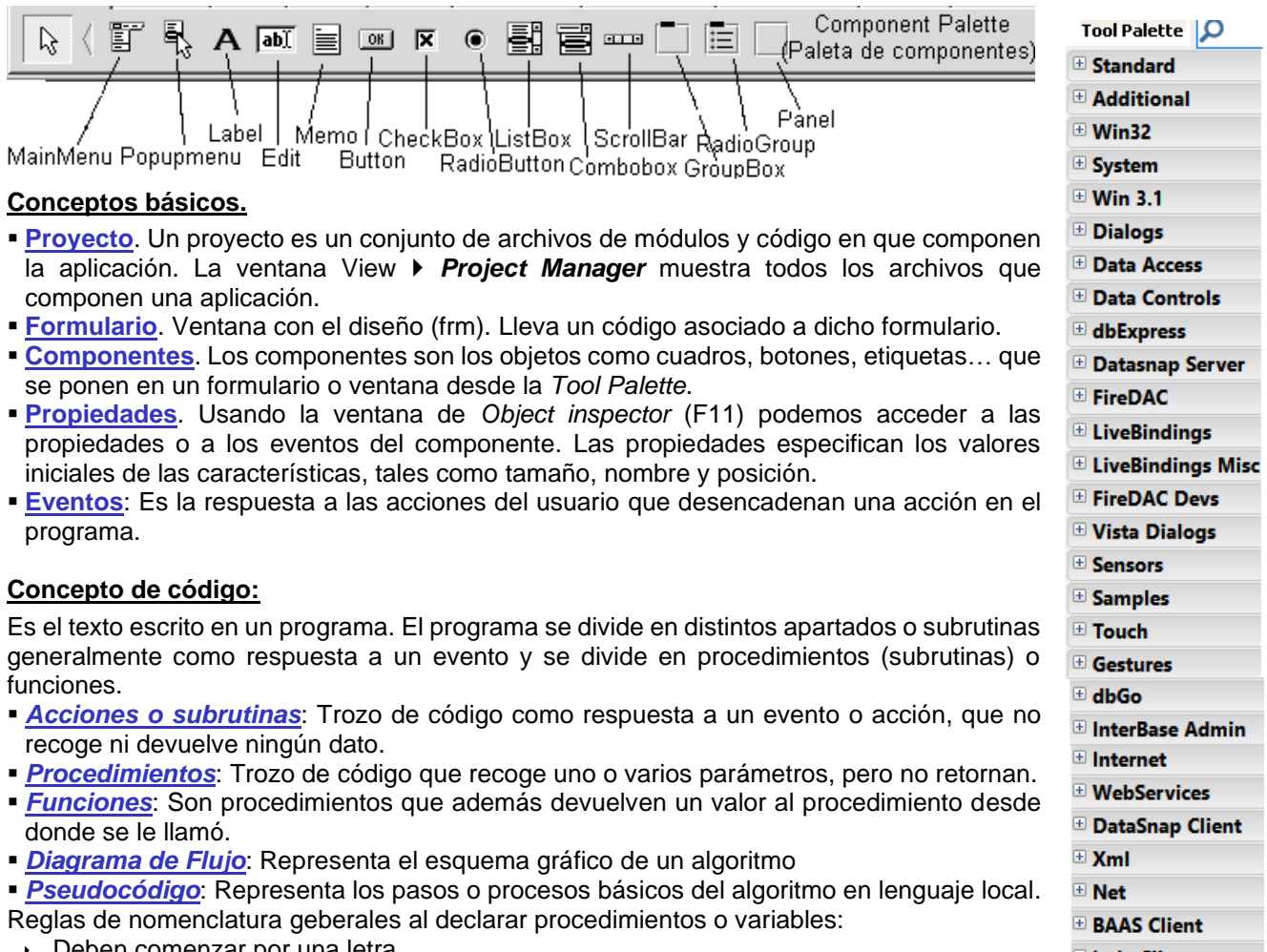


## Introducción a la Programación Object Pascal con Rad Studio Delphi XE & Lazarus

### Delphi – Object Pascal para versiones © Embarcadero XE & Lazarus free Pascal

Delphi/Lazarus son entornos de programación del tipo RAD (Diseño Rápido de Aplicaciones) basado en el lenguaje Object Pascal, creado por *Borland* y continuado por *Embarcadero*. Su característica principal es que posee un control de bases de datos muy amplio, compatible con *Oracle* y a partir de las versiones XE permite aplicaciones multiplataforma dentro de su framework *Firemonkey*. Rad Studio incorpora también **C++ Builder**. **Lazarus** es la versión gratuita, multiplataforma y open source compatible y similar a la versión *Delphi 7*.



#### Conceptos básicos.

- **Proyecto.** Un proyecto es un conjunto de archivos de módulos y código en que componen la aplicación. La ventana View ▶ **Project Manager** muestra todos los archivos que componen una aplicación.
- **Formulario.** Ventana con el diseño (frm). Lleva un código asociado a dicho formulario.
- **Componentes.** Los componentes son los objetos como cuadros, botones, etiquetas... que se ponen en un formulario o ventana desde la *Tool Palette*.
- **Propiedades.** Usando la ventana de *Object inspector* (F11) podemos acceder a las propiedades o a los eventos del componente. Las propiedades especifican los valores iniciales de las características, tales como tamaño, nombre y posición.
- **Eventos:** Es la respuesta a las acciones del usuario que desencadenan una acción en el programa.

#### Concepto de código:

Es el texto escrito en un programa. El programa se divide en distintos apartados o subrutinas generalmente como respuesta a un evento y se divide en procedimientos (subrutinas) o funciones.

- **Acciones o subrutinas:** Trozo de código como respuesta a un evento o acción, que no recoge ni devuelve ningún dato.
- **Procedimientos:** Trozo de código que recoge uno o varios parámetros, pero no retornan.
- **Funciones:** Son procedimientos que además devuelven un valor al procedimiento desde donde se le llamó.
- **Diagrama de Flujo:** Representa el esquema gráfico de un algoritmo
- **Pseudocódigo:** Representa los pasos o procesos básicos del algoritmo en lenguaje local.

Reglas de nomenclatura generales al declarar procedimientos o variables:

- › Deben comenzar por una letra
- › No pueden contener puntos
- › No puedes superar los 255 caracteres
- › Nombres de formularios no sobrepasar 40 caracteres
- › No pueden tener el mismo nombre que una palabra clave
- › Comentarios en el código se inician con: // o entre {}

#### Eventos y acciones (sobre un componente):

Lista de eventos (Desencadena la acción de un objeto)

Evento	Acción
Click	Al hacer clic con el mouse (o el teclado)
DragDrop	Al soltar un control arrastrado con el mouse
DragOver	Al Arrastrar un control con el mouse.
KeyDown	Presionar una tecla mientras el objeto tiene el foco.
KeyPress	Presionar una tecla y devolver su valor ASCII.
KeyUp	Liberar una tecla mientras el objeto tiene el foco.
MouseDown	Presionar el botón del mouse sobre el objeto.
MouseMove	Mover el puntero del mouse por encima del objeto
MouseUp	Liberar el botón del mouse en el objeto.

#### Propiedades básicas de los objetos:

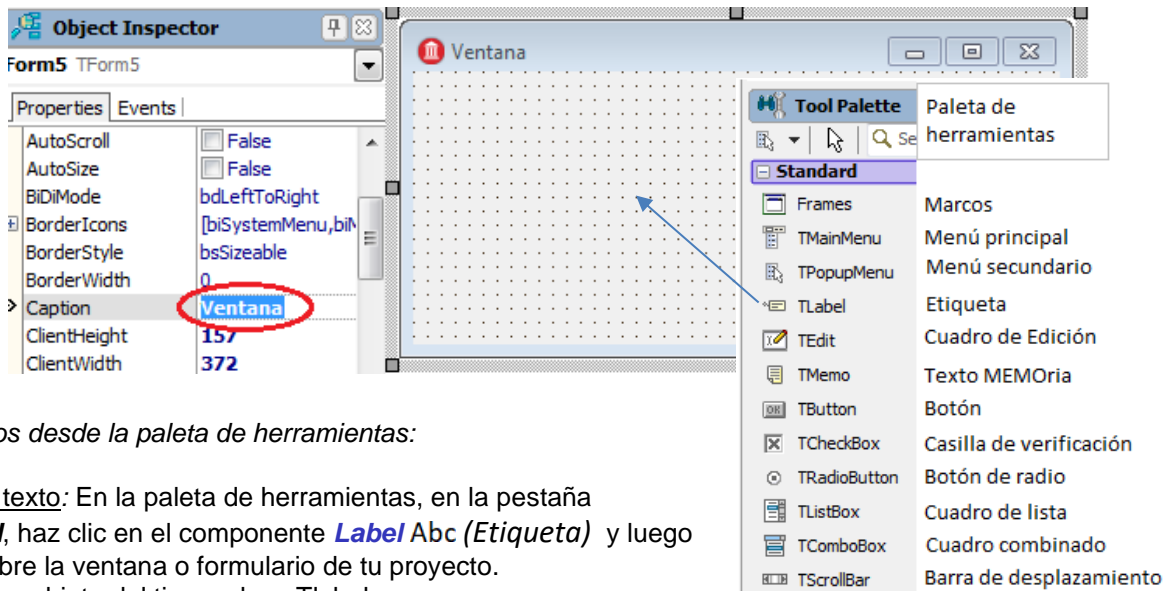
Alignment	- Alineación
BevelInner	- borde interior
BorderStyle	- estilo del borde
Caption	- rótulo
TabStop	- Salto tabulación
Color, Cursor	- Color, puntero
Enabled	- Activo
Font	- Tipo letra
Height	- Altura del objeto
Hint	- Etiqueta ayuda
Left	- Posición izquierda
Name	- Nombre del objeto
PopupMenu	- Menú contextual
ShowHint	- Mostrar etiquetas
Top	- Posición vertical
Visible	- Visible
Width	- Ancho del objeto

## Ejercicio 1. Crear una ventana “hola mundo” y un botón para cerrar.

En primer lugar, antes de trabajar con Delphi, es aconsejable que crees una carpeta (subdirectorio) con tu nombre para guardar ahí los ejercicios.

- Crear un proyecto nuevo:
  - › *Delphi XE:* Elige del menú: **File ▶ New ▶ Aplicación** (o *VCL Forms application*)
  - › *Lazarus:* Elige del menú: Archivo – Nuevo -Proyecto – Aplicación

Te aparecerá una ventana o formulario vacío (*Form*). En el panel izquierdo *Inspector de objetos (Object Inspector)*, verás las propiedades (*Properties*) de la ventana y de los objetos que en ella pongas.
- Cambiar el nombre de la ventana: En el panel *Inspector de objetos (Object Inspector)*, pestaña *Properties (Propiedades)*, haz clic en **Caption** y en su casilla escribe: Ventana para poner un rótulo a la ventana.



Añadir objetos desde la paleta de herramientas:

- Poner un texto: En la paleta de herramientas, en la pestaña **Standard**, haz clic en el componente **Label** *Abc (Etiqueta)* y luego un clic sobre la ventana o formulario de tu proyecto. Pondrás un objeto del tipo o clase *TLabel*

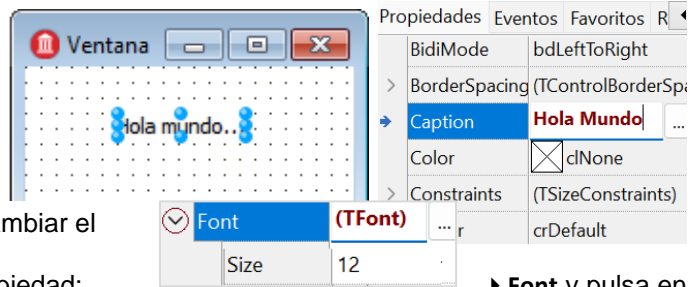
Cambia la propiedad *Caption* de la etiqueta (*Label1*) del inspector de objetos por: **“Hola mundo”**.

Arrastra el componente con el mouse para centrarlo en la ventana.

Puedes cambiar la propiedad *Color* y *Font* para cambiar el color y tipo de letra favorita:

Con la etiqueta *Label1* seleccionada, busca la propiedad:

la flecha izquierda para ver sus subpropiedades. Cambia el valor **Size** a 12.



► Font y pulsa en

- Poner el botón de Salir: En la ficha *Standard* de la paleta, selecciona el componente **Button** (*Ok*) y haz un clic en la ventana de tu proyecto para soltarlo encima este botón. Cambia su propiedad *Caption* por: **&Salir** (El & subraya la letra principal hotkey) Para cambiar el tamaño botón puedes usar sus propiedades *Width* y *Height* o arrastra los tiradores de sus esquinas con el ratón.



- Asignar la acción al botón Salir: Pulsa **dobles clic** sobre el botón **Salir** para abrir la ventana de código. Observa que ya nos ha escrito el inicio y final del subprograma o procedimiento:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    → aquí escribimos el código
end;
```

Para indicar una acción a un objeto escribir el nombre del objeto, un punto y su acción: **Form1.close;**

Si no pones el nombre del objeto, te refieres a la ventana en general: sólo deberás escribir la palabra: **close** (cerrar la ventana) en el lugar indicado:

<pre>Procedure TForm1.Button1Click(Sender: TObject); // -&gt; en Delphi Begin     close; // -&gt; Al final de una línea de código, poner punto y coma (;) end;</pre>	<pre>// -&gt; en C++ Builder: {     Close(); }</pre>
--	--

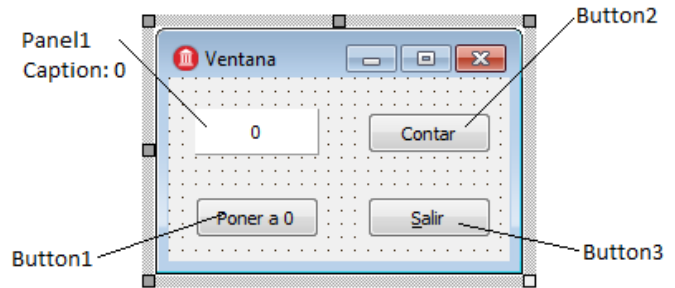
- Compilar y probar el programa: Pulsa el botón **Run** ▶ o eligelo del menú *Ejecutar (Run)*. En el modo de *ejecución* puedes comprobar su funcionamiento: al pulsar en el botón se cierra la ventana.
- Guardar el proyecto:
  - Para guardar la ventana del formulario y su código asociado elige del menú: **File ▶ Save as... Hola1.pas**
  - Para guardar el proyecto completo: elige del menú: **File ▶ Save project as... Hola (\*.dpr o \*.pdroj)**

## Ejercicio 2. Crear un contador de botones. Cambiar el nombre de los botones.

- Crea un nuevo proyecto.  
*Delphi:* File ▶ New ▶ Application (VCL Forms)  
*Lazarus:* Archivo ▶ Nuevo Proyecto ▶ Aplicaci3n

- Busca en la paleta *Estándar* los siguientes componentes y ponlos sobre el formulario, (ventana vacía) como en la imagen: →

- 3 componentes tButton
- 1 componente tPanel



- Selecciona cada componente ańadido y en el *Objet inspector* (F11) cambia las siguientes propiedades:  
En el Button1.Caption: Poner a cero  
En el Button2.Caption: Contar  
En el Button3.Caption: Salir  
En el Panel1. Caption: 0 ;  
En el Panel1. Color: clWhite (blanco)
- Selecciona cada componente creado y en el *Objet inspector* (F11) cambia los siguientes eventos, pulsando doble clic sobre el evento: On clic o directamente sobre el componente:

### Eventos: (pulsar doble clic en cada objeto)

	C3digo en Delphi / Pascal	C3digo en En C++ Builder
En el Bot3n1:	<pre>procedure TForm1.Button1Click(Sender: TObject); begin     Panel1.Caption:='0'; end;</pre>	<pre>Panel1-&gt;Caption="0";</pre>
En el Boton2:	<pre>procedure TForm1.Button2Click(Sender: TObject); begin     Panel1.caption:=IntToStr(StrToInt(Panel1.caption)+1); end;</pre>	<pre>Panel1-&gt;Caption= IntToStr(StrToInt(Panel1-&gt;Caption)+1);</pre>
En el Boton3:	<pre>procedure TForm1.Button3Click(Sender: TObject); begin     Close; end;</pre>	<pre>Close();</pre>

Cambia la variable del tipo numérico entero a cadena de texto

### Comentarios al ejercicio:

Algunos componentes tienen propiedades del tipo de texto, numérico o l3gicas.

No se pueden igualar o comparar propiedades de tipos diferentes. Para ello se debe realizar una conversi3n de un formato a otro:

- ▶ **IntToStr** convierte un valor numérico en un valor de texto para que se pueda mostrar en el panel1
- ▶ **StrToInt** Convierte un valor de tipo texto en su valor numérico

### Probar la aplicaci3n:

Pulsa el bot3n **Run** ▶ o elígelo del men3 *Ejecutar* (Run) para comprobar su funcionamiento.

**Guardar el proyecto:** Crea una carpeta que se llame: *Contador*.

- Escoge: **File** ▶ **Save as...** Para guardar el archivo de c3digo en la carpeta con el nombre: **Contador1.pas**
- Escoge: **File** ▶ **Save project as...** Para guarda el *proyecto* en la carpeta con el nombre **contador.dproj**

En *Lazarus*: Escoge del men3 *Archivo* - *Guardar todo*.

Primero nos pide el proyecto: **Contador.lpi** (*Lazarus Project*) y luego la unidad: **Contador1.pas**

### Conversiones de tipos de datos

**IntToStr** (Integer to String) ▶ Cambia del tipo de n3mero entero a cadena de texto  
**StrToInt** (String to Integer) ▶ Cambia del tipo cadena de texto a n3mero entero

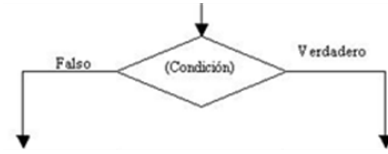
Otros:

StrToDate	(Texto a fecha)	StrToFloat	(texto a decimal flotante)
InToEx	(Entero a Extended)	DateToStr	(Fecha a Texto)
DateTimeToStr	(Fecha/hora a texto)	FloatToDecimal	(Decimal flotante a decimal)
<b>FloatToStr</b>	(Decimal flotante a texto)	StrToTime	(Texto a hora)
<b>Formatfloat</b>	(Decimal flotante a texto con formato)		

**Teoría: Objetos, componentes y clases:**

- Un componente es una clase de objeto. Por eso, un componente le solemos llamar a un objeto disponible en la Toolbar.
- Un objeto es un conjunto de datos de la misma clase o tipo. Por eso empieza generalmente por la letra T, porque pertenecen a clases de tipo genérica (T class) (TForm es la clase de los formularios, TButton la clase de los botones, etc.)
- Una clase es una plantilla o estructura que contiene la definición del tipo de datos que emplea. Ejemplo:  
TForm1 = class(TForm)

El objeto Formulario1 pertenece a la clase Formularios



**Estructura condicional. Toma de decisión**

if comparación then acción\_verdadera else acción\_falsa ;  
Realizará una acción u otra según si cumple una comparación

Nota. Asignar o comparar un valor.

- Para asignar un valor a un objeto o variable se indica con `:=` Ejemplo: `ProgressBar1.position := 100;`
- Para comparar el valor de un objeto o variable se indica con `=` Ejemplo: `if ProgressBar1.position = 100..`  
(En lenguajes C, C++, para asignar un valor es `=` y para comparar `==`)

Otros comparadores son:

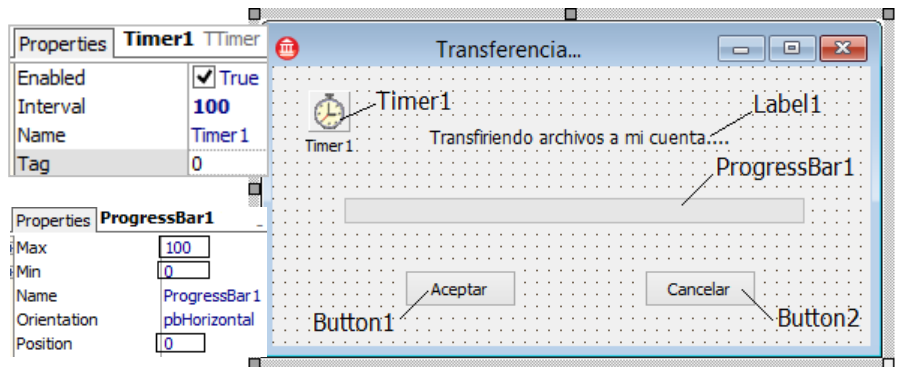
Comparador	símbolo	ejemplo
Igual	=	if ProgressBar1.position=100 then Showmessage ('Has llegado a 100');
Mayor	>	if ProgressBar1.position>100 then Showmessage ('Te has pasado de 100');
Menor	<	if ProgressBar1.position<100 then Showmessage ('Aún no has llegado a 100');
Mayor o igual	>=	if edad>=100 then Showmessage ('Has llegado o te has pasado de 100');
Menor o igual	<=	if edad<=100 then Showmessage ('Aún no has llegado a 100');
Distinto	<> ó !=	if edad <> 100 then Showmessage ('No es 100');

**Ejercicio ventana con barra de progreso**

- Crea un nuevo proyecto: File ▶ New ▶ Application VCL (o VCL Forms application).

**Diseño:**

- Añade al formulario un objeto barra de progreso *tProgressBar* (de la paleta win32), dos botones (*tButtons*), una etiqueta *tLabel* (de la paleta Standard) y un componente *tTimer* (Paleta System)
- Distribuye y cambia las propiedades de la ventana-formulario como en la imagen.



**Eventos:** (Suceso que desencadena una acción)

- Pulsa *doble clic* sobre el evento **OnTimer** del **Timer1** para mostrar el código de su procedure y añade:

