

Manual Básico de Gambas2:

Desarrollar un programa “PASO A PASO” programa Listin v.0.0.2 (2010/09)



realizado por Jsbsan



Sevilla, Septiembre de 2010

Índice de contenido

Prólogo.....	4
Instalación e introducción a la programación con Gambas.....	6
Instalación mediante “Añadir y quitar“	6
Instalación mediante comandos de consola.....	7
Conociendo el entorno IDE de Gambas.....	8
Vamos a crear nuestro proyecto: LISTIN.....	8
Explicación del Ide de Gambas2.....	9
gráfica 1.....	9
Antes de nada... programación de Objetos: Propiedades, métodos y eventos.....	10
Propiedades.....	10
Métodos.....	10
Eventos.....	11
Hacer Formularios y Públicos:.....	12
Empezamos: objetivo de nuestro programa Listín.....	13
Definir el entorno visual de nuestro programa: Formulario Fmain.....	14
Organizar nuestro código fuente: Módulos	18
Módulo VAR: Declarar Variables.....	20
Definir el gridview: GridViewDatos.....	22
Introducir Datos: Textos	24
SetFocus.....	24
Introducir Datos: Comprobación de correo electrónico.....	28
Introducir Datos: Imagen.....	29
Botón Aceptar: evento .add().....	29
Módulo Gestión: Borrar datos antes de pulsar Aceptar	33
Módulo Gestión: Editar.....	34
¿como elegimos el registro a editar? Propiedad .Row.....	34
Registro Único: el campo ID.....	34
El botón Aceptar en modo Edición.....	37
Modulo Gestion: Cancelar.....	39
Módulo Archivo: Salvar.....	45
Módulo Archivo: Leer.....	48
Modulo Archivo: Salir.....	51
Módulo Filtro. Buscar y Filtrar InStr.....	57
Módulo Ordenar. Ordenar un gridviews con _ColumnClick.....	61
Módulo Importar y Exportar desde el Portapapeles.....	64
Exportar: Clipboard.Copy.....	64
Importar: Clipboard.Paste.....	67
Formulario y Módulo: Fechas.....	72
Anexos.....	74
Anexo 1: Galería de Fotos del Recuerdo.....	74
Anexo 2: convert, miniaturas en nuestro gridViewDatos.....	77

Instalación:	77
Uso en nuestro programa de convert: SHELL	77
Anexo 3: Introducción al Gambas. Tipo de variables y datos:	82
Declaración de variables locales	82
Declaración de variables locales	82
[DIM] Identificador AS TipoDeDato	82
[STATIC] (PUBLIC PRIVATE) Identificador [Array Declaration] AS [NEW] TipoDeDato	82
Subrutinas o funciones	83
Tipos de datos	83
Determinar que tipo de dato almacena una variable	83
Conversión de tipos de datos	83
Anexo 4: Empaquetado/Desempaquetado de datos	85
Anexo 5: Optimizando nuestro código	88
Anexo 6: Añadir recientes	92
Anexo 7: Índice Alfabético	99

Prólogo

Todo el mundo sabe usar una calculadora....¿por que no un lenguaje de programación?

Hace mucho tiempo, mucho tiempo, aproximadamente el año 1995.....La primera vez que vi el entorno de Visual Basic 3.0¹, acostumbrado a la programación en el Gw-Basic/QBasic del Ms-Dos, aquello me parecía otro mundo, no sabia ni como empezar, con tantas ventanas, botones, objetos, eventos, propiedades y un extraño “Form” en medio de la pantalla.

Me lleve varias semanas viendo ese entorno intentando comprender como funciona, y como hacer mi primer programa... pero no hubo manera...incluso lo abandone y seguí haciendo mis programitas con el Basic del Ms-Dos.

Pero un buen día tuve suerte. Encontré en la biblioteca de informática un libro que estaba dedicado a la programación del Visual Basic 3,0, pero de una manera in habitual, no solo explicaba las propiedades del lenguaje, sino que durante los capítulos del mismo iba explicando como ir haciendo un “pequeño” programa.

El programa consistía en hacer una base de datos para castores... si.. **¡¡castores!!**. Explicaba el proceso de introducir/editar/imprimir/salvar/leer los datos en un archivo de texto plano (tipo .txt), creando los formularios y el código necesario para que funcionara ¡y tener controlados a los castores!.

¿Os parecerá un poco tonto?, pero la verdad es que no, para mi fue muy util, ya que era la primera vez que veía el código de un programa completo **“desde cero y paso a paso”** desarrollado y explicando en cada capítulo las ordenes (y **aplicándolas a mi programa concreto** y no de manera general) y en ese nuevo entorno de programación “visual y de objetos”.

Este libro pretende hacer algo parecido, pero aplicándolo a GAMBAS2, para GNU-LINUX. Espero que os sea tan útil como lo fue para mi aquel libro.

Yo no soy un programador profesional ni he estudiado dicha carrera, disculpen si no soy ortodoxo a la hora de explicar las cosas, pero mi intención es ayudar y explicar a los usuarios “no profesionales”, aficionados a la programación, autodidactas o simplemente que le pique la curiosidad, como se puede hacer un programa, en Gambas2.

1 Puedes ver una captura de imagen del Visual Basic 3.0 y del Gw-Basic en el Anexo 1: Galería de Fotos del Recuerdo

Hoy en día nadie es autodidacta, gracias a Internet con sus foros, blog, paginas, etc, todos podemos aprender de todos.

Y por último, queria dar las gracias a:

Daniel Campos Fernández y José Luis Redrejo, por su estupendo libro.

David Asorey Álvarez (http://davidasorey.net/static/gambas-tutorial/gambas_tutorial_es.html), que sin su pagina me hubiera sido imposible iniciarme en el Gambas.

Al foro www.gambas-es.org

A Jose Miguel Amaya Camacho (<http://inkarri.wordpress.com/>)

A Soplo (el primero que yo conozca que hizo un foro sobre gambas y a sus tutoriales.)

Al autor de la pagina web <http://gambaslinux.wordpress.com/>

Por supuesto al autor del Gambas: Benoît Minisini

y un largo etc..

Bueno y también a mi mujer Eva, que siempre me dice que estoy haciendo el “Gamba”

*Julio Sánchez Berro
Sevilla (España), Julio de 2010*

Instalación e introducción a la programación con Gambas

*Buscamos y descargamos en el Emule, Ares o en el Bittorrent: gambas2+crack
ja,ja,ja... es broma.*

Gambas es un ambiente libre de desarrollo basado en un interprete *Basic* con extensiones de objetos, un poco como *Visual Basic™* (pero **NO** un clon!).

Con **Gambas**, puede diseñar rápidamente su programa IGU con QT o GTK+, acceder a bases de datos MySQL, PostgreSQL, Firebird, ODBC y SQLite, aplicaciones piloto KDE con *DCOP*, traducir su programa a cualquier lenguaje, crear aplicaciones de red fácilmente, hacer aplicaciones 3D OpenGL, hacer aplicaciones web CGI, y así sucesivamente...

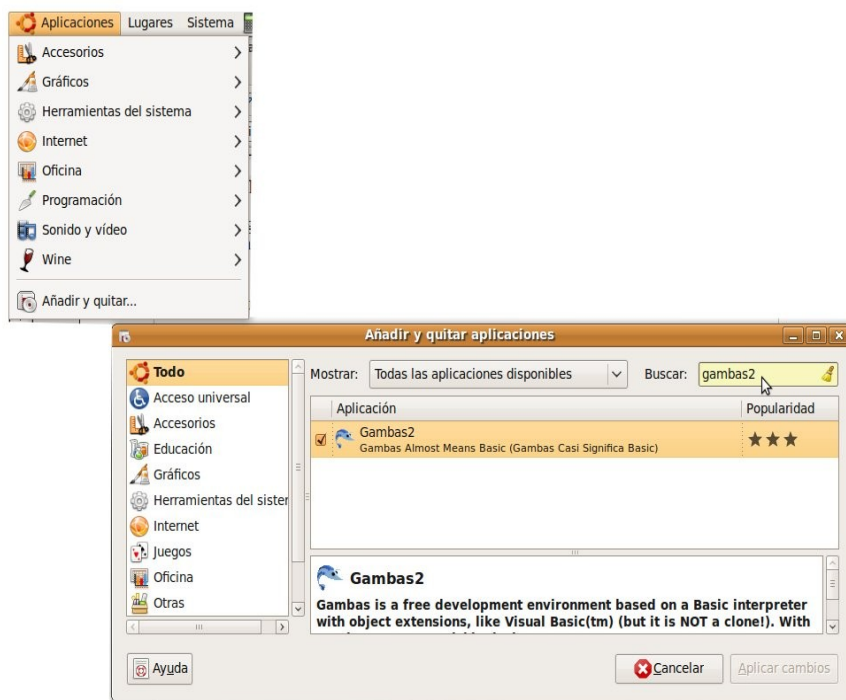
Así lo define su autor **Benoît Minisini** en la página web del proyecto

<http://gambas.sourceforge.net/es/main.html>

El programa es libre y gratuito, pero podemos ayudar al proyecto con donativos mediante [PayPal](#).

Instalación mediante "Añadir y quitar"

Si tienes la distribución Gnu-Linux, Ubuntu con escritorio Gnome, la forma más fácil de instalar el programa es mediante el menú Aplicaciones/Añadir y Quitar



Al dar aceptar se instalará el programa.

Instalación mediante comandos de consola

También puede hacer la instalación desde la línea de comandos de la consola:

1) Instalar los paquetes y librerías necesarios para la compilación (seguramente te pedirá descargar algunos archivos de Internet, pulsa “Y” de sí, cuando te pregunte):

```
sudo aptitude install build-essential g++ automake autoconf bzip2 debhelper dpatch  
firebird2.0-dev gettext kdelibs4-dev libbz2-dev libcurl3-dev libgtk2.0-dev libjpeg62-dev  
libmysqlclient15-dev libpcre3-dev libpng12-dev libpoppler-dev libpq-dev libqt3-compat-  
headers libqt3-mt-dev librsvg2-dev libsdl-gfx1.2-dev libsdl-image1.2-dev libsdl-mixer1.2-dev  
libsdl-sound1.2-dev libsdl1.2-dev libsqlite0-dev libsqlite3-dev libssl-dev libxml2-dev libxtst-  
dev mesa-common-dev unixodbc-dev zlib1g-dev libffi-dev libace-dev libomniorb4-dev
```

2) Creas un directorio (por ejemplo /home/usuario/gambas2):

```
mkdir /home/usuario/gambas2
```

3) Bajar de la página web de gambas <http://gambas.sourceforge.net/> el archivo de código fuente: **gambas2-2.20.2.tar.bz2**

4) Muevelo desde la carpeta en la que se ha descargado (seguramente en el Escritorio se habrá descargado) a la carpeta gambas2:

```
mv /home/usuario/Escritorio/gambas2-2.20.2.tar.bz2 /home/usuario/gambas2/gambas2-  
2.20.2.tar.bz2
```

5) Entra en el directorio gambas2

```
cd /home/usuario/gambas2
```

6) Descomprimirlo en el directorio :

```
tar xvfj gambas2-2.20.2.tar.bz2
```

7) Se creará un nuevo directorio (nos metemos en él) y compilamos:
Primero:

```
cd gambas2-2.20.2
```

ejecuta el siguiente comando:

```
./configure
```

Luego,

```
make
```

Y finalmente:

```
sudo make install
```

De esta manera desde el terminal podemos ejecutar gambas con el comando:

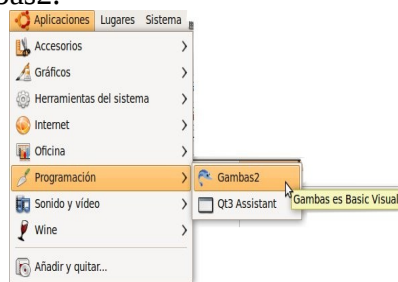
```
gambas2
```

Nota:

Dependiendo del ordenador se puede tardar algo más de una hora en realizar todo el proceso.

Conociendo el entorno IDE de de Gambas

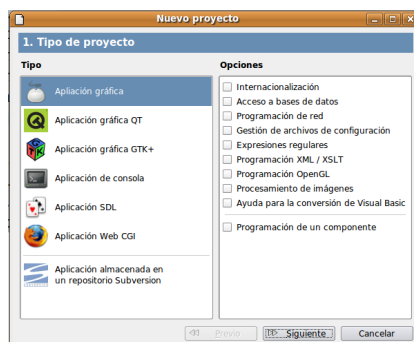
Una vez instalado el programa, lo podremos utilizar, pulsando en el menú Aplicaciones/Programación/Gambas2.



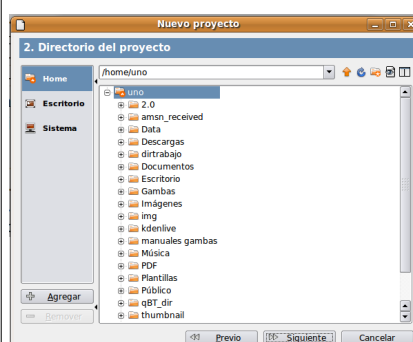
Vamos a crear nuestro proyecto: LISTIN



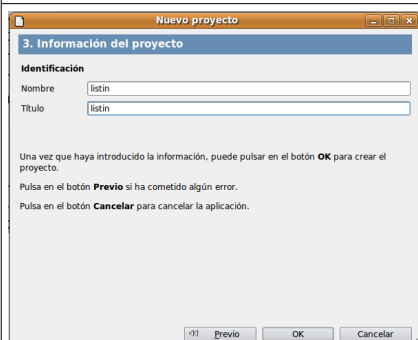
Pantalla Inicial. Pulsamos en Nuevo Proyecto



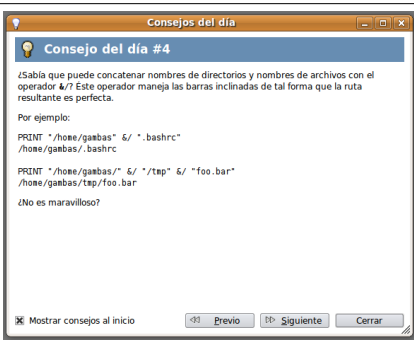
Pulsamos en aplicación grafica, y botón siguiente



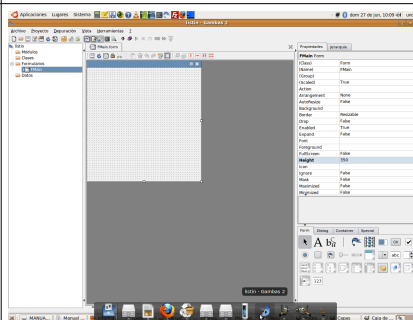
Elegimos la carpeta donde va a estar, y botón siguiente



Escribimos el nombre y titulo del proyecto: Listin

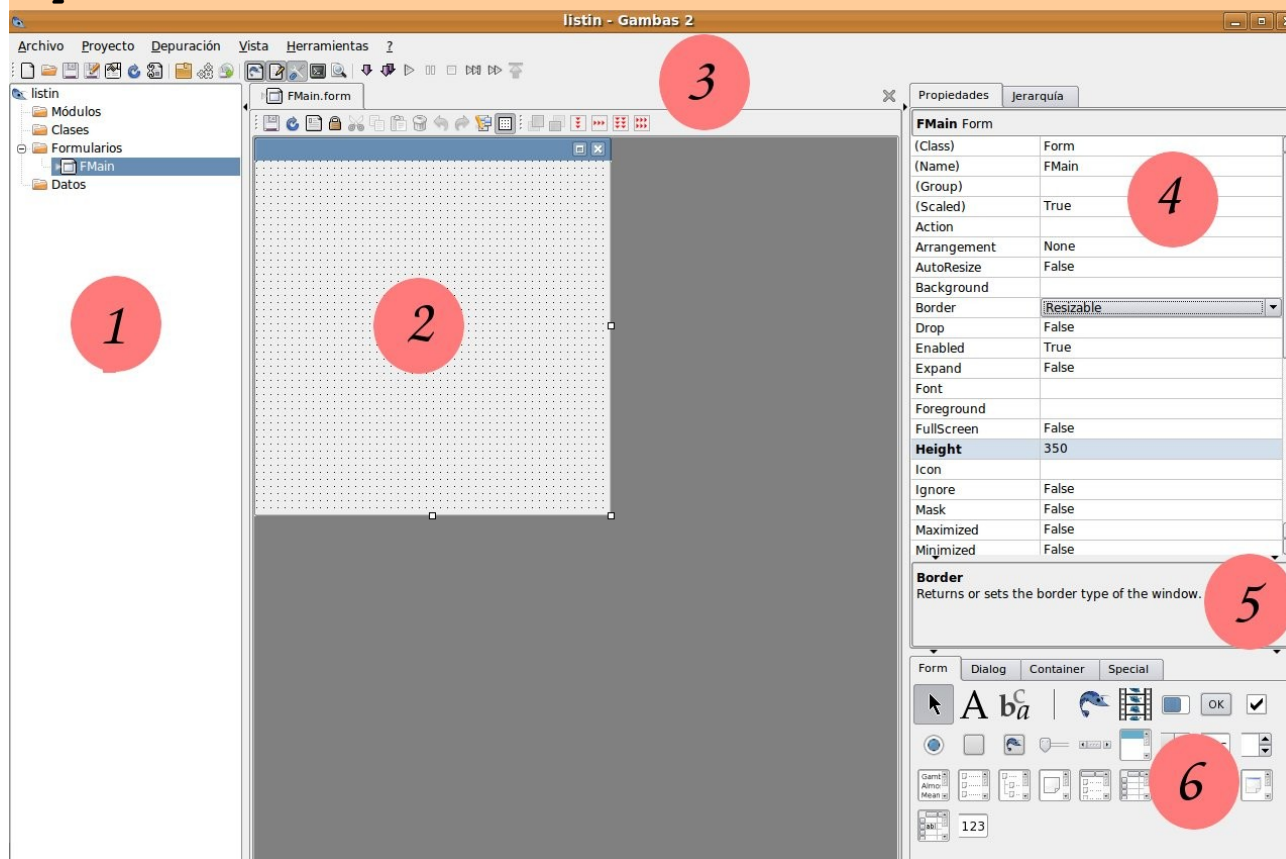


Nos saldrá el consejo diario, pulsamos el botón cerrar



Y finalmente aparece el IDE de gambas2. Pulsando en el Fmain, nos saldra el formulario con sus propiedades.

Explicación del Ide de Gambas2



gráfica 1

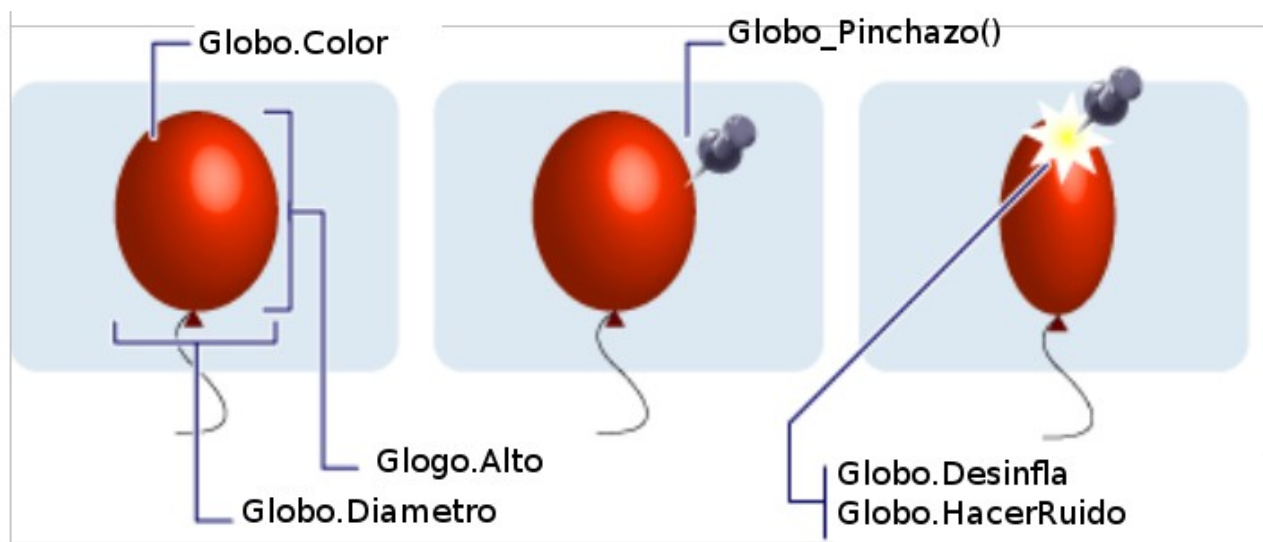
- 1) En este Arbol de organizan los modulos, clases, formularios y datos (iconos, imagenes archivos de datos,etc.)
- 2) Muestra el Formulario que tenemos activado (Fmain, es este caso)
- 3) Menus y botones para diversas acciones a realizar con nuestro proyecto (ejecución, depuración, etc).
- 4) Propiedades del formulario que tengamos elegido en 2 y jerarquia de los elementos del formulario (botones,textbox,etc), para el orden que llevarán al pulsar la tecla "tab"
- 5) Ayuda de la propiedad que tengamos elegida en 4
- 6) Componentes: botones, label, textbox, gridviews, etc.

