Gambas: Basic para Linux

by JonÃ¡s Alvarez
<jalvarez(en)eitb.com>

About the author:

JonÃ¡s Alvarez ha trabajado como programador de aplicaciones en entornos UNIX y Windows durante varios años. Entre otras cosas, ha dado varios cursos de Sistemas Operativos, redes y programación.

Abstract:

Gambas es uno de los Basic disponibles hoy en día para Linux. En este artículo desarrollamos un ejemplo en el que vemos su simplicidad y potencia para tareas cotidianas.

Introducción

Uno de los lenguajes de programación más extendidos y fáciles de usar, sobre todo para los principiantes, es el Basic. Hasta ahora el entorno más habitual para el desarrollo de aplicaciones Basic era el IDE del Visual Basic de Microsoft. Últimamente se está extendiendo el uso de Linux hasta el escritorio del usuario. De estar limitado a servidores y ser manejado únicamente por gurús, está pasando a ser el sistema operativo que encontramos en los ordenadores de los clientes, dando respuesta a necesidades tales como poder leer el correo electrónico, navegar por la web y editar textos. En esa tendencia nos encontramos varios entornos de desarrollo. El que vamos a tratar en este artículo es Gambas, un entorno gráfico de desarrollo en Basic. Con un estilo de programación muy parecido a Visual Basic, como veremos más adelante, tiene también sus diferencias. La versión que voy a emplear es la 0.64a, incluida en mi distribución SuSE 9.0. En el momento de escribir estas líneas en la página del proyecto van por la 0.81, aunque esto no debe afectar a lo que sigue.

A quién le puede interesar

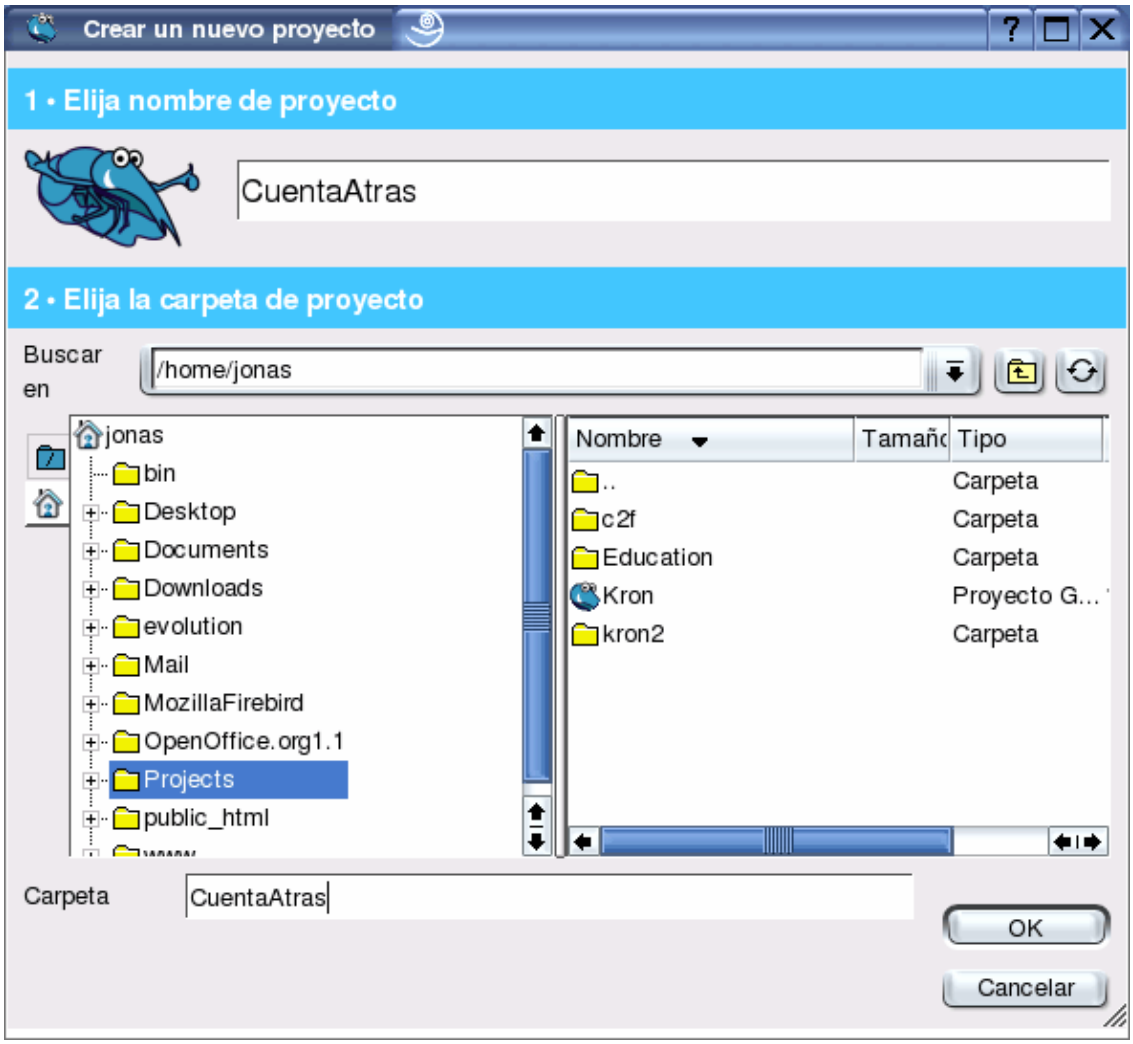
Como programador durante un tiempo de aplicaciones en Visual Basic, no he necesitado más que ponerme a ello para desarrollar este ejemplo. Además es la primera vez que toco Gambas, lo que demuestra que para cualquiera que haya usado alguna vez VB es más que suficiente. Para los demás, puede valer como ejemplo de lo sencillo que es el Basic para muchas cosas.