```

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  ProgressBar1.position := ProgressBar1.position+1; //-> incrementa la barra una posición
  if ProgressBar1.position=100 then //-> si llega a 100...
  begin
    Timer1.Enabled:=False; // -> Para el temporizador
    ShowMessage('Transferencia realizada'); // -> muestra un mensaje
    Application.Terminate; //-> termina el programa
  end;
end;

```

Ofimega

Opcional: Puedes impedir cerrar la ventana si en el evento del form: Closequery pones la variable canClose:=false;  
Ejemplo: ShowMessage('No puedes cerrar la ventana mientras se transfieren los archivos'); canClose:=false;

- **Probar la aplicación:** Pulsa Run ▶ para comprobar su funcionamiento.
- **Guardar el proyecto:** Crea una carpeta que se llame: Transferencia
  - Escoge: File ▶ Save as... Para guardar el archivo: Transferencia1.pas
  - Escoge: File ▶ Save project as... Para guarda el proyecto Transferencia.dpr

## Uso del Timer y posici3n del objeto:

- Crea una nueva aplicaci3n VCL (librería visual) escogiendo del menú: File - New – VCL Form Applicaton.
- Añade un bot3n *Button1* en un formulario nuevo. Cambiar su propiedad **Caption** por: *Púlsame*.
- En el evento del bot3n *OnMouseMove*:

```
Button1.Left := Button1.Left + 1;
```

**Ejecuta y prueba (Run ▶):** Al intentar pulsar el bot3n, este se desplaza hacia la derecha.

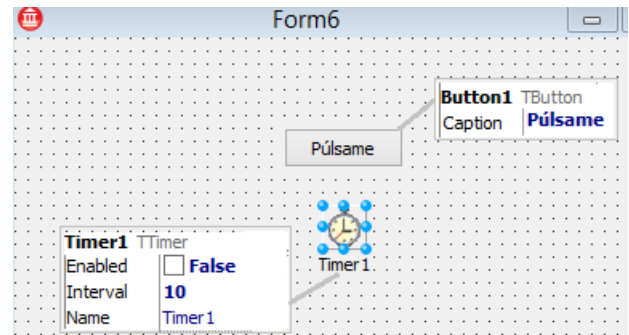
- Anula esta última línea de código convirtiéndola en un comentario: añade // delante
- Añade un componente **TTimer** de la paleta System. Poner su propiedad *Interval* a 10 y si propiedad *enabled* en *False*:

En el evento del bot3n *OnClick*:

```
Timer1.Enabled := True;
```

En el evento del Timer *OnTimer*:

```
Button1.Left := Button1.Left + 1;
```



**Ejecuta y prueba (Run ▶):** Al pulsar el bot3n, activamos el temporizador Timer que desplaza un pixel a la derecha la posici3n del bot3n a cada pulso de su intervalo, de duraci3n 10 milisegundos.

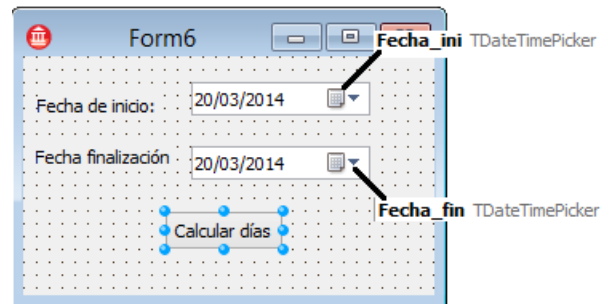
Para guardar primero la unidad y luego el proyecto, escoge del menú: File ▶ Save project as...

## Calcula tu edad. Uso de variables de fecha y decimal

Una variable almacena un valor temporalmente para utilizarlo después.

Se nombran o declaran al inicio del procedimiento en la secci3n **var** y pueden ser de fecha (*datetime*), numéricas (*integer*), decimales sencillas o dobles (*single* o *double*), de texto (*string*), etc...

- Crea una nueva aplicaci3n:  
*Delphi*: menú: File - New – VCL Form Applicaton.  
*Lazarus*: Archivo – Nuevo – Proyecto/Aplicaci3n
- Añade un bot3n *Button1* y dos etiquetas *labels* *Abc*
- Añade un selector de fecha. *Delphi*: *DatetimePicker* de la paleta *Win32*. *Lazarus*: *DateEdit* de la paleta *Misc*
- Cambia el nombre de los *DatetimePicker* por:  
*Fecha\_ini* y *Fecha\_fin*



- Pulsa doble clic sobre el bot3n y escribe en el evento del bot3n *OnClick*:

```
Procedure ...Button1Click(Sender: TObject);
```

```
var
```

```
  días:double;      //-> creo una variable decimal doble
```

```
  hoy:Tdatetime;  //-> creo una variable de fecha-hora
```

```
begin
```

```
  días:=Fecha_Fin.Date-Fecha_ini.date;    // -> resto las dos fechas
```

```
  hoy:=date();
```

```
  Showmessage('Hoy es '+ datetostr(hoy)+ ' y han transcurrido '+floattostr(días)+' días');
```

```
end;
```

- Pulsa Run ▶ o F9 para comprobar su funcionamiento.

### Variante del ejercicio. Calcula tu edad:

- Con el componente *Calendar* o *MonthCalendar* cambia el formulario como en la imagen →

- Escribe en el evento del bot3n *OnClick*:

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var
```

```
  días:double;      //-> creo una variable para guardar los días
```

```
  años:double;     // -> creo una variable para guardar los años
```

```
  hoy:Tdatetime;  //-> creo una variable para guardar el día de hoy
```

```
begin
```

```
  hoy:=date();    // -> guardo en la variable el día de hoy
```

```
  días:=hoy-Calendar1.Date;  //-> calculo los días transcurridos
```

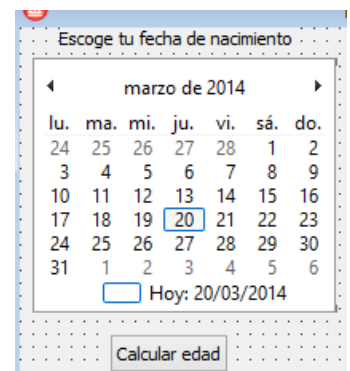
```
  años:=días/365;          //-> calculo los años transcurridos
```

```
  Showmessage('Tienes '+FormatFloat('###.##',días)+' días o '+FormatFloat('###.##',años)+' años');
```

```
end;
```

- **Ejecuta y guarda (Run):** Pulsa Run ▶ o F9 para comprobar su funcionamiento.

Para guardar primero la unidad y luego el proyecto, escoge del menú: File ▶ Save project as...



## Teoría: Conceptos básicos de programación.

### Procedimientos y funciones:

Un *Procedimiento* (procedure) es una subrutina o subprograma que puede ser llamado desde varias partes del programa principal. A diferencia de una función, éste no devuelve ningún valor de respuesta, pero sí puede recoger datos o parámetros del lugar desde el que se llamó utilizando el "Sender" como paso de parámetros.

- **Función:** Es un procedimiento o subrutina que además devuelve un valor de respuesta (Result)
- **Parámetro:** Es una variable o un valor que es pasado a un procedimiento o función.
- **Sender:** Es un parámetro que informa a un procedimiento o función desde qué componente (objeto) ha sido llamado. Se utiliza mucho para agrupar varios botones con un procedimiento parecido (líneas de código comunes).

#### Ejemplo de sender:

En este ejemplo, asignando el mismo evento a ambos botones, una ventana de mensaje mostrará el texto de cada botón.

```
Procedure TForm1.Button1Click(Sender: TObject);
begin
  if Sender = Button1 then showmessage('Has pulsado' +
    button1.caption)
  else if Sender = Button2 then showmessage('Has pulsado' +
    button2.caption);
end;
```

#### Ejemplo de función:

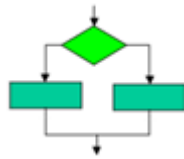
La función recoge un parámetro de entrada y devuelve otro de salida.

Se especifica en su declaración el tipo y valor de entrada (entre paréntesis) y de retorno. En este caso el tipo es boolean.

Luego puedes ser llamada desde cualquier procedure si se declara en la sección pública o privada.

```
Function EsPrimo (N: Integer): Boolean
var;
  i: Integer;
begin
  EsPrimo := True;
  For i := 2 to N - 1 do
    if (N mod Test) = 0 then //mod resto de la division es cero
      begin
        EsPrimo := False;
        break; //salta fuera del bucle
      end;
  end;
end;
Llamada a la función:
If EsPrimo(7) = true then showmessage('Es primo');
```

### Bucles y bifurcación:



#### Bifurcación simple:

- **if ... then ... else:**  
Equivale al **SI** condicional  
No se añade ";" de final de línea antes del Else. Si la condición ocupa varias líneas es necesario añadir Begin – End

```
IF edad >= 18 THEN label1.caption:='Mayor de edad'
ELSE label1.caption:='Mayor de edad';
```

#### Bifurcación múltiple:

- **Case:**  
*Bifurca* las acciones dependiendo de diversos resultados de un parámetro del tipo ordinal.  
(Número o carácter ordenado alfab.)

```
case edad of
  0..18: Label1.caption:='menor edad';
  18..60: Label1.caption:='mayor edad';
  60..99: Label1.caption:='anciano';
else Label1.caption:='no vale';
end;
```

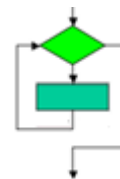
### Bucles:

#### Infinito:

**While:** Repite la ejecución infinitas veces hasta que se cumpla la condición.

#### Finito (Contador):

- **For ... To, o For ... Downto:**  
Ejecuta una acción un número determinado de veces.
  - *To:* Incrementa cada vez el bucle
  - *Downto:* Decrementa cada vez el bucle.

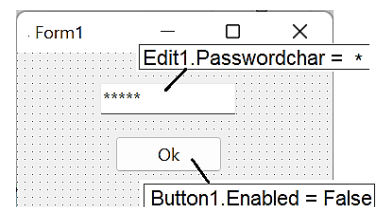


```
while i<10 do i := i+1;
```

```
for i:=1 to 100 do
  Begin
    Label1. caption:='Paso número' + i;
    Beep; // tono sonoro
  End;
```

### Ejemplo-ejercicio Password:

- Crea un nuevo proyecto (File – New – **Windows Application**) (*Archivo – Nuevo – Aplicación*)
- Añade un **Button**  de la paleta *Standard* y cambia sus propiedades:  
Enabled = false ; Caption = OK
- Añade por encima un **Edit**  con *propiedad* PasswordChar = \* ;
- Pulsa *doble clic* en el **Edit1** y escribe, dentro de su procedure, el código:  
If Edit1.text = 'danone' then button1.enabled:=true;
- Pulsa doble clic en el **Button1** y escribe: **Showmessage('Vale');**  
Comprueba ► y guarda la unidad con el nombre *Clave1* y el proyecto: **Clave**




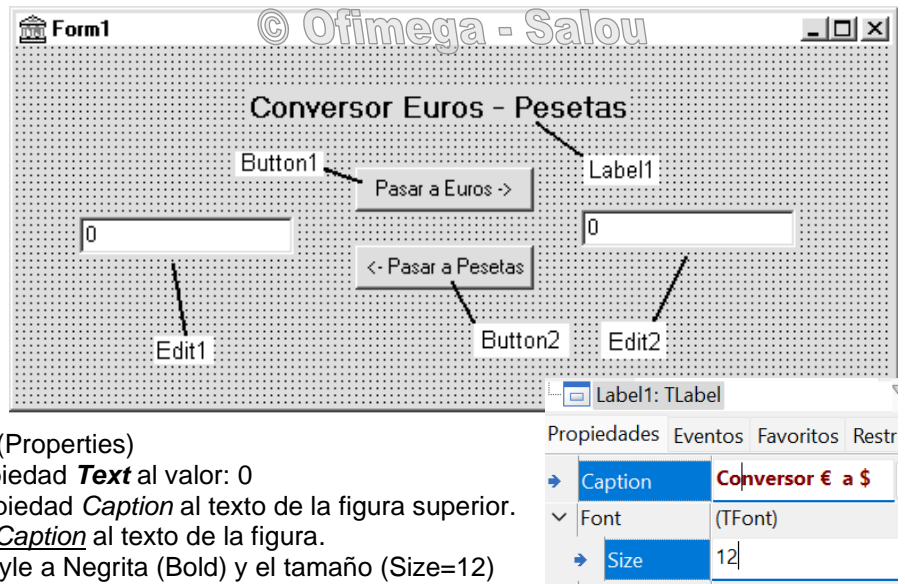
## Ejercicio de conversión de variables. Convertidor de Euros. Para Delphi 7+ Lazarus

### • Ingredientes:

Empieza un proyecto nuevo Windows Aplicación en Delphi o Nuevo – Proyecto – Aplicación en Lazarus.

Añade al formulario los siguientes componentes de la paleta Standard:

- 2 botones (**TButton**) 
- 2 cuadros de texto (**TEdit**)
- 1 etiqueta (**TLabel**) Abc



### • Preparación:

En el panel: *Object Inspector* (F11), pestaña de propiedades (Properties)

**Edit 1 y Edit2:** Cambiar su propiedad **Text** al valor: 0

**Botones 1 y 2:** Cambiar su propiedad **Caption** al texto de la figura superior.

**Label 1:** Cambiar la propiedad **Caption** al texto de la figura.

La propiedad **Font – Style** a Negrita (Bold) y el tamaño (Size=12)

### • Cocción:

Pulsar doble clic sobre el **botón 1** para añadir al evento **OnClick** las siguientes líneas de comando:

(Líneas a añadir en negrita)

```
procedure TForm1.Button1Click(Sender: TObject);
var
  numero1,numero2:double;
begin
  numero1:=strtofloat(edit1.text);
  numero2:=numero1/166.386;
  edit2.text:=floattostr(numero2);
end;
```

Comentarios

*variables a utilizar:*

*número1 y número2 del tipo numéricas con decimales*

*Número1 vale el texto del Edit1 convertido*

*Número2=número2/166,386 (poner punto decimal)*

*El texto del Edit2 será el valor de la variable número2 convertida a string*

### Ejercicio Propuesto:

Pulsa doble clic sobre el **botón 2** y añade el código al botón2 para que multiplique el valor de Edit1 por **166,386**

### Variantes al programa:

Para evitar tener que crear variables que almacenen el valor temporalmente, podríamos haber realizado la conversión directamente del siguiente modo:

```
Edit2.text:=floattostr(strtof(float(edit1.text)/166.386))
```

Sin embargo, es más práctico y analizable, dividir el código en varias líneas.

Para controlar error de introducir texto en un Edit para valor numérico:

Se puede controlar con un **MaskEdit**, ó usando la función **Format**, **FormatFloat/FloatToStrF** o evaluando si es numérico antes de realizar operaciones matemáticas: **IsNan** (Is Not a Number). Véase la ayuda.