Para mas detalles véase en el anexo, manuales Existentes de Gambas.

Antes de nada... programación de Objetos: Propiedades, métodos y eventos

Gambas es un lenguaje orientado a objetos, es decir, tenemos unos “objetos” (botones, textbox, formularios, etc), donde el usuario de nuestro programa va a interactuar con ellos (haciendo click, por ejemplo).

Explicamos con un sencillo ejemplo los conceptos de la programación orientada a objetos.



Un globo tiene propiedades (Color, Alto y Diámetro), responde a eventos (Pinchazo) y puede ejecutar métodos (Desinfla, HacerRuido).

Propiedades

Si pudiera programar un globo, el código de podría parecerse al siguiente "código" que establece las propiedades de un globo.

```
Globo.Color = Color.Red  
Globo.Diametro = 10  
Globo.Inflado = True
```

Observe el orden del código: el objeto (Globo) seguido por la propiedad (Color) seguida por la asignación del valor (= Color.Red). Puede cambiar el color del globo sustituyendo un valor diferente.

Métodos

Los métodos de un globo se *denominan* de este modo.

```
Globo.Inflar  
Globo.Desinflar  
Globo.HacerRuido(5)
```

El orden es parecido al de una propiedad: el objeto (un nombre), seguido por el método (un verbo). En el tercer método, hay un elemento adicional, llamado *argumento*, que especifica la distancia a que se elevará el globo. Algunos métodos tendrán uno o más argumentos para describir aún más la

acción que se va a realizar.

Eventos

El globo podría responder a un evento de la siguiente manera.

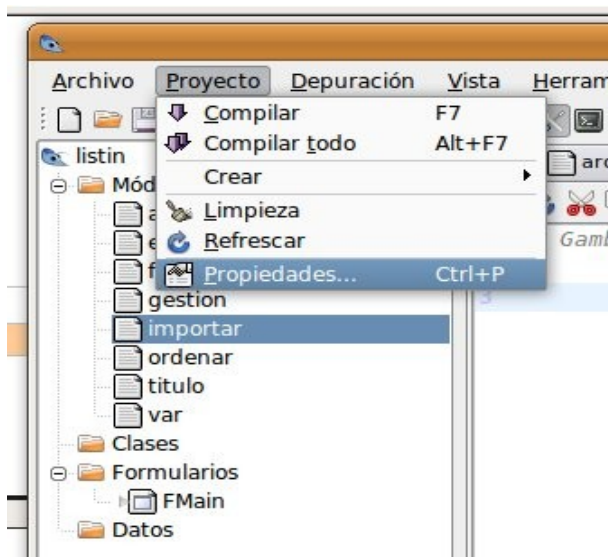
```
Sub Globo_Pinchazo()  
    Globo.HacerRuido("Bang")  
    Globo.Desinfla  
    Globo.inflado = False  
End Sub
```

En este caso, el código describe el comportamiento del globo cuando se produce un evento Pinchazo: llama al método HacerRuido con un argumento "Bang", (el tipo de ruido a realizar), luego llama al método Desinfla. Puesto que el globo ya no está inflado, la propiedad inflado se establece en **False**.

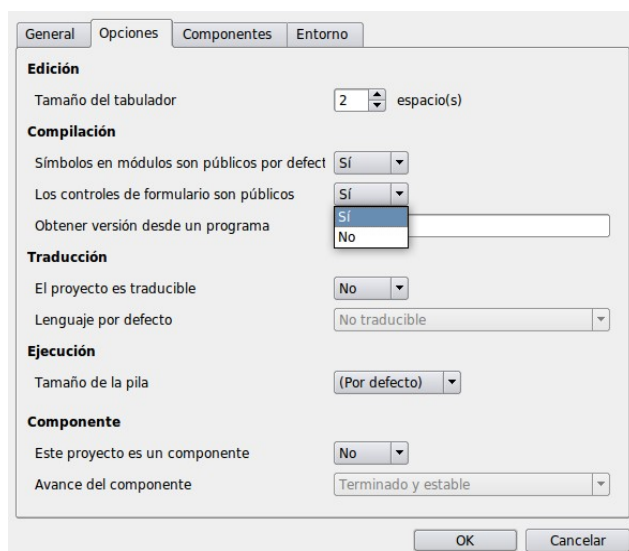
Si bien, en realidad no puede programar un globo, puede programar un formulario o control. Como programador, es el responsable. Decida las propiedades que se deben cambiar, los métodos que se deben invocar o los eventos que se deben responder para lograr la apariencia y el comportamiento deseados.

Hacer Formularios y Públicos:

Para poder acceder a los formularios desde los módulos hay que hacer definirlos como públicos
Para ello hacemos:



En el menu Proyecto /
Propiedades



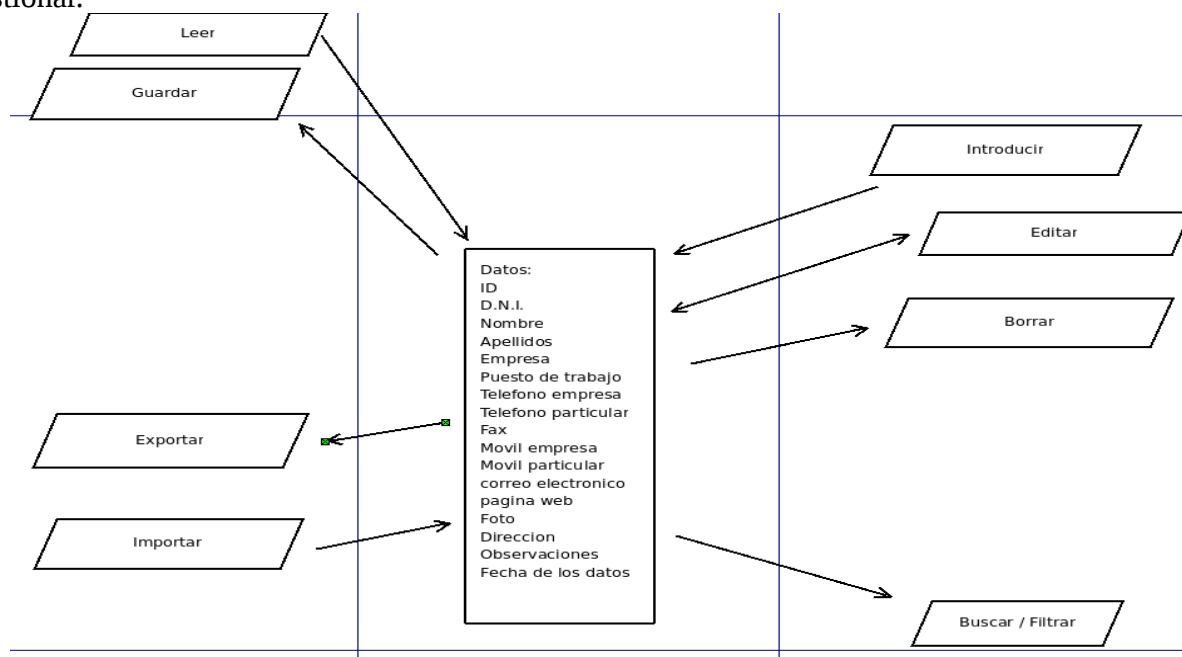
Elegimos la pestaña Opciones.

Y ponemos **Si** en “Símbolos
en módulos son publicos..”
y **Si** en “Los controles de
formularios son públicos”

Empezamos: objetivo de nuestro programa Listín

El objetivo es crear un programa que nos gestione los contactos (ya sea de personas o empresas) con sus teléfonos, dirección, foto, etc. Además de añadir, editar, borrar, nos tiene que ayudar también a buscar los datos de una manera sencilla y exportar e importar datos desde otras aplicaciones (lo haremos mediante el porta papeles) y que tenga la capacidad de salvar y leer los datos introducidos o modificados cada vez que arranquemos el programa.






Por lo tanto, lo primero que tenemos que hacer es un esquema de los datos y acciones que vamos a gestionar.

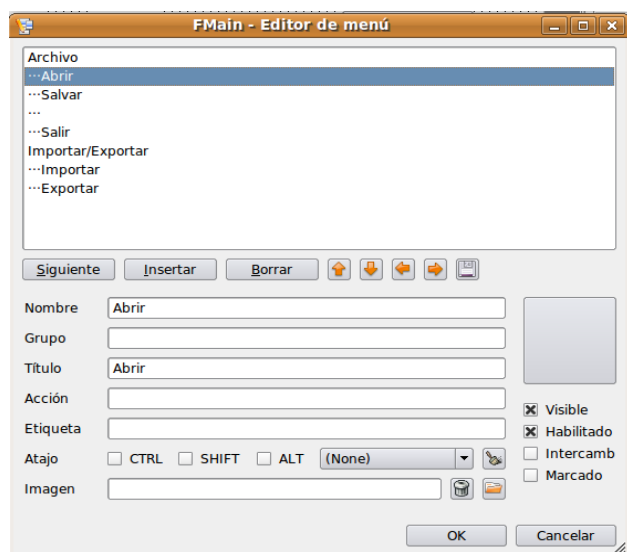


Definir el entorno visual de nuestro programa: Formulario Fmain

Vamos a definir nuestro formulario principal.

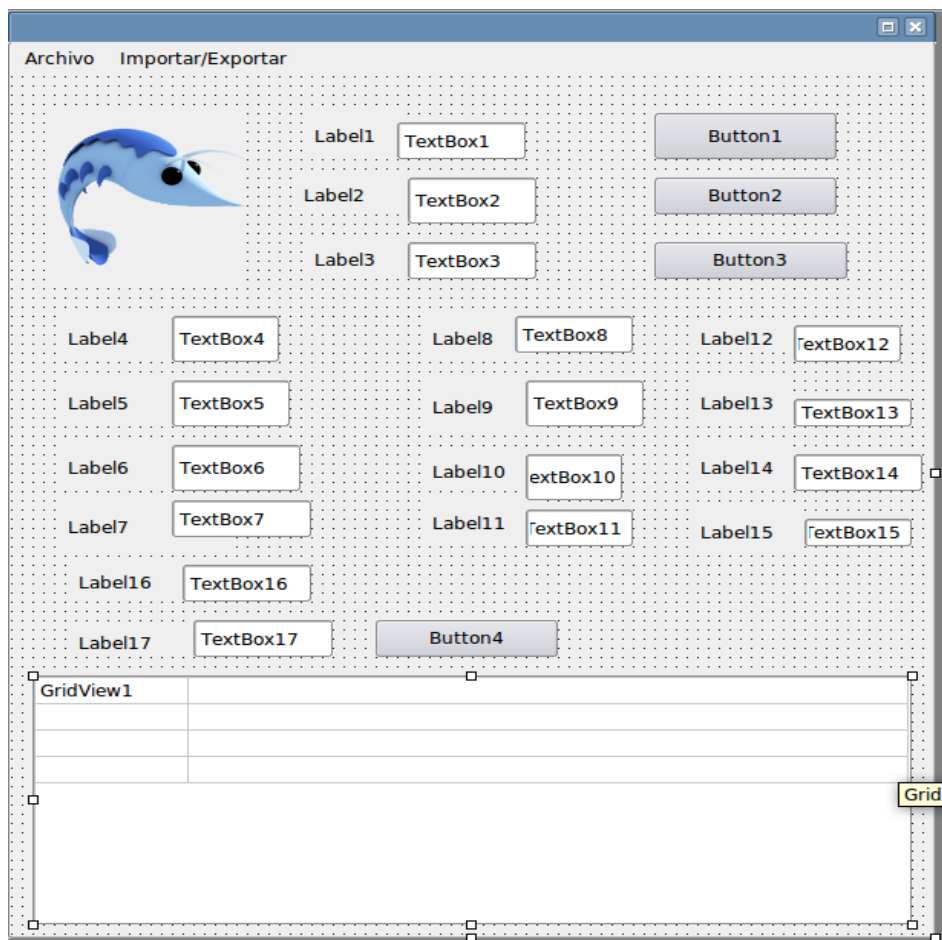
Lo dividimos en 4 zonas:

Entrada de datos:		
	Label	
	Textbox	
	PictureBox	
Botones para las distintas acciones: Editar/ Borrar/ Buscar		
	Botones Aceptar / Cancelar / Papelera / Filtrar	
Muestra de datos		
	Gridviews	
Menus: Para Salvar / Leer / Importar / Exportar		
Lo hacemos con el editor de Menus (Ctrl+E)		



Nota: al nombre de cada opción del menú, le pondremos un nombre referido a lo que hace.

Nos quedara una cosa así:



Lo siguiente que tenemos que hacer es poner en los label el texto que queremos aparezca (propiedad label.text).

En los textboxt, tenemos que poner en la propiedad (**NAME**) el nombre que sera lo haremos coincidir con lo que vaya a contener. Por ejemplo textboxNombre, textboxApellidos, etc. Y el la propiedad textbox.text la dejamos en blanco (borramos el texto que contenga).

Al pictureBox, le cambiamos las siguientes propiedades:

NAME: "pictureBoxFoto"

Border: "plain"

Stretch: "True" ' para que aparezca la imagen completa reducida al tamaño de nuestro pictureBox

Picture: "icon:/96/gambas"

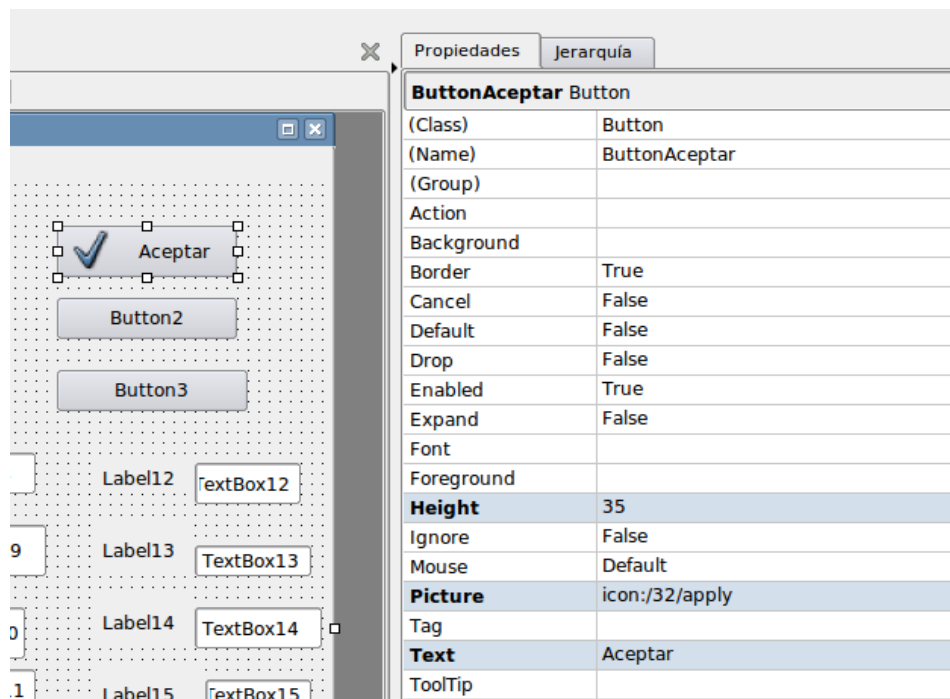
Y a los botones, tambien le cambiamos las propiedades, por ejemplo:

NAME: botonAceptar

BotonAceptar.text: Aceptar

botonAceptar.picture : elegimos el icono apply

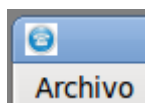
Quedando así el botón



Al gridview, tambien le cambiamos el nombre de gridview1, le ponemos GridViewDatos

Al terminar este proceso conseguimos el siguiente formulario:

También podemos añadir un icono a nuestra aplicación. En las propiedades del formulario fmain en **icon** podemos decirle que icono tendrá la aplicación (se vera es la esquina superior izquierda, una vez elegido)



Nota:



Es importante darle desde el principio (desde el momento de la creación del formulario y de los componentes), sus nombres (**NAME**) , para poder llevar en orden y claramente luego la programación de los eventos. Y QUE SEA LEGIBLE Y FACILMENTE MODIFICABLE nuestro código.

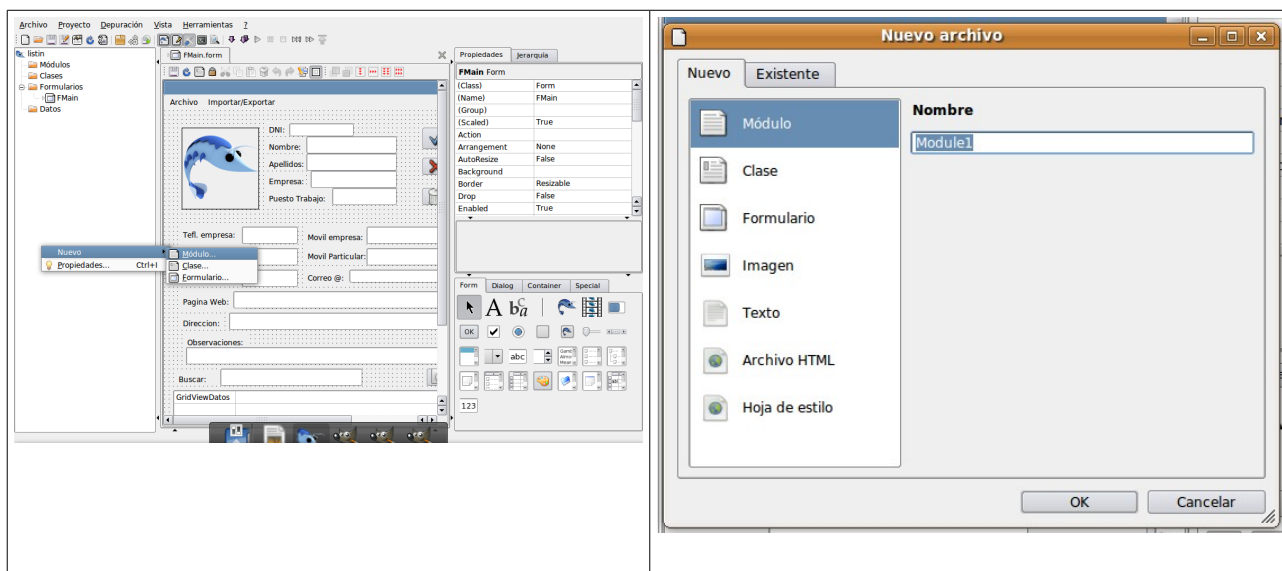
Organizar nuestro código fuente: Módulos

Antes de introducir una sola línea de código fuente vamos a organizarlo con la ayuda de los módulos.

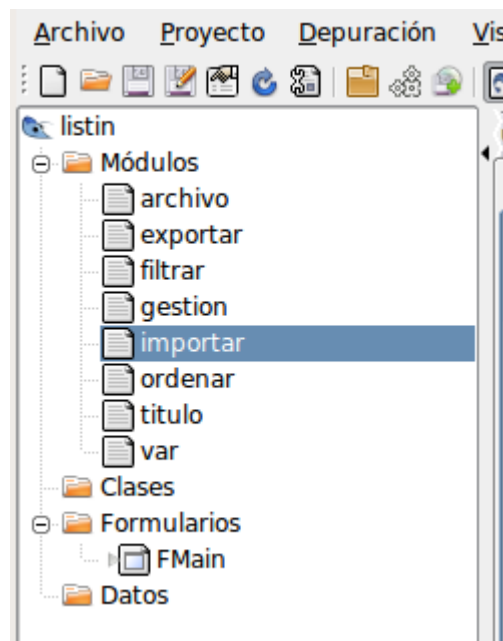
Crearemos los siguientes módulos:

var	Contiene todas las variables globales que vamos a utilizar en el programa.
titulo	Define los títulos de los gridviews y rellena los datos
archivo	Donde se encuentran las funciones de Abrir, Guardar de los archivos de datos
gestión	Funciones de gestionar la agenda: nuevos registros, edición y borrado
filtrar	
importar	Importar datos del portapapeles
exportar	Exportar datos al portapapeles

Para crear módulos, hacemos click en la zona del directorio de nuestro proyecto, nos saldrá el menu nuevo / Módulo, y nos preguntara por el nombre del nuevo modulo



Una vez creados los módulos , quedaran ordenados asi:



Módulo VAR: Declarar Variables

En este proyecto vamos a utilizar variables tipo globales (osea que se pueden acceder a sus valores desde cualquier parte del programa).

Para ello en el módulo “var”, es donde vamos a declararlas (**PUBLIC**) y iniciarlas (**RESIZE**) (los arrays o matrices de datos hay que iniciarlos para poderlos utilizar).

