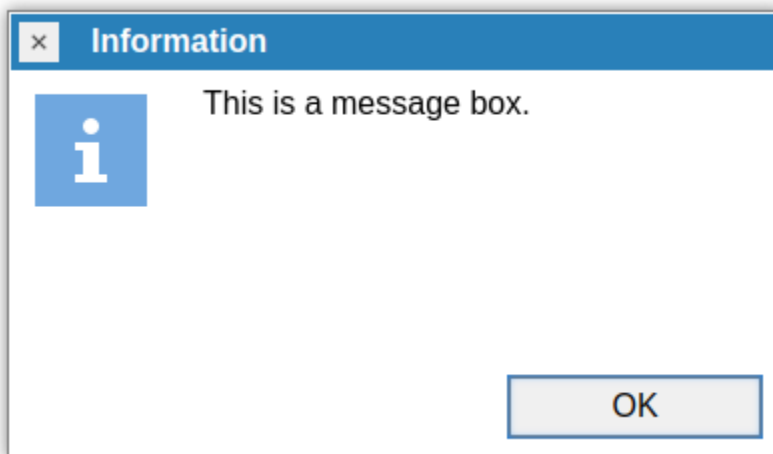
Después de haber configurado el Depurador, puedes ejecutar nuevamente el programa y veras que se abre el navegador elegido con la WebApp de ejemplo:

Podrás observar que la WebApp tiene tres objetos en su webform:

- ✓ Un WebHtml mostrando el texto "Hello world".
- ✓ Un WebTimer mostrando el texto de un reloj que cambia a cada segundo.
- ✓ Un WebButton mostrando el texto "Click me"



Cuando pulses el botón "Click me" este es el mensaje que te saldrá:



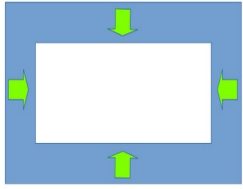
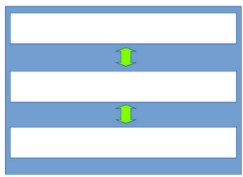


Felicidades ya has creado tu primer WebApp en Gambas.

### **Presentación (Maquetado):**

El contenedor principal de una WebApp es el WebForm, pero dentro de un WebForm también se pueden agrupar objetos por medio de los contenedores que aparecen en la pestaña Contenedores.

Hay varias combinaciones para configurar la presentación tu Webform, te recomiendo que practiques cada una para que veas las formas que van tomando los objetos según cada una de las configuraciones, las propiedades que mas afectan la presentaciones son:

Propiedad	Valor	Acción	
Arrangement	Vertical	Provocara que los objetos se <b>apilen</b> de arriba a abajo y que se <b>expandan</b> automáticamente en el eje horizontal de su contenedor.	
	Horizontal	Provocara que los objetos se <b>apilen</b> de izquierda a derecha y que se <b>expandan</b> automáticamente en el eje vertical de su contenedor.	
Margin	True	Colocara un <b>espacio</b> entre el borde interior del contenedor y cada objeto contenido.	
	False	Eliminara el <b>espacio</b> entre el borde interior del contenedor y cada objeto contenido.	
Spacing	True	Colocara un <b>espacio</b> entre cada objeto dentro del contenedor.	
	False	Eliminara el <b>espacio</b> entre cada objeto dentro del contenedor.	

Las propiedades de la tabla anterior se aplican al WebForm y los contenedores de la pestaña Contenedores excepto los WebHBox y WebVBox ya que estos por defecto tienen un Arrangement asignado según su nombre.

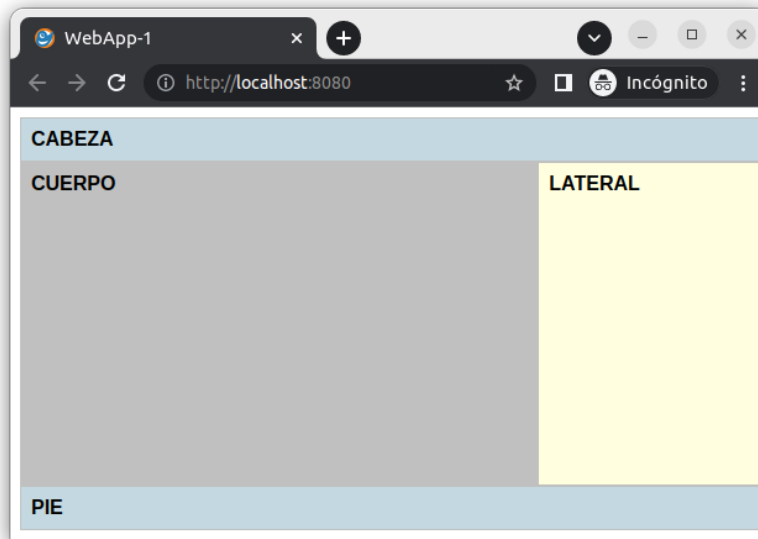
Como vimos en el ejemplo inicial, cada vez que inicies un proyecto del tipo Web Form Application el formulario inicial tendrá configurado estas propiedades:

Arrangement	Vertical
Margin	True
Spacing	True

Como te mencione anteriormente puedes agregar a tu WebForm contenedores para agrupar los objetos, todo depende del maquetado de tu WebApp.

Por ejemplo, para un maquetado del tipo cabeza, cuerpo, lateral, pie, puedes tener en Gambas algo similar:

cabeza	
cuerpo	lateral
pie	



Para lograr este maquetado, primero debemos configurar las siguientes propiedades del WebForm:

Objeto	Propiedad	Valor
WebForm	Arrangement	Vertical
	Height	100%
	Margin	True

Luego agregar los siguientes contenedores:

- ✓ Un WebHBox1 para la cabeza y dentro de este agregar un WebLabel.
- ✓ Un WebHBox2 para contener los contenedores (cuerpo y lateral)
- ✓ Dentro del WebHBox2 agregar dos WebContainer uno para el cuerpo y el otro para el lateral y en cada WebContainer agregar un WebLabel.
- ✓ Un WebHBox3 para el pie y dentro de este agregar un WebLabel.

Ahora debes configurar las propiedades de los otros objetos:

Objeto	Propiedad	Valor
WebHBox1	Background	&HC3D8E0
	Border	True
	Margin	True
WebLabel1	Font	bold
	Text	CABEZA
WebHBox2	Expand	True
Webcontainer1	Arrangement	Vertical
	Background	&HC0C0C0
	Border	True
	Margin	True
	Width	70%
WebLabel2	Font	bold
	Text	CUERPO
Webcontainer2	Background	&HFFFFDF
	Border	True
	Margin	True
	Width	30%
WebLabel3	Font	bold
	Text	LATERAL
WebHBox3	Background	&HC3D8E0
	Border	True
	Margin	True
WebLabel4	Font	bold
	Text	PIE

Finalmente ejecute su proyecto y vera un maquetado igual al mostrado en este manual.

## Crear WebApp de varios WebForm

En algún momento se tendrá la necesidad de crear alguna WebApp que requiera el uso de varios WebForm.

Hasta ahora he visto la propiedad **Current** y una function **Show**, con las que Gambas presenta cada WebForm en una ventana independiente de su padre, pero también hay otra forma incrustando las hijas con el uso de la palabra clave **New** para instancias de objetos .