**Guardar el proyecto:** Guarda el proyecto y su unidad (convertor1.pas) en una carpeta independiente

\Projects\Convertor con el nombre del proyecto: **Convertor.dpr**

Crear y asignar un icono a la aplicación:

- Para Windows 7: Delphi 7 Incorporaba *Image Editor* que permitía crear imágenes \*.ico. Para Delphi XE: Utilizar un programa como **Microangelo** ([www.microangelo.us](http://www.microangelo.us)) o similar para crear iconos \*.ico
- Para Windows 10: Delphi XE 10.2 puede cargar un logo en formato PNG a 150 y 44 píxels.
- *Poner título al programa y cargar el Icono correspondiente:*
- Elige del menú: **Proyect ▶ Options** - Solapa: *Application*. *Load Icon* que previamente has guardado.
- En la solapa *VersionInfo*: Rellenar la Versión 1.0 y el nombre de la compañía.
- Vuelve a compilar el proyecto y sal del programa Delphi guardando todos los cambios.
- Comprueba cómo ha cambiado el archivo de la aplicación (convertor.exe). Es el único archivo que necesitas para el funcionamiento de esta aplicación.

### Distribución e instalación de la aplicación:

Instalación independiente para Windows:


Utiliza el Programa **InstallShield** o **InnoSetup** ([www.jrsoftware.org](http://www.jrsoftware.org)) para generar el programa de instalación: Indicando en este, dónde se encuentra el proyecto **Convertor.dpr** y/o el ejecutable.

Instalación multiplataforma: Más adelante se explica cómo se genera el archivo \*.apk en la subcarpeta Android (debug o release) y se sube a la *PlayStore* de Google. Delphi XE 10.2 permite generar también para la tienda de Windows 10 o la *Appstore* de Apple.

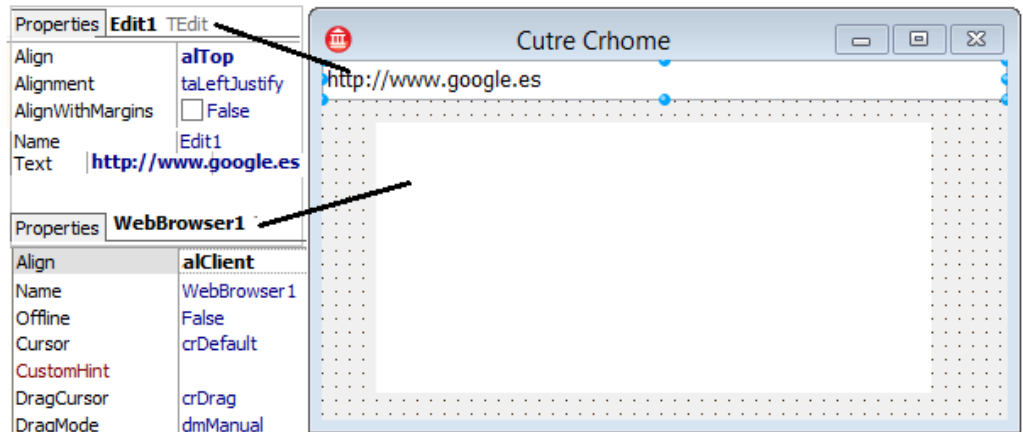
## Ejercicio: Navegador OfiChrome (CutreChrome)

- Crea un nuevo proyecto File ▶ New ▶ Application VCL (o *VCL Forms application*).

Diseño:

- Añade al formulario un objeto **Edit**  (de la paleta *Standard*) y un Objeto **WebBrowser** (de la paleta *Internet*)

- Distribuye y cambia las propiedades de la ventana-formulario como en la imagen



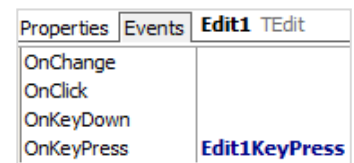
- Selecciona el *Edit 1* y en la pestaña *Events* pulsa doble clic sobre el evento: *OnKeyPress*. Escribe el código:

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
```

```
begin
```

```
    if key=#13 then WebBrowser1.Navigate(Edit1.Text); //si la tecla presionada es la nº 13 = INTRO
```

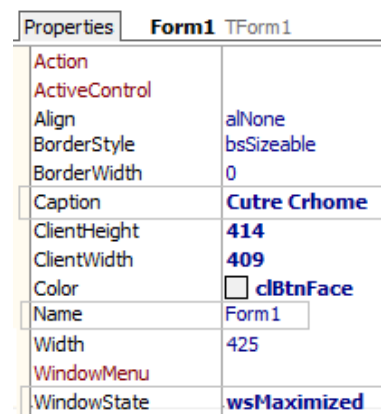
```
end;
```



Comentarios: Al pulsar la tecla INTRO en el edit1 pone en la dirección del navegador, el texto del Edit1

Algunos Códigos de tecla ASCII:	Valor
Retroceso {backspace},	#8
Tab {tab}	#9
Supr {delete} o {del}	#127
Entrar {enter} o ~	#13
Esc {esc}	#27

Ofimega

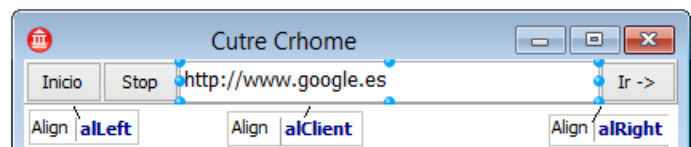


- **Probar la aplicación:** Pulsa Run ▶ para comprobar su funcionamiento.
- **Guardar el proyecto:** Crea una carpeta que se llame: OfiChrome
  - Escoge: File ▶ Save as... Para guardar el archivo: OfiChrome.pas
  - Escoge: File ▶ Save project as... OfiChrome.dpr

Mejoras:

En la ventana: En la propiedad de la ventana Form1 cambia su estado por maximizado: WindowState: wsMaximized y el texto de la ventana (Caption) por CutreChrome

- Añade un **Panel1** alineado al *Top*
- Donde pondremos tres botones y el Edit1 como de muestra en la imagen.



- Añadimos las acciones a los botones:

En el botón parar: WebBrowser1.Stop;

En el botón Inicio: WebBrowser1.Navigate('http://www.google.es');

En el botón Ir: WebBrowser1.Navigate(Edit1.Text);

- Puedes cambiar el objeto **Edit1** por un **ComboBox1**, añadiendo los eventos:

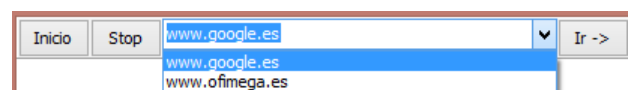
**KeyPress:** if key=#13 then

begin

WebBrowser1.Navigate(ComboBox1.Text);

ComboBox1.Items.Add(ComboBox1.Text);

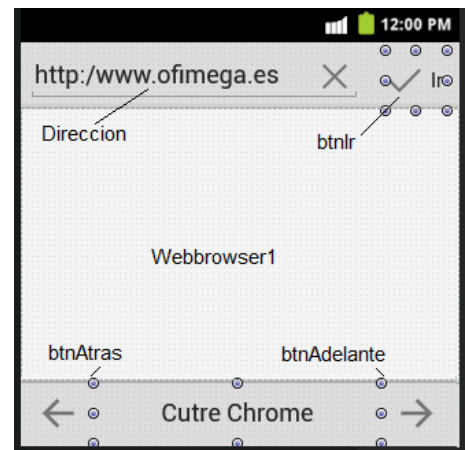
end;



**ComboBox1Select:** WebBrowser1.Navigate(ComboBox1.Text);

## Ejercicio Navegador CutreChrome para móvil Firemonkey (versiones XE para móviles)

- Crea un nuevo proyecto mobil en XE5/6: (File ► New ► FireMonkey Mobile Application). En XE7: *New – Multidevice application*
- Escoge la plantilla Header/Footer con encabezado y pie. Mobil Samsung Galaxy S2
- En el cuerpo, añade un componente WebBrowser alineado al cliente.
- En el encabezado, quita la etiqueta Label y añade de la paleta adicional un componente *ClearingEdit* llamado *Direccion* y en su propiedad *text* pon tu web de inicio favorita y añade un botón con el texto: Ir, nombre: *btnIr* y con el estilo lookup: *donetoolbutton*
- En el *footer* añade dos botones para ir adelante y atrás alineados a izquierda y derecha con apariencia *stilelookup*: *prriotoolbuttonbordered* y *nexttoolbuttonbordered* llamados: *btnAtras* y *btnAdelante* y un label alineado al cliente y *textAlign Center* con el nombre de la aplicación: Cutre Chrome.



### El código:

Botones adelante y atras:

```
WebBrowser1.GoBack; y WebBrowser1.GoForward;
```

Botón para ir:

```
procedure THeaderFooterForm.btnIrClick(Sender: TObject);
begin
  WebBrowser1.URL := Direccion.Text;
end;
```

Al teclear en dirección:

```
procedure THeaderFooterForm.DireccionKeyDown(Sender: TObject; var Key: Word;
var KeyChar: Char; Shift: TShiftState);
begin
  if Key = vkReturn then
  begin
    WebBrowser1.URL := Direccion.Text;
    btnIr.SetFocus; //pone el foco para quitar el teclado
  end;
```

### Mejoras:

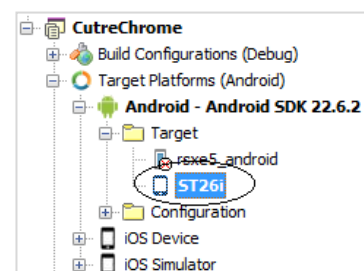
- En el control Edit, optimizar el Virtual Keyboard seleccionando *vktURL* como el tipo de teclado virtual adecuado en la propiedad *KeyboardType*.
- Establecer la propiedad *KillFocusByReturn* a *True*
- Cambiar el Edit por un *EditCombobox* al que podamos añadir elementos a modo de historial al pulsar en Ir:

KeyboardType	vktURL
KillFocusByReturn	<input checked="" type="checkbox"/> True

```
encontrado:=false;
for i := 0 to Direccion.Items.count-1 do
begin
  if Direccion.Items[i]=Direccion.Text then encontrado:=true;
end;
if encontrado=false then
begin
  //limita la lista a 10
  if Direccion.Items.count>10 then Direccion.Items.Delete(0);
  Direccion.Items.Add(Direccion.Text); //añadir a la lista...
end;
```

### Compilar:

- Compilar por emulador: Para probar la emulación de Android se debe instalar las **android SDK manager** y escoger en el *Target* el emulador: *rx5\_android* (muy lento)
- Compilar en el móvil: El teléfono o dispositivo android debemos conectarlo al PC y activar las *opciones del desarrollador* (pulsando varias veces en el numero de compilación) y activar en el modo desarrollador: **Depuración USB** . Entonces, al instalar el **driver ABD** en Windows, nos buscará y mostrará nuestro dispositivo en la lista. Luego activar el target para compilar en el móvil Android.



Guardar proyecto: **CutreChrome.dproj**

## Ejercicio Firemonkey 1. Calcular el área de un triángulo con estilo

### Versión para Firemonkey:

- Crea un nuevo proyecto. (File ▶ New ▶ **Multidevice o Firemonkey desktop application**). HD Application. En XE7: *New – Multidevice application*
- En la etiquetas Labels, poner el rótulo en la propiedad *Text*.
- Poner los componentes de la imagen:
- En el evento: **OnClick** del botón *Button1* poner:

```
Edit3.Text:=FormatFloat('###.##', StrtoFloat(Edit1.Text) *
StrtoFloat(Edit2.Text) / 2 );
```

**FormatFloat:** Convierte un valor decimal de coma flotante en un texto con el formato de número indicado.

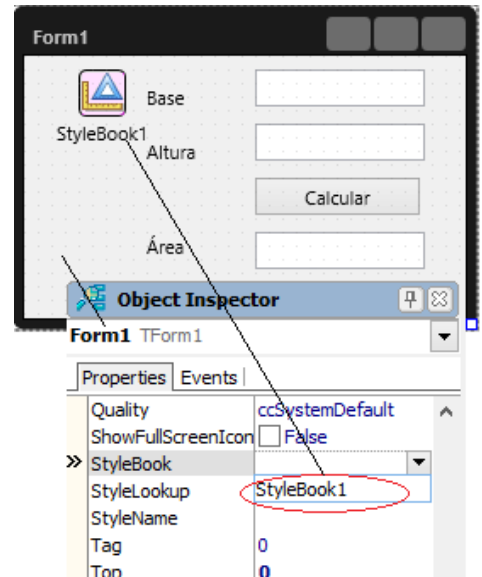
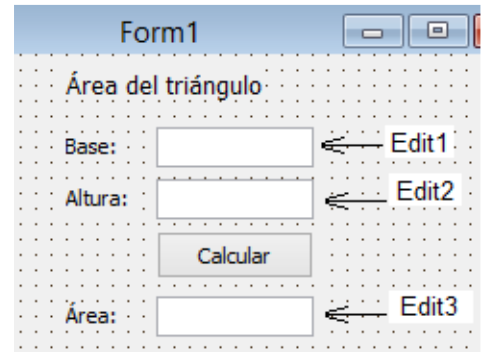
- Añadir al Form1 un componente: **StyleBook1**
- Pulsar doble clic sobre el componente **StyleBook1** y cargar (Load...) de la carpeta:  
Users\Public\Documents\RAD Studio\12.0\Styles el estilo:  
**AquaGraphite.style** – Apply and close
- Cambia la propiedad del formulario *Form1*. *Stylebook:* por **StyleBook1**

### Probar la aplicación:

Pulsa el botón **Run** ▶ o eligelo del menú *Ejecutar / Run* para comprobar su funcionamiento.

**Guardar el proyecto:** Crea una carpeta que se llame: *Contador*.

- Escoge: **File ▶ Save as...** Para guardar el archivo de código en la carpeta con el nombre: **Areas1.pas**
- Escoge: **File ▶ Save project as...** Para guarda el proyecto en la carpeta con el nombre **Areas.dpr**

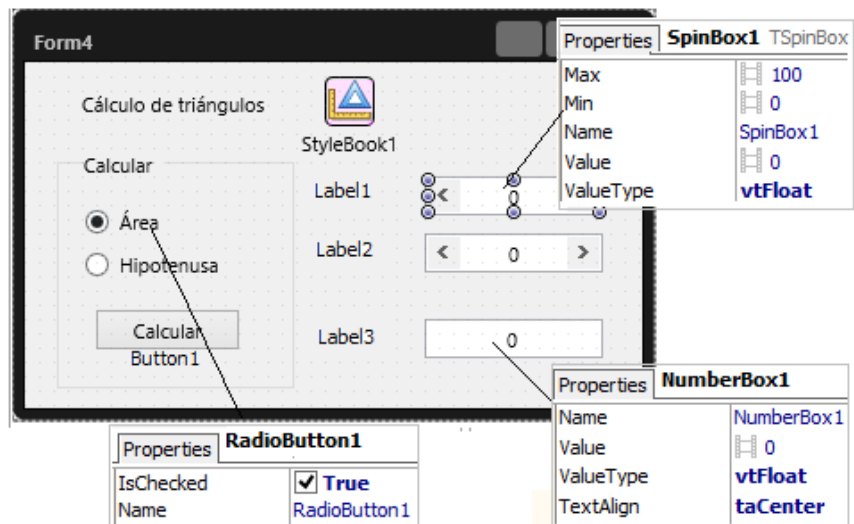


## Ejercicio FireMonkey2. Uso de los componentes de FireMonkey

Variación al ejercicio anterior de áreas del triángulo:

Los componentes con valor numérico como *SpinBox* y *NumberBox*, utilizan su propiedad *Value* y evitan tener que cambiar el formato de número.

- Crea un nuevo proyecto. (File ▶ New ▶ **Multidevice o Firemonkey desktop application**). HD Application
- Poner los componentes de la imagen:  
1 GropupBox, 2 RadioButtons, 2 SpinBox, 1 NumberBox, 1 Button, 4 Labels y un StyteBook →



El tipo de Valor *ValueType* de los cuadros numéricos será *Float* para permitir decimales (decimal de coma flotante).

- Cambia la propiedad del formulario Form1. *Stylebook:* por **StyleBook1**. Resource: **Transparent**
- Pulsa doble clic sobre el botón *Button1* y en el evento: **OnClick** del botón *Button1* añadir el código en negrita:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    if RadioButton1.IsChecked = true then //si esta activado Área...
        NumberBox1.Value:=SpinBox1.Value*SpinBox2.Value/2
    else NumberBox1.Value:=Sqrt(Sqr(SpinBox1.Value)+Sqr(SpinBox2.Value));
end;
```

- Llamaremos al mismo evento: **OnChange** en los dos RadioButton y al evento OnCreate del formulario Form1 para que se cambie el texto de las etiquetas al empezar el programa o al cambiar de selector:

```

procedure TForm4.RadioButton1Change(Sender: TObject);
begin
if RadioButton1.IsChecked then
begin
Label1.Text:='Base: ';
Label2.Text:='Altura: ';
Label3.Text:='Área: ';
end
else
begin
Label1.Text:='Cateto a: ';
Label2.Text:='Cateto b: ';
Label3.Text:='Hipotenusa: ';
end;
end;
end;

```



### Probar la aplicaci3n:

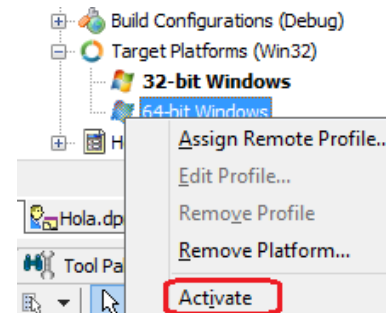
Pulsa **Run** para comprobar su funcionamiento →

**Guardar el proyecto:** Crea una carpeta que se llame: *Contador*.

- Escoge: **File** ▶ **Save as...** Para guardar el archivo de c3digo en la carpeta con el nombre: **triangulos1.pas**
- Escoge: **File** ▶ **Save project as...** Para guarda el proyecto en la carpeta con el nombre **triangulos.dpr**

### Activar la versi3n para Windows 64-bit:

- En el panel *Project Manager*, pulsa con el bot3n *Derecho* del mouse sobre *Target Platforms*.  
Add Platform...  
Escoge 64 bits Windows.
- Pulsa con el bot3n *derecho* del mouse sobre la plataforma 64 bits y escoge: *Activate*.
- Compila de nuevo la aplicaci3n y comprueba que se muestra en la carpeta *Win64\Debug* la aplicaci3n exe.