El ejemplo

Como el movimiento se demuestra andando, vamos a verlo con un ejemplo. Se trata de una aplicación sencillita que tiene un cronómetro con una cuenta atrás en pantalla. Podemos cambiar el tiempo del crono, detenerlo y arrancarlo cuando queramos. Al meollo.

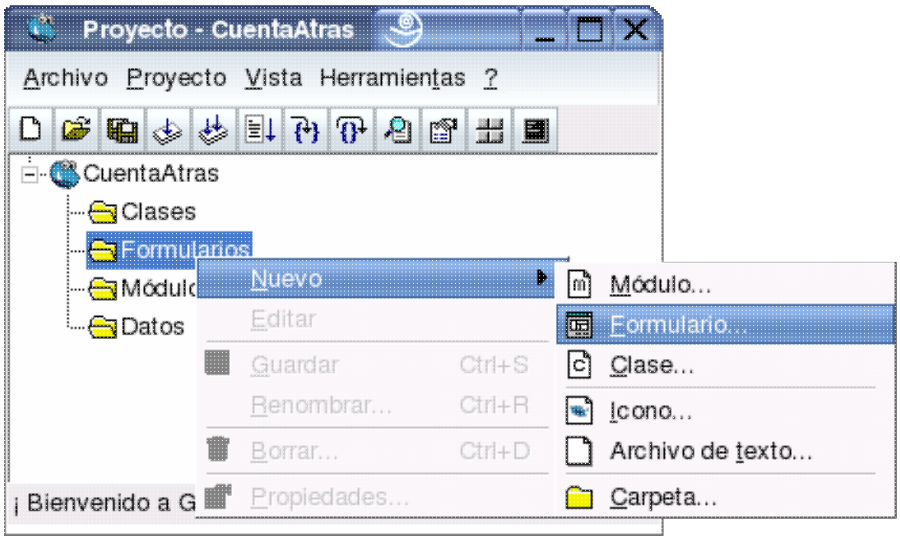
Nada más abrir Gambas nos encontramos con su asistente:

Elegimos **Nuevo Proyecto**. En la siguiente ventana, en el **Paso 1** nos pide **Nombre del Proyecto**. Nuestro proyecto se llamará *CuentaAtras*. En el **Paso 2** debemos elegir la **Carpeta del Proyecto**. Seleccionamos nuestro directorio de trabajo, y en la caja de edición de la parte inferior, escribimos el nombre de la carpeta que vamos a crear.