Nuestros datos son las siguientes variables:

Datos	Nombre de la variable	Declaración
Id	id()	PUBLIC id AS NEW String[]
D.N.I.	dni()	PUBLIC dni AS NEW String[]
Nombre	nombre()	PUBLIC nombre AS NEW String[]
Apellidos	apellidos()	PUBLIC apellidos AS NEW String[]
Empresa	empresa()	PUBLIC empresa AS NEW String[]
Puesto de trabajo	puesto()	PUBLIC puesto AS NEW String[]
Telefono empresa	telf_empresa()	PUBLIC telf_empresa AS NEW String[]
Telefono particular	telf_parti()	PUBLIC telf_parti AS NEW String[]
Fax	fax()	PUBLIC fax AS NEW String[]
Movil Empresa	movil_empresa()	PUBLIC movil_empresa AS NEW String[]
Movil Particular	movil_parti()	PUBLIC movil_parti AS NEW String[]
Página web	pag()	PUBLIC pag AS NEW String[]
foto	foto()	PUBLIC foto AS NEW String[]
direccion	direccion()	PUBLIC direccion AS NEW String[]
observaciones	observaciones()	PUBLIC observaciones AS NEW String[]
Fecha de los datos	fecha_datos()	PUBLIC fecha_datos AS NEW String[]
Nota: Los datos serán cadenas de caracteres (string) y arrays (o matrices) por ello la palabra clave “ NEW ” y los corchetes “ [] ”		

Ademas vamos a crear dentro de este módulo, una subrutina que reinicie las variables (esto se hace cada vez que se inicia el programa Listin, o cuando carguemos un archivos de datos).

Quedaría una cosa así: Modulo **VAR**

```
' Gambas module file
PUBLIC id AS NEW String[]
PUBLIC dni AS NEW String[]
PUBLIC nombre AS NEW String[]
PUBLIC apellidos AS NEW String[]
PUBLIC empresa AS NEW String[]
PUBLIC puesto AS NEW String[]
PUBLIC telf_empresa AS NEW String[]
PUBLIC telf_parti AS NEW String[]
PUBLIC fax AS NEW String[]
PUBLIC movil_empresa AS NEW String[]
PUBLIC movil_parti AS NEW String[]
PUBLIC pag AS NEW String[]
PUBLIC foto AS NEW String[]
PUBLIC direccion AS NEW String[]
PUBLIC observaciones AS NEW String[]
PUBLIC fecha_datos AS NEW String[]
PUBLIC correo AS NEW String[]

PUBLIC SUB reinicio()
  var.id.Resize(0)
  var.dni.Resize(0)
  var.nombre.Resize(0)
  var.apellidos.Resize(0)
  var.empresa.Resize(0)
  var.puesto.Resize(0)
  var.telf_empresa.Resize(0)
  var.telf_parti.Resize(0)
  var.fax.Resize(0)
  var.movil_empresa.Resize(0)
  var.movil_parti.Resize(0)
  var.pag.Resize(0)
```

```

var.foto.Resize(0)
var.direccion.Resize(0)
var.observaciones.Resize(0)
var.fecha_datos.Resize(0)
var.correo.Resize(0)
END

```

Para terminar en el formulario Fmain, cuando se produzca el evento abrir el formulario Form (“Open”) se debe ejecutar la subrutina reinicio del modulo var . Para ello introducimos el siguiente código en el formulario Fmain:

```

PUBLIC SUB Form_Open()
var.reinicio()
END

```

Definir el gridview: GridViewDatos

En el gridViewDatos, vamos a utilizarlo para que el usuario vea los datos del listin. Para esto primero tenemos que definir el numero de columnas, anchura, tipo de letra, etc.

En el módulo titulo crearemos un procedimiento llamado **definir** con el siguiente código:

```

PUBLIC SUB definir()
  WITH FMain.GridViewDatos
    .header = 3
    .Rows.COUNT = 1
    .columns.COUNT = 16
    .Columns[0].title = "Imagen"
    .Columns[1].title = "Nombre"
    .Columns[2].title = "Apellidos"
    .Columns[3].title = "DNI"
    .Columns[4].title = "Empresa"
    .Columns[5].title = "Puesto de Trabajo"
    .Columns[6].title = "Telf. Empresa"
    .Columns[7].title = "Movil Empresa"
    .Columns[8].title = "Telf. Particular"
    .Columns[9].title = "Movil Particular"
    .Columns[10].title = "Fax"
    .Columns[11].title = "Correo"
    .Columns[12].title = "Pagina Web"
  END WITH
END

```

```
.Columns[13].title = "Dirección"
.Columns[14].title = "Observaciones"
.Columns[15].title = "Fecha"
.Columns[0].width = 74
.Columns[1].width = 80
.Columns[2].width = 131
.Columns[3].width = 80
.Columns[4].width = 80
.Columns[5].width = 113
.Columns[6].width = 87
.Columns[7].width = 95
.Columns[8].width = 90
.Columns[9].width = 98
.Columns[10].width = 80
.Columns[11].width = 80
.Columns[12].width = 80
.Columns[13].width = 80
.Columns[14].width = 165
.Columns[15].width = 81
.font.name = "Sans"
.font.size = 9
.Background = 16777215
.Foreground = 0
END WITH
END
```

Y añadimos en el procedimiento Open del formulario la llamada a esta subrutina:

```
PUBLIC SUB Form_Open()
ME.CENTER()
var.reinicio()
titulo.definir()
END
```

Nota:

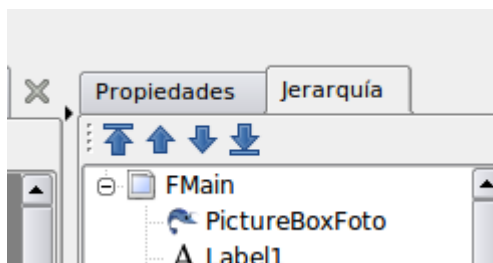
ME.CENTER: provoca que el formulario se centre en la pantalla.

Introducir Datos: Textos

SetFocus

Para introducir los datos nos tenemos que mover por los textbox.

Si pulsamos Tab de Jerarquía: [\(ver gráfica 1, zona del punto 4 rojo\)](#)



Podemos ordenar con los botones flecha arriba y flecha abajo el orden de los controles cuando pulsemos la tecla “tabulador”

Para conseguir el efecto de cuando pulso “Enter” o “Return” nos desplazemos al siguiente textbox (el que nos interese para introducir los datos) hay que utilizar el siguiente código:

```

PUBLIC SUB TextBoxDondeEstoy_KeyPress()
  IF Key.code = Key.enter OR Key.code = Key.Return THEN
    TextBoxSiguiente.SetFocus
  ENDIF
END

```

Siendo el TextBoxDondeEstoy , la TextBox donde esta actualmente el cursor, y TextBoxSiguiente el TextBox a donde quiero dirigir el cursor cuando abandone el actual TextBox.

En nuestro programa seria el siguiente código:

```

PUBLIC SUB TextBoxDNI_KeyPress()
  IF Key.code = Key.enter OR Key.code = Key.Return THEN
    TextBoxNombre.SetFocus
  ENDIF
END

PUBLIC SUB TextBoxNombre_KeyPress()
  IF Key.code = Key.enter OR Key.code = Key.Return THEN
    TextBoxApellidos.SetFocus
  ENDIF

```


END

```
PUBLIC SUB TextBoxApellidos_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxEmpresa.SetFocus  
  ENDIF
```

END

```
PUBLIC SUB TextBoxEmpresa_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxPuesto.SetFocus  
  ENDIF
```

END

```
PUBLIC SUB TextBoxPuesto_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxTelfEmpresa.SetFocus  
  ENDIF
```

END

```
PUBLIC SUB TextBoxTelfEmpresa_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxMovilEmpresa.SetFocus  
  ENDIF
```

END

```
PUBLIC SUB TextBoxMovilEmpresa_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxTelfParticular.SetFocus  
  ENDIF
```

END

```
PUBLIC SUB TextBoxTelfParticular_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxMovilParticular.SetFocus  
  ENDIF
```

END

```
PUBLIC SUB TextBoxMovilParticular_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxFax.SetFocus  
  ENDIF  
END  
  
PUBLIC SUB TextBoxFax_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxCorreo.SetFocus  
  ENDIF  
END  
  
PUBLIC SUB TextBoxCorreo_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxFecha.SetFocus  
  ENDIF  
END  
  
PUBLIC SUB TextBoxFecha_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxWEB.SetFocus  
  ENDIF  
END  
  
PUBLIC SUB TextBoxWEB_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxDireccion.SetFocus  
  ENDIF  
END  
  
PUBLIC SUB TextBoxDireccion_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN  
    TextBoxObs.SetFocus  
  ENDIF  
END  
  
PUBLIC SUB TextBoxObs_KeyPress()  
  IF Key.code = Key.enter OR Key.code = Key.Return THEN
```

```
ButtonAceptar.SetFocus
```

```
ENDIF
```

```
END
```

Introducir Datos: Comprobación de correo electrónico

Vamos a introducir un pequeño código para comprobar que la dirección de correo electrónico esta bien escrita. Comprobamos que tenga una “@” y “.”, aunque eso no implica de que exista esa dirección.

```
PUBLIC SUB TextBoxCorreo_LostFocus()  
'comprobar si el “@” y el “.” está en el textboxcorreo  
  IF InStr((TextBoxCorreo.text), (“@”)) = 0 or InStr((TextBoxCorreo.text), (“.”)) = 0 THEN  
Message.info("escriba una dirección de correo válida,falta @ o .")  
TextBoxCorreo.SetFocus  
ELSE  
ENDIF  
END
```

Lo coloqué en el evento lostfocus para que lo compruebe apenas pases a editar otro campo.

Añadiendo luego de la salida del mensaje, TextBoxCorreo.SetFocus hago que vuelva a editar el campo.

Introducir Datos: Imagen

Para introducir la imagen, vamos a utilizar un cuadro de dialogo. Vamos a crear una nueva variable global que contiene la ruta elegida de la imagen. (añadirla al modulo var)

```
PUBLIC rutaimagen AS String
```

Al hacer click en el cuadro del picture se ejecutara el siguiente codigo: (poner este codigo en el formualrio FORM)

```
PUBLIC SUB PictureBoxFoto_MouseDown()
```

'la variable Stretch que controla el redimensionamiento de la imagen la pongo como true para que se vea complea

```
PictureBoxFoto.Stretch = TRUE
```

```
Dialog.Title = "Seleccione un archivo para abrir"
```

```
Dialog.Path = User.NAME
```

```
Dialog.Filter = [ "*.jpg", "JPG", ".png", "PNG", "*.bmp", "BMP" ]
```

```
IF NOT Dialog.OpenFile() THEN
```

```
    PictureBoxFoto.Picture = Picture[Dialog.Path]
```

'variable global que controla la ruta donde esta la imagen

```
var.rutaimagen = Replace$(Dialog.Path, " ", Chr$(92) & " ")
```

```
ENDIF
```

```
END
```

Mejora con la versión 2: Replace\$.



Esta orden reemplaza el " " (espacio) del nombre de una ruta por "\" (barra invertida + espacio) que es la forma en que entiende linux los espacios entre las palabras que formen el nombre de la ruta o archivo.

Botón Aceptar: evento .add()

Al hacer click en el botón aceptar, entonces es cuando realmente tenemos que añadir la información al array. Eso lo realizamos con el evento .add():

```
PUBLIC SUB ButtonAceptar_Click()
```

'añadimos a la matriz los datos

```
var.id.add("Id" & Str$(Now))
var.dni.add(TextBoxDNI.text)
var.nombre.add(TextBoxNombre.Text)
var.apellidos.add(TextBoxApellidos.Text)
var.empresa.add(TextBoxEmpresa.Text)
var.puesto.add(TextBoxPuesto.Text)
var.telf_empresa.add(TextBoxTelfEmpresa.Text)
var.telf_parti.add(TextBoxTelfParticular.text)
var.fax.add(TextBoxFax.Text)
var.movil_empresa.add(TextBoxMovilEmpresa.Text)
var.movil_parti.add(TextBoxMovilParticular.Text)
var.pag.add(TextBoxWEB.Text)
```

'en foto guardo la ruta donde esta la imagen

```
var.foto.add(var.rutaimagen)
var.direccion.add(TextBoxDireccion.Text)
var.observaciones.add(TextBoxObs.Text)
var.fecha_datos.add(TextBoxFecha.Text)
var.correo.Add(TextBoxCorreo.Text)
```

'ponemos en blanco la propiedad .text de los textbox

```
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
TextBoxDNI.text = ""
TextBoxNombre.text = ""
TextBoxApellidos.text = ""
TextBoxEmpresa.text = ""
TextBoxPuesto.text = ""
TextBoxTelfEmpresa.text = ""
TextBoxTelfParticular.text = ""
TextBoxFax.text = ""
TextBoxMovilEmpresa.text = ""
TextBoxMovilParticular.text = ""
TextBoxWEB.text = ""
PictureBoxFoto.Picture = ""
TextBoxDireccion.text = ""
TextBoxObs.text = ""
TextBoxFecha.text = ""
TextBoxCorreo.text = ""
```

'escribimos en el gridviews el dato introducido

titulo.rellena()

'el setfocus lo ponemos justo al inicio de los datos una vez que hemos pulsado el boton Aceptar

TextBoxDNI.SetFocus

END

Y para rellenar el dato introducido en nuestro gridviews, vamos a llamar a una subrutina que esta en el módulo titulo: **rellena**

```
PUBLIC SUB rellena()
DIM a AS Integer
FMain.GridViewDatos.Rows.COUNT = var.id.COUNT
FOR a = 0 TO var.id.COUNT - 1
  WITH FMain
    .GridViewDatos[a, 0].Picture = Picture[var.foto[a]]
    .GridViewDatos[a, 1].text = var.nombre[a]
    .GridViewDatos[a, 2].text = var.apellidos[a]
    .GridViewDatos[a, 3].text = var.dni[a]
    .GridViewDatos[a, 4].text = var.empresa[a]
    .GridViewDatos[a, 5].text = var.puesto[a]
    .GridViewDatos[a, 6].text = var.telf_empresa[a]
    .GridViewDatos[a, 7].text = var.movil_empresa[a]
    .GridViewDatos[a, 8].text = var.telf_parti[a]
    .GridViewDatos[a, 9].text = var.movil_parti[a]
    .GridViewDatos[a, 10].text = var.fax[a]
    .GridViewDatos[a, 11].text = var.correo[a]
    .GridViewDatos[a, 12].text = var.pag[a]
    .GridViewDatos[a, 13].text = var.direccion[a]
    .GridViewDatos[a, 14].text = var.observaciones[a]
    .GridViewDatos[a, 15].text = var.fecha_datos[a]
  END WITH
NEXT
FMain.GridViewDatos.Refresh
END
```

Notas sobre este código:

1) Tenemos que definir una variable “a” que es al responsable de ir aumentando en el For-Next para ello utilizamos

```
Dim a as integer
```

2. El numero de filas sera igual al numero de datos Existentes.

```
FMain.GridViewDatos.Rows.COUNT = var.id.COUNT
```

3. For – Next: como empezamos de “0” el .COUNT lo reducimos en uno

```
FOR a = 0 TO var.id.COUNT - 1
```

De este modo rellena tantas filas como numero de datos haya (propiedad .COUNT)

4. WITH – END WITH: nos ahorramos de escribir varias veces fmain

5. Hay que tambien refrescar los datos del gridviewDatos, una vez definido, normalmente lo hace, pero no siempre, por lo tanto mejor forzarlo a que refresque:

```
Fmain.GridViewDatos.Refresh
```


Módulo Gestión: Borrar datos antes de pulsar Aceptar

Bueno, y que ocurre que si mientras estamos introduciendo los datos queremos borrar lo que hemos introducido. Habría dos soluciones

- 1) Borrar manual mente todos los textbox
- 2) Utilizar el siguiente código al hacer click en el botón “Borrar”

```
PUBLIC SUB ButtonBorrar_Click()
```

```
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
```

```
TextBoxDNI.text = ""
```

```
TextBoxNombre.text = ""
```

```
TextBoxApellidos.text = ""
```

```
TextBoxEmpresa.text = ""
```

```
TextBoxPuesto.text = ""
```

```
TextBoxTelfEmpresa.text = ""
```

```
TextBoxTelfParticular.text = ""
```

```
TextBoxFax.text = ""
```

```
TextBoxMovilEmpresa.text = ""
```

```
TextBoxMovilParticular.text = ""
```

```
TextBoxWEB.text = ""
```

```
PictureBoxFoto.Picture = ""
```

```
TextBoxDireccion.text = ""
```

```
TextBoxObs.text = ""
```

```
TextBoxFecha.text = ""
```

```
TextBoxCorreo.text = ""
```

```
END
```

Módulo Gestión: Editar

Una vez vez introducidos los datos de los registros, con el paso del tiempo, tenemos la necesidad de cambiarlos para actualizarlos (el típico cambio de móvil de empresa) o simplemente complementarlos con información que no teníamos. (por ejemplo la foto)

Pues bien, en este modulo voy a desarrollar el código necesario para hacerlo.

¿como elegimos el registro a editar? Propiedad .Row

Pues habria varias maneras, una de ellas seria pedir por ejemplo el dni del usuario y pasar a editarlo, a mi se me ocurre una “mas intuitiva” al usuario que es haciendo CLICK en la fila del gridViewDatos, y editar la fila donde hemos hecho el click, los datos que contiene.

Para ello tenemos la propiedad .Row del gridViewDatos, que nos indica en que fila hemos pulsado.

Registro Único: el campo ID

Pero todavia nos faltaria un dato muy importante para poder editar el registro, sabiendo la fila, tenemos que saber el var.id del registro (numero que identifica “unicamente” un registro), ya que si elegimos otro campo (nombre, apellidos, etc), pueden coincidir varios registros con esos datos, pero no asi el ID.

Para ello a nuestro gridViewDatos, le vamos a añadir una columna (no hace falta darle anchura ni titulo) y en esa columna escribiremos el id del registro. Para ello hay que modificar:

MODULO **TITULO**:

Procedimiento: definir()

En la linea:

```
.columns.COUNT = 16
```

Habria que cambiarla por :

```
.columns.COUNT = 17
```

Procedimiento: rellena()

Añadir la linea:

```
.GridViewDatos[a, 15].text = var.fecha_datos[a]  
.GridViewDatos[a, 16].text = var.id[a]  
END WITH
```

De esta forma la columna 17, tendrá el dato id que nos ayudara a identificar el registro para editarlo.



Los gridviews empiezan a contar desde 0, pero las columnas visibles empiezan desde 1, por lo tanto un gridview que tenga 5 columnas (`.columns.COUNT = 5`), realmente se cuenta “0,1,2,3,4” y si queremos poner un dato en la columna 5, debemos indicar al gridviews el 4. `.GridViewDatos[a, 4].text = lo_que_sea`

Ahora pasamos al “editar” la fila-registro elegido y para indicar de algun modo que estamos en modo edicion, el gridviewdatos lo pondremos con la propiedad `.enabled=False`. Posteriormente cuanto terminemos la edicion volvera a su estado normal (`.enabled=True`).:

En el Formulario **Fmain**:

```
PUBLIC SUB GridViewDatos_Click()  
    gridViewDatos.enabled=false  
    gestion.editar(GridViewDatos[GridViewDatos.Row, 16].text)  
END
```

De esta manera llamamos a un procedimiento que esta en el modulo gestion, llamado editar , que va a hacer la funcion de editar el registro con el ID que contiene la fila clickeada.

En el modulo **Gestion**:

Necesitamos buscar el numero de registro que contenga ese ID, eso lo resolvemos con una funcion que dado el id devuelva el nº de registro, o en el peor de los casos un mensaje de error:

```
PUBLIC FUNCTION buscarDadoId(id AS String) AS Integer  
    DIM a AS Integer  
    FOR a = 0 TO var.id.COUNT - 1  
        IF id = var.id[a] THEN  
            RETURN a  
        ENDIF  
    NEXT  
    Message.Error("No encuentro registro con este Id: " & ID)  
    RETURN -1  
END
```



Nota Importante: los paranteceis () no son lo mismo que los corchetes []

Los parentesis () indica que son funciones que les pasa un valor, el que esta entre parentesis

Los corchetes [] el el numero que ocupa el datos en una matriz o array.

Esta función es llamada dentro de la subrutina principal editar.

```
PUBLIC SUB editar(id AS String)
  DIM registro AS Integer
  registro = buscarDadoId(id)
  IF REGISTRO = -1 THEN
    Message.Info("No es posible editar, datos corruptos")
  ELSE
    var.estado = "Edicion"
    var.RegistroEditado = registro
    WITH FMain
      .PictureBoxFoto.Picture = picture[var.foto[registro]]
      .TextBoxDNI.text = var.dni[registro]
      .TextBoxNombre.text = var.nombre[registro]
      .TextBoxApellidos.text = var.apellidos[registro]
      .TextBoxEmpresa.text = var.empresa[registro]
      .TextBoxPuesto.text = var.puesto[registro]
      .TextBoxTelfEmpresa.text = var.telf_empresa[registro]
      .TextBoxTelfParticular.text = var.telf_parti[registro]
      .TextBoxFax.text = var.fax[registro]
      .TextBoxMovilEmpresa.text = var.movil_empresa[registro]
      .TextBoxMovilParticular.text = var.movil_parti[registro]
      .TextBoxWEB.text = var.pag[registro]
      .TextBoxDireccion.text = var.direccion[registro]
      .TextBoxObs.text = var.observaciones[registro]
      .TextBoxFecha.text = var.fecha_datos[registro]
      .TextBoxCorreo.text = var.correo[registro]
    END WITH
    var.rutaimagen = var.foto[registro]
  ENDIF
END
```

Ademas vamos a definir otra variable global (en el modulo **var**)

```
PUBLIC estado AS String
PUBLIC registroeditado as integer
```

que indicara al programa que estamos “editando” y no “introduccion datos” eso servira para que los botones de Aceptar / Borrar / Cancelar actuen de diferente manera.