Veamos los tres escenarios con un pequeño Ejemplo:

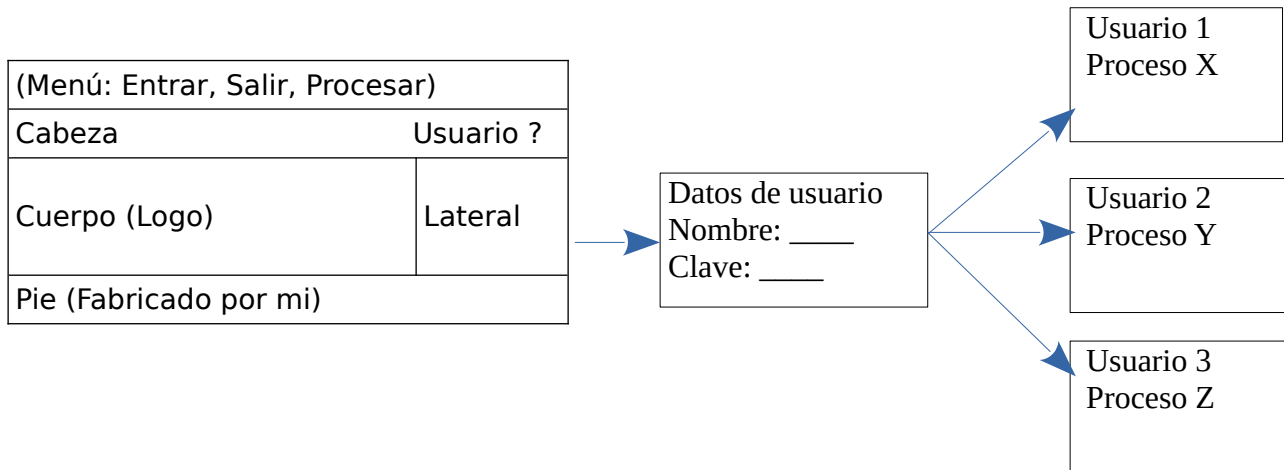
Suponga que el dueño de la empresa XYZ nos solicita una WebApp:

- Para tres usuarios de diferentes departamentos.
- Todos deben ingresar con usuario y contraseña.
- También desea que se vean mensajes generales para sus colaboradores.

Aceptamos el proyecto y le decimos que se lo vamos desarrollar con una súper herramienta que se ejecuta en Linux... al día siguiente en la mañana después de una buena taza de café (si lo toma), empezamos por el maquetado.., bueno... umm... heee..., ¡ha si!, crearemos lo siguiente:

1. Un WebForm principal que tendrá:
  1. Un menú con las opciones de ingresar, salir y procesar.
  2. La pantalla de inicio con el logo de la empresa
2. Un WebForm para solicitar los datos del usuario y validar su departamento.
3. Un WebForm para los usuarios del proceso 1 (X).
4. Un WebForm para los usuarios del proceso 2 (Y).
5. Un WebForm para los usuarios del proceso 3 (Z).

Finalmente tenemos un maquetado que nos tomo unos largos 21 segundos...

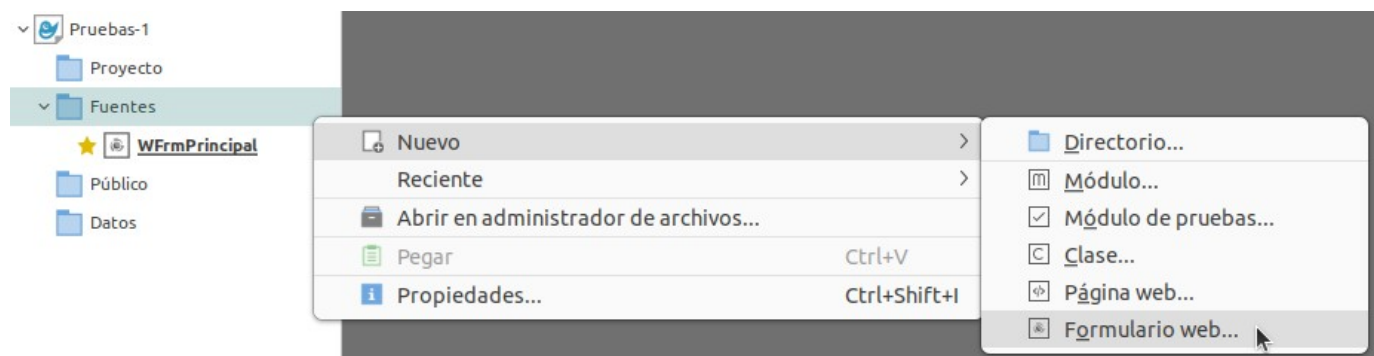


Rápidamente ejecutamos nuestro espectacular IDE de Gambas y creamos 5 WebForm:

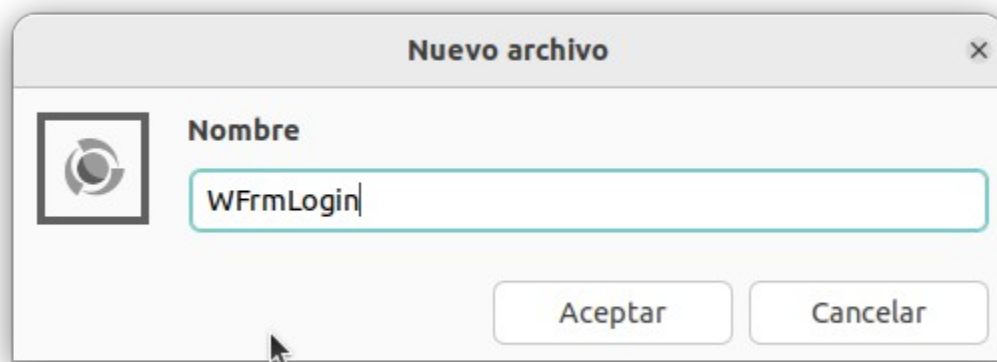
- ✓ WFrmPrincipal
- ✓ WFrmLogin
- ✓ WFrmProcesoX
- ✓ WFrmProcesoY
- ✓ WFrmProcesoZ

Nota: colega, puedes usar el proyecto actual o crear uno nuevo, si usas el actual solo debes cambiarle el nombre al archivo Webform1, para ello seleccionalo con el mouse y pulsa la tecla F2 o con el botón derecho del mouse selecciona Renombrar y le cambias el nombre a WFrmpPrincipal.

Para agregar nuevos formularios al proyecto solo debes seleccionar el directorio Fuentes con el botón derecho del mouse y luego seleccionar Nuevo y después Formulario web...

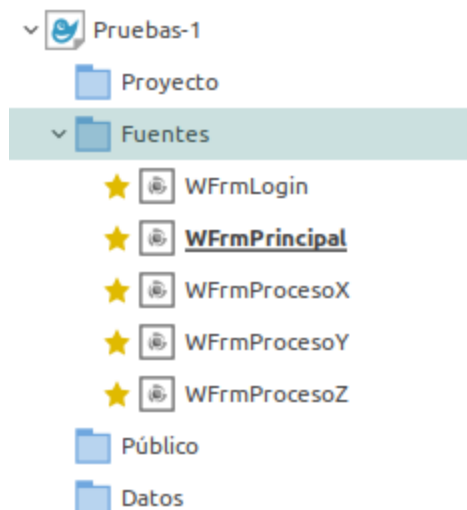


El IDE te mostrara una ventana solicitando el nombre del nuevo formulario:



Ingresa el nombre y luego pulsas el botón aceptar, repite estos pasos para crear los WebForm del ejemplo.