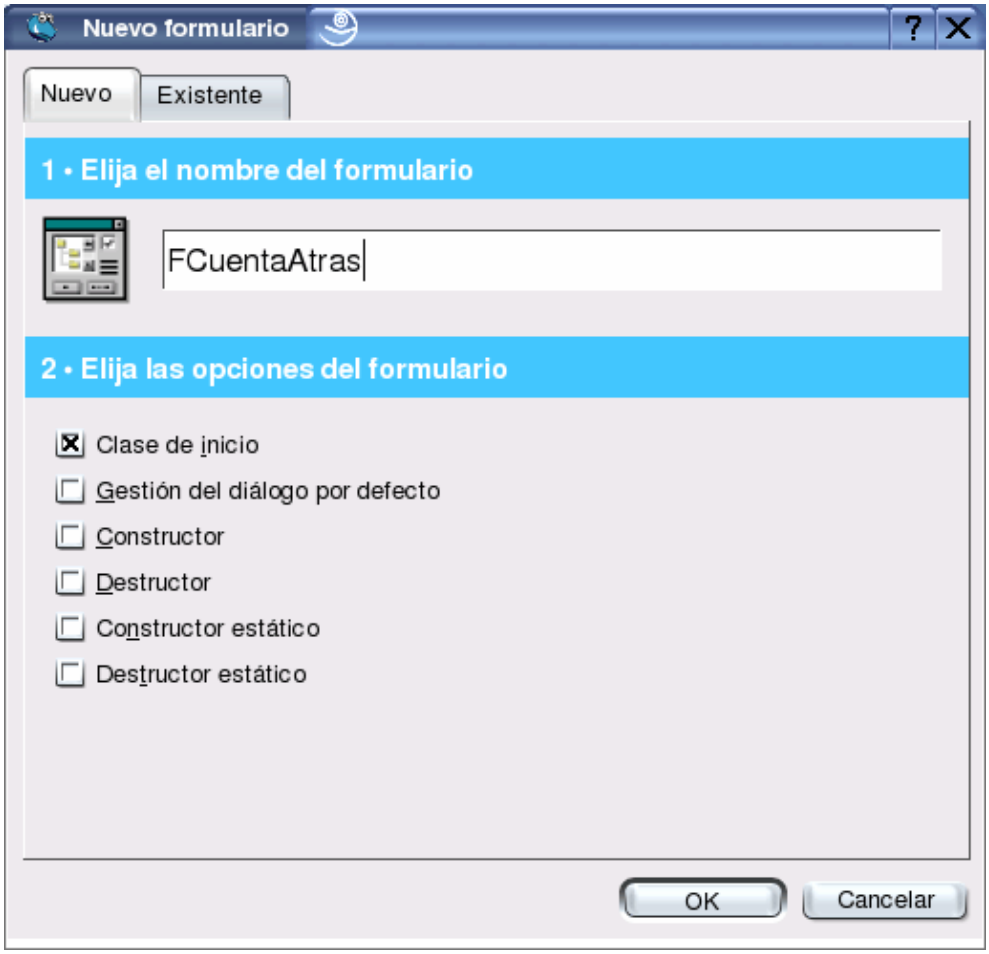


Si es la primera vez que abrimos Gambas o no hemos desactivado la opción, veremos los Consejos del día. Leemos lo que nos interese y cerramos su ventana. Ya estamos en el entorno dispuestos a empezar a trabajar. Vemos varias ventanas en nuestro escritorio. Si estamos en el entorno gráfico KDE, con varios escritorios, seguramente nos interesará dedicar uno de estos a Gambas y así tendremos todas sus ventanas controladas. Personalmente una de las primeras opciones que suelo activar en KDE, es que en cada escritorio sólo me aparezcan sus iconos.

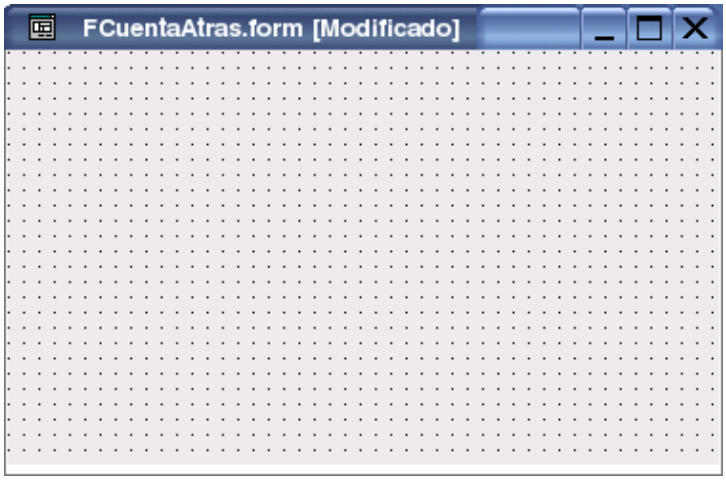
Vamos a crear el formulario principal de la aplicación. Para ello, en cualquier parte de la ventana del Proyecto hacemos clic con en botón derecho del ratón y creamos un formulario nuevo.