El botón Aceptar en modo Edición.

El boton aceptar se debe de compartir de una forma distinta que cuando estamos introduciendo datos. ¿como lo hacemos? Pues con la variable global “estado” y un simple “IfThen”, si el estado es “editando” haremos que al pulsar el boton aceptar este sobre escriba los nuevos datos en el mismo registro, si no estamos editando, pues añadiremos como explicamos anteriormente.

Os pongo el codigo modificado:

Codigo del formulario FMAIN

```
PUBLIC SUB ButtonAceptar_Click()
```

```
IF var.estado<> "Edicion" then
```

```
'añadimos a la matriz los datos, no estamos editando... solo insertando nuevo registro
```

```
var.id.add("Id" & Str$(Now))
var.dni.add(TextBoxDNI.text)
var.nombre.add(TextBoxNombre.Text)
var.apellidos.add(TextBoxApellidos.Text)
var.empresa.add(TextBoxEmpresa.Text)
var.puesto.add(TextBoxPuesto.Text)
var.telf_empresa.add(TextBoxTelfEmpresa.Text)
var.telf_parti.add(TextBoxTelfParticular.text)
var.fax.add(TextBoxFax.Text)
var.movil_empresa.add(TextBoxMovilEmpresa.Text)
var.movil_parti.add(TextBoxMovilParticular.Text)
var.pag.add(TextBoxWEB.Text)
'en foto guardo la ruta donde esta la imagen
var.foto.add(var.rutaimagen)
var.direccion.add(TextBoxDireccion.Text)
var.observaciones.add(TextBoxObs.Text)
var.fecha_datos.add(TextBoxFecha.Text)
var.correo.Add(TextBoxCorreo.Text)
```

```
endif
```

```

if var.estado= "Edicion" then
  var.dni[var.RegistroEditado] = TextBoxDNI.text
  var.nombre[var.RegistroEditado] = TextBoxNombre.Text
  var.apellidos[var.RegistroEditado] = TextBoxApellidos.Text
  var.empresa[var.RegistroEditado] = TextBoxEmpresa.Text
  var.puesto[var.RegistroEditado] = TextBoxPuesto.Text
  var.telf_empresa[var.RegistroEditado] = TextBoxTelfEmpresa.Text
  var.telf_parti[var.RegistroEditado] = TextBoxTelfParticular.text
  var.fax[var.RegistroEditado] = TextBoxFax.Text
  var.movil_empresa[var.RegistroEditado] = TextBoxMovilEmpresa.Text
  var.movil_parti[var.RegistroEditado] = TextBoxMovilParticular.Text
  var.pag[var.RegistroEditado] = TextBoxWEB.Text
  'en foto guardo la ruta donde esta la imagen
  var.foto[var.RegistroEditado] = var.rutaimagen
  var.direccion[var.RegistroEditado] = TextBoxDireccion.Text
  var.observaciones[var.RegistroEditado] = TextBoxObs.Text
  var.fecha_datos[var.RegistroEditado] = TextBoxFecha.Text
  var.correo[var.RegistroEditado] = TextBoxCorreo.Text
  'se da por compluido el estado de edicion
  var.estado=""
endif

```

'ponemos en blanco la propiedad .text de los textbox

```

PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
TextBoxDNI.text = ""
TextBoxNombre.text = ""
TextBoxApellidos.text = ""
TextBoxEmpresa.text = ""
TextBoxPuesto.text = ""
TextBoxTelfEmpresa.text = ""
TextBoxTelfParticular.text = ""
TextBoxFax.text = ""
TextBoxMovilEmpresa.text = ""
TextBoxMovilParticular.text = ""
TextBoxWEB.text = ""
PictureBoxFoto.Picture = ""

```

```

TextBoxDireccion.text = ""
TextBoxObs.text = ""
TextBoxFecha.text = ""
TextBoxCorreo.text = ""
'escribimos en el gridviews el dato introducido
titulo.rellena()
'el setfocus lo ponemos justo al inicio de los datos
TextBoxDNI.SetFocus
'hacemos de nuevo accesible el gridViewDatos
gridViewDatos.enabled = True
END

```

Nota:

La parte del código en azul, es la parte añadida.

Modulo Gestion: Cancelar

Cuando estemos editando (var.estado="Editando") y pulsemos este botón, dejaremos de editar y borraremos los datos del formulario, sin que se cambien los valores de los registros.

```

PUBLIC SUB ButtonCancelar_Click()
'ponemos en blanco la propiedad .text de los textbox
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
TextBoxDNI.text = ""
TextBoxNombre.text = ""
TextBoxApellidos.text = ""
TextBoxEmpresa.text = ""
TextBoxPuesto.text = ""
TextBoxTelfEmpresa.text = ""
TextBoxTelfParticular.text = ""
TextBoxFax.text = ""
TextBoxMovilEmpresa.text = ""
TextBoxMovilParticular.text = ""
TextBoxWEB.text = ""
PictureBoxFoto.Picture = ""
TextBoxDireccion.text = ""
TextBoxObs.text = ""
TextBoxFecha.text = ""

```

```
TextBoxCorreo.text = ""  
'escribimos en el gridviews el dato introducido  
titulo.rellena()  
'el setfocus lo ponemos justo al inicio de los datos  
TextBoxDNI.SetFocus  
'se da por compluido el estado de edicion  
var.estado = ""  
'hacemos de nuevo accesible el gridViewDatos  
gridViewDatos.enabled = True  
END
```



Como observareis mucho código se repite en varias zonas del programa, os propongo la optimización de este código a través de subrutinas.

Módulo Gestión. Borrar datos editados: Remove()

Nos referimos esta vez a borrar un registro por completo (de la matriz o del array que con lo contenga).

Borraremos el registro de datos que estemos editando (eliminacion total de la matriz). Para ello tenemos el metodo .REMOVE(indice), que indicandole el indice que queramos eliminar/borrar lo hara automaticamente y automaticamente modificará tambien el número de elementos .COUNT

```
PUBLIC SUB ButtonBorrar_Click()
```

```
IF var.estado<> "Edicion" then
```

```
'si estamos en mode "introduccion de datos"
```

```
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
```

```
TextBoxDNI.text = ""
```

```
TextBoxNombre.text = ""
```

```
TextBoxApellidos.text = ""
```

```
TextBoxEmpresa.text = ""
```

```
TextBoxPuesto.text = ""
```

```
TextBoxTelfEmpresa.text = ""
```

```
TextBoxTelfParticular.text = ""
```

```
TextBoxFax.text = ""
```

```
TextBoxMovilEmpresa.text = ""
```

```
TextBoxMovilParticular.text = ""
```

```
TextBoxWEB.text = ""
```

```
PictureBoxFoto.Picture = ""
```

```
TextBoxDireccion.text = ""
```

```
TextBoxObs.text = ""
```

```
TextBoxFecha.text = ""
```

```
TextBoxCorreo.text = ""
```

```
ENDIF
```

```
IF var.estado= "Edicion" then
```

```
var.id.REMOVE(var.RegistroEditado)
```

```
var.dni.REMOVE(var.RegistroEditado)
```

```
var.nombre.REMOVE(var.RegistroEditado)
```

```
var.apellidos.REMOVE(var.RegistroEditado)
```

```
var.empresa.REMOVE(var.RegistroEditado)
```

```
var.puesto.REMOVE(var.RegistroEditado)
```

```
var.telf_empresa.REMOVE(var.RegistroEditado)
```

```
var.telf_parti.REMOVE(var.RegistroEditado)
```

```

var.fax.REMOVE(var.RegistroEditado)
var.movil_empresa.REMOVE(var.RegistroEditado)
var.movil_parti.REMOVE(var.RegistroEditado)
var.pag.REMOVE(var.RegistroEditado)
var.foto.REMOVE(var.RegistroEditado)
var.direccion.REMOVE(var.RegistroEditado)
var.observaciones.REMOVE(var.RegistroEditado)
var.fecha_datos.REMOVE(var.RegistroEditado)
var.correo.REMOVE(var.RegistroEditado)
Y PONEMOS TODOS LOS CAMPOS EN BLANCO...
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
TextBoxDNI.text = ""
TextBoxNombre.text = ""
TextBoxApellidos.text = ""
TextBoxEmpresa.text = ""
TextBoxPuesto.text = ""
TextBoxTelfEmpresa.text = ""
TextBoxTelfParticular.text = ""
TextBoxFax.text = ""
TextBoxMovilEmpresa.text = ""
TextBoxMovilParticular.text = ""
TextBoxWEB.text = ""
PictureBoxFoto.Picture = ""
TextBoxDireccion.text = ""
TextBoxObs.text = ""
TextBoxFecha.text = ""
TextBoxCorreo.text = ""
ENDIF
escribimos en el gridviews el dato introducido
titulo.rellena()
el setfocus lo ponemos justo al inicio de los datos
TextBoxDNI.SetFocus
se da por cumplido el estado de edicion
var.estado = ""
hacemos de nuevo accesible el gridViewDatos
gridViewDatos.enabled = True
END

```

```
PUBLIC SUB ButtonBorrar_Click()
```

```
IF var.estado<> "Edicion" then
```

```
  'si estamos en mode "introduccion de datos"
```

```
  PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
```

```
  TextBoxDNI.text = ""
```

```
  TextBoxNombre.text = ""
```

```
  TextBoxApellidos.text = ""
```

```
  TextBoxEmpresa.text = ""
```

```
  TextBoxPuesto.text = ""
```

```
  TextBoxTelfEmpresa.text = ""
```

```
  TextBoxTelfParticular.text = ""
```

```
  TextBoxFax.text = ""
```

```
  TextBoxMovilEmpresa.text = ""
```

```
  TextBoxMovilParticular.text = ""
```

```
  TextBoxWEB.text = ""
```

```
  PictureBoxFoto.Picture = ""
```

```
  TextBoxDireccion.text = ""
```

```
  TextBoxObs.text = ""
```

```
  TextBoxFecha.text = ""
```

```
  TextBoxCorreo.text = ""
```

```
ENDIF
```

```
IF var.estado= "Edicion" then
```

```
  var.id.REMOVE(var.RegistroEditado)
```

```
  var.dni.REMOVE(var.RegistroEditado)
```

```
  var.nombre.REMOVE(var.RegistroEditado)
```

```
  var.apellidos.REMOVE(var.RegistroEditado)
```

```
  var.empresa.REMOVE(var.RegistroEditado)
```

```
  var.puesto.REMOVE(var.RegistroEditado)
```

```
  var.telf_empresa.REMOVE(var.RegistroEditado)
```

```
  var.telf_parti.REMOVE(var.RegistroEditado)
```

```
  var.fax.REMOVE(var.RegistroEditado)
```

```
  var.movil_empresa.REMOVE(var.RegistroEditado)
```

```
  var.movil_parti.REMOVE(var.RegistroEditado)
```

```
  var.pag.REMOVE(var.RegistroEditado)
```

```
  var.foto.REMOVE(var.RegistroEditado)
```

```

var.direccion.REMOVE(var.RegistroEditado)
var.observaciones.REMOVE(var.RegistroEditado)
var.fecha_datos.REMOVE(var.RegistroEditado)
var.correo.REMOVE(var.RegistroEditado)
'Y PONEMOS TODOS LOS CAMPOS EN BLANCO....
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
TextBoxDNI.text = ""
TextBoxNombre.text = ""
TextBoxApellidos.text = ""
TextBoxEmpresa.text = ""
TextBoxPuesto.text = ""
TextBoxTelfEmpresa.text = ""
TextBoxTelfParticular.text = ""
TextBoxFax.text = ""
TextBoxMovilEmpresa.text = ""
TextBoxMovilParticular.text = ""
TextBoxWEB.text = ""
PictureBoxFoto.Picture = ""
TextBoxDireccion.text = ""
TextBoxObs.text = ""
TextBoxFecha.text = ""
TextBoxCorreo.text = ""
ENDIF
'escribimos en el gridviews el dato introducido
titulo.rellena()
'el setfocus lo ponemos justo al inicio de los datos
TextBoxDNI.SetFocus
'se da por compluido el estado de edicion
var.estado = ""
'hacemos de nuevo accesible el gridViewDatos
gridViewDatos.enabled = True
END

```



Nota: Ver mejoras en presentacion de imagenes
Anexo II: convert, miniaturas en nuestro gridViewDatos

Módulo Archivo: Salvar

¿y ahora que?, pues habrá que guardar los datos que hemos introducidos claro esta...

En el modulo **ARCHIVO**

Crearemos una funcion para comprobar que no Exista y evitemos sobreescribiremos otro archivo de datos:

```
PUBLIC FUNCTION compruebaExistencia(destino AS String, tipo AS String) AS Integer
DIM DEVUELVE AS Integer
'Exist DEVUELVE TRUE SI ESISTE EL ARCHIVO
IF Exist(destino) = FALSE THEN
'no Existe el archivo, por lo tanto podemos continuar con el salvado con el mismo nombre
RETURN 1
ENDIF
'esta parte se ejecuta cuando Existe el archivo.
IF Message.Warning("Existe el archivo de " & tipo, "Cancelar", "Sobreescribir") = 1 THEN
' ha cancelado, pedir de nuevo el nombre
RETURN 2
ENDIF
IF Message.Warning("Ha decidido sobreescribir ¿esta seguro?", "Cancelar", "Aceptar") = 1 THEN
' ha cancelado, pedir de nuevo el nombre
RETURN 2
ELSE
'el usuario quiere sobreescribir y esta seguro con ello ya que lo ha dicho dos veces
RETURN 1
ENDIF
END
```

Y ahora haremos la rutina de salvado de datos. Tenemos que organizar como guardar los datos. ¿de que forma lo podemos hacer?

1. Con la orden Dialog.SaveFile(), leemos la ruta donde el usuario quiere guardar los datos.
2. Vamos a utilizar una cadena de caracteres que llamaremos **lineas** , donde se vayan añadiendo los datos separados por **codigofinline**. El codigo seria

Lineas &=dato & codigofinline

3. Separación de los datos: para separar los datos unos de otros utilizaremos un carácter poco usado: “|”, y la variable que lo contiene la llamaremos **codigofinline**
4. El primer dato que guardamos sera la versión del archivo de datos “v0.0.1”, y el programa que lo ha realizado. ¿por que? Con el tiempo iremos haciendo mejoras en el programa y seguramente guardaremos datos distintos, con lo cual el programa deberá saber que versión es la que lee para tratarlos adecuadamente en la memoria.
5. Guardamos el numero de datos que tenemos (propiedad **.COUNT**)
6. Realizamos un bucle For...Next para ir añadiendo a lineas los datos
7. Con el comando File.Save(ruta,lineas) salvamos los datos en un archivo.

```
PUBLIC SUB salva(OPTIONAL ruta AS String)
```

```

DIM destino AS String
DIM a AS Integer
DIM lineas AS String
DIM codigofinline AS String

inicio: 'etiqueta utilizada para en el caso de que no deseemos sobrecribir vuelva el programa al inicio del
Dialog.SaveFile()

IF ruta = "" THEN
Dialog.Title = "Escriba un nombre de archivo para guardar los datos"
Dialog.Path = ""
Dialog.Filter = ["*.lis", "Datos de Listin"]
a = Dialog.SaveFile()
IF a = -1 THEN GOTO fins
IF Right$(Dialog.Path, Len(".lis")) <> ".lis" THEN
destino = Dialog.Path & ".lis"
ELSE
destino = Dialog.Path
ENDIF
ELSE
destino = ruta
ENDIF

'Comprueba si Existe el archivo, en tal caso pide confirmación de sobre escritura
IF compruebaExistencia(DESTINO, "listin") = 2 THEN
GOTO inicio
ENDIF

'continua ejecutando el programa
codigofinline = "|"
Lineas = "v0.0.1" & codigofinline 'informo versión
Lineas &= "listin.20100718" & codigofinline ' programa que ha echo el archivo
lineas &= var.id.COUNT & codigofinline 'numero de registros Existentes
FOR a = 0 TO var.id.COUNT - 1
    Lineas &= var.id[a] & codigofinline
    Lineas &= var.dni[a] & codigofinline
    lineas &= var.nombre[a] & codigofinline
    lineas &= var.apellidos[a] & codigofinline
    lineas &= var.empresa[a] & codigofinline
    lineas &= var.puesto[a] & codigofinline
    lineas &= var.telf_empresa[a] & codigofinline

```

```

lineas &= var.telf_parti[a] & codigofinline
lineas &= var.fax[a] & codigofinline
lineas &= var.movil_empresa[a] & codigofinline
lineas &= var.movil_parti[a] & codigofinline
lineas &= var.pag[a] & codigofinline
lineas &= var.foto[a] & codigofinline
lineas &= var.direccion[a] & codigofinline
lineas &= var.observaciones[a] & codigofinline
lineas &= var.fecha_datos[a] & codigofinline
lineas &= var.correo[a] & codigofinline
NEXT
File.Save(destino, lineas)
fins: ' hemos pulsado el botón de cancelar en el cuadro de dialogo Dialog.SaveFile()
'fin de la subrutina
END

```

Y desde el formulario FMAIN en su menú Archivo/Salvar ejecutaremos el código:

```

PUBLIC SUB Salvar_Click()
    archivo.salva()
END

```



Nota Importante:

Como os dareos cuenta, los archivos de imagen no se guardan, solo las rutas donde estan ubicados. Cuando llevemos las datos de un ordenador a otro, estas imagenes no apareceran. Ver Anexo 4: empaquetado/desempaquetado de datos, donde se explica una posible solución

Módulo Archivo: Leer*Leemos nuestros datos.... nuestra agenda casi funcional*

Bueno, una vez que guardemos nuestros datos, habra que leerlos. Como siempre, y para ser organizados nos iremos al modulo “Archivo”, y crearemos la siguiente subrutina:

```

PUBLIC SUB abrir(OPTIONAL ruta AS String)
DIM a AS Integer 'contador de conjunto de datos
DIM b AS Integer 'contador de dato
DIM arr_cadenas AS String[]
DIM codigofinline AS String
DIM numero_de_datos AS Integer
codigofinline = "|"
Dialog.Title = "Seleccione un archivo de datos listin"
Dialog.Filter = ["*.lis", "Datos de Listin"]
IF NOT Dialog.OpenFile() THEN
arr_cadenas = Split(File.LOAD(Dialog.Path), codigofinline)

if arr_cadenas[0]<>"v0.0.1" then
'se trata de una version incompatible con la version de este programa, abandono el procedimiento
    Message.Error("Error: Version incompatible de datos")
    goto finlectura
endif
numero_de_datos=arr_cadenas[2] ' la arr_cadenas[1] contiene el programa que lo hizo
b=2
'redimensiono los datos....
    var.id.Resize(numero_de_datos)
    var.dni.Resize(numero_de_datos)
    var.nombre.Resize(numero_de_datos)
    var.apellidos.Resize(numero_de_datos)
    var.empresa.Resize(numero_de_datos)
    var.puesto.Resize(numero_de_datos)
    var.telf_empresa.Resize(numero_de_datos)
    var.telf_parti.Resize(numero_de_datos)
    var.fax.Resize(numero_de_datos)
    var.movil_empresa.Resize(numero_de_datos)
    var.movil_parti.Resize(numero_de_datos)
    var.pag.Resize(numero_de_datos)
    var.foto.Resize(numero_de_datos)
    var.direccion.Resize(numero_de_datos)

```



```
var.observaciones.Resize(numero_de_datos)
var.fecha_datos.Resize(numero_de_datos)
var.correo.Resize(numero_de_datos)

'leo los datos
for a=0 to numero_de_datos-1
    b+=1
    var.id[a]=arr_cadenas[b]
    b+=1
    var.dni[a]=arr_cadenas[b]
    b+=1
    var.nombre[a]=arr_cadenas[b]
    b+=1
    var.apellidos[a]=arr_cadenas[b]
    b+=1
    var.empresa[a]=arr_cadenas[b]
    b+=1
    var.puesto[a]=arr_cadenas[b]
    b+=1
    var.telf_empresa[a]=arr_cadenas[b]
    b+=1
    var.telf_parti[a]=arr_cadenas[b]
    b+=1
    var.fax[a]=arr_cadenas[b]
    b+=1
    var.movil_empresa[a]=arr_cadenas[b]
    b+=1
    var.movil_parti[a]=arr_cadenas[b]
    b+=1
    var.pag[a]=arr_cadenas[b]
    b+=1
    var.foto[a]=arr_cadenas[b]
    b+=1
    var.direccion[a]=arr_cadenas[b]
    b+=1
    var.observaciones[a]=arr_cadenas[b]
    b+=1
    var.fecha_datos[a]=arr_cadenas[b]
```

```
b+=1
var.correo[a]=arr_cadenas[b]
next
titulo.rellena()
ENDIF
finlectura:
END
```



Orden Split:

Es capaz de crear un array desde una cadena o archivo (file.load(dialog.path) cuando los elementos que esten separados por un caracter (en nuestro caso el “codigofinline”)

```
arr_cadenas = Split(File.LOAD(Dialog.Path), codigofinline)
```

Y en el formulario **FMAIN**, añadimos el código siguiente al pulsar en el Menu Archivo/Abrir

```
PUBLIC SUB Abrir_Click()
    archivo.abrir()
END
```

Modulo Archivo: Salir

Para salir del programa, utilizaremos la orden ME.Close.