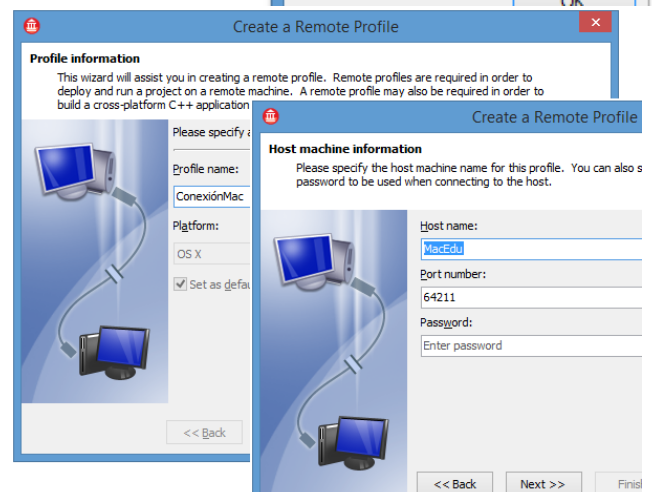
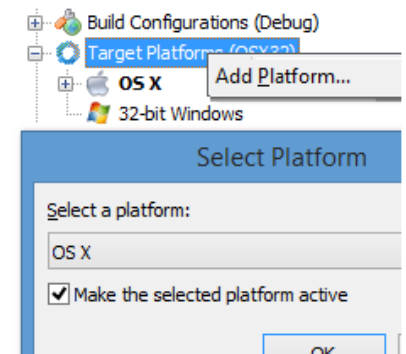


Nota: Si tu FM RAD Studio corre bajo x86 (Win32) podr3s compilar en 64 bits pero no podr3s ejecutarlo.

### Activar la plataforma Mac OsX y ejecutar en un Mac:

- Escoge del men3: **Tools > Options > Environment Options > Connection Profile Manager** page. Click en **Add**
- Escribe el nombre del profile: "**ConexionMac**". Plataforma OsX
- Pon el nombre de tu equipo Mac en la red o su IP
- En el panel *Project Manager*, pulsa con el bot3n *Derecho* del mouse sobre *Target Platforms*. Add Platform...Escoge: OSX.
- Pulsa con el bot3n *derecho* del mouse sobre la plataforma OSX y escoge: *Assign Remote Profile*.
- Selecciona **Add...** para abrir la ventana de conexi3n con un equipo mac por red.
- Escribe el nombre de la conexi3n: **ConexionMac**




Entonces, el IDE estar3 conectado con tu Mac.  
Si pulsas F9 se ejecutar3 la aplicaci3n en tu Mac



## Ejercicio Else If – variables y crear objetos. Adivina un número.

Creamos un nuevo proyecto VCL aplicación en Delphi y añadimos al formulario los siguientes...

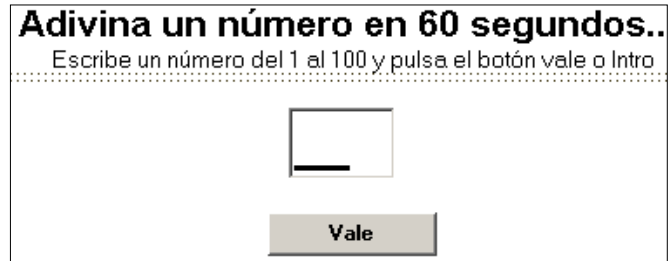
### Ingredientes:

- 2 TLabel 
- 1 TEdit 
- 1 TButton 

### Preparación:

Creamos 2 variables en la sección de variables públicas(al principio): Bueno e Intentos:

```
var
  Form1: TForm1;
→ Bueno: integer;
→ Intentos: integer;
```



### Cocción:

Al crearse el formulario se genera un número aleatorio del 1 al 100 y se guarda en la variable buena:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Randomize; // activa el generador aleatorio (baraja)
  bueno:=random(100); //crea un num aleatorio entre 0 y 100 y lo guarda en la variable bueno
  intentos:=0; //pone la variable intentos a cero
end;
```



Acciones: Al pulsar el botón *Vale*, se crea una variable num y se evalúa si la variable num es igual o mayor a la variable bueno:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  num:integer;
begin
  intentos:=intentos+1;
  If Edit1.text='' then showmessage('Escribe un número') // -> si no escriben nada = ' '
  else
  begin
    num:=strtoint(Edit1.text); //-> Variable num guarda el dato del Edit1
    if num>bueno then showmessage('Es más bajo')
    else if num<bueno then showmessage('Es más alto')
    else if num=bueno then acierto;
    Edit1.text:=' '; //-> borra el núm del Edit1
  end;
end;
```

Se crea un procedimiento manual llamado **acierto** sin parámetros:

```
procedure TForm1.acierto;
var
  i:integer;
begin
  Edit1.visible:=false;
  Label2.Caption:='Acertaste, es el '+inttostr(bueno);
  Label1.Caption:='Te ha costado '+ inttostr(intentos)+' intentos y has tardado '+ inttostr(60-
  ProgressBar1.position)+' segundos';
  for i:=1 to 100 do // Bucle finito: repite el bucle 100 veces
  begin
    with TButton.Create(self) do //se autocrea un botón que aparecerá aleatoriamente en el form
    begin
      SetBounds (Random (panel1.Width-100), Random (Panel1.Height-50), 100, 50);
      Parent := Panel1;
      Caption := Label2.Caption;
    end;
  end;
end;
```

### Mejoras:

- Añadir una barra de progreso: **progressbar1**  (Paleta win32) Progressbar: Max: 60 Position: 60
- Añadir un Temporizador: **Timer**  (Paleta System)
- Interval: 1000 (Repetirá el evento cada 1 segundo)

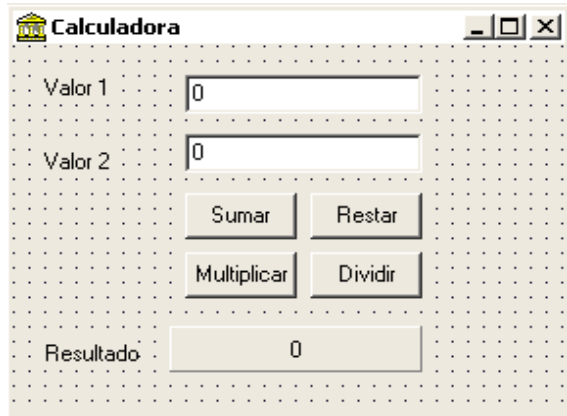
En el evento del timer **OnTimer** añadir el código de la derecha para evaluar el tiempo tardado en acertar y decrementar la barra de progreso.

- Guarda la unidad (Adivina1.pas) y el proyecto (Adivina.DPR)

```
Procedure TForm1.Timer1Timer
begin
  ProgressBar1.Position:=ProgressBar1.Position-1;
  if ProgressBar1.Position=0 then
  begin
    Showmessage('Se acabó el tiempo');
    Timer1.enabled:=false;
  close;
  end;
end;
```

## Ejercicio calculadora. Uso del parámetro sender y de las variables

Creemos un nuevo formulario. File – New - Application



### Ingredientes: (Componentes)

3 Labels: Valor 1 – Valor 2 - Resultado

2 Edits [ab]: Edit1 – Edit2

4 botones:

Button1: Sumar

Button2: Restar

Button3: Multiplicar

Button4: Dividir

1 Panel: Caption: 0

### Acciones:

Pulsar doble clic en el botón 1 para escribir la acción dentro del procedure:

**panel1.caption:=edit1.text + edit2.text;** → da error

Primer programa:

- Escribe el código correcto en negrita: Comprueba que funciona el botón, (Pulsar Run ► ó F9)
- Ídem deberíamos escribir los otros tres botones, pero en lugar de ello vamos a crear dos variables públicas para cada número. Busca la sección **Var** al inicio de la unit1.pas y escribe el código en negrita:
- Ahora cambia el código del botón 1 por este otro: Comprueba que funciona el botón, (pulsar Run ► ó F9)

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  panel1.caption:=inttostr(strtoint(edit1.text) +
strtoint(edit2.text));
end;
```

```
var
  Form1: TForm1;
numero1:double;
numero2:double;
```

Segundo programa:

### Llamada al mismo

#### procedure:

Ídem deberíamos escribir los otros tres botones, pero en lugar de ello vamos a utilizar el mismo procedure para todos los botones, utilizando el parámetro **sender** y evaluando cuál es el botón pulsado con la función IF ... ELSE IF... Cambia el código por el que se muestra. Luego escogemos para los otros tres botones el evento "On clic" y le asignamos el mismo procedure que para el botón 1:



```
procedure TForm1.Button1Click(Sender: TObject);
begin
  numero1:=strtofloat(edit1.text); // convierte a decimal flotante el edit1
  numero2:=strtofloat(edit2.text); // convierte a decimal flotante el edit2
  panel1.caption:=floattostr(numero1+numero2);
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  numero1:=strtofloat(edit1.text); // convierte a decimal flotante el edit1
  numero2:=strtofloat(edit2.text); // convierte a decimal flotante el edit1
  if sender=button1 then
    panel1.caption:=floattostr(numero1+numero2)
  else if sender=button2 then
    panel1.caption:=floattostr(numero1-numero2)
  else if sender=button3 then
    panel1.caption:=floattostr(numero1*numero2)
  else if sender=button4 then
    panel1.caption:=floattostr(numero1/numero2)
end;
```

De este modo, los tres botones realizan el mismo procedure pero utilizamos el parámetro Sender con IF ... else if para evaluar qué botón se ha pulsado en cada caso.

**Mejoras:** ¿Qué pasa si dividimos un número por cero? → Devuelve un error el programa.

Escoge del menú: Program → Reset para desbloquear el programa.

En el caso de dividir deberíamos evaluar que si numero2=0 entonces muestre un mensaje de que no se puede realizar la operación. Para ello, añadir la siguiente línea:

```
If numero2=0 then showmessage('No se puede dividir por cero');
```

(Sin olvidar poner un **begin – end** cuando escribimos varias líneas de código dentro de un **IF**)

**Guarda** este proyecto en tu carpeta con el nombre: **Calculadora.DPR** y su unidad pas: **Calculadora1.PAS**

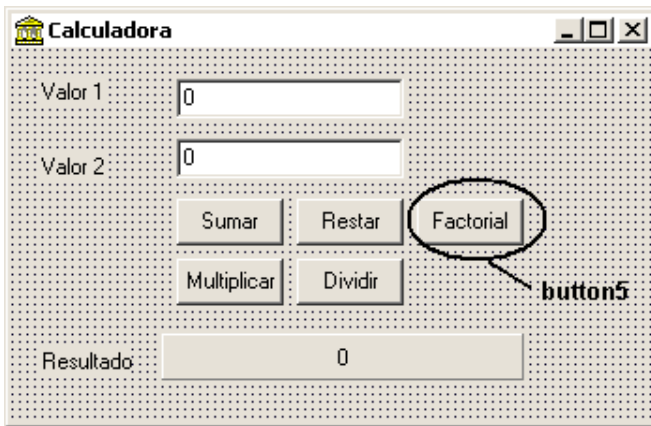
### Ejercicio While. Factorial en la calculadora.

**While:** Repite la ejecución infinitas veces hasta que se cumpla la condición.

En matemáticas, factorial o permutación multiplica un número por el siguiente inferior hasta 1.

Ejemplo:  $5! = 5 * 4 * 3 * 2 * 1 = 120$

Abre el proyecto anterior: CALCULADORA y añade el siguiente botón: *Button5*. Añade el código que se muestra y comprueba su funcionamiento ▶:



```
procedure TForm1.Button5Click(Sender: TObject);
var
  resultado:double;
begin
  resultado:=strtofloat(edit1.text);
  numero1:=resultado-1;
  while numero1>0 do
    begin
      resultado:= resultado*numero1;
      numero1:=numero1-1;
    end;
  panel1.caption:=floattostr(resultado);
end;
```

Al finalizar, guarda este proyecto en tu carpeta con el nombre: CALCULADORA2

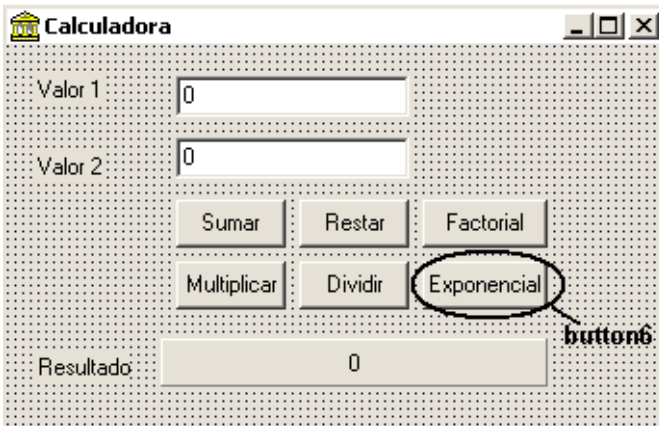
### Ejercicio FOR. Exponencial en la calculadora.

**For ... To:** Ejecuta una acción un número determinado de veces.

En matemáticas, exponencial multiplica un número por sí mismo tantas veces como diga el exponente.

Ejemplo:  $5^3 = 5 * 5 * 5 = 625$

Abre el proyecto: CALCULADORA2 y añade el siguiente botón: *Button6*. Añade el código que se muestra y comprueba su funcionamiento ▶:



```
procedure TForm1.Button6Click(Sender: TObject);
var
  resultado:double;
  x,veces:integer;
begin
  resultado:=strtofloat(edit1.text);
  numero1:=resultado;
  veces:=strtoint(edit2.text);
  for x:=1 to veces do
    begin
      resultado:= resultado*numero1;
    end;
  panel1.caption:=floattostr(resultado);
end;
```

Guarda este proyecto en tu carpeta con el nombre: CALCULADORA3

### Mejora en el aspecto de los componentes:

**Botones planos con imágenes:**

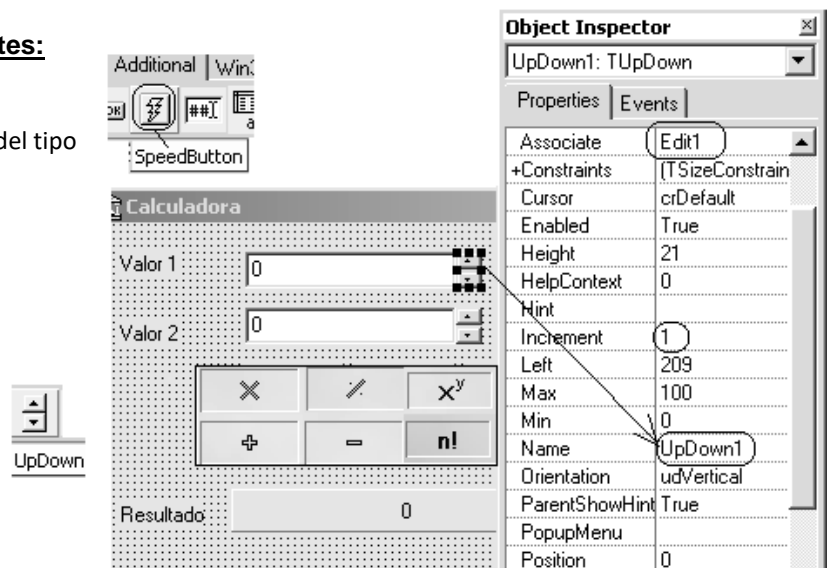
Sustituye los botones del tipo *button* por los del tipo *SpeedButton* de la paleta adicional.

Pon su propiedad *Flat*: *True*.

Cambia el texto del botón por una imagen (propiedad *Glyph*) diseñada por tí en Paint.

**Añadir incrementadores de valor:**

Añade dos componentes *UpDown* de la paleta Win32 y cambia las propiedades que se indican en la figura para incrementar los valores del edit con flechas.

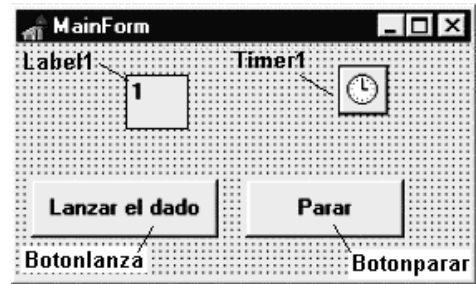


## Ejercicio 2. Lanzar un dado en un juego:

En este ejercicio al pulsar el bot3n **Lanzar** se activa el timer y al pulsar el bot3n **Parar** desactiva el timer y muestra el dato del timer. Por lo tanto, el programa dispondr3 de tres eventos o acciones:

1. procedure botonLanzarClick(Sender: TObject);
2. procedure botonpararClick(Sender: TObject);
3. procedure Timer1Timer(Sender: TObject);

Necesitamos crear antes de los *procedures* una variable **X** que es la que contendr3 el valor del dado. Ser3 del tipo num3rica entera (Integer) y la declararemos en el apartado VAR al principio del programa.