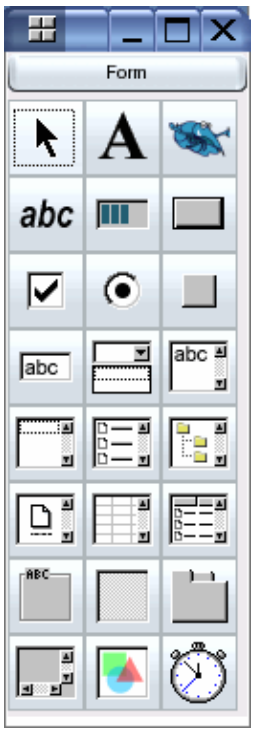
En el diálogo posterior le indicamos el nombre del formulario, en este caso *FCuentaAtras*, con todos los valores por defecto.



Ya tenemos nuestro primer formulario, que por ahora está vacio.



Aquí vamos a incluir los controles que va a tener nuestro cronómetro. Hacemos clic en los elementos de la barra de herramientas que incluiremos en nuestro formulario. Si pasamos el ratón por encima de cada control veremos su nombre. Con doble clic se situará al control en la parte superior izquierda del formulario. Con un sólo clic pondremos el control, cambiándolo de tamaño, en la parte del formulario que queramos. Para nuestro programa vamos a necesitar un Label, un TextBox, un Timer, dos Button y un ToggleButton.



Una vez situados todos los controles nos debe quedar algo parecido a esto (más o menos, cada uno lo puede poner como le dé la gana):

Una vez que tenemos los controles en nuestro formulario les cambiamos los nombres por algo que tenga lÃ³gica para nosotros. Para ello editamos la propiedad **Name** de la **Hoja de Propiedades**. Si no vemos la **Hoja de Propiedades** en pantalla podemos activarla desde la ventana del proyecto en el botón al efecto. Para buscarlo pasamos el ratón por encima de los botones y así localizaremos el que buscamos.

Al control *Label1* le llamo *lblContador:* hago un clic en el control y a continuación puedo cambiar su nombre en la **Hoja de Propiedades**. Para ello edito la propiedad **Name** y le pongo como valor *lblContador.* Después cambio su tipo de letra por otro más grande y visible ya que va a visualizar el cronómetro en esta cuenta descendente. Para ello, en el botón ... de su propiedad **Font** elijo el Tipo de Letra *Courier Bold de 72* y Acepto (**OK**). De la misma manera, al *ToggleButton1* le cambio el nombre por *tglFuncionando*. Al control *TextBox1* le llamo *txtSegundos*, al control *Timer1* le llamo *clkMiReloj*, a *Button1 cmdPonerSegundos* y por último renombro *Button2* por *cmdSalir*. Además cambio el **Alignment** de *txtSegundos* por *Right*.

Y empezamos con el código Basic. Es muy sencillo y no es muy estricto en lo que se refiere a la sintaxis. Lo primero que vamos a hacer es cambiar los textos que vemos en el formulario por valores más reales. Aunque muchas de las opciones se están cambiando desde Basic, también lo podríamos haber hecho en la misma hoja de propiedades de cada control, con cualquiera de las opciones obtendríamos el mismo efecto.

Nada mas abrirse el formulario rellenamos los títulos que queremos que tenga cada control. Al decir *nada mas abrirse el formulario* estamos hablando de manejar un evento: la apertura del formulario. Para ello hacemos doble clic en una parte de nuestro formulario que no tenga ningún control. Se abre una ventana de edición y se sitúa el cursor dentro de un procedimiento nuevo: **Public Sub Form_Open()** (si hemos programado anteriormente en Visual Basic, hubiéramos empleado el evento Form_Load). Vamos a hacer que el control *lblContador* presente los segundos restantes de la cuenta atrás. Las primeras líneas de código de la clase formulario en la que nos encontramos quedan de esta manera:

```
' Gambas class file
CONST fSegundosPorDefecto AS Float=120.0
fSegundos AS Float

PRIVATE SUB VerValores()
  DIM nMinutos AS Integer

  nMinutos = Int(Int(fSegundos) / 60)
  lblContador.Caption = nMinutos & ":" & Format (fSegundos - nMinutos * 60, "00.0")
END

PRIVATE SUB VerActivarDesactivar()
  IF tglFuncionando.Value THEN
    tglFuncionando.Text = ("&Detener")
```

```
  ELSE
    tglFuncionando.Text = ("&Arrancar")
  ENDIF
END

PUBLIC SUB Form_Open()
  fSegundos = fSegundosPorDefecto
  VerValores
  tglFuncionando.Value = FALSE
  VerActivarDesactivar
  txtSegundos.Text = fSegundos
  cmdPonerSegundos.Text = ("&Reiniciar")
  cmdSalir.Text = ("&Salir")
END
```