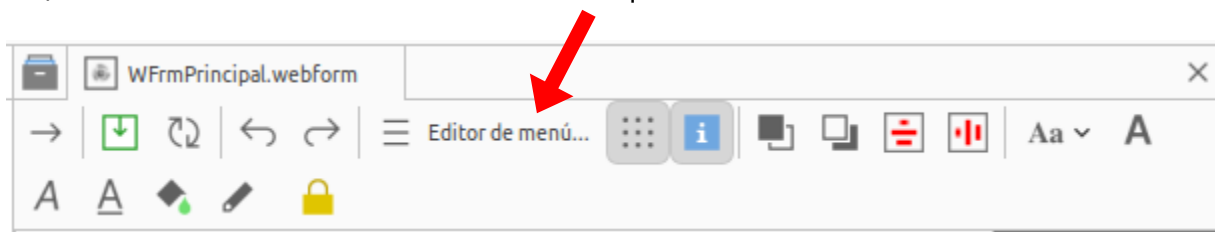
Bien ya hemos creado los cinco WebForm, el árbol del proyecto se debería ver como en la siguiente imagen:



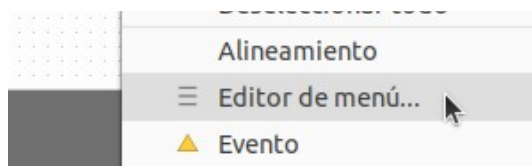
## WfrmPrincipal:

Ahora agregaremos algunos objetos y algo de código, abrir el WebForm WfrmPrincipal y seguir los siguientes pasos:


1. Agregaremos un menú y para esto tenemos 3 opciones:
  - a) Pulsando las combinación de teclas CTRL+E.
  - b) Dando clic en el botón Editor de Menú que se encuentra en la barra de botones



- c) Dando clic en el WebForm con el botón derecho del mouse y seleccionado la opción Editor de menú



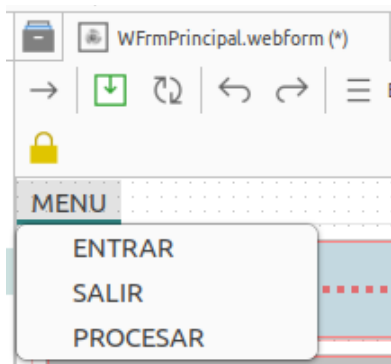
Con cualquiera de las tres opciones se abrirá una ventana para ingresar los elementos del menú:

- ✓ En la barra de botón daremos clic en el botón insertar  luego en el lateral derecho, editaremos la propiedad **Caption** que tiene el texto WebMenu1 por el texto MENU.
- ✓ Pulsaremos tres veces mas el botón insertar y editaremos la propiedad Name con los textos WebMenuEntrar y WebMenuSalir y WebMenuProcesar, también editaremos la propiedad Caption de cada elemento con los textos ENTRAR, SALIR y PROCESAR.
- ✓ Seleccionaremos el elemento ENTRAR y luego damos clic en el botón Identificar



- ✓ Seleccionamos el elemento SALIR y luego damos clic en el botón Identificar
- ✓ Seleccionamos el elemento PROCESAR y luego damos clic en el botón Identificar
- ✓ Desmarcaremos las propiedades Visible y Enabled de los elementos SALIR y PROCESAR.

Finalmente damos click en el botón Aceptar y ahora el menú aparece en el WebForm:

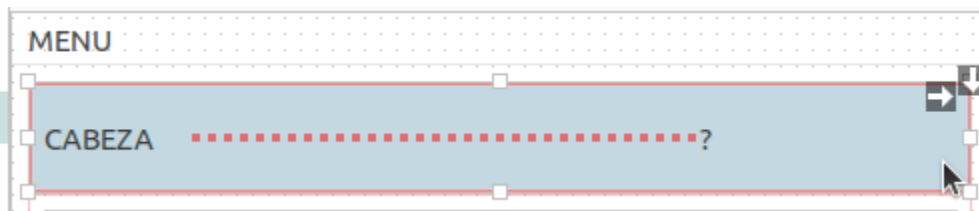


2. En el contenedor WebHBox1 (cabeza) agregamos en este orden: un WebContainer y un WebLabel luego modificamos las siguientes propiedades:

WebContainer	
Propiedad	Valor
Expand	True

WebLabel	
Propiedad	Valor
Name	WebLblUsuario
Alignment	Center
Text	?
Border	True

El contenedor webHBox1 debe quedar como en la siguiente imagen:

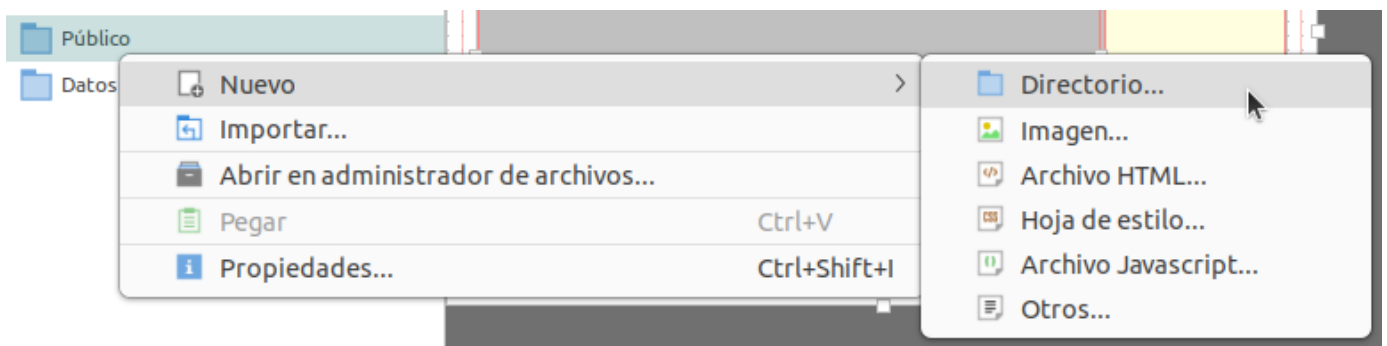


Nota: colega, como abras observado, agregaremos un WebContainer al que solamente le editamos la propiedad Expand, el objetivo de esto es desplazar el objeto WebLblUsuario al extremo derecho, el IDE de Gambas te deja saber eso colocando los puntos rojos.