```
var
  MainForm: TMainForm;
  x: integer; // -> variable para contador
implementation
```

### Los comentarios:

Puedes apuntar comentarios en tus programas utilizando:  
 // -> La doble barra: Para una l3nea  
 {} -> Las llaves para varias l3neas

En los tres procedimientos siguientes, s3lo tendremos que escribir el texto en **negrita**. El resto se escribe autom3ticamente al pulsar doble clic en el componente o sobre el evento adecuado del componente (F11)

```
procedure TMainForm.botonLanzarClick
begin
  timer1.enabled:=true;
end;
```

```
procedure TMainForm.botonpararClick
begin
  timer1.enabled:=false;
  label1.caption:='has sacado un '+ inttostr(x);
end;
```

```
procedure TMainForm.Timer1Timer(Sender: TObject);
{poner interval del timer1 a 50
 poner en el var general x: integer}
begin
  if x<6 then x:=x+1 else x:=1; {Si X es menos que 6 suma 1 y si pasa de 6 valdr3 1 }
  label1.caption:=inttostr(x); {Ponemos al texto de la etiqueta el valor de x }
end;
```

### Uso de una variable como "Flag" (bandera de permiso)

#### Utilizar el mismo bot3n para lanzar y parar:

Borra el bot3n 2, haciendo clic en 3l y pulsando **Suprimir**

Crea otra variable llamada **lanzar** del tipo integer, por defecto, esta variable valdr3 0: **Lanzar:integer;**

La variable lanzar es un "flag" o bandera que al comparar con ella, el mismo bot3n realizar3 una acci3n u otra. Cambia el c3digo del bot3n lanzar por este otro: (se aconseja escribir tambi3n los comentarios)


#### Variantes y mejoras:

1.- Añade el componente **Imagelist (W32)** al que se le añaden (add) las im3genes *bmp* de las 6 caras de un dado.(que previamente has dibujado con la aplicaci3n: Tools - **Image Editor** - File - New - Bitmap)

```
procedure TForm1. TMainForm.botonLanzarClick (Sender: TObject);
begin
  if lanzar=0 then// Si lanzar vale cero ...
  begin
    timer1.enabled:=true; // Activa el timer
    button1.caption:='Parar'; // En el bot3n 1 pondr3 Parar
    lanzar:=1; // y sube la bandera
  end
  else // Y si no...
  begin
    timer1.enabled:=false; // Desactiva el timer
    button1.caption:='Lanzar'; // En el bot3n 1 pondr3 Lanzar
    lanzar:=0; // y baja la bandera
  end
end;
```

2.- Añade un componente **Image1 (additional)** y añade al timer: **imagelist1.GetBitmap(x, image1.picture);**

#### Extensi3n: Librer3as Jedi:

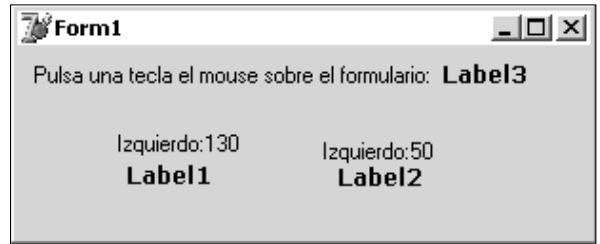
Si tienes instaladas las librer3as JVCL - JEDI, utiliza el componente en la paleta: JVVisual: **JVDice**  con sus propiedades: Autostop Interval: 800 y Rotate true.

## Ejercicio de Captura y pulsación de botón de ratón:

En los eventos del ratón (mouse):

**On mouse down, On mouse up y On mouse Move**, los parámetros X y Y son automáticos y nos devuelven la información de la posición del mouse.

En los eventos del ratón (mouse) *On mouse down* y *On mouse up*, también existe el parámetro *Button* que nos informa si el botón pulsado o liberado es el izquierdo (mbleft) o el derecho (mbright).



- ▶ Crear un proyecto nuevo llamado: *Mouse.dpr* y añada al formulario, los componentes que se muestran en la figura.
- ▶ Con el formulario seleccionado, busca en pestaña *Events*, el evento *onMouseDown* y pulsa doble clic para crear el procedure.
- ▶ Agrega el código siguiente:

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft then
  begin
    label1.Caption:='Izquierdo:' + inttostr(X); //guarda posición X del ratón
    label2.Caption:='Izquierdo:' + inttostr(Y); //guarda posición Y del ratón
  end
  else if Button = mbRight then
  begin
    label1.Caption:='derecho:' + inttostr(X);
    label2.Caption:='derecho:' + inttostr(Y);
  end;
end;
```

- ▶ Comprueba (*Run* ▶ ó F9) y guarda el proyecto y la unidad en tu carpeta al finalizar.

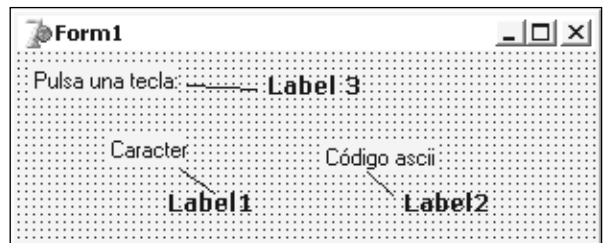
## Ejercicio de Captura y pulsación de Teclado. Tecla

Para detectar cuándo el usuario pulsa una tecla, se emplea los eventos: *on key press* y *on key down* (en la pestaña *Events* del *Object Inspector*)

En los eventos *on key press* y *on key down*. Los procedures nos informan, con la variable *key*, del código de la tecla pulsada.

**Ord (Char)** → devuelve el valor numérico ASCII que le corresponde un caracter

- ▶ Crear un proyecto nuevo llamado: *Tecla.dpr* y añada al formulario, tres *labels* con los *captions* (rótulos) que se muestran en la figura:→
- ▶ Con el formulario seleccionado, busca en pestaña *Events*, el evento *onkeypress* y pulsa doble clic para añadir el procedure.
- ▶ Agrega el código siguiente:



```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  label1.caption:=key;
  label2.caption:=IntToStr(Ord(key)); //Ord(key) devuelve el número ordinal de la tecla pulsada (1 a 256)
  if key = #27 then close; //Si pulsan la tecla escape se termina el programa
end;
```

- ▶ Comprueba (*Run* ▶ ó F9) y guarda el proyecto y la unidad en tu carpeta al finalizar.

## **Variante al ejercicio de la calculadora:**

- ▶ Capturar la tecla de la operación con el teclado:
- ▶ Recupera el proyecto de la calculadora
- ▶ Asigna al formulario la propiedad **KeyPreview** = True
- ▶ Añade el siguiente procedure al evento *OnKeyPress* del formulario →

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
var
  num: integer;
begin
  case key of
    #13, #61:operarClick(opeigual);{teclas intro}
    #43 : Button1Click(mas);
    #45 : Button1Click (menos);
    #42 : Button1Click (por);
    #47 : Button1Click (div);
  end;
```

### Ejercicio 3. Crear un contador de tiempo. Practicar con tipos de variables.

**Variables pùblicas y privadas:** Crearemos dos variables para almacenar dos datos: la hora de inicio y la hora final. Si estas variables van a ser utilizadas en todo el programa, se crea al principio, en el apartado var (pùblicas), si no se crea el apartado var dentro del procedure (privadas).



ejemplo:

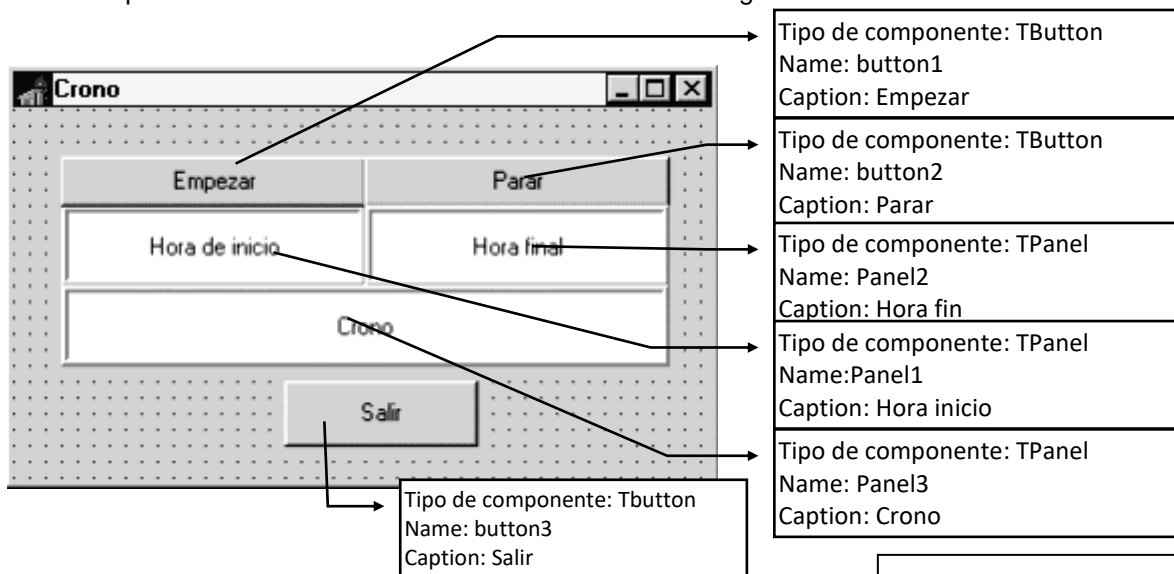
```
var
  Form1: TForm1;
  horainicio: double; //no existe el tipo time, se elige el tipo numérico doble. Permite decimales doble
  precision
  horafin: double;
  I: Integer; // numérico entero)
  S: string; // del tipo cadena de texto
```

**otros tipos de variables:**

Integer	Shortint	SmallInt	Longint	Byte	Word
Cardinal	Boolean	ByteBool	WordBool	LongBool	Char

#### Práctica:

- Crear una nueva aplicaci3n VCL llamada **CRONO**. Añade al formulario (ventana) los siguientes componentes:
  - 3 botones 
  - 3 paneles  de color blanco como se muestra en la figura:



- Añadir las siguientes variables al inicio de la unidad:
- Añadir las líneas de código a los procederes:

```
var
  Form1: TForm1;
  horainicio: double;
  horafin: double;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  horainicio:=time;
  panel1.Caption := 'Hora de inicio: ' + TimeToStr(Time);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  horafin:=time;
  panel2.Caption := 'Hora de paro: ' + TimeToStr(Time);
  panel3.caption:= 'Crono: ' + TimeToStr(horainicio-
  horafin);
end;

procedure TForm1.Button3Click (Sender: TObject);
begin
  close;
end;
```

#### Nivel avanzado

##### **Cómo crear un archivo INI que guarde la hora de entrada.**

- Es necesario añadir INIFILES a la sección USES de la unidad.

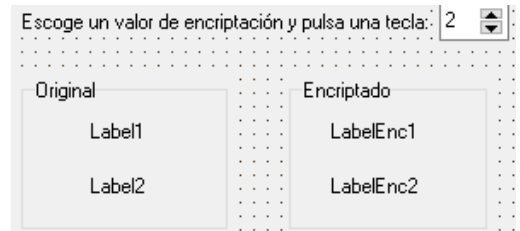
```
procedure TForm1.Button1Click(Sender: TObject);
var
  datosini:TIniFile;
  HORAENTRADA:STRING;
begin
  HORAENTRADA:=TIMETOSTR(TIME);
  datosini:= TIniFile.Create('DATOS.INI');
  -->crea el archivo datos.ini
  datosini.WriteString('Eduardo', 'Entrada',
  HORAENTRADA); -->añade a la sección
  [Eduardo] entrada=12:15
  DATOSINI.FREE; --> cierra el archivo datos.ini
end;
```

## Encriptación de caracteres por teclado

Crear un proyecto nuevo llamado: Tecla\_encriptada.dpr, similar al anterior pero con las variaciones de la imagen.

Añade el siguiente procedure al evento *OnKeyDown* del formulario:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var
  codigo:byte; //variable numérica de 256 valores
begin
  label2.Caption:=inttostr(key);
  if (Key=VK_RETURN) then Label2.caption:='Has pulsado Enter';
  if (Key=VK_ESCAPE) then Label2.caption:='Has pulsado Esc';
  codigo:=key+spinedit1.Value;
  labelEnc1.Caption:=inttostr(codigo);
  labelEnc2.Caption:=chr(codigo);
end;
```

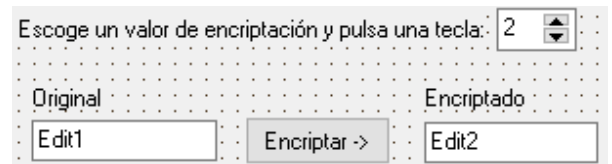


Virtual Keys Codes	
VK_RETURN	Enter key
VK_SHIFT	Shift key
VK_CONTROL	Ctrl key
VK_LBUTTON	Left mouse
VK_RBUTTON	Right mouse
VK_CANCEL	Control+Break
VK_BACK	Backspace key
VK_TAB	Tab key
VK_ESCAPE	Esc key
VK_UP	Up Arrow key
VK_RIGHT	Right Arrow key
VK_DOWN	Down Arrow key

## Encriptación de caracteres de texto

- ▶ Crear un proyecto nuevo llamado: *Texto\_encriptado.dproj*, similar al anterior pero con las variaciones de la imagen →
- ▶ Añade el siguiente procedure al pulsar el botón:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  original,encriptado:string; //variables para almacenar los textos
  i:integer; //var numerica entera para el vontador for
begin
  original:=edit1.Text; // almacena el valor del edit1
  encriptado:=original;
  for i := 1 to Length(original) do //lo hace las veces del ancho del texto
    encriptado[i]:=chr(ord(original[i])+spinedit1.Value); //convierte de char a número y le suma el del spin
  edit2.Text:=encriptado; //muestra el resultado
end;
```



## **Encriptación de texto en un Memo o en un Richedit:**

En este caso llamamos a la propiedad *lines* para para completar las líneas del memo mediante un *for*. Hasta finalizar todas las líneas (*linescount*):

```
for i:=0 to memo1.Lines.Count do
begin
  memo2.lines.add(memo1.lines[i]);
end;
```

## Ventanas de Mensajes y cuadros de diálogo directos.

### Ventanas de mensaje:

Modo Delphi: `Showmessage('Mensaje');`  
`MessageDlg('Mensaje', mtInformation, [mbOk], 0)`

Modo Windows: `Application.MessageBox('Mensaje', 'Textoventana', MB_OK);`

### Ventana de pregunta:

`Application.MessageBox('Pregunta', 'Textoventana', MB_YESNO)`

`MessageDlg('Mensaje', mtConfirmation, [mbYes, mbNo], 0)` → Devuelve: mrYes o mrNo

`InputBox('Nombre_ventana', 'Texto_Dato', 'Respuesta_predeterminada');` → Devuelve Var String

`InputQuery('Identificate!', 'Pon tu nombre, por favor...', Nombre);` → Devuelve Var Boolean

### Otros mensajes:

`ShowMessageFmt(const Msg: string; Params: array of const);`

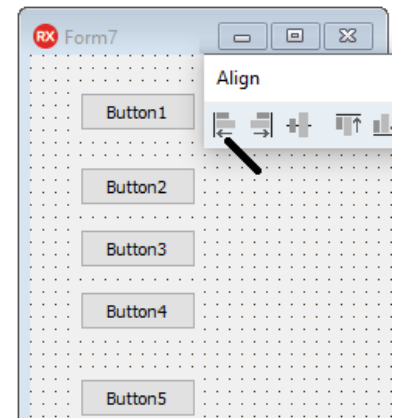
`CreateMessageDialog(const Msg: string; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons)`

En la plataforma FMX Firemonkey multidevice & mobile:

Véase [MessageDialogSync](#) y [ShowMessageSync](#) de la librería IFMXDialogServiceSync.MessageDialogSync

### Ejercicio de mensajes:

- En una nueva aplicaci3n VCL, añaede 5 botones como en la imagen:
- Con la paleta de alineaci3n, puedes situarlos todos verticalmente alineados.
- Asigna a cada uno de ellos el siguiente c3digo:



```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var
  Dato: string;
begin
  Dato:= InputBox('Nombre ventana', 'Nombre:', 'Antonio');
  showmessage('Hola, '+ Dato);
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
```

```
var
  Nombre:string;
  PulsoOk:boolean;
begin
  PulsoOk:= InputQuery('Identificate!', 'Pon tu nombre, por favor...', Nombre);
  if PulsoOk=true then ShowMessage('Envantado de conocerte, '+Nombre)
  else ShowMessage('Has de pulsar Ok...');
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
```

```
begin
if (Application.MessageBox('¿Eres tonto?', 'Tontería', MB_YESNO) = IDYES) then showmessage('Lo sabía')
else showmessage('Respuesta incorrecta')
end;
```

```
procedure TForm1.Button4Click(Sender: TObject);
```

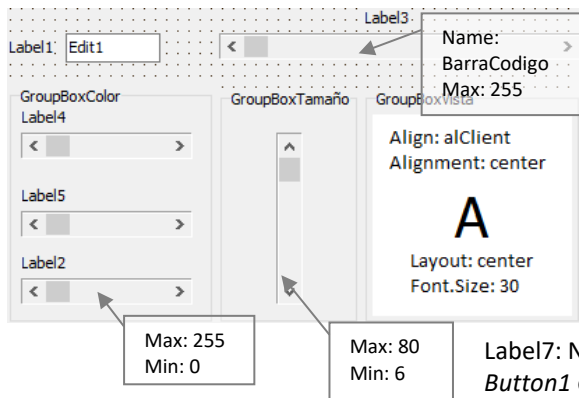
```
begin
Application.MessageBox('Este es un mensaje con botón Aceptar', 'Informaci3n', MB_OK);
end;
```

```
procedure TForm1.Button5Click(Sender: TObject);
```

```
var
  Nombre:string;
  PulsoOk:boolean;
  pregunta:pchar;
begin
  PulsoOk:=false;
  while PulsoOk=false do
  begin
  PulsoOk:= InputQuery('Identificate!', 'Pon tu nombre, por favor...', Nombre);
  if PulsoOk=false then ShowMessage('Has de pulsar Ok...');
  end;
  Pregunta:=Pchar('Hoye ' + Nombre + ' , ¿eres tonto?');
  while Application.MessageBox(Pregunta, 'Tontería', MB_YESNO) = IDNO do
  begin
  MessageDlg('Respuesta incorrecta', mtError, [mbOk], 0);
  end;
  MessageDlg('Respuesta correcta', mtInformation, [mbOk], 0);
end;
```

## Ejercicio de controles.

### Resumen de controles usuales. (Válido para Visual basic & Delphi VCL)



En una nueva aplicación (File – New – VCL Application – Delphi )  
Añade desde la paleta los controles que se muestran en la figura  
y cambia las **propiedades** como se indica, para que quede como  
en la 2ª figura:

Label 1: Nombre: Carácter

Label2: Nombre: ASCII

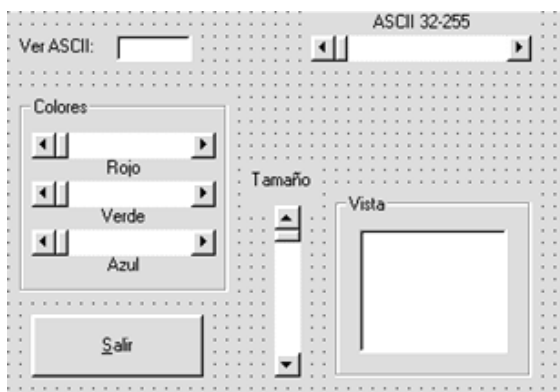
Label 8: Nombre: VerAscii - Backcolor: Blanco - Center

Label5: Nombre: Rojo Caption: Rojo

Label6: Nombre: Verde Caption: Verde

Label7: Nombre: Azul Caption: Azul

Button1 o Command1: Caption: &Salir



Las Barras de desplazamiento, en Visual basic tiene la propiedad  
*Value* mientras que en Delphi tienen la propiedad *Position*.

El rango de caracteres ASCII y de combinación de colores RGB  
son de 256

- **Chr(Valor):** Convierte un numero en un carácter ASCII
- **RGB(valor1, valor2, valor3)** -> Convierte tres números en un color

#### Código para Delphi (Vcl)

```
procedure TForm4.BarraCodigoChange...  
begin  
VerAscii.Caption :=Chr(Barracodigo.Position);  
end;
```

```
procedure TForm4.BarraRojoChange...  
begin  
VerAscii.Color:= RGB(BarraRojo.Position,  
BarraVerde.Position, BarraAzul.Position);  
Rojo.Caption = "Rojo:" +  
inttostr(BarraRojo.Position);  
end;
```

```
procedure TForm4.BarraTamanoChange...  
begin  
VerAscii.Font.Size:= BarraTamano.Position;  
end;
```

#### Código para Microsoft Visual Basic:

```
Private Sub Form Load()  
VerAscii.FontName = "Times New Roman" //tipo de letra  
End Sub
```

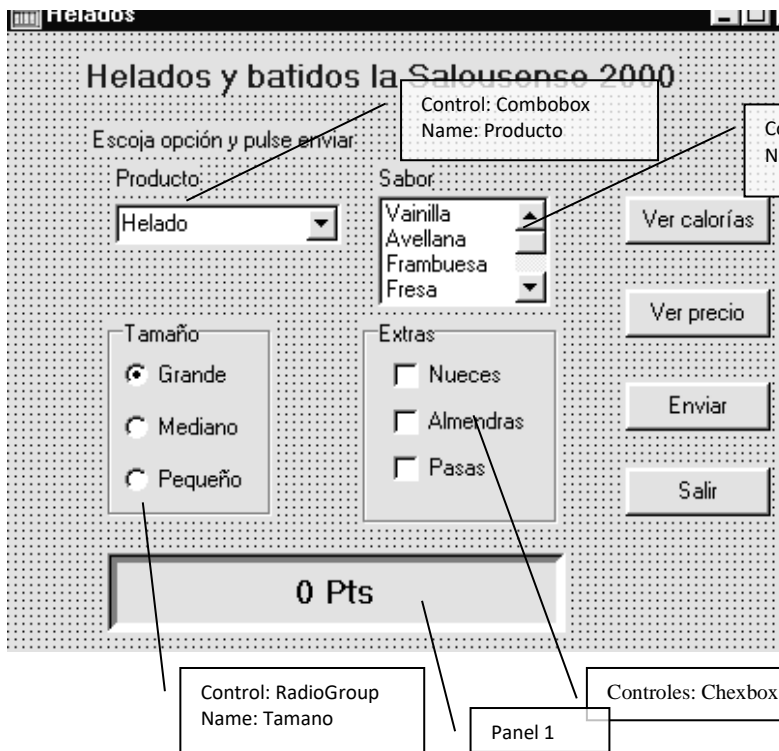
```
Private Sub TamCaracter Change()  
VerAscii.FontSize = TamCaracter.Value 'tamaño  
Label4.Caption = "Tamaño-" + Str(TamCaracter.Value)  
End Sub
```

```
Private Sub (Procedure) Colorrojo Change()  
VerAscii.ForeColor = RGB(Colorrojo.Value,  
Colorverde.Value, ColorAzul.Value)  
Rojo.Caption = "Rojo -" + Str(Colorrojo.Value)  
End Sub
```

```
Private Sub (Procedure)Barracodigo Change()  
ASCII.Text = Chr(Barracodigo.Value)  
VerAscii.Caption = Chr(Barracodigo.Value)  
Label3.Caption = "Valor Ascii-" + Str(Barracodigo.Value)  
End Sub (end;)
```

```
Private Sub (Procedure) ASCII Change()  
VerAscii.Caption = Chr(Barracodigo.Value)  
If ASCII.Text <> "" Then Barracodigo.Value =  
Asc(ASCII.Text)  
End Sub
```

## Ejercicio 5. Prácticas con controles de lista y elecci3n.



1.- Crear el formulario **Helados.frm** con los nombres y controles que se indican en la figura.

2.- Crear la variable pública:  
precio: integer;

3.- Crear un segundo formulario llamado **Envio**:

(File – New – Form) Name: *Envio*

Al hacer clic en el botón enviar:

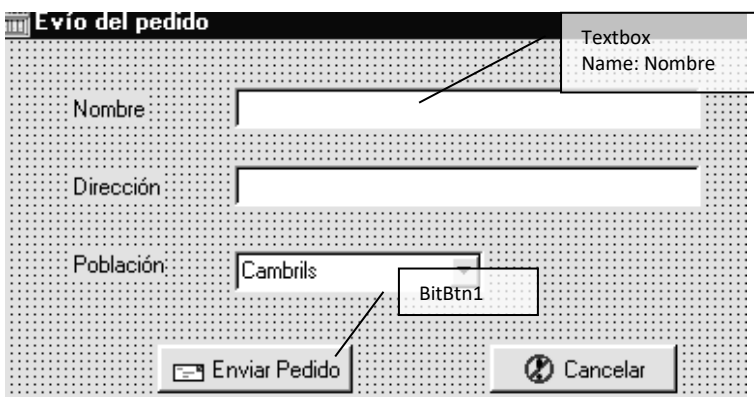
**envio.showmodal;**

Guarda el proyecto con el nombre:

**Helados1.pas** y **Helados.dpr**

**Al hacer clic en el botón calorías:**

```
procedure THelados.Button1Click
var
  calorias, calorias1:integer;
begin
  if sabor.itemindex=-1 then
  begin
    ShowMessage('elige un sabor');
    calorias:=0;
  end
  else
  begin
    Button2Click(Sender);
    calorias:=precio*10;
    case sabor.itemindex of
      0:calorias1:=10;
      1:calorias1:=50;
      2:calorias1:=100;
      3:calorias1:=150;
      4:calorias1:=200;
      5:calorias1:=250;
    end;
    calorias:=calorias+calorias1;
    panel1.caption:=inttostr(calorias)+'
    Calorías';
  end;
end;
```



**Al enviar el pedido:**

```
procedure Tenvio.BitBtn1Click(Sender: TObject);
begin
  if nombre.text = '' then
    ShowMessage('Haga el favor de rellenar sus datos,
    gracias')
  else
  begin
    ShowMessage('Señor ' + nombre.text + ', su pedido ha
    sido enviado a ' + poblacion.text + ', en breves
    minutos recibira el producto. Gracias');
  end
end;
```

**Al cancelar el pedido:**

```
procedure Tenvio.BitBtn2Click(Sender: TObject);
begin
  ShowMessage('Su pedido ha sido cancelado');
close;
end;
```

Llamada extena al enviar el pedido: **ShellExecute**  
(añadir librería *shellapi*)

```
ShellExecute(GetDesktopWindow(), nil, pChar('mailto:'+
  direc), nil, nil, SW_SHOWNORMAL);
//mailto:' + direc + '?Subject=' + asunto + '&Body=' + cuerpo
```

**Al hacer clic en el botón Ver Precio:**

```
procedure THelados.Button2Click(Sender: TObject);
begin
  precio:=0;
  if producto.text='Helado' then precio:=200;
  if producto.text='Batido' then precio:=150;
  if producto.text='Refresco' then precio:=100;
  if nueces.State=cbChecked then precio:=precio+50;
  if almendras.State=cbChecked then
    precio:=precio+50;
  if pasas.State=cbChecked then precio:=precio+50;
  case tamano.itemindex of
    0:precio:=precio+100;
    1:precio:=precio+50;
    2:precio:=precio;
  end;
  panel1.caption:=inttostr(precio)+' Pts';
end;
```

## Ejercicio 4. Visor de imágenes.

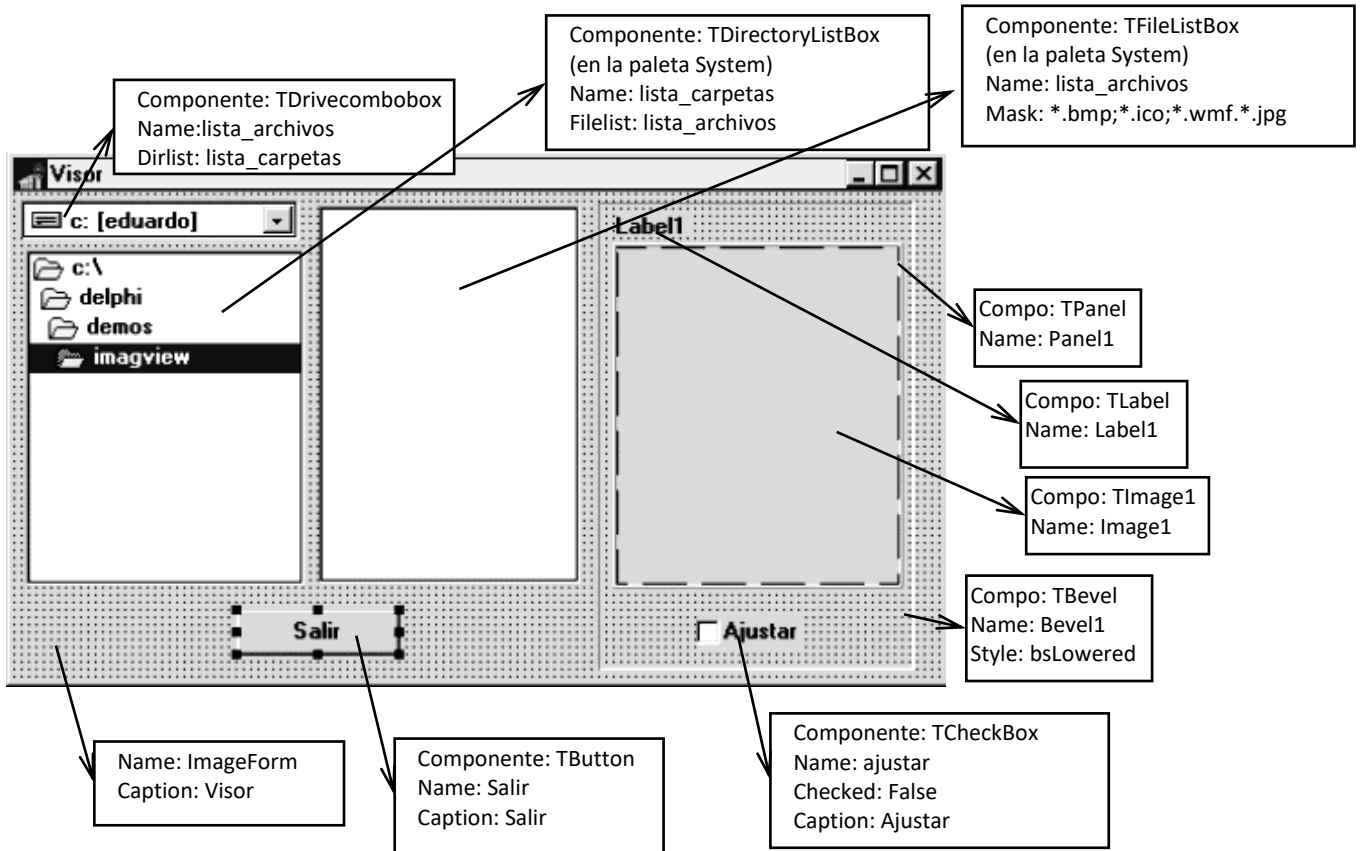
Válido sólo para el tipo de aplicación formulario VCL Form para Windows.

Vamos a crear una aplicación visor de imágenes del tipo \*.pcx;\*.cur;\*.gif;\*.ani;\*.jpg;\*.jpeg;\*.bmp;\*.ico;\*.emf;\*.wmf.

### Paso 1. Poner los componentes y establecer las propiedades.

Crear **Nuevo** proyecto/aplicación del tipo formulario VCL Delphi. Añade 2 paneles y un split para distribución vertical.

Busca en la *tool palette Win31* los componentes de la figura y modifica las siguientes propiedades:



### Paso 2. Escribir los códigos de los eventos

Al hacer clic en una unidad de disco (*Drive combobox*) se cambiarán la lista DirectoryListBox (*lista\_carpetas*) porque le hemos dicho que Dirlist: *lista\_carpetas*, lo mismo ocurre al cambiar de directorio, se cambia la lista de archivos (Componente: FileListBox) y en la lista de archivos sólo aparecerán los archivos del tipo que se pongan en la máscara (Mask: \*.bmp;\*.ico;\*.wmf;\*.jpg). Para ver el tipo de imagen jpg es necesario añadir la librería *jpeg* a la lista de USES. La parte más difícil de código está al hacer clic en un archivo de la lista *lista\_archivos*:

```
procedure TForm1. lista_archivosClick(Sender: TObject);
var
  Extension: string[4];           {creamos la variable Extensión que será de tres 4 caracteres: letras más el punto}
begin
  Extension := UpperCase(ExtractFileExt(lista_archivos.FileName)); {detecta la extensión del archivo en mayúsc}
  if (Extension = '.BMP') or (Extension = '.JPG') or (Extension = '.GIF') or (Extension = '.WMF') then
  begin
    Image1.Picture.LoadFromFile(lista_archivos.FileName);           {Carga el archivo en la imagen}
    Label1.Caption := ExtractFilename(lista_archivos.FileName);    {Cambia el texto de la etiqueta}
  end;
end;
```

Código al pulsar el botón de Salir: (SalirClick): **close;**

Código al pulsar en ajustar imagen (StretchCheck): **Image1.Stretch := Ajustar.Checked;**

### Paso 3. Ejecutar y comprobar.

- Pulsa en el botón **Run** o en el menú Run (*Ejecutar*). Comprueba y ya está.
- Guarda el proyecto en tu carpeta con el nombre *Visor.dproj*

## Mejoras al visor de imágenes - Uso de las librerías Jedi - JVCL

Si tienes instaladas las librerías gratuitas JEDI-VCL (<http://jvcl.sourceforge.net>), sustituye los componentes anteriores por los de la ficha: **JVList, combos, trees**: *TjvFileListBox*, *TjvDriveList*, *TjvDirectoryListBox*, más potentes.

En *JVFileListBox*: Activa Filetype: ftHidden (para ver imágenes ocultas) y Showglyphs a true (Ver iconos).

*Imagen mejorada*:

Sustituye Image1 por **JvSpecialImage1** (Paleta: JVimages, animators) y añade junto a la casilla de ajustar la de Reflejo (Flip) y un deslizador para el brillo (Brightness) y otros controles de la imagen como en la figura

## Ejercicio 6 . Reproductor multimedia

Componente: TOpenDialog  
Paleta: Dialogs  
Name: Abrirdialogo  
Filter: AVI|\*.avi|WAV|\*.wav|MIDI|\*.mid

Componente: Mainmenu  
Paleta: Standard  
Name: MainMenu1

Componente: TMediaPlayer  
Name: Player Carpeta: System  
Display: VideoPanel  
AutoOpen: False

Componente: Panel  
Name: Videopanel

Formulario  
Name: Form1  
Caption: Reproductor multimedia

Evento: Abrir1click

```

Procedure Form1.Abrir1Click(Sender: TObject);
begin
with Abrirdialogo do
  if Execute and (FileName <> "") then
  begin
    Player.Close;
    Player.FileName := FileName;
    Player.Open;
  end;
end;
end;

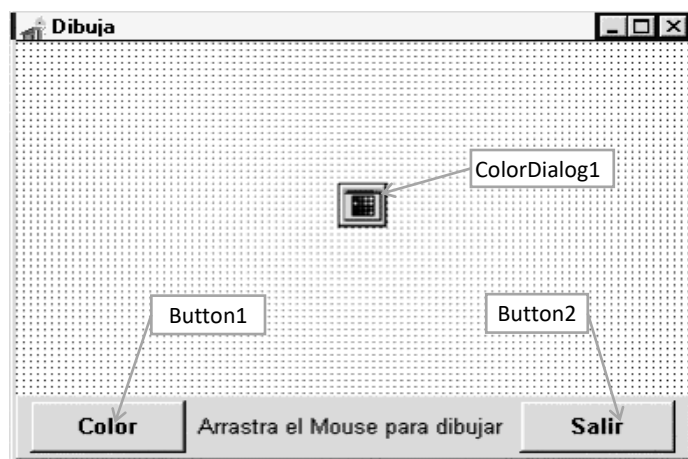
```

Librería añadida en la main: Mplayer  
**uses**  
SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,  
Controls, Forms, Dialogs, ExtCtrls, Menus, Mplayer;

**Ejercicio propuesto 1:** Añade al proyecto *Visor de imágenes*, del ejercicio anterior, la posibilidad de ver archivos de vídeo y música además de imágenes; añadiendo el componente mediaplayer. 🚀

**Ejercicio propuesto 2:** Si tienes instaladas las librerías gratuitas JVCL. Puedes crear un capturador de vídeo fácilmente con el componente: JVAVICapture1 (Paleta: JVimages, animators).

## Ejercicio 7 Dibujar: Controles de los botones del mouse. Uso de una variable "flag"



En un **Nuevo** proyecto/aplicaci3n VCL Delphi Creamos una variable tipo l3gica (booleana) y p3blica llamada:

*Pinta*  
(para uso como *flag* o permiso)

```

public
var
  Form1: TForm1;
  Pinta: Boolean;

```

### procedure TForm1.FormMouseDown

```

begin
  Pinta := True; // Permiso para pintar al pulsar
  Canvas.MoveTo(X, Y); //Fija el origen
end;

```

### procedure TForm1.FormMouseUp

```

begin
  Pinta := False; //Quita permiso para pintar al liberar el bot3n del mouse
end;

```

### procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);

```

begin
  If Pinta=True then Canvas.LineTo(X, Y); // Si la variable Pinta es True entonces traza linea desde el CP al puntoXY
end;

```

### procedure TForm1.Button1Click