Hemos añadido después del comentario que nos había generado Gambas, ' *Gambas class file*, una constante que contiene el número de segundos por defecto de la cuenta atrás, *fSegundosPorDefecto*, con un valor de 120 segundos (dos minutos), y una variable, *fSegundos* que va a contener la cuenta atrás. Hemos creado también dos procedimientos: *VerValores*, que visualiza los valores de la cuenta atrás, y *VerActivarDesactivar*, que cambia el texto del botón de Marcha/Paro.

En este momento ya tenemos un formulario que funciona. No hace nada útil, a parte de hacernos entender lo que hemos hecho hasta ahora, por lo que vale la pena que lo probemos. Guardamos los cambios a todo desde la ventana principal del proyecto, *Proyecto – CuentaAtras*, y arrancamos la aplicación con **F5**, o con el botón ***Ejecutar*** de la barra de botones de la misma ventana. Esto es lo que debemos ver:

Si no se parece a esto o nos da cualquier error debemos revisar lo que hemos hecho hasta ahora. Aunque pulsemos **Arrancar**, **Reiniciar** o **Salir** no ocurre nada. Esa será nuestra siguiente tarea: asignar eventos a estos botones para que cuando el usuario pulse cualquiera de ellos esto se mueva. Antes de seguir jugamos un poco con nuestra aplicación y descubrimos todo lo que ya contiene. Para cerrarla pulsamos en la X de la parte superior derecha. Yo estoy con KDE, con el tema de SuSE, como habrás podido comprobar en los formularios y puede ser que en tu caso tengas otra manera de cerrar una ventana.

Vamos a por el más fácil de los botones: ¿quÃ© debe pasar cuando el usuario pulse **Salir**? Debemos cerrar la aplicación. Para introducir el código Basic que se ejecutará cuando el usuario pulse ese botón, hacemos doble clic en el botón con el texto **Salir** (*cmbSalir*). Vemos que Gambas nos genera unas líneas de código y que sitúa el cursor entre ellas. Aquí es donde hay que rellenar con código. Este procedimiento se ejecutará cuando el usuario haga clic en este botón. Para cerrar la aplicación debemos ejecutar Me.Close, con lo que el código de este evento es:

```
PUBLIC SUB cmdSalir_Click()
  ME.Close
END
```

El siguiente botón que vamos a controlar es el de **Reiniciar**. De la misma manera: hacemos doble clic en el botón y en la ventana de código que nos presenta Gambas rellenamos:

```
PUBLIC SUB cmdPonerSegundos_Click()
  fSegundos = txtSegundos.Text
  VerValores
END
```

Pero al grano, todavía parece que no pasa nada. Hay que darle acción a nuestra aplicación. Vamos a activar el objeto *Timer* que tenemos situado en el formulario desde el principio. Para ello tenemos que establecer el intervalo para recibir eventos del reloj. O lo hacemos desde código, en el evento anterior *Form_Open*, o lo ponemos en el formulario. Ahora lo haremos de esta manera. En el formulario hacemos un clic en el objeto *Timer* y en su **Hoja de Propiedades** cambiamos su valor **Delay**, que contiene 1000ms, por 100, para recibir un evento cada décima de segundo, que va a ser la precisión de nuestro cronómetro.