3. En el contenedor WebContainer1 (Cuerpo) agregamos un objeto WebContainer y modificamos sus propiedades:

WebContainer	
Propiedad	Valor
Name	WebCLogo
Arrangement	Vertical
Margin	True
Spacing	True
Visible	False

4. Dentro del contenedor WebCLogo agregamos un objeto WebImage, pero antes de modificar sus propiedades debemos importar el archivo de imagen que contiene el logo de la empresa, para eso debemos dar click con el botón derecho del mouse sobre el directorio Público del árbol del proyecto y seleccionar la opción Directorio...:





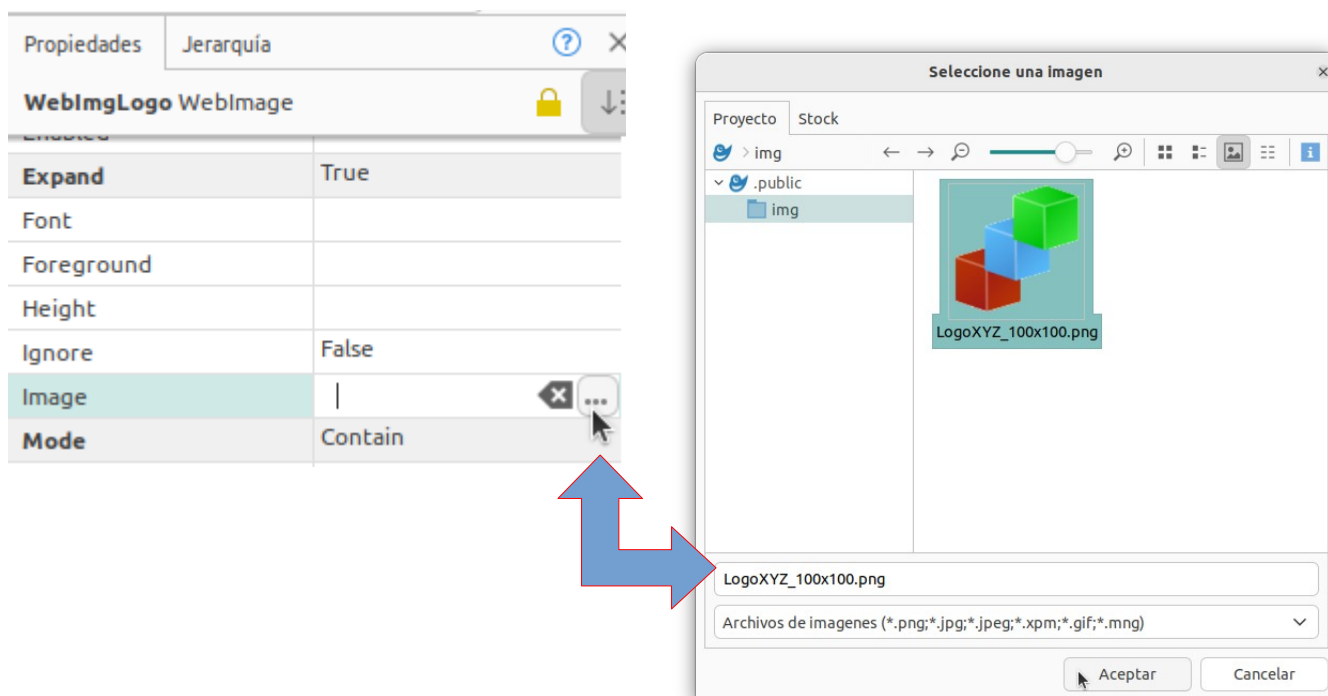
nos aparecerá una ventana solicitando el nombre para el nuevo directorio, ingresamos el nombre img y luego damos click en Aceptar, vemos que el nuevo directorio aparece como directorio hijo del directorio Público.

Ahora con el botón derecho del mouse damos click en el nuevo directorio img y seleccionamos la opción Importar..., nos aparece una nueva ventana para buscar el archivo del logo que importaremos.

Cuando hayamos seleccionado el archivo del logo ya podemos modificar las propiedades del objeto WebImage:

WebImage	
Propiedad	Valor
Name	WebImgLogo
Height	200px
Image	img/LogoXYZ_100x100.png
Mode	Contain

Nota: colega, para asignar una imagen al objeto WebImage, deberás dar click en el botón de los tres puntos de la propiedad Image, para que te aparezca la ventana de selección de archivos de imagen:



Deberás dar clic en el botón Aceptar para asignar la imagen.

5. Dentro del contenedor WebCLogo, debajo del WebImage ahora agregamos un contenedor WebHBox, no es necesario modificar sus propiedades.
6. Dentro del WebHBox anterior agregamos tres objetos en el siguiente orden: un WebContainer, un WebHtml y otro WebContainer, modificamos sus propiedades:

Objeto	Propiedad	Valor
WebContainer	Expand	True
WebHtml	Font	fantasy,bold
	Foreground	White
	Text	<h1>COMPAÑIA XYZ</h1>
WebContainer	Expand	True

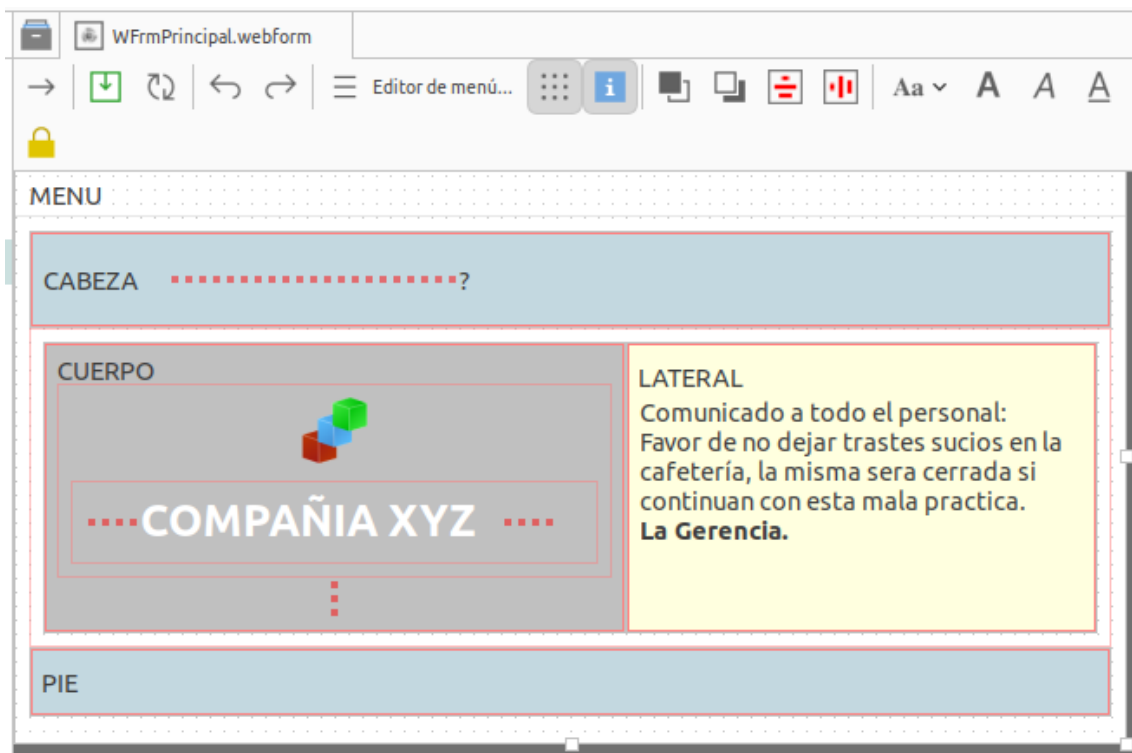
7. Debajo del Contenedor WebCLogo agregamos un contenedor WebContainer y modificamos sus propiedades:

WebContainer	
Propiedad	Valor
Name	WebCHijas
Expand	True
Visible	False

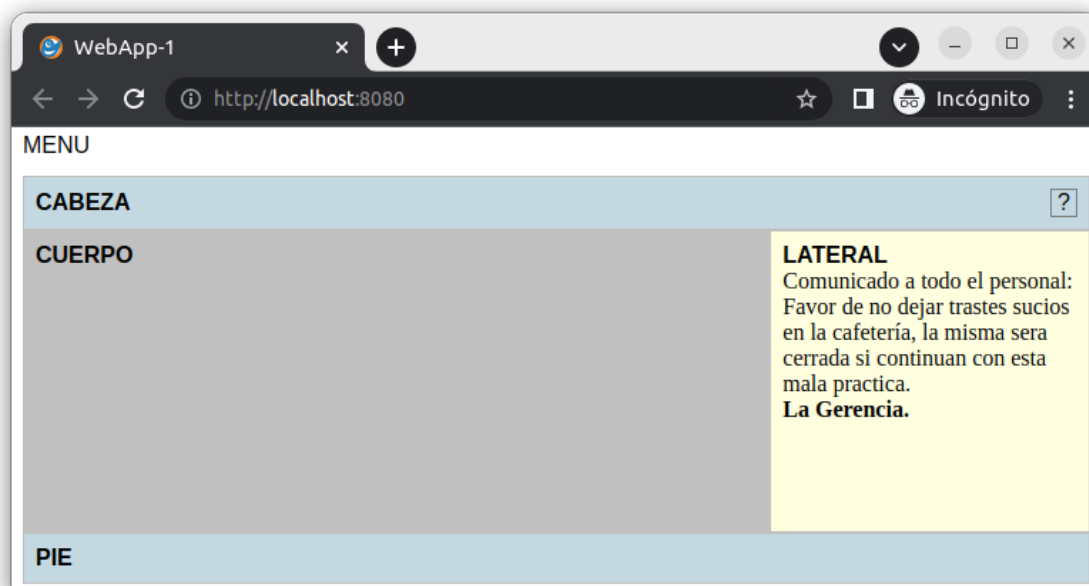
8. En WebContainer (LATERAL) agregamos un objeto WebHtml y modificamos sus propiedades:

WebHtml	
Propiedad	Valor
Name	WebHtmlNotas
Font	cursive
Text	Comunicado a todo el personal:  Favor de no dejar trastes sucios en la cafetería, la misma sera cerrada si continúan con esta mala practica.  <b>La Gerencia.</b>

Finalmente después de todos estos pasos nuestro WebForm WFrPrincipal se debe ver como en la imagen: (supongo que en el logo colocaste una imagen “pifiosa”, disculpa la mía.. :-())



Al ejecutar el proyecto debería verse como en la siguiente imagen:



Nota: colega, como abras observado los objetos logo y nombre de la compañía no se ven porque su contenedor tiene el valor false en su propiedad Visible, esta propiedad la estaremos controlando por programación, eso lo veremos mas adelante.

## WFrmProcesosX, WFrmProcesosY y WFrmProcesosZ

Ahora agregaremos algunos objetos a los WebForm de los Procesos para poder diferenciarlos en la ejecución:

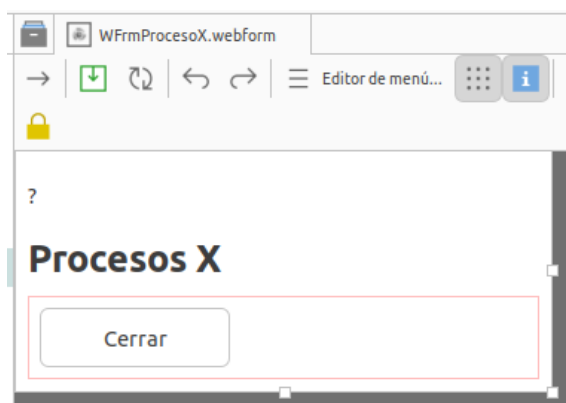
Abrimos el WebForm WFrmProcesosX y editamos las propiedades:

Objeto	Propiedad	Valor
WebForm	Arrangement	Vertical
	BackGround	White
	Height	100%
	Margin	True
	Spacing	True

Agregaremos los objetos: un WebLabel, un WebHtml, un WebHBox y dentro del WebHBox agregaremos un WebButton, editamos las propiedades de todos los objetos:

Objeto	Propiedad	Valor
WebLblUsuario	Text	?
WebHtml	Text	<h1>Procesos X</h1>
WebButton	Name	WebBtnCerrar
	Text	Cerrar

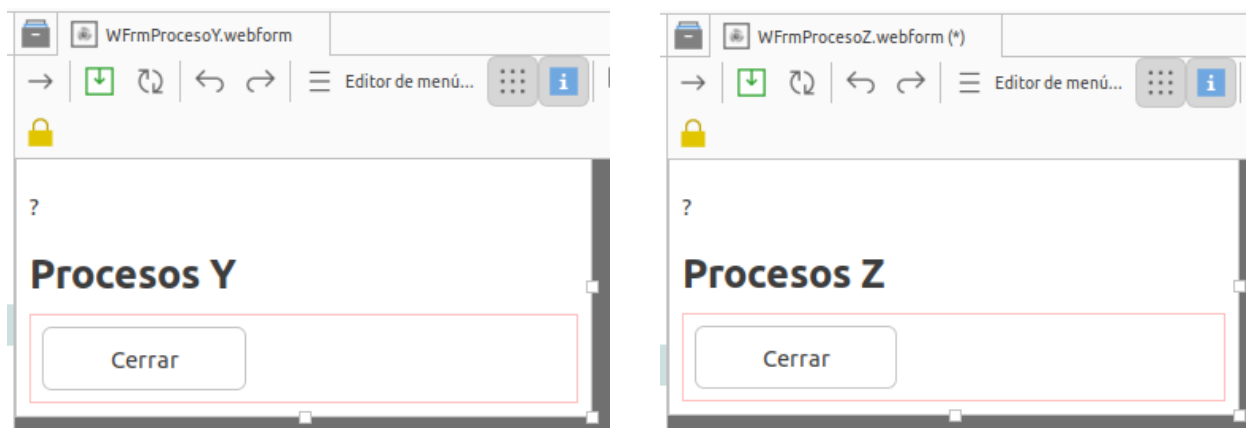
El WebForm deberá verse como en la imagen:



Repetimos estos pasos para los WebForm **WFrmProcesosY** y **WFrmProcesosZ**, con la diferencia de que el objeto WebHtml debe tener en su propiedad Text la letra que hace referencia a su Nombre.

Nota: colega, un súper truco que te permite el IDE es seleccionar todos los objetos del WebForm WFrmProcesosX para copiarlos y luego pegarlos a los otros WebForm y de este modo solo tienes que editar la propiedad Text del objeto WebHtml (Y, Z) . (Tranquilo esto no es trampa... solo rapidez... jeje)

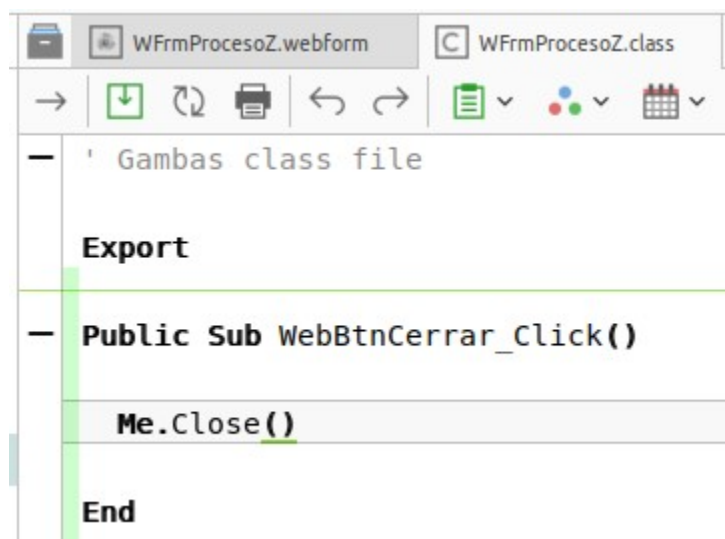
Los WebForm te deben quedar como en la siguientes imágenes:



**Nota:** colega, puedes adelantarte y dar doble click en los botones Cerrar de cada WebForm de procesos y agregarle el siguiente código:

Me.Close()

Cuando das doble click a un objeto de Gambas el IDE abrirá la ventana de código con el evento que afecta al objeto y entonces puedes agregar el código:



Por ahora este sera el único código que tendrán las ventanas de procesos, después lo cambiaremos según la lógica de presentación de ventanas hijas.