```

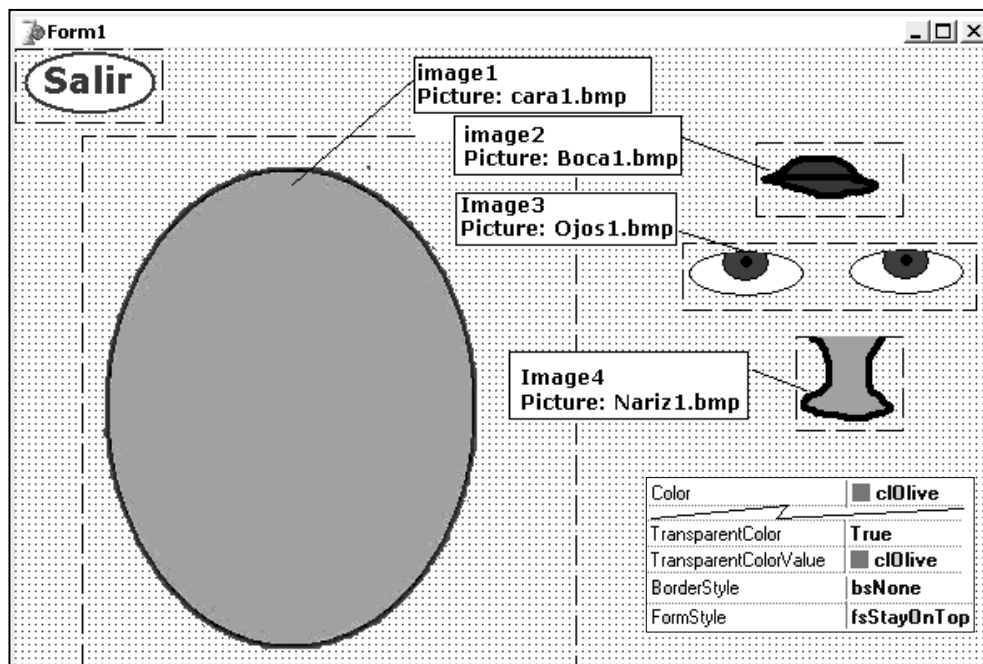
begin
  If Colordialog1.execute then Form1.Canvas.Pen.Color := colordialog1.color; //Abre ventana color y asigna
End

```

- Comprueba ▶ y guarda el Proyecto con el nombre Pintar.dproj

## Ejercicio Arrastrar y soltar. Propiedades de los formularios y del mouse.

1º. En un **Nuevo** proyecto/aplicación VCL Delphi: **CARAS**. Añadir al formulario los ingredientes que se muestran:



### Poner el formulario transparente:

1. En las propiedades del formulario, poner un color que no vayas a usar, como por ejemplo, el oliva.
2. Cambiar las propiedades del formulario: **TransparentColor=True** y **TransparentColorValue=el oliva**. Además, quitar el borde de la ventana (**BorderStyle=bsNone**) y poner siempre encima: (**StayOnTop**).
3. Añadir otra imagen o botón para salir cuyo código sea: Close;
4. Compila y corre el programa (F9) observarás que sólo se muestran las imágenes sobre el escritorio.

### Para arrastrar las imágenes sobre la cara:

- En los eventos del *mouse* al apretar y soltar, los parámetros *X* y *Y* nos informan de la posición del cursor.
- Al mover el mouse, deberemos detectar si el botón izquierdo del mouse está apretado o no. Para ello crearemos una variable de permiso (flag) del tipo booleana, llamada: *Apretado*

1. Creamos las variables públicas:

```
var
  Form1: TForm1;
  Xini,Yini :integer;    // Guardará la posición inicial del ratón al apretar
  Apretado: boolean;    // Variable de permiso (flag) para arrastrar o mover la imagen
```

2. Seleccionamos la imagen 2 y en sus eventos, escribiremos el código al apretar, al soltar y al mover el mouse:

```
Procedure TForm1.Image2MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
begin
  if Button = mbLeft then
  begin
    Xini := X;           //guarda posición X del raton
    Yini := Y;           //guarda posición Y del raton
    Apretado:= true;    //da permiso para arrastrar
  end
end;
```

```
procedure TForm1.Image2MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
begin
  Apretado:= false;    //quita el permiso para arrastrar
  Screen.Cursor := crdefault;
end;
```

```

procedure TForm1.Image2MouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  Screen.Cursor := CrHandPoint;
  if Apretado = true then
    begin
      if sender=image1 then
        begin
          image1.Top := image1.Top + Y - Yini; //mueve pos esquina imagen menos pos del rat3n
          image1.Left := image1.Left + X - Xini
        end
      else if sender=image2 then
        begin
          image2.Top := image2.Top + Y - Yini; //mueve pos esquina imagen menos pos del rat3n
          image2.Left := image2.Left + X - Xini
        end
      end;
    end;
end;

```

Asigna a cada imágen los mismos 3 eventos o podemos aprovechar los mismos procedures, para las otras imágenes, aprovechando el parámetro: **Sender**

### Delphi 7: Juego Antiestress: Cargar archivo de recursos (res). Formulario semi-transparente

Busca en la carpeta **delphi7\Demos\Swat** de el archivo de recursos: **extrares.res**. Ábrelo con el editor de imágenes de delphi: Tools ► Image Editor. Observa los iconos e imágenes cargadas. (puedes cambiar la imagen Live por una rotura de cristal)

Crea un Proyecto nuevo en una carpeta nueva, llamado Stress.dpr.

En el formulario, únicamente pondremos una imágen o botón para salir. Pero utilizaremos la propiedad **AlphaBlend** a **100** para variar la transparencia del formulario.

```

var
  Form1: TForm1;
  Live : TBitmap;

implementation

{$R *.DFM}
{$R extrares.res} // <- aquí se incluye el recurso

```

```

procedure TForm1.FormCreate(Sender: TObject);

```

```

begin
  Screen.Cursors[5] := LoadCursor(HInstance, 'Malet');
  Screen.Cursors[6] := LoadCursor(HInstance, 'MaletDown');
  Screen.Cursor := TCursor(5);
  Live := TBitmap.Create;
  Live.LoadFromResourceName(HInstance, 'Live');
  Live.Transparent:=true;
end;

```

```

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

```

```

begin
  Screen.Cursor := 6; // Al pulsar el mouse aparece el cursor 6 (martillo bajado)
  Canvas.Draw(x-50, y-35, Live); //Al pulsar el mouse Dibuja la imagen Live en los pixels de canvas
end;

```

```

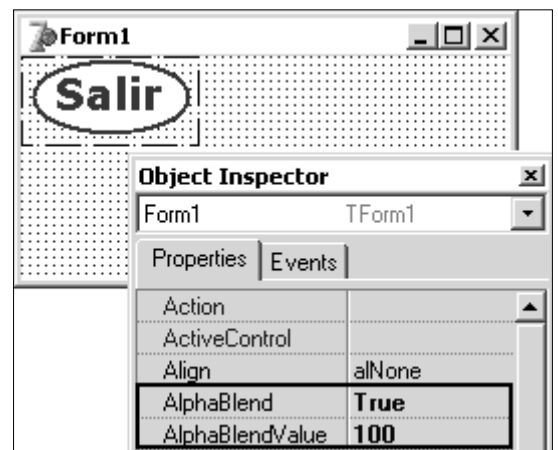
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);

```

```

begin
  Screen.Cursor := 5;
end

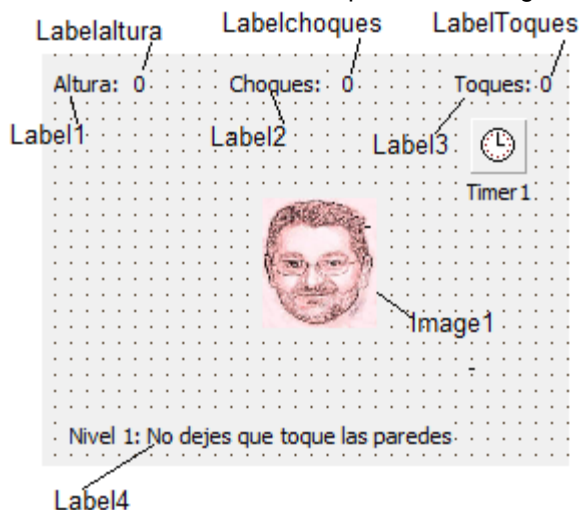
```



//carga el cursor nº 5 del recurso Malet  
 //carga el cursor nº 6 del recurso MaletDown  
 // Al pulsar el mouse aparece el cursor 5 (martillo)  
 // Crea una imágen imagen Bitmap: Live  
 // Carga la imagen Bitmap Live del recurso: Live

## Ejercicio Else If y variables. Mover un objeto y controlar su posición.

**Objetivo:** Se trata de la estructura base para el movimiento de un objeto en un juego evaluando con un IF el estado de la variable *subir* que como es lógica (booleana) puede ser verdadera (true) o falsa (false).



### Ingredientes:

- 7 **Labels** de la paleta *Estandar*
- 1 **Image** de la paleta *Adicional*
- 1 **Timer** (temporizador) de la paleta *System*

### Preparación:

En sus propiedades (Properties), los objetos tendrán el nombre (Name) que se indica en la imagen de la izquierda. Cambia las propiedades (Properties) de los elementos:

**Label 1:** Cambiar propiedad *Caption:* Altura

**Label 2:** Cambiar propiedad *Caption:* Choques

**Label 3:** Cambiar propiedad *Caption:* Toques

**Timer1:** Poner su estado *Enabled:* False y su *Interval:* 1

**Labelaltura, Labeltoques y Labelchoques:** poner caption a cero.

**Image1:** *Picture:* Cargar el dibujo de una pelota. *Stretch:* True

### Cocción:

Pulsar doble clic sobre el fondo de la ventana formulario y escribir dentro del procedure el texto en negrita del recuadro derecho → Esto sirve para inicializar las variables y ponerlas a cero al crearse la ventana.

Estas variables anteriores deben estar declaradas al principio como públicas, porque se utilizarán en varios procedimientos. Subir a la sección **Var** del código y añadir el texto que se muestra en negrita cursiva del cuadro →

Volver al formulario pulsando **F12** y luego pulsar doble clic sobre la **image1** y escribir dentro lo que se muestra en el recuadro derecho →

Cada vez que se da un clic en la imagen, la pone en movimiento, cuenta un toque y cambia el sentido de subida.

Pulsar doble clic sobre el **Timer1** y escribir dentro del procedure lo que se muestra en cursiva a la derecha de cuadro derecho →

Según el estado de la variable *subir*, sumará o restará la posición vertical de la imagen y esto ocurrirá cada décima de segundo.

Si toca arriba o abajo, se parará el timer y contará un choque

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  subir:=false; //ponemos en falso la variable subir.
  choques:=0; //ponemos a cero los contadores
  Toques:=0;
end;
```

```
var
  Form1: TForm1;
  subir:Boolean; //variable para controlar el sentido
  choques:Integer; //variable para contar los choques
  Toques:Integer; //variable para contar los toques
```

implementation

```
procedure TForm1.image1Click(Sender: TObject);
begin
  Timer1.Enabled:=True; //activamos el timer
  if subir=true then subir:=false else subir:=True;
  toques:=toques+1; //sumamos un toque
  LabelToques.caption:=inttostr(toques);
end;
```

```
Procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if subir = True then image1.Top:=image1.top-1
  else image1.Top:=image1.top+1;
  labelaltura.caption:=inttostr(Form1.clientHeight-
image1.top);
  if (image1.top<=0)
  or (image1.top>=Form1.clientHeight-image1.height)
  then
  begin
    choques:=choques+1;
    Labelchoques.caption:=inttostr(choques);
    Timer1.Enabled:=False;
  end;
end;
```



Comprueba pulsando **F9** o el botón **Run** su funcionamiento. Si se bloquea escoge del menú: Program – Reset.

Escoge del menú: **File – Save all** para guardar el proyecto y su unidad. Escoge tu carpeta de trabajo.

Guarda la unidad con el nombre: **toques1.pas** y luego guarda el proyecto con el nombre: **Toques.pdr**

## **Animaci3n. Protector de pantalla (Uso del estamento CASE)**

### **Ingredientes:**

- Shape  (Paleta Additional): Name: Bola; Shape: stCircle
  - Timer  (paleta system): Timer1 – Interval: 20
  - Botones: 2 (Empezar y parar)
- Formulario: Si lo deseas, puedes asignar color transparente.

### **Variables:**

Mover: integer;

### **Acciones:**

En el Button1: Empezar : Timer1.enabled=true ;

En el Button2: Parar: Timer1.enabled=false;



### **procedure TForm1.FormCreate(Sender: TObject);** //Al crearse el formulario

begin

mover:=1; // -> importante: La variable: Mover, debe tener cualquier valor

end;

### **procedure TForm1.Timer1Timer(Sender: TObject);** //En el timer1

begin

case Mover of

**1:** // Mueve la bola a la izquierda y hacia arriba

begin

Bola.left:=Bola.Left - 20;

Bola.Top:=Bola.Top - 20;

// Si el gráfico alcanza el borde izquierdo del formulario, se mueve a la derecha y hacia arriba.

If Bola.Left <= 0 Then Mover := 2

// Si el gráfico alcanza el borde superior del formulario, se mueve a la izquierda y hacia abajo.

Else If Bola.Top <= 0 Then Mover := 4;

End;

**2:** // Mueve la bola la derecha y hacia arriba

begin

Bola.left:=Bola.Left + 20;

Bola.Top:=Bola.Top - 20;

{Si el gráfico alcanza borde derecho del formulario, se mueve a la izquierda y hacia arriba.

Se determina el borde derecho del formulario restando el ancho del gráfico del ancho del formulario.}

If Bola.Left >= (Form1.Width - Bola.Width) Then Mover := 1

// Si el gráfico alcanza el borde superior del formulario, se mueve a la derecha y hacia abajo.

Else If Bola.Top <= 0 Then Mover := 3;

end;

**3:** // Mueve la bola a la derecha y hacia abajo

begin

Bola.left:=Bola.Left + 20;

Bola.Top:=Bola.Top + 20;

// Si el gráfico alcanza el borde derecho del formulario, se mueve a la izquierda y hacia abajo.

If Bola.Left >= (Form1.Width - Bola.Width) Then Mover := 4

// Si el gráfico alcanza el borde inferior del formulario, se mueve a la derecha y hacia arriba. Se

determina el borde inferior del formulario restando la altura del gráfico de la altura del formulario menos 680 twips debido a la altura de la barra de título la barra de menús.

Else If Bola.Top >= (Form1.Height - Bola.Height) Then Mover := 2;

End;

**4:** // Mueve la bola a la izquierda y hacia abajo

begin

Bola.left:=Bola.Left - 20;

Bola.Top:=Bola.Top + 20;

// Si el gráfico alcanza el borde izquierdo del formulario, se mueve a la derecha y hacia abajo.

If Bola.Left <= 0 Then Mover := 3

// Si el gráfico alcanza el borde inferior del formulario, se mueve a la izquierda y hacia arriba.

Else If Bola.Top >= (Form1.Height - Bola.Height) Then Mover := 1;

End;

### **Case:**

Bifurca las acciones dependiendo de diversos resultados de un parámetro del tipo ordinal

### **Variantes y mejoras:**

Este protector de pantalla puede convertirse en un juego de habilidad (mezclado con el anterior) si al pulsar doble clic sobre el objeto (que puede ser una imagen) cuenta los golpes y a su vez se acelera el objeto.

## Animación. (Juego Ping) Avanzado Firemonkey

```
var
  Form1: TForm1;
  p1x, p1y, p2x, p2y, p3x, p3y: real; //posic. de los
recuadros
  b: byte;
  modowin:boolean;
  mx,my:integer; //posic. del mouse

procedure TForm1.ButtonPausaClick(Sender: TObject);
begin
  Timer1.Enabled := False ;
  Timer2.Enabled := False ;
  Timer3.Enabled := False ;
  Timer4.Enabled := False ;
  buttonSalir.Visible := True ;
  buttonIniciar.Visible := True ;
end;

procedure TForm1.ButtonIniciarClick(Sender: TObject);
begin
  Timer1.Enabled := True ;
  Timer2.Enabled := True ;
  Timer3.Enabled := True ;
  Timer4.Enabled := True ;
  buttonSalir.Visible := False ;
  buttonIniciar.Visible := False ;
{$IFDEF MSWINDOWS}
modowin:=true; //si es windows
{$ELSE}
modowin:=false; //si no es windows
{$ENDIF}
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Enabled:=False;
  Timer2.Enabled:=False;
  Timer3.Enabled:=False;
  Timer4.Enabled:=False;
  p1x := form1.Width / 2; //situa al centro inferior
  p1y := form1.Height-20;
end;

procedure TForm1.FormMouseMove
begin
  mx:=round(x);
  my:=round(y);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  rectangle1.Position.X := p1x; //mueve los objetos
  rectangle1.Position.Y := p1Y;
  rectangle2.Position.X := p2X;
  rectangle2.Position.Y := p2Y;
  rectangle3.Position.X := p3X;
  rectangle3.Position.Y := p3Y;
  Application.ProcessMessages; //procesa pero espera a que se
acabe la tarea anterior
end;

procedure TForm1.Timer2Timer(Sender: TObject);
var
  a: integer;
begin
  if p1y < 0 then
  begin
    b := 0; //modo bajar
    p1y := p1y + 15;
  end;
  if ((trunc(p1y) < trunc(p3y) - rectangle1.Height)) and
((trunc(p1y) > trunc(p3y) - rectangle1.Height - 5))and (p1x >
p3x - rectangle1.Width / 1.3) and (p1x < p3x +
(rectangle3.Width - rectangle1.Width) + rectangle1.Width /
1.3) then
  begin
    if b = 0 then //si hay colisión con primera pala
bajando...puntu
begin
label4.Text := inttostr(strtoint(label4.text) + 1); end;
    b := 1;
  end;

  if ((trunc(p1y) < trunc(p2y) - rectangle1.Height)) and
((trunc(p1y) > trunc(p2y) - rectangle1.Height - 5))and (p1x >
p2x - rectangle1.Width / 1.3) and (p1x < p2x +
(rectangle2.Width - rectangle1.Width) + rectangle1.Width /
1.3) then
  begin
    if b = 0 then //si hay colisión con segunda pala
bajando...puntu
begin
label4.Text := inttostr(strtoint(label4.text) + 1);
end;
    b := 1;
  end;

  if p1y > form1.Height then //toca fondo pierde
begin
  b := 1; //pone en modo subir
  p1y := form1.Height - 63;
  p1x := form1.Width / 2;
  if strtoint(label4.Text) > strtoint(label2.Text) then
label2.Text := label4.Text; //fija la máxima puntuación y pone
puntos a cero.
  label4.Text := '0';
end;

  if b = 1 then //si esta en modo subir
begin
  p1y := p1y - 5; //resta posición vertical
  if modowin=false then //si esta jugando en android
controla con el sensor
begin
  if (rectangle1.Position.X + 20 *
motionsensor1.Sensor.AccelerationY < form1.Width) and
(rectangle1.Position.X + 20 *
motionsensor1.Sensor.AccelerationY > 0) then
  p1x := rectangle1.Position.X + 20 *
motionsensor1.Sensor.AccelerationY;
  end
  else //si esta jugando en windows
begin
  if (mx - p1x) > Rectangle1.Width then p1x :=
rectangle1.Position.X + 5 //mouse a la derecha
  else if (mx - p1x) < 5 then p1x :=
rectangle1.Position.X - 5; ///mouse a la izda
  end;
end;

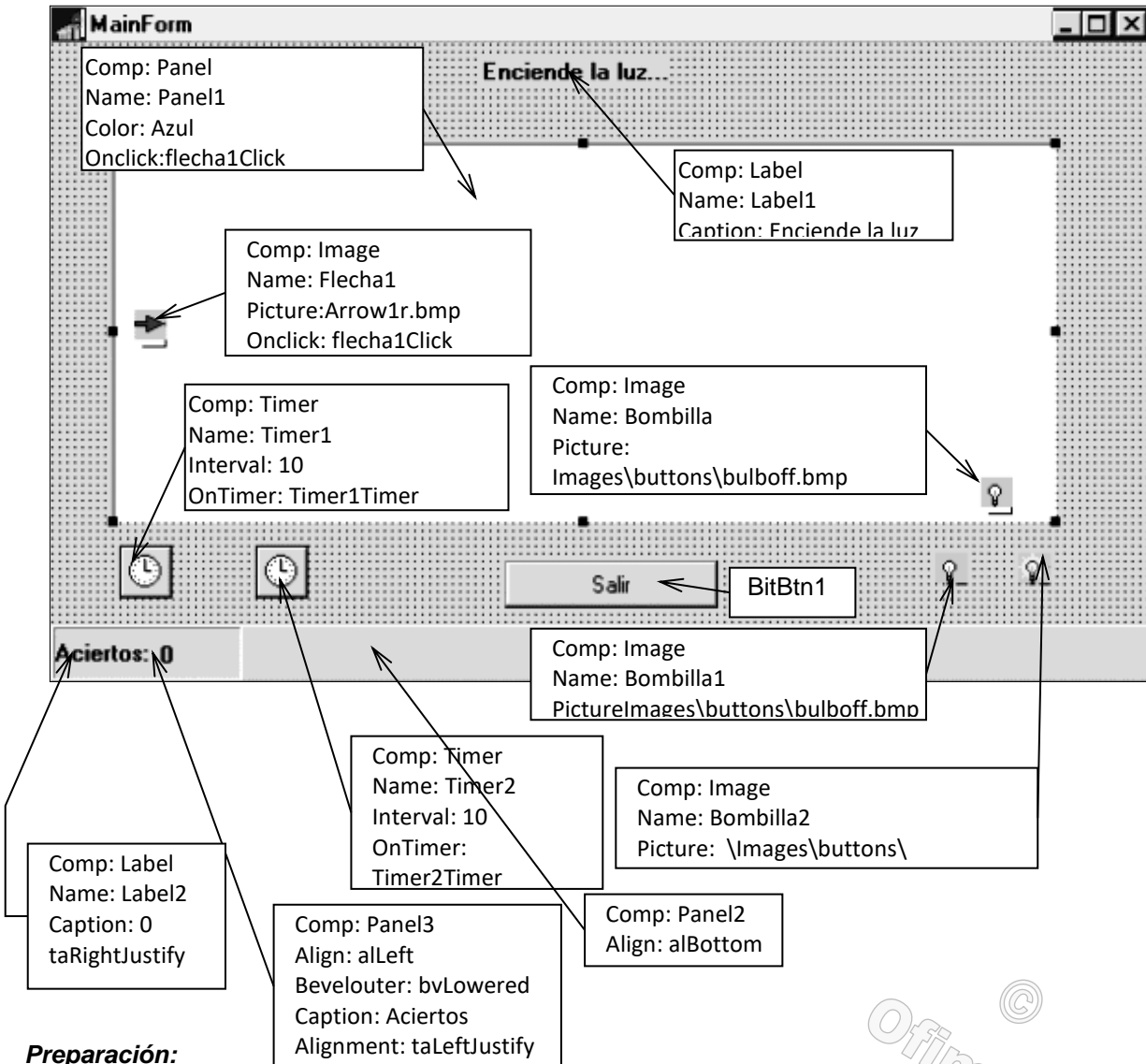
  if b = 0 then //si esta en modo bajar
begin
  p1y := p1y + 5; //suma posición vertical
  if modowin=false then
begin
  if (rectangle1.Position.X + 20 *
motionsensor1.Sensor.AccelerationY < form1.Width) and
(rectangle1.Position.X + 20 *
motionsensor1.Sensor.AccelerationY > 0) then
  p1x := rectangle1.Position.X + 20 *
motionsensor1.Sensor.AccelerationY;
  end
  else //si esta jugando en windows
begin
  if (mx - p1x) > Rectangle1.Width then p1x :=
rectangle1.Position.X + 5 //mouse a la derecha
  else if (mx - p1x) < 5 then p1x :=
rectangle1.Position.X - 5; ///mouse a la izda
  end;
end;
end;

procedure TForm1.Timer3Timer(Sender: TObject); //mueve la
pala 1 aleatoriamente
begin
  p2y := form1.Height / 1.3;
  p2x := rectangle2.Position.X - 10;
  if rectangle2.Position.X < -150 then
  begin
    p2x := form1.Width;
    timer3.Interval := Random(150)+10;
  end;
end;

procedure TForm1.Timer4Timer(Sender: TObject); //mueve la
pala 2 aleatoriamente
begin
  p3y := form1.Height / 1.3;
  p3x := rectangle3.Position.X + 10;
  if rectangle3.Position.X > form1.Width + 150 then
  begin
    p3x := - 150;
    timer4.Interval := Random(150) + 10;
  end;
end;
end;
```