En el Menus/Archivo/Salir, introducimos el siguiente código:

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

Esta salida, la hace “bruscamente”, osea que si no hemos salvado los ultimos datos que hemos introducido o modificados se perderan.... ¿que podemos hacer “corregirlo”?

Pues creamos una variable global llamada “cambio”.

En el módulo var:

```
PUBLIC cambio AS String
```

Cada vez que **introducamos,editamos o borramos** ,esta variable cambiara de valor a **cambio=”si”**,

```
PUBLIC SUB ButtonAceptar_Click()
IF var.estado <> "Edicion" THEN
    'añadimos a la matriz los datos
    var.id.add("Id" & Str$(Now))
    var.dni.add(TextBoxDNI.text)
    var.nombre.add(TextBoxNombre.Text)
    var.apellidos.add(TextBoxApellidos.Text)
    var.empresa.add(TextBoxEmpresa.Text)
    var.puesto.add(TextBoxPuesto.Text)
    var.telf_empresa.add(TextBoxTelfEmpresa.Text)
    var.telf_parti.add(TextBoxTelfParticular.text)
    var.fax.add(TextBoxFax.Text)
    var.movil_empresa.add(TextBoxMovilEmpresa.Text)
    var.movil_parti.add(TextBoxMovilParticular.Text)
    var.pag.add(TextBoxWEB.Text)
    'en foto guardo la ruta donde esta la imagen
    var.foto.add(var.rutaimagen)
    var.direccion.add(TextBoxDireccion.Text)
    var.observaciones.add(TextBoxObs.Text)
    var.fecha_datos.add(TextBoxFecha.Text)
    var.correo.Add(TextBoxCorreo.Text)
    var.cambio = "si"
ENDIF
```

```

IF var.estado = "Edicion" THEN
    var.dni[var.RegistroEditado] = TextBoxDNI.text
    var.nombre[var.RegistroEditado] = TextBoxNombre.Text
    var.apellidos[var.RegistroEditado] = TextBoxApellidos.Text
    var.empresa[var.RegistroEditado] = TextBoxEmpresa.Text
    var.puesto[var.RegistroEditado] = TextBoxPuesto.Text
    var.telf_empresa[var.RegistroEditado] = TextBoxTelfEmpresa.Text
    var.telf_parti[var.RegistroEditado] = TextBoxTelfParticular.text
    var.fax[var.RegistroEditado] = TextBoxFax.Text
    var.movil_empresa[var.RegistroEditado] = TextBoxMovilEmpresa.Text
    var.movil_parti[var.RegistroEditado] = TextBoxMovilParticular.Text
    var.pag[var.RegistroEditado] = TextBoxWEB.Text
    'en foto guardo la ruta donde esta la imagen
    var.foto[var.RegistroEditado] = var.rutaimagen
    var.direccion[var.RegistroEditado] = TextBoxDireccion.Text
    var.observaciones[var.RegistroEditado] = TextBoxObs.Text
    var.fecha_datos[var.RegistroEditado] = TextBoxFecha.Text
    var.correo[var.RegistroEditado] = TextBoxCorreo.Text
    var.cambio = "sí"
    'se da por compluido el estado de edicion
ENDIF

'ponemos en blanco la propiedad .text de los textbox
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
TextBoxDNI.text = ""
TextBoxNombre.text = ""
TextBoxApellidos.text = ""
TextBoxEmpresa.text = ""
TextBoxPuesto.text = ""
TextBoxTelfEmpresa.text = ""
TextBoxTelfParticular.text = ""
TextBoxFax.text = ""
TextBoxMovilEmpresa.text = ""
TextBoxMovilParticular.text = ""
TextBoxWEB.text = ""
PictureBoxFoto.Picture = ""
TextBoxDireccion.text = ""
TextBoxObs.text = ""
TextBoxFecha.text = ""

```

```

TextBoxCorreo.text = ""
'escribimos en el gridviews el dato introducido
titulo.rellena()
'se da por compluido el estado de edicion
var.estado = ""
'el setfocus lo ponemos justo al inicio de los datos
TextBoxDNI.SetFocus
'hacemos de nuevo accesible el gridViewDatos
gridViewDatos.enabled = TRUE
END

```

Y cuando borramos:

```

PUBLIC SUB ButtonBorrar_Click()
IF var.estado <> "Edicion" THEN
'si estamos en mode "introduccion de datos"
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
TextBoxDNI.text = ""
TextBoxNombre.text = ""
TextBoxApellidos.text = ""
TextBoxEmpresa.text = ""
TextBoxPuesto.text = ""
TextBoxTelfEmpresa.text = ""
TextBoxTelfParticular.text = ""
TextBoxFax.text = ""
TextBoxMovilEmpresa.text = ""
TextBoxMovilParticular.text = ""
TextBoxWEB.text = ""
PictureBoxFoto.Picture = ""
TextBoxDireccion.text = ""
TextBoxObs.text = ""
TextBoxFecha.text = ""
TextBoxCorreo.text = ""
endif
IF var.estado = "Edicion" THEN
'si estamos en modo "edicion"
var.id.REMOVE(var.RegistroEditado)
var.dni.REMOVE(var.RegistroEditado)
var.nombre.REMOVE(var.RegistroEditado)

```

```

var.apellidos.REMOVE(var.RegistroEditado)
var.empresa.REMOVE(var.RegistroEditado)
var.puesto.REMOVE(var.RegistroEditado)
var.telf_empresa.REMOVE(var.RegistroEditado)
var.telf_parti.REMOVE(var.RegistroEditado)
var.fax.REMOVE(var.RegistroEditado)
var.movil_empresa.REMOVE(var.RegistroEditado)
var.movil_parti.REMOVE(var.RegistroEditado)
var.pag.REMOVE(var.RegistroEditado)
var.foto.REMOVE(var.RegistroEditado)
var.direccion.REMOVE(var.RegistroEditado)
var.observaciones.REMOVE(var.RegistroEditado)
var.fecha_datos.REMOVE(var.RegistroEditado)
var.correo.REMOVE(var.RegistroEditado)
var.cambio = "Si"
' Y PONEMOS TODOS LOS CAMPOS EN BLANCO
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
TextBoxDNI.text = ""
TextBoxNombre.text = ""
TextBoxApellidos.text = ""
TextBoxEmpresa.text = ""
TextBoxPuesto.text = ""
TextBoxTelfEmpresa.text = ""
TextBoxTelfParticular.text = ""
TextBoxFax.text = ""
TextBoxMovilEmpresa.text = ""
TextBoxMovilParticular.text = ""
TextBoxWEB.text = ""
PictureBoxFoto.Picture = ""
TextBoxDireccion.text = ""
TextBoxObs.text = ""
TextBoxFecha.text = ""
TextBoxCorreo.text = ""
'hacemos de nuevo accesible el gridViewDatos
gridViewDatos.enabled = TRUE
ENDIF
'escribimos en el gridviews el dato introducido
titulo.rellena()

```

'el setfocus lo ponemos justo al inicio de los datos

TextBoxDNI.SetFocus

'se da por compluido el estado de edicion

var.estado = ""

END

Cada vez que se **guarde o abra** un archivo esta variable valdra **cambio="no"**

En el módulo Archivar / salva():

.....

File.Save(destino, lineas)

var.cambio="no" *' se ha grabado toda la informacion, ha dejado de haber cambios*

fins: *' hemos pulsado el boton de cancelar en el cuadro de dialogo Dialog.SaveFile()*

'fin de la subrutina

END

En el módulo Archivar / abrir()

.....

NEXT

titulo.rellena()

var.cambio="no" *' se ha abierto un archivo de datos, no hay cambios*

ENDIF

finlectura:

END

y cuando salgamos, mediante un If...Then y un Message.Info, se nos preguntara si deseamos salir o no, sin guardar cambios.

PUBLIC SUB Salir_Click()

Form_close()

END

PUBLIC SUB Form_Close()

DIM res AS Integer

IF var.cambio = "si" THEN

res = Message.Question("¿Desea salir sin salvar?", "si", "no")

IF res = 1 THEN

```
ME.Close
ELSE
STOP EVENT 'paramos este evento y no salgo del programa
ENDIF
ELSE
ME.Close
ENDIF
'no salgo del programa
End
```



De esta manera conseguiremos que cada vez que se modifique o altere los datos siempre nos pregunte si queremos salir sin guardar los datos.

!!! Un toque profesional a nuestro programa !!!!



Nota Importante:

Ahora con `Form_Close()` , cuando pulsamos el boton “X” el programa te pregunta si quieres salir o no. El formulario puede tener borde, lo suyo es que sea fijo. **`Fmain.Border=fixed`**

Módulo Filtro. Buscar y Filtrar InStr

Vamos a empezar a utilizar nuestra agenda: buscar los datos que nos interesen...

Para ello vamos a utilizar el TextBoxBuscar donde introducimos las palabras (hasta 3) (completas o no) , separadas por comas. Ejemplo: “Sev,954,43”, buscaremos todos los registros que contengan la parte de la palabra “Sev”, como “sevilla”, “951 954 953” y “942 485 43”, sin importar el orden como aparezcan.

```
PUBLIC SUB TextBoxBuscar_KeyPress()
  IF Key.code = Key.enter OR Key.code = Key.Return THEN
    ButtonBuscar.SetFocus
  endif
```

y el botón ButtonBuscar, que es el que contendrá la llamada al procedimiento de búsqueda en el módulo **Filtrar**.


```
PUBLIC SUB ButtonBuscar_Click()
  DIM c AS String
  DIM resultado AS Integer
  DIM patrones AS NEW String[]
  DIM numero_patrones AS Integer
  numero_patrones = 0
  patrones = Split(Upper$(TextBoxBuscar.text), ",")
  numero_patrones = patrones.COUNT
  IF numero_patrones = 0 THEN
    patrones.Resize(3)
    patrones[0] = ""
    patrones[1] = ""
    patrones[2] = ""
  endif
  IF numero_patrones = 1 THEN
    patrones.Resize(3)
    patrones[1] = ""
    patrones[2] = ""
  endif
  IF numero_patrones = 2 THEN
    patrones.Resize(3)
    patrones[2] = ""
```

```

ENDIF
IF numero_patrones > 3 THEN
    Message.Info("solo se admiten 3 patrones de busqueda como maximo")
    TextBoxBuscar.SetFocus
    GOTO finfiltro
ENDIF

resultado = filtrar.ConceptoFiltro(patrones[0], patrones[1], patrones[2])
IF RESULTADO = 1 THEN
    gridViewDatos.Background = Color.cyan
ELSE
    GridViewDatos.Background = Color.White
ENDIF
finfiltro:
END

```

Como veis cada vez que se devuelva el RESULTADO=1, el gridViewDatos cambiara a color cyan, para demostrar que solo vemos los datos que contienen el filtro. Para anular este filtro crearemos un boton  que haga borrar el filtro y vuelva el color blanco al gridViewDatos.

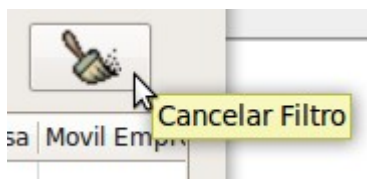
```

PUBLIC SUB ToolButtonCancelarFiltro_Click()
    TextBoxBuscar.text = ""
    ButtonBuscar_Click
END

```

Tambien podemos utilizar la propiedad “ToolTip” para indicar al usuario que hace dicho boton (cuando pasamos por encima del boton salda un mensaje en amarillo)

En el IDE de Gambas, buscamos la propiedad ToolTip del boton ToolbuttonCancelarFiltro y escribimos "Cancelar Filtro"



Por otro lado en el módulo **filtro** crearemos dos subrutinas, las verdaderas responsables de hallar los parecidos y rellenar el gridViewDatos

```

PUBLIC FUNCTION ConceptoFiltro(patron AS String, OPTIONAL patron1 AS String, OPTIONAL patron2 AS String) AS Integer
DIM lineas AS String
DIM a AS Integer

```

```

DIM numero AS Integer
DIM validar AS Boolean

fmain.GridViewDatos.Rows.COUNT = 0
numero = 0 'contador de filas validas (sin estar borradas)
IF patron = "" OR patron = "" THEN
    'dibujar todos los datos
    titulo.rellena()
    RETURN 0
ENDIF

IF patron1 = "" THEN patron1 = " "
IF patron2 = "" THEN patron2 = " "
FOR a = 0 TO var.id.COUNT - 1
    lineas = ""
        Lineas &= var.id[a]
        Lineas &= var.dni[a]
        lineas &= var.nombre[a]
        lineas &= var.apellidos[a]
        lineas &= var.empresa[a]
        lineas &= var.puesto[a]
        lineas &= var.telf_empresa[a]
        lineas &= var.telf_parti[a]
        lineas &= var.fax[a]
        lineas &= var.movil_empresa[a]
        lineas &= var.movil_parti[a]
        lineas &= var.pag[a]
        lineas &= var.foto[a]
        lineas &= var.direccion[a]
        lineas &= var.observaciones[a]
        lineas &= var.fecha_datos[a]
        lineas &= var.correo[a]
    lineas = Upper$(lineas)
    validar = dentro(lineas, patron) AND dentro(lineas, patron1) AND dentro(lineas, patron2)
    IF validar THEN
        numero += 1
        FMain.GridViewDatos.Rows.COUNT = numero
    WITH FMain
        .GridViewDatos[numero - 1, 0].Picture = Picture[var.foto[a]]
        .GridViewDatos[numero - 1, 1].text = var.nombre[a]
    END WITH
END FOR

```

```

.GridViewDatos[numero - 1, 2].text = var.apellidos[a]
.GridViewDatos[numero - 1, 3].text = var.dni[a]
.GridViewDatos[numero - 1, 4].text = var.empresa[a]
.GridViewDatos[numero - 1, 5].text = var.puesto[a]
.GridViewDatos[numero - 1, 6].text = var.telf_empresa[a]
.GridViewDatos[numero - 1, 7].text = var.movil_empresa[a]
.GridViewDatos[numero - 1, 8].text = var.telf_parti[a]
.GridViewDatos[numero - 1, 9].text = var.movil_parti[a]
.GridViewDatos[numero - 1, 10].text = var.fax[a]
.GridViewDatos[numero - 1, 11].text = var.correo[a]
.GridViewDatos[numero - 1, 12].text = var.pag[a]
.GridViewDatos[numero - 1, 13].text = var.direccion[a]
.GridViewDatos[numero - 1, 14].text = var.observaciones[a]
.GridViewDatos[numero - 1, 15].text = var.fecha_datos[a]
.GridViewDatos[numero - 1, 16].text = var.id[a]

END WITH
ENDIF
Finnext:
NEXT
IF numero > 0 THEN
  'SE HA ENCONTRADO PARECIDOS
  RETURN 1
ELSE
  RETURN 0
ENDIF
END

'-----
'funcion dentro
'-----

SUB dentro(frase AS String, patron AS String) AS Boolean
IF patron = " " THEN
  RETURN 1
ELSE
  RETURN InStr(frase, patron)
ENDIF
END

```

La funcion dentro() con la InStrucción **InSr** devuelve si la frase contiene el patron (true o false)

Módulo Ordenar. Ordenar un gridviews con _ColumnClick

Y por que no, vamos a ordenar nuestro gridviews al estilo Excel o Calc. Cuando pulsemos en el titulo de la cabecera de la columna, nos lo ordenara (alfabeticamente).

¿Como lo hacemos?

Vamos a utilizar el evento _ColumnClick, que produce cuando hacemos click en el titulo de la columna. También crearemos una variable **ordenlistado**, que indicara como se ordenara (creciente de la A a la Z, o decreciente de la Z a la A). Esta variable es global, y la crearemos en el modulo **var**.

En el módulo **Ordenar**, crearemos dos procedimientos según el orden creciente (ord_AZ) o decreciente (ord_ZA), y le pasamos a los dos el grid que queremos ordenar (en nuestro caso el Fmain.Gridviewdatos) y la columna (Column), que le da valor al producirse el evento _ColumnClick.

En el módulo **Var**:

```
PUBLIC ordenlistado AS Integer
```

En el módulo **Fmain**:

```
PUBLIC SUB GridViewDatos_ColumnClick(Column AS Integer)
  IF var.ordenlistado = 0 THEN
    Ordenar.ord_AZ(GridViewDatos, Column)
    var.ordenlistado = 1
  ELSE
    var. ordenlistado = 0
    Ordenar.ord_ZA(GridViewDatos, Column)
  ENDIF
END
```

En el módulo **Ordenar**:

' Gambas module file

```
PUBLIC SUB ord_AZ(grid AS GridView, a AS Integer)
  'a: indica la columna a la que ordenamos
  'grid: es la rejilla que ordenamos
  DIM limite AS Integer
  DIM cambio1 AS String
  DIM i AS Integer
```

```

DIM j AS Integer
DIM col AS Integer
grid.visible = FALSE
limite = grid.Rows.COUNT
IF (grid.Columns.COUNT < a + 1) OR a < 0 THEN
    Message.Error("Error en la columna introducida para ordenar")
    GOTO salidaordenar
ENDIF
FOR i = 0 TO limite - 1
    FOR j = 0 TO limite - 2
        IF UCase$(Grid[j, a].Text) > UCase$(Grid[j + 1, a].Text) THEN
            FOR col = 0 TO grid.Columns.COUNT - 1
                cambio1 = Grid[j, col].Text
                Grid[j, col].text = Grid[j + 1, col].Text
                Grid[j + 1, col].text = cambio1
            NEXT
        ENDIF
    NEXT
NEXT
salidaordenar:
grid.visible = TRUE
END

'-----
PUBLIC SUB ord_ZA(grid AS GridView, a AS Integer)
    'a: indica la columna a la que ordenamos
    'grid: es la rejilla que ordenamos
    DIM limite AS Integer
    DIM cambio1 AS String
    DIM i AS Integer
    DIM j AS Integer
    DIM col AS Integer
    grid.visible = FALSE
    limite = grid.Rows.COUNT
    IF (grid.Columns.COUNT < a + 1) OR a < 0 THEN
        Message.Error("Error en la columna introducida para ordenar")
        GOTO salidaordenar
    ENDIF

```

```
FOR i = 0 TO limite - 2

  FOR j = i TO limite - 1
    IF UCase$(Grid[i, a].Text) <= UCase$(Grid[j, a].Text) THEN
      FOR col = 0 TO grid.Columns.COUNT - 1
        cambio1 = Grid[j, col].Text
        Grid[j, col].text = Grid[i, col].Text
        Grid[i, col].text = cambio1
      NEXT
    ENDIF
  NEXT
NEXT
salidaordenar:
grid.visible = TRUE
END
```

Varias cosas a comentar:

1. El método de ordenación elegido es el llamado “Burbuja”.

En internet podéis encontrar la explicación de este algoritmo de ordenación

(yo lo saque de allí 🤖)

2. No diferencio entre mayúsculas o minúsculas, usando **UCase\$**, convierto las cadenas a mayúsculas y las comparo. Si hago click en la columna de las imágenes, se ordenan por la ruta donde estén ubicadas.

Módulo Importar y Exportar desde el Portapapeles

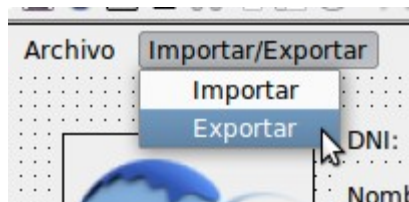
Reutilizar los datos de otros programas o utilizar los nuestros en otros programas, es una de las cosas que todos nuestro programas deben de tener para así ser mas util. ¿os imagináis que cada vez que tengais el listin con un monton de datos, no podais utilizarlo para por ejemplo, aprovechar la direccion de cada uno?? ¿tendrias que volverlo a escribir???

Bueno pues en con este módulo vamos a utilizar la forma mas sencilla que Existe de pasar datos de una programa a otro: EL PORTAPAPELES.

Exportar: *Clipboard.Copy*

En su dia creamos el modulo exportar, es ahora cuando vamos a añadir el codigo necesario para transferir la informacion del gridViewDatos al portapapeles.