**WFrmLogin:**

Abrimos el WebForm WFrmLogin y editamos las propiedades:

Objeto	Propiedad	Valor
WebForm	Arrangement	Vertical
	Margin	True
	Spacing	True

Ajustamos el ancho y alto del WebForm hasta obtener las siguientes dimensiones:



Ahora agregamos tres contenedores WebHBox y a cada uno modificamos las propiedades:

Objeto	Propiedad	Valor
WebHBox1 WebHBox2 WebHBox3	Margin	True
	Spacing	True

Dentro del WebHbox1 agregamos dos objetos: un WebLabel y un WebTextBox, modificamos sus propiedades:

Objeto	Propiedad	Valor
WebLabel1	Text	Usuario
	Width	120px
WebTextBox1	Name	WebTxtUsuario
	Expand	True

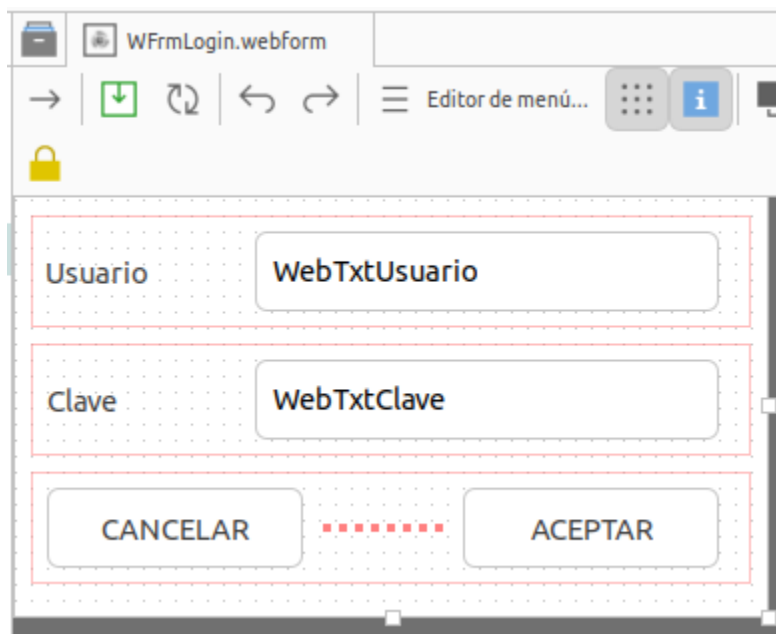
Dentro del WebHbox2 agregamos dos objetos: un WebLabel y un WebTextBox, modificamos sus propiedades:

Objeto	Propiedad	Valor
WebLabel2	Text	Clave
	Width	120px
WebTextBox1	Name	WebTxtClave
	Expand	True
	Password	True

Dentro del WebHbox3 agregamos tres objetos: un WebButton, un WebContainer y un WebButton, modificamos sus propiedades:

Objeto	Propiedad	Valor
WebButton1	Name	WebBtnCancelar
	Text	CANCELAR
Webcontainer1	Expand	True
WebButton2	Name	WebBtnAceptar
	Text	ACEPTAR

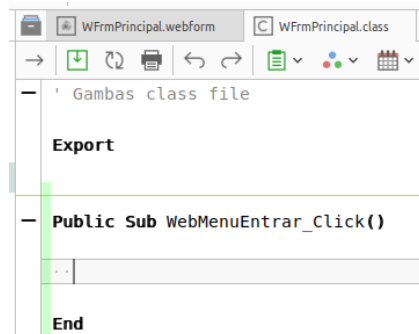
Al finalizar creación de los objetos del WFrmlLogin el diseño debe parecerse al de la imagen:



## **WFrmLogin con ShowModal()**

Según lo solicitado por nuestro cliente la única forma de mostrar los WebForm de los procesos (X,Y y Z) es después de haber ingresado con usuario y contraseña en el WebApp, así que vamos a codificar el elemento ENTRAR del Menú del WebForm WFrmPrincipal.

Abrimos el WebForm WFrmPrincipal y luego damos click el elemento ENTRAR del MENU, de inmediato se nos abre una nueva pestaña con el evento Click del WebMenuEntrar, esta pestaña representa un archivo de clase osea WFrmPrincipal.class:



En la subrutina (Sub) WebMenuEntrar\_Click() es donde invocaremos al WebForm WebFrmLogin, para esto escribimos las líneas de código:

```
WFrmLogin.ShowModal(Me)
```

y luego puede ejecutamos el proyecto y verificamos que al dar click en el elemento ENTRAR se abre la ventana para el Login.

Pero todavía no es funcional nuestro proyecto, solo hemos codificado para probar que se muestra la ventana del Login de una forma Modal: (Ventana padre bloqueada).



Ahora codifiquemos las subrutinas del WFrmlLogin, para esto dar click con el botón derecho del mouse sobre el WebFrmlLogin en el árbol del proyecto y seleccionar abrir código, en la pestaña de la clase agregar estas subrutinas:

```
Public Sub WebForm_Open()
    WebTxtUsuario.SetFocus()
End

Public Sub WebBtnCancelar_Click()
    Me.Close("")
End

Public Sub WebBtnAceptar_Click()
    ,
    If Not Trim(WebTxtUsuario.Text) Then
        Message.Warning("Usuario Invalido", "OK")
        WebTxtUsuario.SetFocus()
        Return
    Endif
    ,
    If Not Trim(WebTxtClave.Text) Then
        Message.Warning("Clave Invalida", "OK")
        WebTxtClave.SetFocus()
        Return
    Else If Trim(WebTxtClave.Text) <> "123"
        Message.Warning("Datos Invalidos...", "OK")
        Return
    Endif
    ,
    Select Case LCase(Trim(WebTxtUsuario.Text))
        Case "usuario1"
            Me.Close("1")
        Case "usuario2"
            Me.Close("2")
        Case "usuario3"
            Me.Close("3")
        Case Else
            Message.Warning("Usuario No Existe", "OK")
    End Select
    ,
End
```

Con el código de arriba la ventana de Login retornara los datos que espera la ventana principal y esta invocara la ventana de procesos que le corresponde a cada usuario.

Colega por ahora todo bien verdad..., como te comente anteriormente tenemos que elegir cual sera la forma de presentar las ventanas, ya que Gambas realizara procesos diferentes, según el código que se utilice para mostrar las ventanas hijas, por ahora las que tenemos son tres que son:

- ✓ Con la propiedad **Current**.
- ✓ Con la function **Show**.
- ✓ Con la palabra clave **New** para crear instancias de Objetos.

### **Invocando ventanas hijas con la propiedad Current:**

Esta propiedad devuelve o establece el formulario actual, es decir, el formulario principal de la aplicación que llena la página web.