Nos falta el código que se ejecutará cada vez que salte el reloj y la manera de activar este último, que todavía está parado. Para generar el código del evento del reloj nada más sencillo, como siempre, que hacer doble clic en el reloj del formulario. Nos lleva a la ventana de código en la posición correcta. Después de introducir nuestro código debe quedarnos así:

```
PUBLIC SUB clkMiReloj_Timer()
  IF fSegundos < 0.1 THEN
    tglFuncionando.Value = FALSE
    tglFuncionando_Click
  ELSE
    fSegundos = fSegundos - 0.1
    VerValores
  END IF
END
```

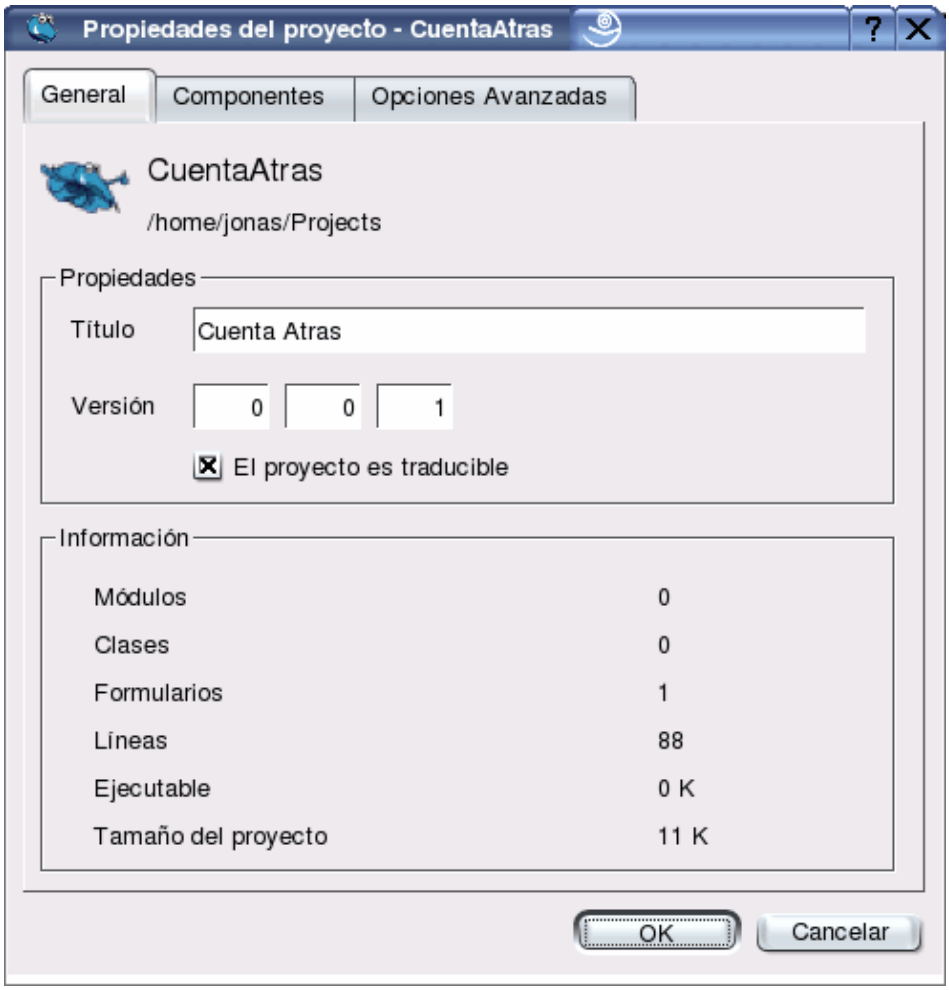
Y finalmente activamos a voluntad del usuario el cronómetro con el botón Toggle, que es el que nos quedaba por manejar. Con doble clic en el botón podemos rellenar el código de su evento:

```
PUBLIC SUB tglFuncionando_Click()
  clkMiReloj.Enabled = tglFuncionando.Value
  VerActivarDesactivar
END
```