En nuestro Fmain creamos un menu, elegimos exportar:



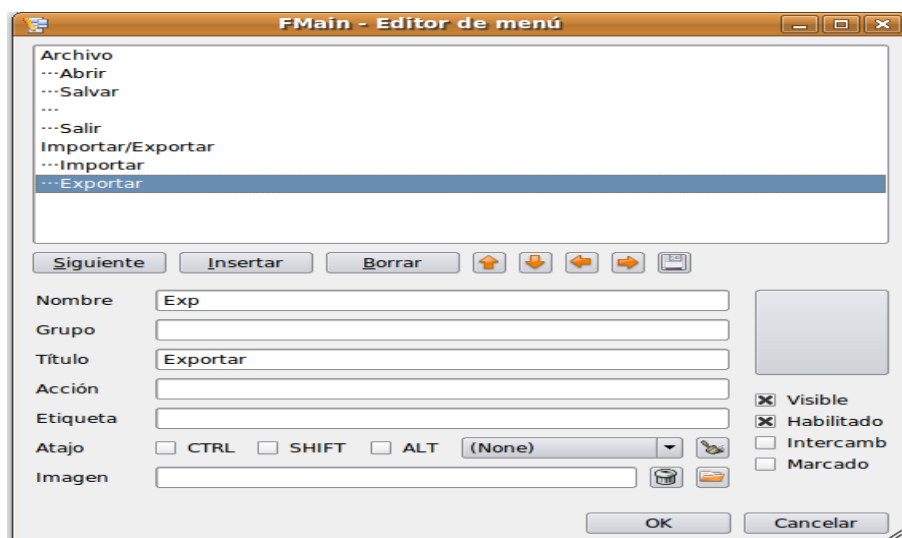
Y definimos el siguiente codigo en **Fmain**:

```
PUBLIC SUB Exp_Click()  
  exportar.copiaraportapapeles(gridViewDatos)  
END
```



Nota:

A la hora de definir el editor de menus, así se definiria el menu/importar exportar / exportar



Y en el módulo **Exportar**:

```
PUBLIC SUB copiaraportapapeles(rejilla AS GridView)
```

```
    DIM texto AS String
```

```
    DIM a AS Integer
```

```
    DIM b AS Integer
```

```
    IF rejilla.header = 1 OR rejilla.header = 3 THEN
```

```
        FOR a = 0 TO rejilla.Columns.COUNT - 1
```

```
            texto &= rejilla.Columns[a].Title & "\t"
```

```
        NEXT
```

```
        texto &= "\n"
```

```
    ENDIF
```

```
    FOR a = 0 TO rejilla.Rows.COUNT - 1
```

```
        FOR b = 0 TO rejilla.Columns.COUNT - 1
```

```
            texto &= revisa(rejilla[a, b].text) & "\t"
```

```
        NEXT 'b
```

```
    texto &= "\n"
```

```
    NEXT 'a
```

```
    Clipboard.Copy(texto)
```

```
END
```

```
-----
PUBLIC SUB revisa(cadena AS String) AS String
```

```
    DIM a AS Integer
```

```
    DIM letra AS String
```

```
    DIM devuelta AS String
```

```
    DIM prueba AS Integer
```

```
    'compruebo si tiene solo numero
```

```
    FOR a = 1 TO Len(cadena)
```

```
        letra = Mid$(cadena, a, 1)
```

```
        prueba = InStr("0123456789.", letra)
```

```
    IF prueba = 0 THEN
```

```
        RETURN cadena
```

```
    'contiene letras
```

```
    ENDIF
```

```
    NEXT
```

```

FOR a = 1 TO Len(cadena)
  letra = Mid$(cadena, a, 1)
  IF letra = "." THEN
    letra = ","
  ENDIF
  devuelta &= letra
NEXT
RETURN devuelta
END

```

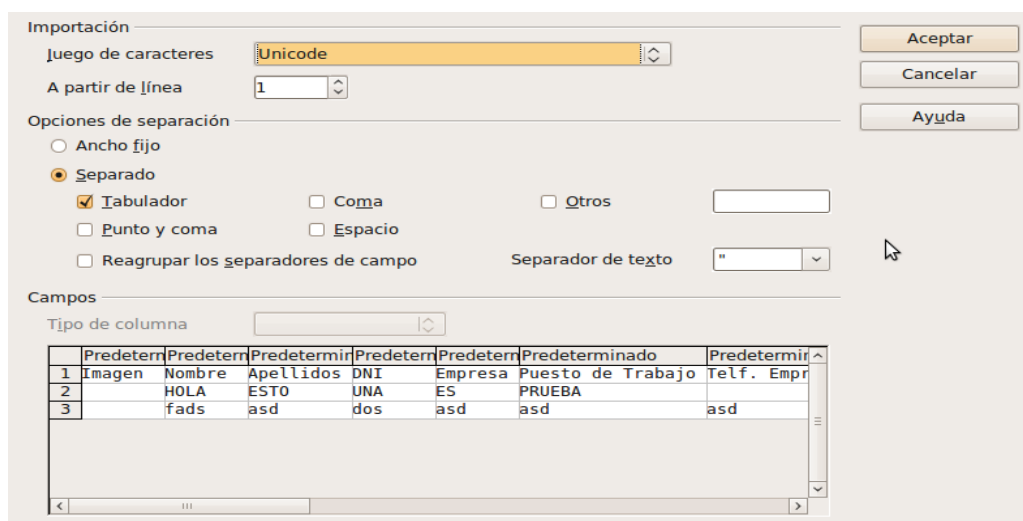
Como vemos, se divide en dos subrutinas:

La primera **“copiaraportapapeles”**, se le pasa la rejilla que queremos que copie (rejilla **AS GridView**), y va leyendo los valores de los títulos de columnas y de las distintas filas, añadiéndolo a una variable llamada **texto**, y con el comando **Clipboard.Copy(texto)**, lo pasa al portapapeles.

La segunda subrutina, **“revisa”** convierte “el punto decimal” en coma decimal, para que por ejemplo una hoja de calculo lea correctamente los numeros.

Una vez pulsado en el menu “Exportar”, el contenido gridViewDatos actual (si esta filtrado, solo lo filtrado), pasa al portapapeles, pudiendose pegar en (por ejemplo openoffice Calc):

Abrimos openoffice Calc, pulsamos en el menu “Editar/Pegar”, nos saldra esta pantalla:



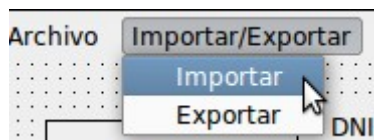
Pulsamos “Aceptar”, y obtenemos:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Imagen	Nombre	Apellidos	DNI	Empresa	Puesto de Trabajo	Tel. Empresa	Móvil Empresa	Tel. Particular	Móvil Particular	Fax	Correo	Página Web	Dirección	Observación	Fecha	
2		HOLA	ESTO	UNA	ES	PRUEBA	periquito	asd	asd	asd	asd	asd	asd	sf	asd	asdf	Id19/07/2019
3		fads	asd	dos	asd	asd	asd	asdf	asd	asd	asd	asd	asd				Id19/07/2019
4																	
5																	

Importar: *Clipboard.Paste*

Para importar datos, también vamos a utilizar el Clipboard.Paste, por ejemplo lo vamos a hacer para que desde una hoja de cálculo podamos volcar los datos en nuestro listín.

En el formulario Fmain, pulsamos en el menú Importar/Exportar / Importar



y escribimos el siguiente código:

```
PUBLIC SUB Imp_Click()
  FOpenOfficeCalcGambas2.Show()
END
```

Crearemos un nuevo formulario: **FopenOfficeCalcGambas2**

y el siguiente código que se ejecutará cada vez que lo abramos:

```
PUBLIC SUB Form_Open()
  GridViewLectura.Rows.COUNT = 1001
  GridViewLectura.Columns.COUNT = 20
END
```

 A screenshot of the Gambas form 'Fmain'. At the top, there is a button labeled 'Pulse para leer del portapapeles' and a checkbox labeled 'primera fila contiene titulos'. Below these is a data grid named 'GridViewLectura' with a header row containing the following columns: 'Imagen', 'Nombre', 'Apellidos', 'Dni', 'Empresa', 'Puesto', 'Telf.Emp', 'Movil.Emp', 'Telf.Part', 'Movil Part.', 'Fax', 'Correo', 'Paq', 'Direc', and 'Observ.'. The grid is currently empty. At the bottom of the form, there are two buttons: 'Añada datos al listín' and 'Cancelar'.

El CheckBoxTitulos, si esta “checkqueado” quera decir que el contenido del portapapeles, en su primera linea contiene los titulos (cabeceras de los datos) o no (solo los datos). Leeremos esa propiedad cuando nos interese. (**CheckBoxTitulos.value**)

El boton “Pulse para leer del portapapeles” ejecuta el codigo de lectura del portapapeles

El boton “Añadir datos al listin” es el encargado de pasar los datos del gridviewlectura a la memoria.

El boton “Cancelar” cancelara todo el proceso.

Su codigo es asi de simple:

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

Bueno, pues vamos por partes:

Código de “Pulse para leer del portapapeles” : Clipboard.Paste()

```
PUBLIC SUB leerportapapeles_Click()
DIM titulos AS Boolean
IF CheckBoxTitulos.value = -1 THEN
    FOpenOfficeCalcGambas2.GridViewLectura.Header = 1
    titulos = -1
ELSE
    FOpenOfficeCalcGambas2.GridViewLectura.Header = 0
    titulos = 0
ENDIF
GridViewLectura.Rows.COUNT = 0
GridViewLectura.Columns.COUNT = 0
GridViewLectura.Rows.COUNT = 1001
GridViewLectura.Columns.COUNT = 20
leer(FOpenOfficeCalcGambas2.GridViewLectura, titulos)
END
```

y la subrutina leer:

```
'----- leer rejilla -----
PUBLIC SUB leer(rejilla AS gridview, titulos AS Boolean)
'rejilla: es el gridview donde se van a escribir los datos
'titulos: 0: sin titulo, -1 titulo contiene la 1º fila
DIM lineas AS String[]
DIM linea_procesada AS String
DIM columnas AS String[]
```

```

DIM a AS Integer
DIM c AS Integer
DIM cadena AS String
DIM portapapeles AS String
DIM finlinea AS String
DIM fincolumna AS String
finlinea = "\n" ' retorno de carro (separa las filas)
fincolumna = "\t" ' tabulador (separa las columnas)
portapapeles = Clipboard.Paste()
lineas = Split(portapapeles, finlinea)
FOR a = 0 TO lineas.COUNT - 1
    linea_procesada = lineas[a]
    columnas = Split(linea_procesada, fincolumna)
    FOR c = 0 TO columnas.COUNT - 1
        IF a = 0 AND titulos = -1 THEN
            rejilla.Columns[c].title = columnas[c]
        ELSE
            IF titulos = 0 THEN rejilla[a, c].text = columnas[c]
            IF titulos = -1 THEN rejilla[a - 1, c].text = columnas[c]
        ENDIF
    NEXT 'c
NEXT 'a
END

```

A comentar la orden más importante es `portapapeles=Clipboard.Paste()`

y con la orden `Split()` iremos separando líneas de texto con retorno de carro “\n” y los datos separados por tabulador “\t”

Código para “Añadir datos al listín”

```

PUBLIC SUB ButtonAnadir_Click()
DIM a AS Integer
DIM repite AS Integer
IF CheckBoxTitulos.Value = TRUE THEN
    'la primera linea contiene titulos
    a = 1
ENDIF
FOR repite = a TO 1001

```

```

IF comprobarvacio(repite) <> 1 THEN
    var.id.add("Id" & Str$(Now) & Str$(a))
    var.dni.add(GridViewLectura[repite, 3].text)
    var.nombre.add(GridViewLectura[repite, 1].text)
    var.apellidos.add(GridViewLectura[repite, 2].text)
    var.empresa.add(GridViewLectura[repite, 4].text)
    var.puesto.add(GridViewLectura[repite, 5].text)
    var.telf_empresa.add(GridViewLectura[repite, 6].text)
    var.telf_parti.add(GridViewLectura[repite, 8].text)
    var.fax.add(GridViewLectura[repite, 10].text)
    var.movil_empresa.add(GridViewLectura[repite, 7].text)
    var.movil_parti.add(GridViewLectura[repite, 9].text)
    var.pag.add(GridViewLectura[repite, 12].text)
' la imagen en el unico dato que no leemos, solo le añadimos el simbolo de la gambita
    var.foto.add("icon:/16/gambas")
    var.direccion.add(GridViewLectura[repite, 13].text)
    var.observaciones.add(GridViewLectura[repite, 14].text)
    var.fecha_datos.add(GridViewLectura[repite, 15].text)
    var.correo.Add(GridViewLectura[repite, 11].text)
ELSE
    GOTO finlectura
ENDIF
NEXT
finlectura:
titulo.rellena()
Message.Info("Datos cargados")
ME.Close
END
'-----si la fila no contine datos (todos son "") entonces devuelve 0 -----
PUBLIC FUNCTION comprobarvacio(fila AS Integer) AS Integer
DIM b AS Integer
FOR b = 0 TO 19 'tantas veces como columnas haya, para no ingresar registros en blanco
    IF GridViewLectura[fila, b].text <> "" THEN
        RETURN 0
    ENDIF
NEXT
RETURN 1
END

```

A comentar:

1. La funcion **comprobarvacio**, hace que cuando la linea del gridviewlectura esta completamente vacia (sin texto en ninguna de sus columnas) devuelve 0, y el procedimiento principal **ButtonAnadir_Click()**, dejara de añadir información, en caso contrario devuelve 1, si sigue añadiendo el registro.

```
( IF comprobarvacio(repite) <> 1 THEN .... )
```

2. ' la imagen en el unico dato que no leemos, solo le añadimos el simbolo de la gambita

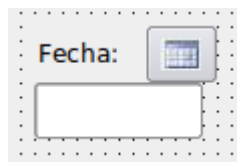
```
var.foto.add("icon:/16/gambas")
```

ya que deberia de ser una ruta, en caso de que no sea una ruta valida provocaria un error que saldria del programa. Os lo dejo como tarea de casa si queréis mejorar esta parte del código....

Formulario y Módulo: Fechas

Ante lo incomodo de introducir una fecha y la distintas formas de introducirlas (la podemos poner en formato español 01/12/2010, o americano 2010/12/01, o “uno de diciembre del dosmil diez”) vamos a mejorar la forma de introducir la fecha de nuestros datos, con un pequeño formulario que nos pasara al formato años/mes/dia (mejor para ordenar por fecha), fácilmente viendo un calendario.

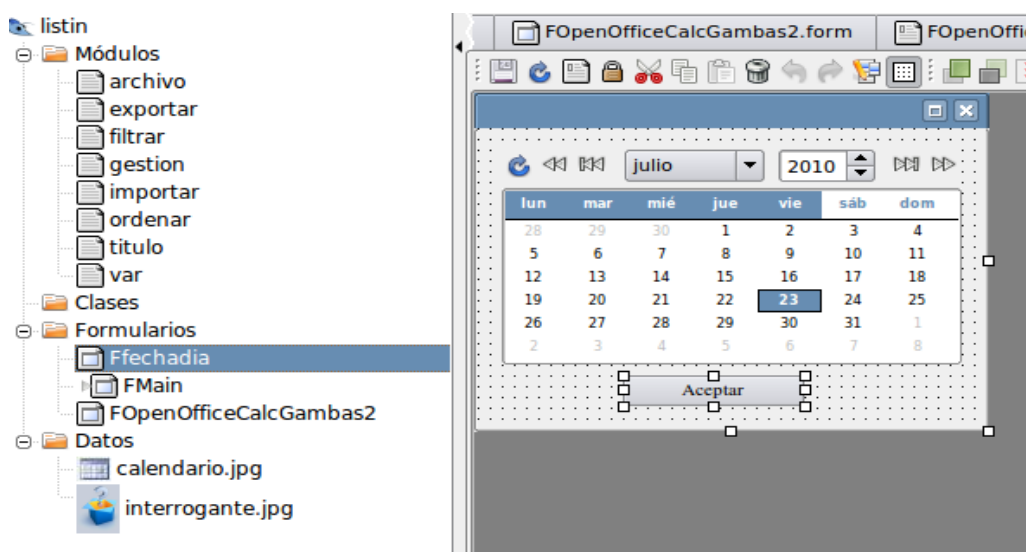
Para ello crearemos un pequeño boton **ToolButtonCalendario**,



que al ser pulsado llame a un nuevo formulario **Ffechadia.pregustarhastafecha**,

```
PUBLIC SUB ToolButtonCalendario_Click()
    Ffechadia.Preguntarhastafecha
END
```

Creamos un nuevo formulario, la llamamos **Ffechadia**, con un boton “aceptar” y un **datechooser** que le llamaremos datechooserfecha, donde eligeremos la fecha.



Y con el siguiente codigo, dentro del formulario Ffechadia:

```
' Gambas class file
PUBLIC SUB DateChooserfecha_Change()
    ME.Caption = DateChooserfecha.Value
END
PUBLIC SUB Form_Open()
    ME.Caption = DateChooserfecha.Value
```



```
END
PUBLIC SUB Preguntarhastafecha()
    Ffechadia.ShowModal
END

PUBLIC SUB ButtonAceptar_Click()
    FMain.TextBoxFecha.text = Format(Ffechadia.DateChooserfecha.Value, "yyyy/mm/dd")
    ME.Close
END
```

La función más interesante es **Format**, que hace que la fecha que lee del **DateChooserfecha**, la convierta en formato año/mes/día "yyyy/mm/dd"

Anexos

Anexo 1: Galería de Fotos del Recuerdo

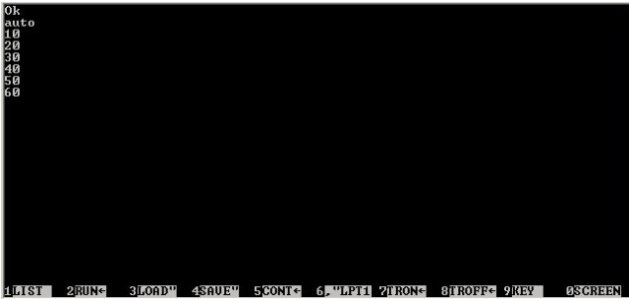
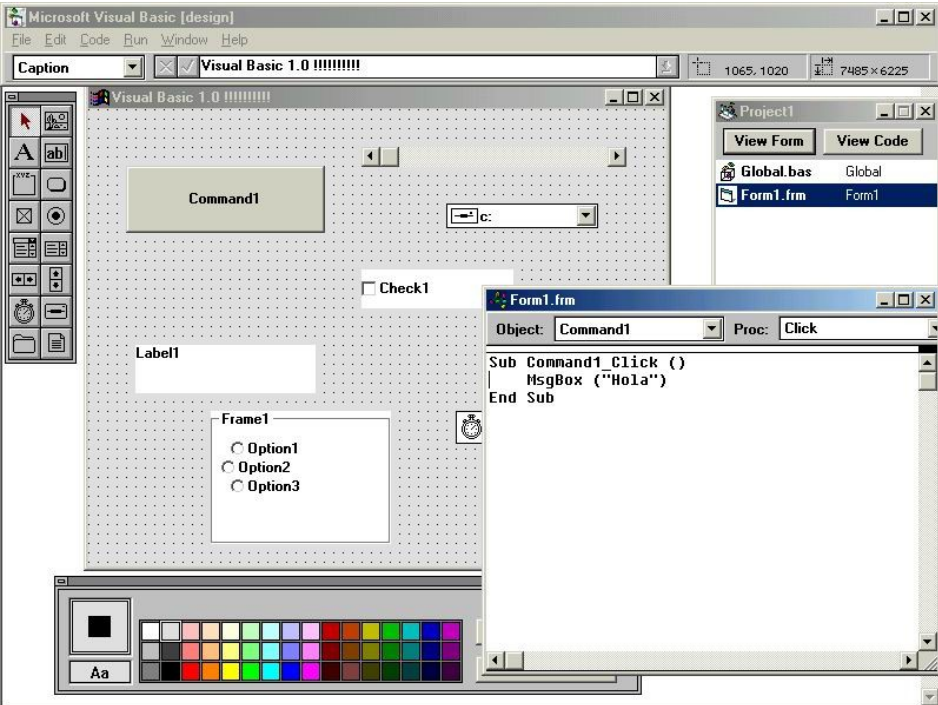
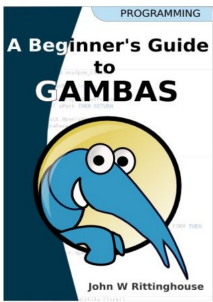
Foto	Observaciones
	Gw-Basic
	Visual Basic 3.0
	

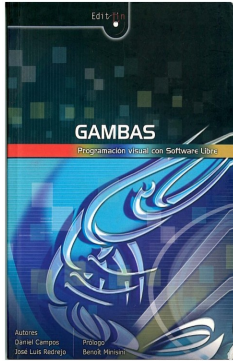

Foto	Observaciones
http://mygnet.net/manuales/gambas/manual_para_gambas_ingles.524	<p>Manual de Gambas en Ingles</p>
 <p>http://www.gambas-es.org/dload.php?action=file&file_id=1</p>	<p>Manual de Gambas en Español</p>
 <p>http://www.wikilearning.com/tutorial/desarrollo_de_aplicaciones_con_gambas/6333</p>	<p>Manual Basic de Desarrollo de aplicaciones en Gambas</p>

Foto	Observaciones
 <p>Desarrollo de aplicaciones con Gambas.</p> <p>Tutorial y ejemplo de un programa hecho con Gambas.</p> <p>Sumario: Vamos a crear una aplicación sencilla con Gambas. Veremos cómo se programan los eventos y algunos trucos y tré el magnífico entorno de desarrollo.</p> <p>David Asorey Álvarez. Febrero de 2005.</p> <ul style="list-style-type: none"> • Introducción • Primeros pasos • Gestión de eventos • Consideraciones relativas al diseño de formularios • Al grano ... • Acción "Limpiar" • Acción "Añadir" • Acción "Modificar" • Acción "Borrar" • Acción "Salir" • Acción "Abrir" • Acción "Guardar" • Un último ajuste • Nuestro programa funcionando • Distribuyendo nuestra aplicación • Conclusiones • Acerca de este documento y del autor • Notas <p>Introducción</p> <p>Gambas es una herramienta de desarrollo visual de aplicaciones muy similar a los conocidos programas comerciales Microsoft Delphi.</p> <p>Con Gambas se pueden hacer aplicaciones o programas con interfaz gráfica de forma muy rápida, pues integran un diseñador de ventanas, un editor de código, un explorador de clases, un visor de ayuda, etc.</p> <p>Este tipo de herramientas han sido siempre muy habituales en la plataforma Microsoft Windows, pero para Linux no existían tan depuradas. Podemos encontrar Kdevelop, Kylix o VDK Builder. Hay que destacar que en el desarrollo de aplicaciones en Linux se acostumbra a emplear muchas herramientas de desarrollo, como el compilador GCC, el linker, etc.</p>	<p>David Asorey</p> <p>http://davidasorey.net/static/gambas-tutorial/gambas_tutorial_es.html</p>

Anexo 2: convert, miniaturas en nuestro gridViewDatos

En nuestro gridViewDatos, vemos que no podemos ver totalmente las imágenes, ¿por que no hacemos pequeñas miniaturas de ellas y las podemos así ver mejor??? linux tiene un comandos y programas free para ello.... Manos a la obra:

El programa **convert** nos ayudara a reducir las imágenes cargadas al tamaño necesario para nuestro gridViewData,

Instalación:

Desde la consola del sistema escribimos “convert”, si no lo tuviésemos instalado habría que instalar el programa:

```
$ convert
```

El programa «convert» puede encontrarse en los siguientes paquetes:

- * imagemagick

- * graphicsmagick-imagemagick-compat

Pruebe: `sudo apt-get install <paquete seleccionado>`

```
bash: convert: orden no encontrada
```

```
$ sudo apt-get install imagemagick
```

Nos pedirá que confirmemos la instalación del paquete , y se instalara.