Si el formulario actual cambia, la página web se vuelve a cargar automáticamente.

Si utilizamos esta propiedad, la ventana principal (padre) desaparecerá y la ventana de proceso (hija) quedara como ventana principal, en este caso tendríamos que colocar los objetos en cada ventana (hija) que sea invocada, ademas los objetos tales como el usuario o todos aquellos que tenga la pagina principal (padre) que se desean que se vean siempre en las hijas del WebApp.

Cuando se cierra la ventana hijas, aparece la ventana padre pero todos los objetos serán recargados y las variables incluso las de Session se limpiaran.

Parece un escenario funcional, pero no tendría sentido colocar en la ventana principal datos de usuario si al retornar de las paginas hijas ya no hay datos de usuarios, por ahora los hemos colocado ya que mas adelante serán usados por los otros escenarios.

### **Invocando ventanas hijas con la function Show():**

De esta forma las ventanas hijas no ocultaran la ventana padre, aparecen de forma similar a las ventas invocadas con la function ShowModal() usada para la ventana del login, la diferencia es que Show no bloquea la ventana padre, solo ocultaran al padre si su tamaño es igual al del Padre.

Según como configuremos sus propiedades, las ventanas hijas pueden ser reubicadas, se le pueden cambiar sus dimensiones, etc.

Con este escenario los objetos el padre conservan sus valores, parece funcional, pero con la incomodidad de que debemos estar ajustando el tamaño y la posición de la ventanas hijas.

### **Invocando ventanas hijas con la palabra clave New:**

Creación de instancias de formularios.

“Ahora viene la salsa...” jejeje... digo esto por que me parece que es el mejor escenario... creo...

En este escenario las ventanas hijas:

- ✓ No ocultan las ventanas padres
- ✓ No provocan recarga el WebApp, evitando perdida de dato en el padre (sin Close).
- ✓ Se incrustan en un objeto contenedor del padre, en el lugar que queremos que aparezcan.
- ✓ Se deben eliminar desde el padre, sin usar comando Close desde ellas mismas.
- ✓ El único inconveniente es que las ventanas hijas al cargarse no se activa su evento \_Open, pero esto lo remediamos por programación.

**Los códigos:**

- ✓ La ventana de Login WFrmlLogin el código actual de esta se queda como esta.
- ✓ Las Ventanas de Procesos (X,Y,Z) por ahora se quedan con la línea para cerrar Me.Close.

En la venta padre WFrmlPrincipal codificaremos lo siguiente su archivo class:

Creamos dos variables privadas:

```
' Gambas class file

Export

Private InvocarCon As String = "New" 'Posibles valores: Current, Show, New
Private WebFormHija As WebForm
```

La variable InvocarCon se utilizara para cambiar a una de las tres dormas de invocar.  
La variable WebFormHija sera utilizada para cargar la venta hija invocada.

Ahora agregamos el evento Open del WebForm y verificamos que un usuario este logueado para mostrar el contenedor del Logo

```
Public Sub WebForm_Open()
    '
    WebCLogo.Visible = IIf(WebLblUsuario.Text <> "?", False, True)
    '
End
```

Agregaremos un subrutina para controlar que elementos del menú se deben mostrar si un usuario esta logueado, Estado de Logueo: Si = True, No = False

```
Private Sub CambiarMenu(Estado As Boolean) "Estado de Logueo: Si = True, No = False
    WebMenuEntrar.Enabled = Not Estado
    WebMenuEntrar.Visible = Not Estado
    WebMenuSalir.Enabled = Estado
    WebMenuSalir.Visible = Estado
    WebMenuProcesar.Enabled = Estado
    WebMenuProcesar.Visible = Estado
End
```

Agregaremos el código al evento Click del elemento de Menú Salir:

```
Public Sub WebMenuSalir_Click()
    Session["Usuario"] = Null
    WebLblUsuario.Text = "?"
    CambiarMenu(False)
    CerrarVentanaHija()
End
```

Agregaremos el código al evento Click del elemento de Menú Procesar:

```
Public Sub WebMenuProcesar_Click() 'Si esta logueado,puede regresar a su proceso.
    LLamarSegunUsuario(WebLblUsuario.Tag)
End
```

Agregaremos el código al evento Click del elemento de Menú Entrar:

```
Public Sub WebMenuEntrar_Click()
    Dim Usuario As String
    Usuario = WFrmLogin.ShowModal(Me)
    LLamarSegunUsuario(Usuario)
End
```

Como la ventana modal del login retorna un valor de usuario, este se utiliza para saber a que ventana que archivo de clase cargaremos a la variable WebFormHija, el usuario a mostrar en el padre.

Agregaremos el código de la Subrutina LLamarSegunUsuario(Usuario As String):

```
Private Sub LLamarSegunUsuario(Usuario As String)
    Select Case Usuario
        Case "1" 'LLamar la ventana de procesos X
            Session["Usuario"] = "Usuario" & Usuario
            WebFormHija = WFrmProcesoX
            LLamarVentanaHija()
        Case "2" 'LLamar la ventana de procesos Y
            Session["Usuario"] = "Usuario" & Usuario
            WebFormHija = WFrmProcesoY
            LLamarVentanaHija()
        Case "3" 'LLamar la ventana de procesos Z
            Session["Usuario"] = "Usuario" & Usuario
            WebFormHija = WFrmProcesoZ
            LLamarVentanaHija()
        Case Else
            Return
    End Select
    WebLblUsuario.Text = Session["Usuario"]
    WebLblUsuario.Tag = Usuario
    CambiarMenu(True)
End
```

Agregaremos el código de la subrutina LlamarVentanaHija(), esta hará uso de la variable InvocarCon para saber de que forma se presentaran las ventanas hijas:

```
Private Sub LLamarVentanaHija()
    ,
    Select Case InvocarCon
        Case "Current"
            WebForm.Current = WebFormHija
        Case "Show"
            WebFormHija.Show()
        Case "New"
            ,
            CerrarVentanaHija()
            ,
            Select Case WebFormHija.Name
                Case WFrmpProcesoX.Name
                    WebFormHija = New WFrmpProcesoX(WebCHijas)
                Case WFrmpProcesoY.Name
                    WebFormHija = New WFrmpProcesoY(WebCHijas)
                Case WFrmpProcesoZ.Name
                    WebFormHija = New WFrmpProcesoZ(WebCHijas)
            End Select
            ,
            WebCLogo.Visible = False
            WebCHijas.Visible = True
            ,
    End Select
End
```

Colega, observa que cuando los valores de la variable InvocarCon son ("Current" ó "Show") solo se requiere una sola línea de programación, pero cuando es "New" se requiere varias líneas por lo general:

- ✓ Eliminar cualquier hija que este en el contenedor donde se incrustara la hija invocada.
- ✓ Saber quien es la hija que se esta invocando para crearla como instancia con New, osea que tendrás una estructura **Select Case** con una cantidad de condicionales **Case**, según el numero de hijas que tenga tu proyecto.

**Importante:** Colega, si llegaras a conocer alguna forma de evitar esta estructura Select Case y crear una instancia a partir de una forma que utilice la variable WebFormHija, pues te invito a que aporte a este manual dicha forma, ya que te imaginas una WebApp de por lo menos 100 hijas y creciendo, umm... sera esta la mejor forma... :-(, bueno es lo que hay y de todas formas no afecta mucho... pero 100...200... jooo parece como que muchísimo para una WebApp... tal vez...