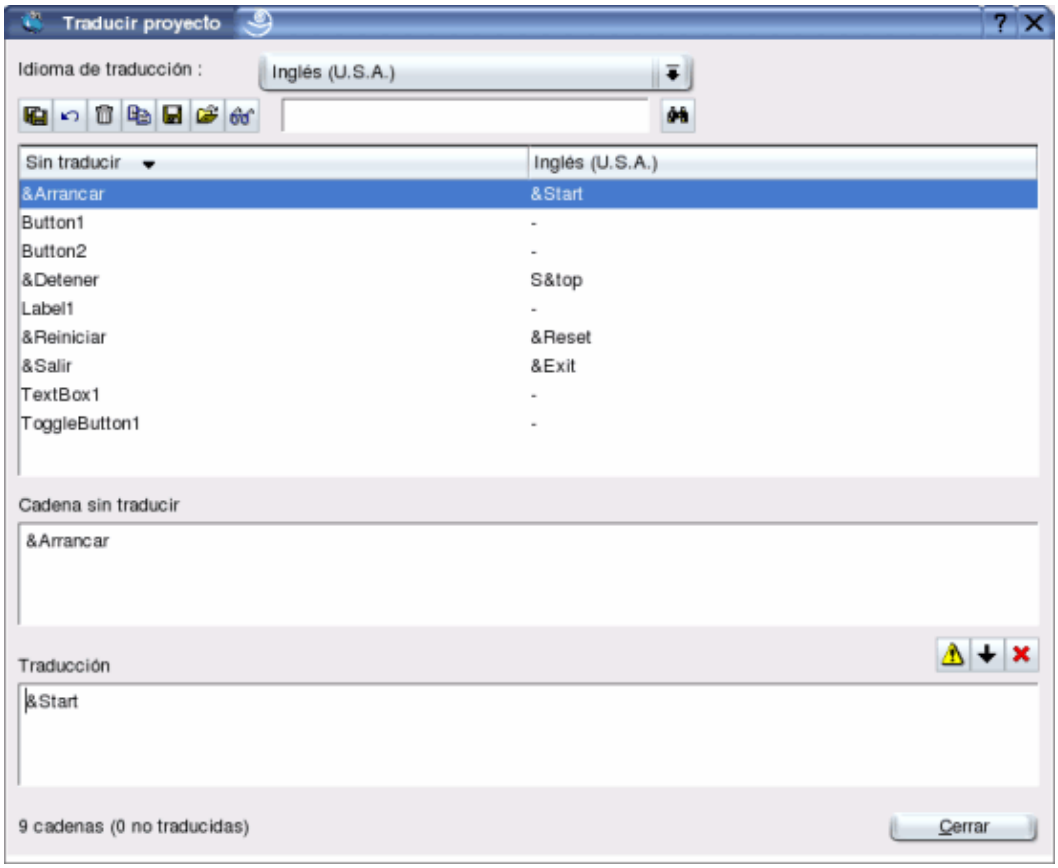
Y ya podemos probar nuestro trabajo, ya hemos terminado.

Y lo mejor: Gambas es multilenguaje, como debe ser

Otra de las capacidades de Gambas en el soporte para mÃºltiples idiomas. Si seguiste el código pudiste ver que las cadenas estaban rodeadas por paréntesis; era para indicar a Gambas que luego iban a ser traducidas. Los textos de los controles de los formularios no necesitan estos paréntesis. Nuestro proyecto se ha convertido en algo muy útil y la gente nos pide que los diálogos aparezcan en su idioma. Nada más sencillo. Vamos al menú **Proyecto / Propiedades** de la ventana del proyecto.



Aquí ponemos un **Título** a nuestro proyecto y activamos la opción **El proyecto es traducible** que nos va a permitir traducir los diálogos. Ahora tenemos activa una nueva opción en los menús: **Proyecto / Traducir**. Si abrimos el diálogo, vemos que ahora es muy intuitiva esta traducción:



Primero elegimos en la parte superior el idioma destino. Cuando queremos traducir una cadena, la seleccionamos y rellenamos la parte inferior. Una vez traducidas todas las cadenas, podemos probarlo arrancando la aplicación desde un terminal si establecemos antes la variable LANG en el idioma al que hemos hecho la traducción. Si quiero ver como me ha quedado la traducción al inglés, cierro gambas y desde una consola ejecuto:

```
$ LANG=en_US; gambas
```

Para volver a la situación anterior arranco Gambas desde el menú del KDE, ya que aquí no tengo esta variable de entorno que sólo vive en su consola.

Conclusión

Aunque es un lenguaje interpretado y necesitamos tener todo el Gambas instalado, es una buena opción para empezar a desarrollar aplicaciones dentro del escritorio de Linux. Como hemos visto es muy sencillo y el desarrollo muy rápido. Para una gran cantidad de aplicaciones del día a día es mas que suficiente.

La ayuda en pantalla es bastante completa, además de los ejemplos de que dispone desde el menú **Archivo/Abrir ejemplo**. También podemos pasarnos por la web del proyecto, que en la sección de enlaces contiene otros proyectos de Basic que también pueden ser interesantes. Esto sólo es el principio de un proyecto al que por mi parte le auguro muy buen porvenir.

2005−01−10, generated by lfparser_pdf version 2.51