Uso en nuestro programa de convert: SHELL

Una vez que hemos instalado el programa, debemos incluir el código que haga:

- cada vez que introducimos o editamos un registro a la hora de rellenar el gridViewDatos, queremos que en vez de aparecer la imagen elegida, aparezca una imagen reducida (por ejemplo de 120x120 pixeles), en nuestra gridViewData.
- Desde la línea de comandos para “redimensionar” una imagen a 120x120, lo haríamos de la siguiente manera

```
$ convert -size 120x120 lena512.bmp -resize 120x120 +profile '*' thumbnailDD.jpg
```

”De esta manera, “-size 120x120” da una idea al decodificador JPEG que la imagen se va a escalar a menor a 120x120, lo que le permite correr mas rápido, evitando el retorno de una imagen de alta resolución. El “-resize 120x120” especifica las dimensiones deseadas de la imagen de salida. Será escala por lo que su mayor dimension es de 120 pixeles. El “” en “+profile” elimina cualquier ICM,EXIF,IPTC, o perfiles de otros que pueden estar presentes en la entrada y no son necesarios en la miniaturas.”*

Al ejecutar este comando la imagen “lena512.bmp” de 257,1 kib, la hemos convertido en otra llamada “thumbnailDD.jpg” de solo 3.7 kib.

Bien, y como lo hacemos en gambas2, pues para ello tenemos la orden “SHELL”

Crearemos un nuevo modulo al que llamaremos **miniaturas**, en el insertaremos este código:

```

public function hacer(fichero AS String, tamano AS String) as string
DIM Proc AS process
DIM linea AS String
DIM mini AS String
'comprobar si Existe directorio de miniaturas
IF Exist("~/listin/mini") THEN
' Existe directorio de miniaturas, no necesito crearlo
ELSE
'no Existe el directorio de miniaturas, hay que crearlo...
'primero del que cuelga
MKDIR "~/listin"
'y luego en "mini"
MKDIR "~/listin/mini"
ENDIF
'nota:
'comando: file.dir(path), saca solo la ruta (sin el nombre del fichero)
'comando: file.name(path), saca solo el el nombre del archivo (sin la ruta)
mini = "~/listin/mini" & "/mini" & file.name(fichero)
IF Exist(mini) then
    'Existe el archivo mini, no lo tengo que crear
    RETURN mini
ELSE
    'tamano puede ser 120x120, 96x96, etc
    linea = "convert -size " & tamano & " " & fichero & " -resize " & tamano & " +profile '*' "
    & mini
    proc = SHELL linea WAIT
    'ejecutar el proceso y controlar el flujo de salida en tu aplicación
    return mini
ENDIF
END

```

Notas:

1- La funcion **miniaturas.hacer()** ademas de realizar el redimensionado de la imagen devuelve la

ruta donde se crea esa miniatura.

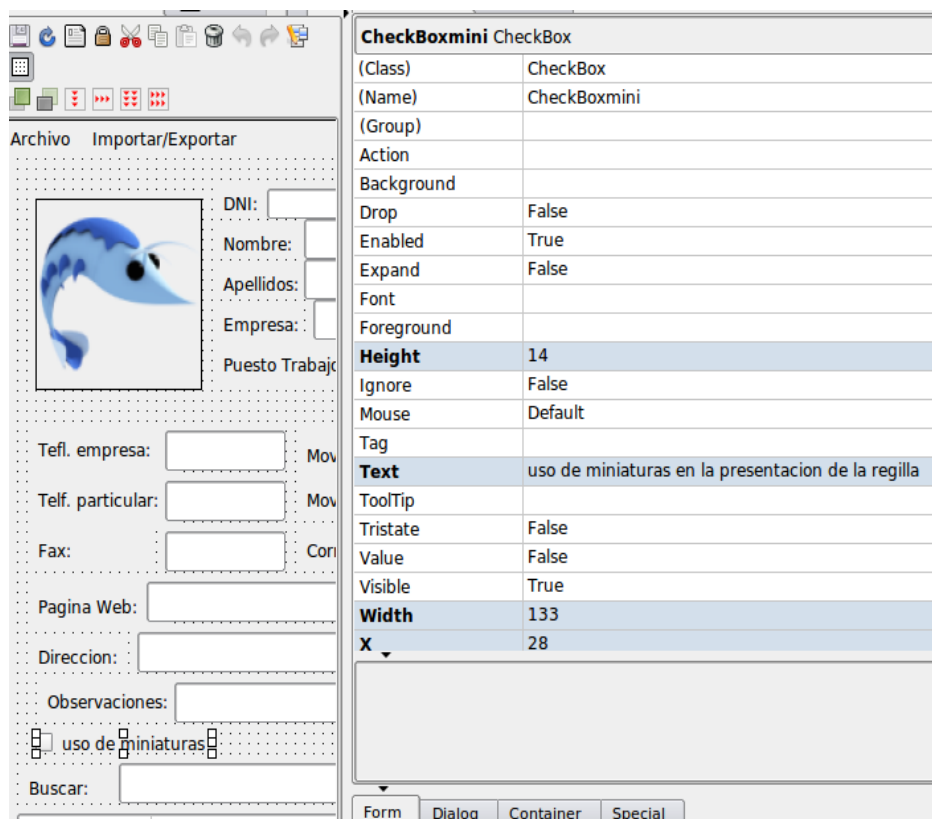
2- “~/I” : el carácter ~ se reemplaza por el directorio home del usuario, indistintamente de que usuario sea, “/home/julio” o “/home/juan”.

3- **Exist()**: **Exist("~/listin/mini")**: En caso de no Existir, vamos a crear un directorio desde nuestro home/usuario, llamado listin/mini, donde de guardaran todas la miniaturas que hagamos.

Exist(mini): En caso de que Exista el archivo miniatura, no vamos a ejecutar **convert**

4. **SHELL**: esta orden ejecuta desde la linea de comando el programa **convert** con todos los atributos que hemos añadido (ruta del archivo, tamaño, y nombre nuevo de la miniatura)

Bien, una vez desarrollado el codigo, y creado el nuevo modulo, vamos a crear un **checkbox** (checkboxmini) en el formulario **Fmain**, para que el usuario “active o desactive” la opcion de utilizar miniaturas en el gridViewDatos, y asi modificaremos tambien la subrutina “titulo.relleno()", para que muestre las miniaturas o los archivos originales.



Tenemos que poner en el formulario **Fmain**, que cuando cambie el valor del checkboxmini, se ejecute tambien la subrutina Titulos.Rellena()

```
PUBLIC SUB CheckBoxmini_Click()
    titulo.rellena()
END
```

Y en el código de **Titulo.Rellena()**:

```
PUBLIC SUB rellena()
DIM a AS Integer
FMain.GridViewDatos.Rows.COUNT = var.id.COUNT
FOR a = 0 TO var.id.COUNT - 1
  WITH FMain
    IF FMain.CheckBoxmini.Value = FALSE THEN
      .GridViewDatos.Rows.height = 25
      .GridViewDatos[a, 0].Picture = Picture[var.foto[a]]
    ELSE
      .GridViewDatos[a, 0].Alignment = 1
      .GridViewDatos.Rows.height = 100
      .GridViewDatos[a, 0].Picture = Picture[miniaturas.hacer(var.foto[a], "96x96")]
    ENDIF
    .GridViewDatos[a, 1].text = var.nombre[a]
    .GridViewDatos[a, 2].text = var.apellidos[a]
  .....
END SUB
```

Nota: La parte amarilla es la que se ha modificado, haciendo que cuando el checkboxmini.value=false dibuje la imagen normal, y cuando sea True realice la miniatura y dimensione la alto de la fila para que se vea la imagen.

Nota importante: En todos los sitios que se rellene el gridViewDatos, hay que añadir el mismo código.


También en el módulo de **filtrar.ConceptoFiltro**, hay que hacer el mismo cambio:

```
.....
.....
validar = dentro(lineas, patron) AND dentro(lineas, patron1) AND dentro(lineas, patron2)
IF validar THEN
  numero += 1
  FMain.GridViewDatos.Rows.COUNT = numero
  WITH FMain
    IF FMain.CheckBoxmini.Value = FALSE THEN
      .GridViewDatos.Rows.height = 25
      .GridViewDatos[numero-1, 0].Picture = Picture[var.foto[a]]
    ELSE
      .GridViewDatos[numero-1, 0].Alignment = 1
      .GridViewDatos.Rows.height = 100
      .GridViewDatos[numero-1, 0].Picture = Picture[miniaturas.hacer(var.foto[a],
"96x96")]
    ENDIF
    .GridViewDatos[numero - 1, 1].text = var.nombre[a]
  .....
END SUB
```


Aquí os pongo un ejemplo de como los registros sin y con miniaturas:

Sin miniaturas


Archivo Importar/Exportar



DNI:
Nombre:
Apellidos:
Empresa:
Puesto Trabajo:

✓ Aceptar

✗ Cancelar

 Borrar

Telf. empresa: Movil empresa:
Telf. particular: Movil Particular:
Fax: Correo @:
Fecha:
Pagina Web:
Direccion:
Observaciones:
☐ uso de miniaturas en la presentacion de la regilla






Buscar:  Buscar 

	Imagen	Nombre	Apellidos	DNI	Empresa	Puesto de Trabajo	Telf. Empresa	Movil E
1		Chica	Guapa	000	en la mia no	PRUEBA		periqui
2		super	iker		el numero U	seleccion es	porter	asdf

Con miniaturas


Archivo Importar/Exportar



DNI:
Nombre:
Apellidos:
Empresa:
Puesto Trabajo:

✓ Aceptar

✗ Cancelar

 Borrar

Telf. empresa: Movil empresa:
Telf. particular: Movil Particular:
Fax: Correo @:
Fecha:
Pagina Web:
Direccion:
Observaciones:
☒ uso de miniaturas en la presentacion de la regilla





Buscar:  Buscar 

	Imagen	Nombre	Apellidos	DNI	Empresa	Puesto de Trabajo	Telf. Empresa	Movil E
1		Chica	Guapa	000	en la mia no	PRUEBA		periqui
2		super	iker		el numero U	seleccion es	porter	asdf



Mucho mejor con miniaturas....¿verdad?

Anexo 3: Introducción al Gambas. Tipo de variables y datos:

Nota: Mezcla de varios artículos aparecidos en la web. Autores: [Alfonso Martínez García](#)

Una **variable** es un elemento que se utiliza para almacenar distintos tipos de datos, como lo son números, letras o ambos, el valor que se almacena en dicha variable puede cambiar en cualquier momento durante la ejecución de un programa, a diferencia de las constantes en donde el valor se mantiene durante la ejecución de dicho programa.

Básicamente hay dos lugares donde se pueden declarar las variables en gambas esto dependiendo del uso que se le quiere dar a la variable. Se pueden declarar dentro de una subrutina o función, esta ultima declaración de variables solo tendrán uso en esa subrutina o función Si se declaran en la parte inicial ya sea en un modulo o clase estarán disponibles esas variables para ese archivo en todas sus funciones o subrutinas.

Para usar variables en gambas tenemos que declararlas, para esto Existen varias maneras.

Declaración de variables locales

[DIM] Identificador AS TipoDeDato

Con esta sintaxis declaramos una variable dentro de un procedimiento o función, esta variable es solo accesible dentro de la función o procedimiento donde fue declarada.

Declaración de una variable “val” de tipo entero

DIM Val AS integer

Declaración de una variable “NAME” de tipo cadena de caracteres

DIM NAME AS String

Declaración de una matriz de tres por tres “Matriz” del tipo float (punto flotante)

DIM Matriz[3,3] AS Float

Declaración de una variable “nObject” de tipo Objeto

DIM nObject AS Object

***[STATIC] (PUBLIC | PRIVATE) Identificador [Array Declaration] AS
[NEW] TipoDeDato***

Esta sintaxis sirve para declarar una variable global en una clase, la variable declarada es accesible desde cualquier lugar en la clase o módulo donde fue declarada

Si la declaración se hace con la palabra PUBLIC, la variable sera accesible desde otras clases que hagan referencia a un objeto de la clase donde la variable fue declarada.

Si lleva la palabra STATIC la variable sera compartida por todos los objetos de la clase. Si se incluye la palabra NEW, la variable es inicializada con una nueva instancia de la clase especificada con TipoDeDato.

Subrutinas o funciones

Subrutina: Es un procedimiento que ejecuta algo pero no devuelve ningún tipo de valor se identifican en gambas por la palabra Reservada SUB y no tiene RETURN.

Función: Es un procedimiento que devuelve algo se identifica por la palabra reservada FUNCTION y tiene RETURN.

Tipos de datos.

Los tipos de datos que se utilizan en gambas son los siguientes:

Boolean: admite dos valores Verdadero (true) y Falso (false)

Byte: admite valores enteros entre 0 y 255

Short: admite valores enteros entre -32768 y +32767

Integer: admite valores enteros entre -2147483648 y +2147483647

Float: admite valores de tipo flotante, equivalente al double de C

Date: admite valores de tipo fecha y tiempo. Internamente la fecha y la hora se almacenan en formato UTC.

String: Se usa para almacenar una cadena de texto. admite valores que incluye tanto letras como números

Variant: admite cualquier tipo de valor

Object: para declarar variables que hagan referencia a un objeto

Determinar que tipo de dato almacena una variable.

IsBoolean(expresión): devuelve TRUE si la expresión es un valor booleano

IsDate(expresión): devuelve TRUE si la expresión almacena un valor DATE

IsFloat(expresión): devuelve TRUE si la expresión almacena un tipo de dato flotante

IsInteger(expresión): devuelve TRUE si la expresión almacena un tipo de dato entero

IsNull(expresión): devuelve TRUE si la expresión es NULA

IsNumber(expresión): devuelve TRUE si la expresión almacena un tipo de dato que sea número

IsObject(expresión): devuelve TRUE si la expresión almacena un OBJETO o una referencia nula

IsShort(expresión): devuelve TRUE si la expresión almacena un tipo de dato short

IsString(expresión): devuelve TRUE si la expresión almacena un tipo de dato string

Conversión de tipos de datos

Las funciones que tiene gambas para la conversión de distintos tipos de datos:

1. Cbool(expresión): Convierte la expresión a un valor booleano.

Ejemplo Cbool(expresión) te devolverá 'true' o 'false'.

2. Cshort(expresión), Cint(expresión), Cinteger(expresión), Clong(expresión): convierte la expresión en short, int o en long.

3. Cdate(expresión) Convierte una expresión en un valor
4. CStr(expresión) Convierte una expresión en una cadena.
5. Val(expresión) Convierte una cadena en un booleano, número o fecha, de acuerdo con el contenido de la cadena.
6. Str\$(expresión) Convierte la expresión en una cadena de texto.

Anexo 4: Empaquetado/Desempaquetado de datos

"Houston, Houston , tenemos un problema"

Como os comente a la hora de guardar los datos, solo se guardaban las rutas de las imagenes y no la imagen en si (bueno, solo las miniaturas se guardan en "~/listin/mini").

Normalmente eso es efectivo, pero...

¿que ocurre si cambiamos de ordenador?

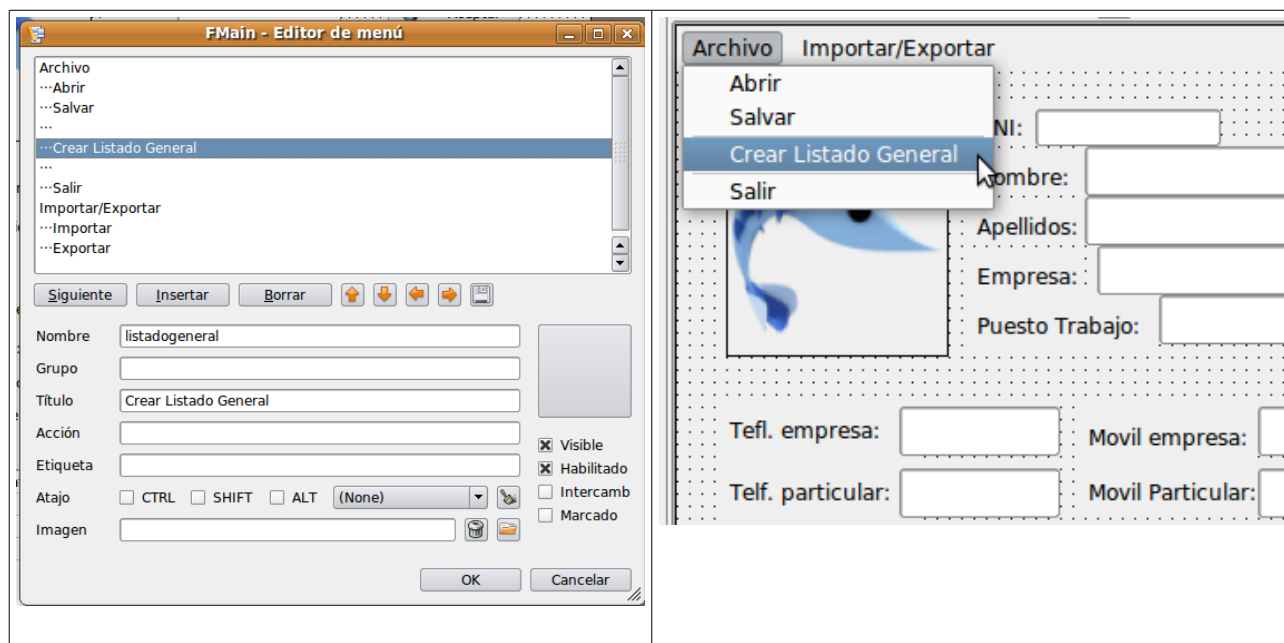
¿y si las imagenes “grandes” las tenemos en un disco duro externo?

¿podemos hacer una copia de seguridad de todos los dastos?