Finalmente, agregaremos el código de la subrutina CerrarVentanaHija():

```
Public Sub CerrarVentanaHija()
  If InvocarCon = "New" Then
    WebCHijas.DeleteChildren()
  Else
    If Object.IsValid(WebFormHija) Then
      WebFormHija.Close()
    Endif
  Endif
  ,
  WebCLogo.Visible = True
  WebCHijas.Visible = False
  ,
End
```

Nota: colega, como abras observado, cuando el valor de la variable InvocarCon es "New", las ventanas hijas deben ser eliminadas del contenedor, pero cuando es "Current" ó "Show" se utiliza la function Close().

Un ultimo detalle muy importante:

Recuerdas que las ventanas hijas de los procesos (X, Y, Z), solo tienen codificado el evento Open, pues eso lo dejamos a propósito para realizar pruebas que cambiaran según el escenario:

Escenarios 1:

- ✓ Ejecuta el proyecto con la variable (InvocarCon = "Current").
- ✓ Loguea con un usuario.
- ✓ Observaras la ventana hija, pero no habrá en ella ninguna información de usuario y cuando la cierres observa que tampoco lo abra en el padre.

Escenarios 2:

- ✓ Ejecuta el proyecto con la variable (InvocarCon = "Show").
- ✓ Loguea con un usuario.
- ✓ Observaras la ventana hija, pero no habrá en ella ninguna información de usuario, en la ventana padre permanecen los datos del usuario y cuando la cierres la hija, observa que no se pierden los datos y desde el menú puedes regresar a la hija desde el elemento PROCESOS.

Escenarios 3:

- ✓ Ejecuta el proyecto con la variable (InvocarCon = "New").
- ✓ Loguea con un usuario.
- ✓ Observaras la ventana hija incrustada, pero no habrá en ella ninguna información de usuario, en la ventana padre los datos permanecen, cuando cierres la hija observa que el padre pierde los datos.

Hasta ahora hemos visto que solamente con Show, te logueas y hasta que no decidas salir, puede regresar a la ventana de procesos sin tener que volver a loguarte.

Ahora utilizaremos el valor de la variable de Session Session["Usuario"] para mostrarlo en las ventanas hijas, para esto agrega en cada venta hija el siguientes código:

```
Public Sub WebForm_Open()
    WebLblUsuario.Text = Session["Usuario"]
End
```

Prueba el proyecto cambiando el valor de la variable **InvocarCon**, veras que aparece el usuario en las ventas hija... pero que sorpresa... no aparece el dato del usuario cuando utilizamos el valor **"New"**... umm heee, bueno esto se debe a que el evento Open no se ejecuta cuando las ventanas hijas están incrustadas en el padre... esto lo podemos resolver cambiando el evento \_Open por el \_New, realizar el siguiente cambio en todas las ventas hijas:

```
Public Sub _new()
    WebLblUsuario.Text = Session["Usuario"]
End

' Public Sub WebForm_Open()
'     WebLblUsuario.Text = Session["Usuario"]
' End
```

Hemos agregado un apostrofe al código del evento \_Open con esto se desactiva y no se ejecutara, ademas agregamos el método \_new() para reemplazar el evento Open.

Probamos y listo ahora si todas las ventas hijas tienen el dato de usuario...

Pero falta un ultimo detalle para el escenario 3 ( InvocarCon = "New"), como mencionamos anteriormente, cuando cierras la ventana hija con el comando **Me.Close()**, observa que el padre pierde los datos por que se da una recarga, pues para resolver este problema, solo hay que cambiar una linea en el evento Click del botón **WebBtnCerrar**:

```
Public Sub WebBtnCerrar_Click()
    'Me.Close()
    WFrmlPrincipal.CerrarVentanaHija()
End
```

Con este cambio hacemos que las hijas le digan al padre que las cierre según el escenario elegido y con esto evitamos que se de un Close() cuando las hijas estas incrustadas.

Con estas pruebas puedes elegir cual seria el escenario que se ajusta a una determinada situación en tus WebApp.

Espero que te sirva este manual y te invito a corregir cualquier error que encuentres o si existe alguna mejor forma de realizar las WebApp con WebForm Application de Gambas.

Saludos.

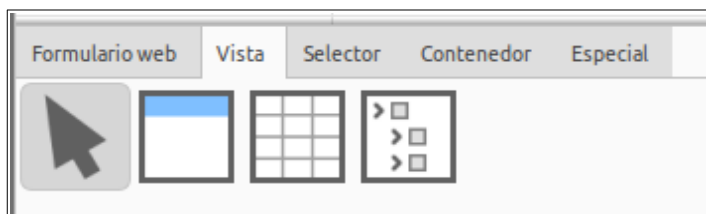
## Pendientes:

A la fecha de creación de este manual esto son los objetos que forman parte de un WebForm de Gambas:

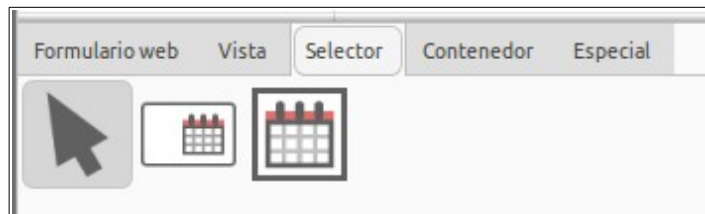
Los objetos están organizados en Tab o pestañas, de izquierda a derecha y de arriba a abajo tenemos:



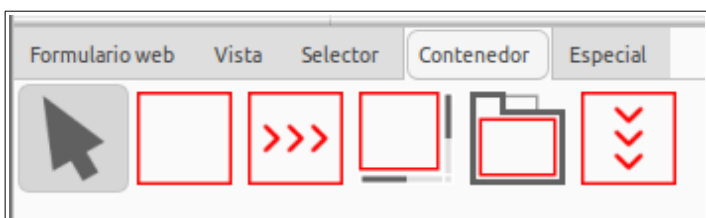
WebButton, WebCheckBox, WebComboBox, WebDrawingArea, WebFileButton, WebHtml, WebImage, WebLabel, WebProgressBar, WebRadioButton, WebSeparator, WebSlider, WebSpinBox, WebTextArea, WebTextBox



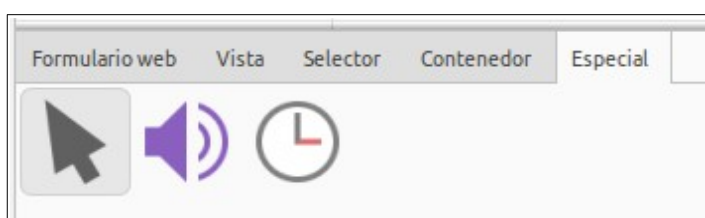
WebListBox, WebTable, WebTree



WebDateBox, WebDateChooser



WebContainer, WebHBox, WebScrollView, WebTabPanel, WebVBox



WebAudio, WebTimer

Lo siguiente sera como explicar el uso de estos objetos, la mayoría son intuitivos, pero los que podrían requerir un poco mas de explicación son los WebTable, WebTree y creo que el WebComboBox que al no tener una propiedad de valores por cada elemento, entonces utilizo su propiedad Tag.

Otros temas:

- ✓ Uso de archivos javascript, archivos CSS
- ✓ Conexiones a bases de datos.
- ✓ Configuración de servidores que soporten cgi-bin.
- ✓ Continuara ....

Nos animamos...?