Pues bien para resolver estos problemas lo que vamos es a realizar es:

- Crear y copiar en una carpeta "~/listin/fotos", todas las fotos con su tamaño original
- modificar las rutas de las fotos y copiar los datos en una carpeta que llamaremos "~/listin/datos"
- crear un archivo "ListadoGeneral.lis", que tendra todos los datos accesibles al contenido de las carpetas "~/listin"

Para ello, en el formulario **Fmain** vamos a crear una nueva opcion en el menu que llamaremos "Crear Listado General"



Con el siguiente código:

```
PUBLIC SUB listadogeneral_Click()  
    importar.completo()  
END
```

Ahora en el módulo **importar**, crearemos el procedimiento completo()

```
SUB completo()
'defino variables locales
DIM a AS Integer
DIM destino AS String
DIM lineas AS String
DIM codigofinline AS String
DIM fotoNuevaRuta AS NEW String[]

IF Exist("~/listin/") THEN
  ' Existe directorio , no necesito crearlo
ELSE
  'no Existe el directorio s, hay que crearlo...
  MKDIR "~/listin/"
ENDIF

'Crear y copiar en una carpeta "~/listin/fotos", todas las fotos con su tamaño original
IF Exist("~/listin/fotos") THEN
  ' Existe directorio , no necesito crearlo
ELSE
  'no Existe el directorio , hay que crearlo...
  MKDIR "~/listin/fotos"
ENDIF

'empiezo a copiar todas las fotos....
FOR a = 0 TO var.dni.COUNT - 1
  'comando copiar desde consola
  IF Exist("~/listin/fotos/" & File.NAME(var.foto[a])) THEN
    'Existe no la copio
  ELSE
    COPY var.foto[a] TO "~/listin/fotos/" & File.NAME(var.foto[a])
  ENDIF
NEXT

'modificar las rutas de las fotos y copiar los datos en una carpeta que llamaremos "~/listin/datos"
fotoNuevaRuta.Resize(var.dni.COUNT)
FOR a = 0 TO var.dni.COUNT - 1
  fotoNuevaRuta[a] = "~/listin/fotos/" & File.NAME(var.foto[a])
NEXT
```

'crear un archivo "ListadoGeneral.lis", que tendra todos los datos accesibles al contenido de las carpetas "~/listin"

destino = "~/listin/ListadoGeneral.lis"

codigofinline = ""

Lineas = "v0.0.1" & codigofinline 'informo version

Lineas &= "listin.20100718" & codigofinline ' programa que ha echo el archivo

 lineas &= var.id.COUNT & codigofinline 'numero de registros Existentes

FOR a = 0 TO var.id.COUNT - 1

 Lineas &= var.id[a] & codigofinline

 Lineas &= var.dni[a] & codigofinline

 lineas &= var.nombre[a] & codigofinline

 lineas &= var.apellidos[a] & codigofinline

 lineas &= var.empresa[a] & codigofinline

 lineas &= var.puesto[a] & codigofinline

 lineas &= var.telf_empresa[a] & codigofinline

 lineas &= var.telf_parti[a] & codigofinline

 lineas &= var.fax[a] & codigofinline

 lineas &= var.movil_empresa[a] & codigofinline

 lineas &= var.movil_parti[a] & codigofinline

 lineas &= var.pag[a] & codigofinline

'aqui meto la nueva ruta de las fotos

 lineas &= fotoNuevaRuta[a] & codigofinline

'modifico la ruta de la foto original a la nueva foto

 var.foto[a] = fotoNuevaRuta[a]

 lineas &= var.direccion[a] & codigofinline

 lineas &= var.observaciones[a] & codigofinline

 lineas &= var.fecha_datos[a] & codigofinline

 lineas &= var.correo[a] & codigofinline

NEXT

File.Save(destino, lineas)

'fin del proceso

END

Ahora toda la información (imagenes, miniaturas, datos), estan en el directorio "~/listin/" (User.Home & "/listin". Se podria copiar a otro ordenador este directorio, y pegarlo en su User.Home, haciendose accesible su informacion completamente.

Anexo 5: Optimizando nuestro código.

Una vez concluido el programa vemos que me podemos organizar y reducir el código repetitivo mediante llamadas a procedimientos. Por ejemplo tenemos parte de código que se repite varias veces:

'ponemos en blanco la propiedad .text de los textbox

```
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
```

```
TextBoxDNI.text = ""
```

```
TextBoxNombre.text = ""
```

```
TextBoxApellidos.text = ""
```

```
TextBoxEmpresa.text = ""
```

```
TextBoxPuesto.text = ""
```

```
TextBoxTelfEmpresa.text = ""
```

```
TextBoxTelfParticular.text = ""
```

```
TextBoxFax.text = ""
```

```
TextBoxMovilEmpresa.text = ""
```

```
TextBoxMovilParticular.text = ""
```

```
TextBoxWEB.text = ""
```

```
PictureBoxFoto.Picture = ""
```

```
TextBoxDireccion.text = ""
```

```
TextBoxObs.text = ""
```

```
TextBoxFecha.text = ""
```

```
TextBoxCorreo.text = ""
```

'escribimos en el gridviews el dato introducido

```
titulo.rellena()
```

'el setfocus lo ponemos justo al inicio de los datos

```
TextBoxDNI.SetFocus
```

Este código se repite muchísimo, podemos crear una subrutina que lo englobe y que la llamemos cada vez que nos haga falta.

```
'-----  
'agrupando código que se repite mucho  
'-----
```

```
PUBLIC SUB limpia()
```

'ponemos en blanco la propiedad .text de los textbox

```
PictureBoxFoto.Picture = Picture["icon:/96/gambas"]
```

```
TextBoxDNI.text = ""
```



```
TextBoxNombre.text = ""
TextBoxApellidos.text = ""
TextBoxEmpresa.text = ""
TextBoxPuesto.text = ""
TextBoxTelfEmpresa.text = ""
TextBoxTelfParticular.text = ""
TextBoxFax.text = ""
TextBoxMovilEmpresa.text = ""
TextBoxMovilParticular.text = ""
TextBoxWEB.text = ""
PictureBoxFoto.Picture = ""
TextBoxDireccion.text = ""
TextBoxObs.text = ""
TextBoxFecha.text = ""
TextBoxCorreo.text = ""

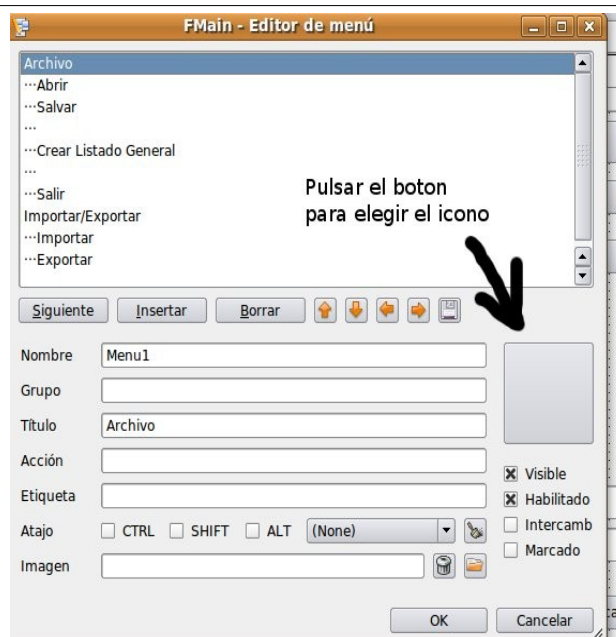
'escribimos en el gridviews el dato introducido
titulo.rellena()

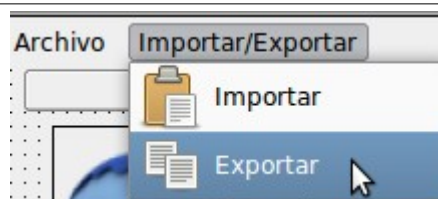
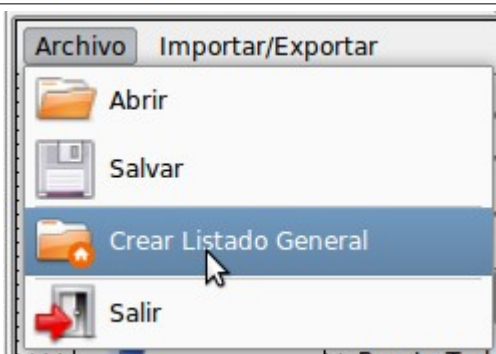
'el setfocus lo ponemos justo al inicio de los datos
TextBoxDNI.SetFocus

END
```

Otro tema sería ponerle iconos a los menus:

Desde el editor de menus podemos ponerles unos iconos a las distintas opciones de los menus, es facil y queda muy bien:





Otro sería poner una etiqueta donde se viera que archivo “.lis” estamos actualmente trabajando.

En el formulario Fmain, crearemos un labeltext, llamado **labelActual**, y en el IDE de gambas, la propiedad Border, le ponemos el valor Raised, para que se viera mejor (se hace un recuadro en el label recuadro)

Cada vez que abramos, guardemos, o generemos “listado General”, se presentaría el nombre de dicho archivo:

Por ejemplo: Módulo **Archivo.Salva**.

```
.....
lineas &= var.fecha_datos[a] & codigofinline
lineas &= var.correo[a] & codigofinline
NEXT
var.cambio = "NO"
File.Save(destino, lineas)
Fmain.LabelActual.text=destino
fins: ' hemos pulsado el boton de cancelar en el cuadro de dialogo Dialog.SaveFile()
'fin de la subrutina
END
```

En el módulo **Archivo.abrir**:

```
.....
b += 1
var.correo[a] = arr_cadenas[b]
NEXT
var.cambio = "NO"
titulo.rellena()
FMain.LabelActual.text = Dialog.Path
ENDIF
```

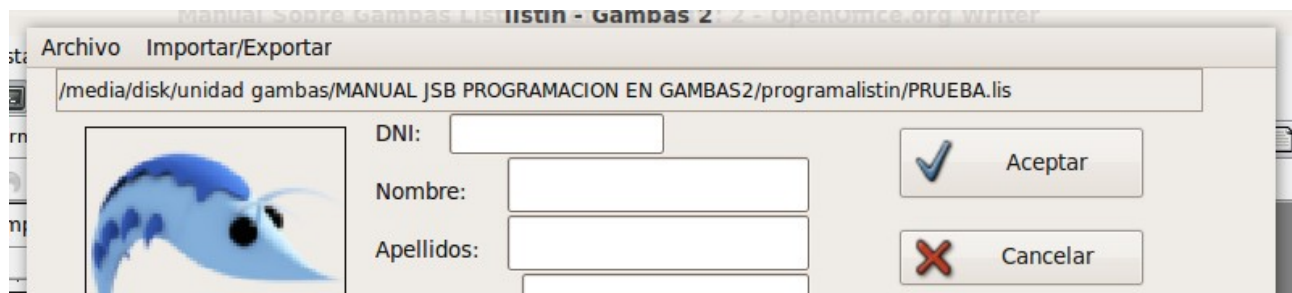
finlectura:

END

Y en el módulo **importar.completo**:

```
.....  
    lineas &= var.correo[a] & codigofinline  
NEXT  
File.Save(destino, lineas)  
var.cambio = "NO"  
FMain.LabelActual.text = destino  
'fin del proceso  
END
```

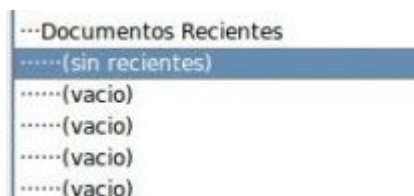
Quedaría mas o menos así:



Anexo 6: Añadir recientes

En este anexo vamos a añadir un nuevo módulo que lo llamaremos **addrecientes**, que se va a encargar de “recordar” los archivos que anteriormente habríamos abierto.

Primero añadiremos en el editor de menus del formulario Fmain este menu:



Y pondremos

Nombre	reciente0
Grupo	
Título	(sin recientes)
Acción	

Para los nombre reciente1,reciente2,reciente3 o reciente4, pondremos en el titulo “(vacio)”.

De esta manera estamos dejando 5 lugares para guardar las rutas de los últimos 5 archivos abiertos.

En el Formulario **Fmain**, añadimos en Form_Open():

```
PUBLIC SUB Form_Open()
ME.CENTER()
var.reinicio()
titulo.definir()
var.rutaimagen = "icon:/96/gambas"
addrecientes.abre()
END
```

y para que cada vez que pulsemos en un menu de recientes abra ese archivo:

```
'-----
'abrir archivos segun recientes
'-----
PUBLIC SUB reciente0_Click()
    archivo.abrir(FMain.reciente0.caption)
END
PUBLIC SUB reciente1_Click()
```

```

archivo.abrir(FMain.reciente1.caption)
END
PUBLIC SUB reciente2_Click()
archivo.abrir(FMain.reciente2.caption)
END
PUBLIC SUB reciente3_Click()
archivo.abrir(FMain.reciente3.caption)
END
PUBLIC SUB reciente4_Click()
archivo.abrir(FMain.reciente4.caption)
END

```

y en Form_Close()

```

PUBLIC SUB Form_Close()
DIM res AS Integer
IF var.cambio = "si" THEN
res = Message.Question("¿Desea salir sin salvar?", "si", "no")
    IF res = 1 THEN
        addrecientes.salva()
        ME.CLOSE
    ELSE
        STOP EVENT 'paramos este evento y no salgo del programa
    ENDIF
ELSE
    ME.close
ENDIF
'no salgo del programa
END

```

En el modulo **archivo**, en la subrutina **abrir()**, al inicio añadimos:

```
PUBLIC SUB abrir(OPTIONAL ruta AS String)
DIM a AS Integer 'contador de conjunto de datos
DIM b AS Integer 'contador de dato
DIM arr_cadenas AS String[]
DIM codigofinline AS String
DIM numero_de_datos AS Integer
codigofinline = "|"
IF ruta = "" THEN
    Dialog.Title = "Seleccione un archivo de datos listin"
    Dialog.Filter = ["*.lis", "Datos de Listin"]
    IF NOT Dialog.OpenFile() THEN
        ruta = Dialog.Path
    ELSE
        STOP EVENT
    ENDIF
ENDIF
arr_cadenas = Split(File.Load(ruta), codigofinline)

IF arr_cadenas[0] <> "v0.0.1" THEN
    'se trata de una version incompatible con la version de este programa, abandono el procedimiento
    .....
```

y al final del módulo añadimos

```
.....
var.correo[a] = arr_cadenas[b]
NEXT
var.cambio = "NO"
titulo.rellena()
FMain.LabelActual.text = Dialog.Path
addrecientes.add(Dialog.path)
```

finlectura:

END

Ahora pasamos a definir el nuevo modulo **addrecientes**:

```
PUBLIC SUB add(ruta AS String) AS String
IF FMain.reciente0.caption = "(sin recientes)" THEN
  FMain.reciente0.caption = ruta
  RETURN
ENDIF
'no adjuntar si el elemento esta repetido
IF FMain.reciente0.caption = ruta THEN
  RETURN
ENDIF
IF FMain.reciente1.caption = ruta THEN
  RETURN
ENDIF
IF FMain.reciente2.caption = ruta THEN
  RETURN
ENDIF
IF FMain.reciente3.caption = ruta THEN
  RETURN
ENDIF
IF FMain.reciente4.caption = ruta THEN
  RETURN
ENDIF
IF FMain.reciente1.caption = "(vacio)" THEN
  FMain.reciente1.caption = ruta
  FMain.reciente1.Visible = TRUE
  RETURN
ENDIF
IF FMain.reciente2.caption = "(vacio)" THEN
  FMain.reciente2.caption = ruta
```

```

FMain.reciente2.Visible = TRUE
RETURN
ENDIF
IF FMain.reciente3.caption = "(vacío)" THEN
FMain.reciente3.caption = ruta
FMain.reciente3.Visible = TRUE
RETURN
ENDIF
IF FMain.reciente4.caption = "(vacío)" THEN
FMain.reciente4.caption = ruta
FMain.reciente4.Visible = TRUE
RETURN
ENDIF
'si llega hasta aqui quiere decir que tenemos relleno los 5 recientes,
'por lo tanto debemos trasladar todos los caption y añadir al ultimo el nuevo
FMain.reciente0.caption = FMain.reciente1.caption
FMain.reciente1.caption = FMain.reciente2.caption
FMain.reciente2.caption = FMain.reciente3.caption
FMain.reciente3.caption = FMain.reciente4.caption
FMain.reciente4.caption = ruta
END

PUBLIC SUB salva()
  DIM lineas AS String
  DIM ruta AS String
  lineas = "***** Contenido de Doc. Recientes *****" & "\n"
  lineas &= "documentosrecientes" & "\n"
  lineas &= "v.0.0.1" & "\n"
  lineas &= FMain.reciente0.caption & "\n"
  lineas &= FMain.reciente1.caption & "\n"
  lineas &= FMain.reciente2.caption & "\n"
  lineas &= FMain.reciente3.caption & "\n"
  lineas &= FMain.reciente4.caption & "\n"

```



```

lineas &= "***** Fin *****" & "\n"
ruta = User.Home
File.Save(ruta & "/documentosrecienteslistin.DocRec", lineas)
END

PUBLIC SUB abre()
DIM arr_cadenas AS String[]
FMain.reciente1.Visible = FALSE
FMain.reciente2.Visible = FALSE
FMain.reciente3.Visible = FALSE
FMain.reciente4.Visible = FALSE
IF Exist(User.Home & "/documentosrecienteslistin.DocRec") = FALSE THEN
'no Existe el archivo, por lo tanto podemos continuar con el salvado con el mismo nombre
ENDIF
TRY arr_cadenas = Split(File.LOAD(User.Home & "/documentosrecienteslistin.DocRec"), "\n")
IF ERROR THEN
Message.Error("error en lectura")
RETURN
ENDIF
'si se produce un error vuelve al programa, no encuentra el archivo documentosrecientes.DocRec
FMain.reciente0.caption = arr_cadenas[3]
FMain.reciente1.caption = arr_cadenas[4]
FMain.reciente2.caption = arr_cadenas[5]
FMain.reciente3.caption = arr_cadenas[6]
FMain.reciente4.caption = arr_cadenas[7]
IF FMain.reciente1.caption <> "(vacio)" THEN
    FMain.reciente1.Visible = TRUE
ENDIF
IF FMain.reciente2.caption <> "(vacio)" THEN
    FMain.reciente2.Visible = TRUE
ENDIF
IF FMain.reciente3.caption <> "(vacio)" THEN
    FMain.reciente3.Visible = TRUE

```

```
ENDIF
IF FMain.reciente4.caption <> "(vacio)" THEN
    FMain.reciente4.Visible = TRUE
ENDIF
END
```

Anexo 7: Índice Alfabético.

Índice alfabético

abrir.....	29, 48, 50, 55, 90
Abrir.....	18, 50
cd.....	7
Clipboard.Copy.....	64 ss.
Clipboard.Paste.....	67 ss.
compruebaExistencia.....	45 s.
ConceptoFiltro.....	58, 80
convert.....	44, 77 ss.
copiaraportapapeles.....	64 ss.
DateChooserfecha_Change().....	72
DateChooserfecha.Value.....	72 s.
dentro.....	59 s., 72, 80
Dialog.Path.....	29, 46, 48, 50, 90
Dialog.SaveFile().....	45 ss., 55, 90
END WITH.....	23, 31 s., 34, 36, 60
Exist.....	32, 45 s., 78 s., 86 s.
Ffechadia.ShowModal.....	73
file.load.....	50
File.LOAD.....	48, 50
File.Save.....	45, 47, 55, 87, 90 s.
fixed	56
Fmain.Border.....	56
Form_Close()	56
gestion.editar.....	35
GridViewDatos.....	16
InStr.....	57, 60, 65
Key.code.....	24 ss., 57
Key.enter.....	24 ss., 57
Key.Return.....	24 ss., 57
ME.CENTER.....	23
Message.Error.....	35, 48, 62
Message.Info.....	36, 55, 58, 70
Message.Question.....	55
Message.Warning.....	45
Mid\$.....	65 s.
mkdir.....	7
MKDIR.....	78, 86
mv.....	7
NAME.....	15, 17
Now.....	30, 37, 51, 70
OPTIONAL.....	45, 48, 58
ord_AZ.....	61
ord_ZA.....	61 s.
Picture.....	14 s., 29 ss., 33, 36, 38 s., 41 ss., 52 ss., 59, 80, 88 s.
PUBLIC.	20 ss., 29, 31, 33, 35 ss., 39, 41, 43, 45, 47 s., 50 s., 53, 55, 57 s., 61 s., 64 s., 67 ss., 72 s.,

79 s., 82, 85, 88	
Replace.....	29
Replace\$.....	29
RESIZE.....	20
revisa.....	65 s.
SetFocus.....	24 ss., 31, 39 s., 42, 44, 53, 55, 57 s., 88 s.
SHELL.....	77 ss.
Split.....	48, 50, 57, 69
Stretch.....	15, 29
tar xvfj.....	7
titulo.rellena().....	31, 39 s., 42, 44, 50, 53 ss., 59, 70, 79, 88 ss.
ToolTip.....	58
UCase\$.....	62 s.
Upper\$.....	57, 59
User.Home.....	87, 97
User.NAME.....	29
Val(.....	84
var.cambio.....	51 s., 54 s., 90 s.
var.estado	36, 40, 42, 44, 51 ss., 55
WAIT.....	78
WITH	22 s., 31 s., 36, 59 s., 80
WITH – END WITH.....	32
.....	26
"\n" ' retorno de carro (separa las filas).....	69
[].....	36
buscarDadoId.....	35 s.
ME.Close.....	51, 56, 68, 73
Ordenar.ord_AZ.....	61
Ordenar.ord_ZA.....	61
Str\$.....	30, 37, 51, 70, 84
.add.....	29 s., 37, 51, 70 s.
.Background.....	23, 58
.COUNT.....	22, 31 s., 34 s., 41, 45 s., 57, 59, 62 s., 65, 67 ss., 80, 86 s.
.enabled.....	35, 39 s., 42, 44, 53 s.
.font.name.....	23
.font.size.....	23
.Foreground.....	23
.Refresh.....	31 s.
.REMOVE.....	41 ss., 53 s.
.Resize.....	21 s., 48 s., 57, 86
.Row.....	22, 31 s., 34 s., 59, 62, 65, 67 s., 80
"\t" ' tabulador (separa las columnas).....	69
"yyyy/mm/dd".....	73
().....	20 ss., 29, 31, 33 ss., 39 ss., 50 s., 53 ss., 57 ss., 64, 67 ss., 78 ss., 85 s., 88 ss.
~/.....	78 s., 85 ss.