## Delphi. Ejercicio 5 . Movimientos y contadores. Juego

En la figura se muestran los componentes y sus propiedades principales.



### Preparaci3n:

var

MainForm: TMainForm;

subir:integer; // => Variable p3blica para controlar el sentido subir ser3 num3rica entera

### Acciones:

**El timer1 se encarga de desplazar la flecha hacia la derecha y comprueba si su posici3n est3 cerca de la bombilla:**

```

procedure TMainForm.Timer1Timer(Sender: TObject);           {solo funciona si el timer1 est3 activado}
begin
if (flecha1.left=460{bombilla.left})and (flecha1.top<bombilla.top+10) and (flecha1.top>bombilla.top-10)
then
begin
bombilla.picture:=bombilla2.picture; {si flecha toca bombilla cambia picture a bombilla2}
puntos.caption:=inttostr(strtoint(puntos.caption)+1); {suma un punto y convierte en cadena}
end;
if flecha1.left<500 then
flecha1.left:=flecha1.left+5           {avanza de izquierda a derecha 5 pixels a cada pulso del timer}
else
begin
timer1.enabled:=false;                 {descativa el temporizador timer1}
flecha1.left:=10;                       {si la flecha pasa de largo empieza de nuevo en posic.
izquierda a 10}
end
end;

```

## El timer2 se encarga de desplazar arriba y abajo la bombilla, (según si la variable subir esté a 0 o 1):

```
procedure TMainForm.Timer2Timer(Sender: TObject);
begin
  if subir=0 then {si no está activado subir...}
  begin
    if bombilla.top<200 then {si no pasa de 200 de profundidad que siga bajando}
    begin
      bombilla.top:=bombilla.top+1;
      bombilla.picture:=bombilla1.picture; {por si está encendida, que se apage}
    end
  else {si pasa de 200 de fondo...}
  {que se active subir}
  subir:=1
  end
  else {si está activada subir..}
  begin
    if bombilla.top>0 then {y si aún no está arriba...}
    {que siga subiendo}
    bombilla.top:=bombilla.top-1
  else {si ya ha llegado arriba...}
  {que se desactive subir}
  subir:=0
  end
  end;
end;
```

## Al empezar el programa, la variable subir debe de valer algo (cero por ejemplo)

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  subir:=0; {empieza con subir desactivado}
end;
```

## Al hacer clic sobre la flecha se activa el timer (o también se puede hacer sobre un botón)

```
procedure TMainForm.flecha1Click(Sender: TObject);
begin
  timer1.enabled:=true; {al hacer clic con el ratón que se active el timer1}
end;
```

## Variantes y mejoras:

### Conexión a Internet.


1.- Crea un botón que nos conecte con la página web de soporte del programa:

Añade la siguiente línea de código:

```
ShellExecute(GetDesktopWindow(),nil, PChar('www.ofimega.es/oficalc/index.htm'), nil, nil, SW_SHOWNORMAL);
```

2.- Crea un botón que nos abra un archivo web de ayuda:

- Crea el archivo web: '**instrucciones.htm**' (con Microsoft Word o Dreamweaver), explicando el funcionamiento del programa. Este archivo debe guardarse en el mismo directorio o carpeta que el ejecutable (.exe).

- Añade el componente **webbrowser**  de la paleta **Internet**.
- Asignar propiedad visible= false
- Añade las siguientes líneas de código:

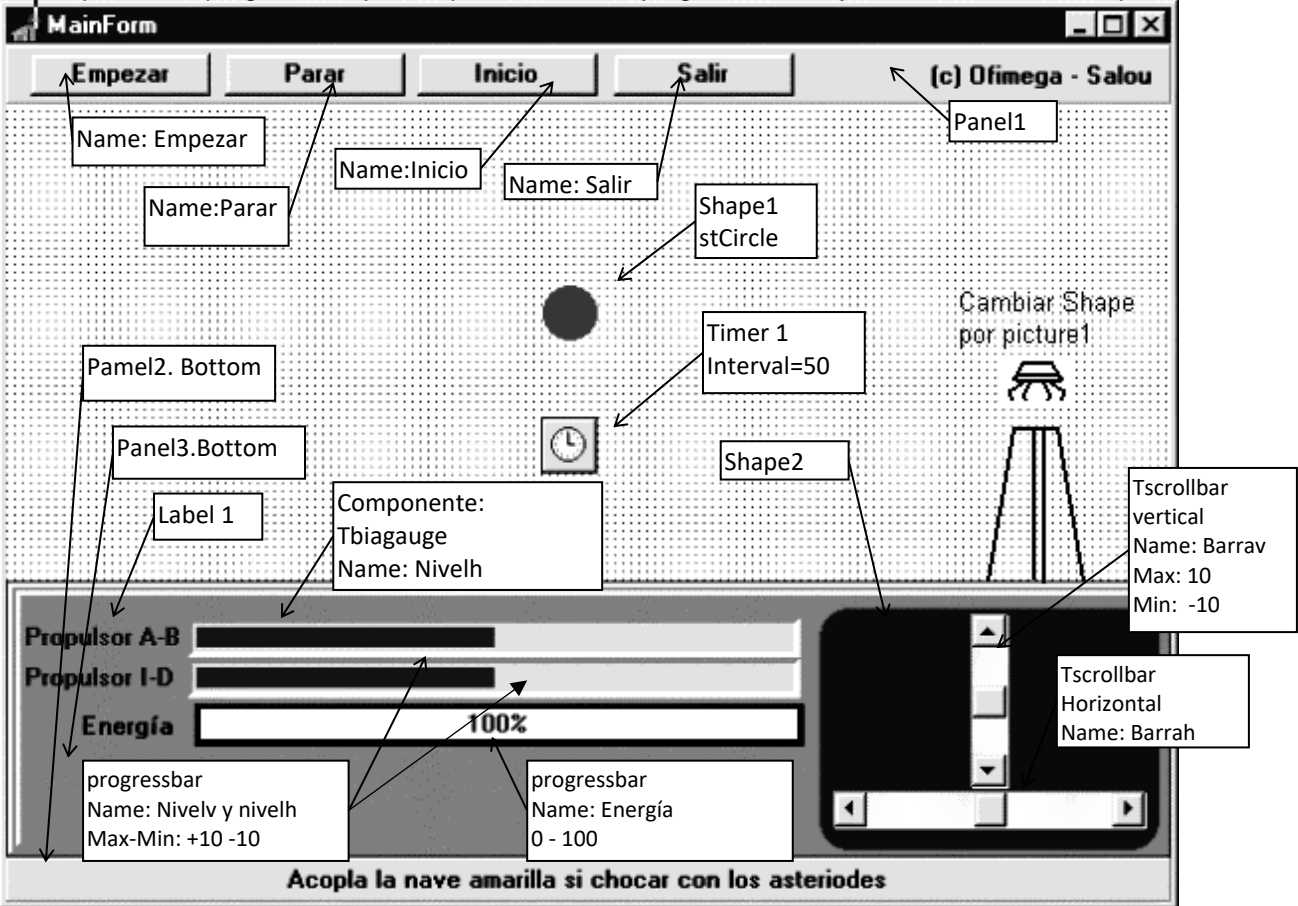
```
var
  dire:string;
begin
  dire:=ExtractFilePath(ParamStr(0)); //Detecta la ruta del programa
  WebBrowser1.visible:=true;
  WebBrowser1.Navigate('file://'+dire+'instrucciones.htm')
end;
```

### Cómo ejecutar un programa externo:

```
var
  dire:string;
begin
  dire:=ExtractFilePath(ParamStr(0));
  ShellExecute(Form1.Handle,nil,PChar(dire+'programa.exe'),'','',SW_SHOWNORMAL)
end;
```

### Ejercicio 7. Control de direcci3n y velocidad de un objeto

Completa este programa *esqueleto* para dise1ar un juego de aterrizaje lunar de una nave espacial:



#### Procedures:

**Procedures de los botones:**

```

EmpezarClick
timer1.enabled:=True;
PararClick
timer1.enabled:=False;
InicioClick
shape1.top:=124;
shape1.left:=184;
SalirClick
Close;
    
```

**Procedure de las barras de desplazamiento:**

```

barravChange
nivelv.value:=barrav.position;
energia.progress:=energia.progress-1;
barrahChange
nivelh.value:=barrah.position;
    
```

**Procedure del timer**

```

Timer1Timer
Shape1.Left := Shape1.Left + barrah.position;
Shape1.Top := Shape1.top + barrav.position;
    
```

#### Variantes:

- El fondo de la ventana ser1 de color negro y se a1adir1n *shapes* de estrellas blancas
- El objeto ser1 el dibujo de una nave en un picture box. (En el programa *esqueleto* el objeto es un shape de forma circular.)
- Se a1adir1n asteriodes que resten puntos y/o desv1en la nave de su ruta.
- Finalizar1 el juego al coincidir el objeto con las coordenadas de la base lunar.
- Puedes sustituir las barras de direcci3n por botones *Updown* de la paleta Win32 o sustituirlos por *ArcDial* y *TrackBar* el en *Firemonkey*

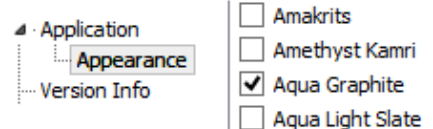
Variante para que no se salga la nave de los l1mites de la ventana a incluir en el procedimiento del timer:

```

If Shape1.Left<0 then Shape1.Left:=299;
If Shape1.Left>300 then Shape1.Left:=1;           ->Idem para control arriba y abajo con la propiedad top
    
```

#### Estilos para XE Rad Studio:

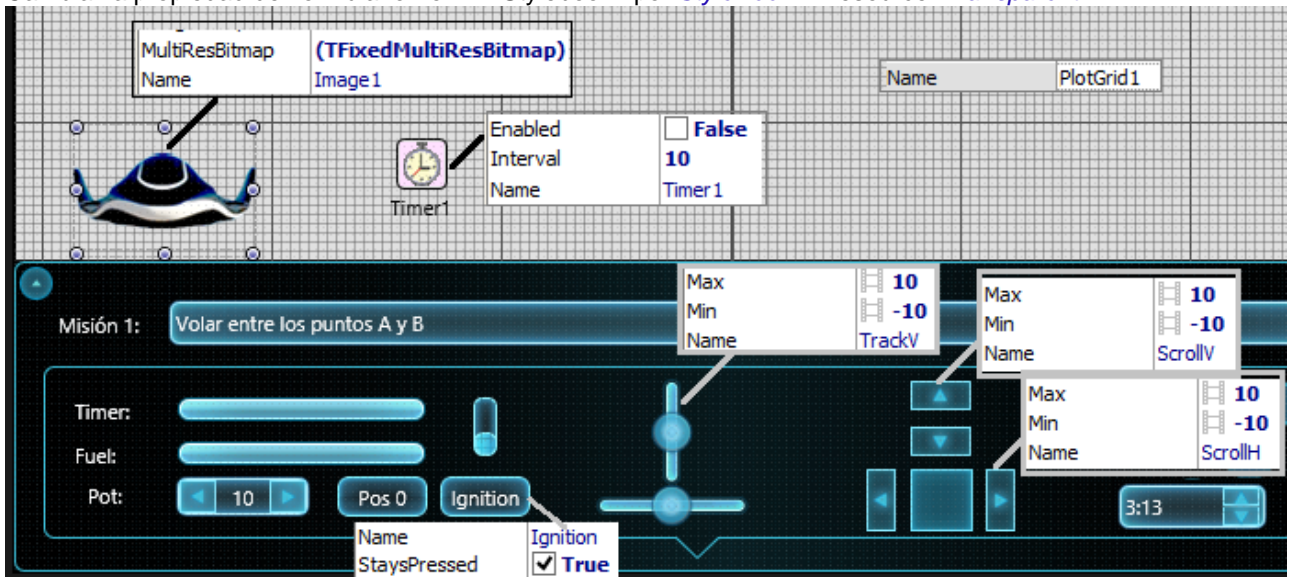
Si usas XE: Escoge del men1: Project – Options: Application – Appearance y activa un estilo como:  Aqua Graphite



**Guarda** el proyecto con el nombre: Aterriz1.dpr y la unidad Aterriz1.pas en la carpeta: **Aterriz1**

## Variante Multidevice FireMonkey: Simulador Dron

- Crea nuevo proyecto. *File* ▶ *New* ▶ *Multidevice application - Delphi*
- Poner los componentes de la imagen:
- Cambia la propiedad del formulario Form1. Stylebook: por *StyleBook1*. Resource: *Transparent*



```

procedure TForm4.ScrollVChange(Sender: TObject);
begin
  if sender= ScrollV then
    TrackV.Value:=ScrollV.Value
  else
    TrackH.Value:=ScrollH.Value;
    ProgressF.Value:=ProgressF.Value-1;
end;

```

```

procedure TForm4.Timer1Timer(Sender: TObject);
begin
  Nave1.Position.X:=Nave1.Position.X+ScrollH.Value;
  Nave1.Position.Y:=Nave1.Position.Y+ScrollV.Value;
  ProgressT.value:=ProgressT.value-1; //--> consumo
  de tiempo
  Nave1.rotationangle:=ScrollH.Value*3; //-->
  inclinación de nave
  ComprobarBordes(); //comprueba si se sale
  //ComprobarAterrizaje(); //comprueba si aterriza
  bien
end;

```

```

procedure TForm4.IgnitionClick(Sender: TObject);
begin
  Timer1.Enabled:=Ignition.IsPressed;
  ScrollV.Value:=-1;
end;

```

```

procedure TForm4.DialVChange(Sender: TObject);
begin
  if sender=TrackV then
    ScrollV.Value:=TrackV.Value
  else
    ScrollH.Value:=TrackH.Value;
    ProgressF.Value:=ProgressF.Value-1;
end;

```

```

procedure TForm4.ComprobarBordes();
begin
  if Nave1.Position.X>PlotGrid1.width then
    Nave1.Position.X:=0;
  if Nave1.Position.X<0 then
    Nave1.Position.X:=PlotGrid1.width;
end;

```

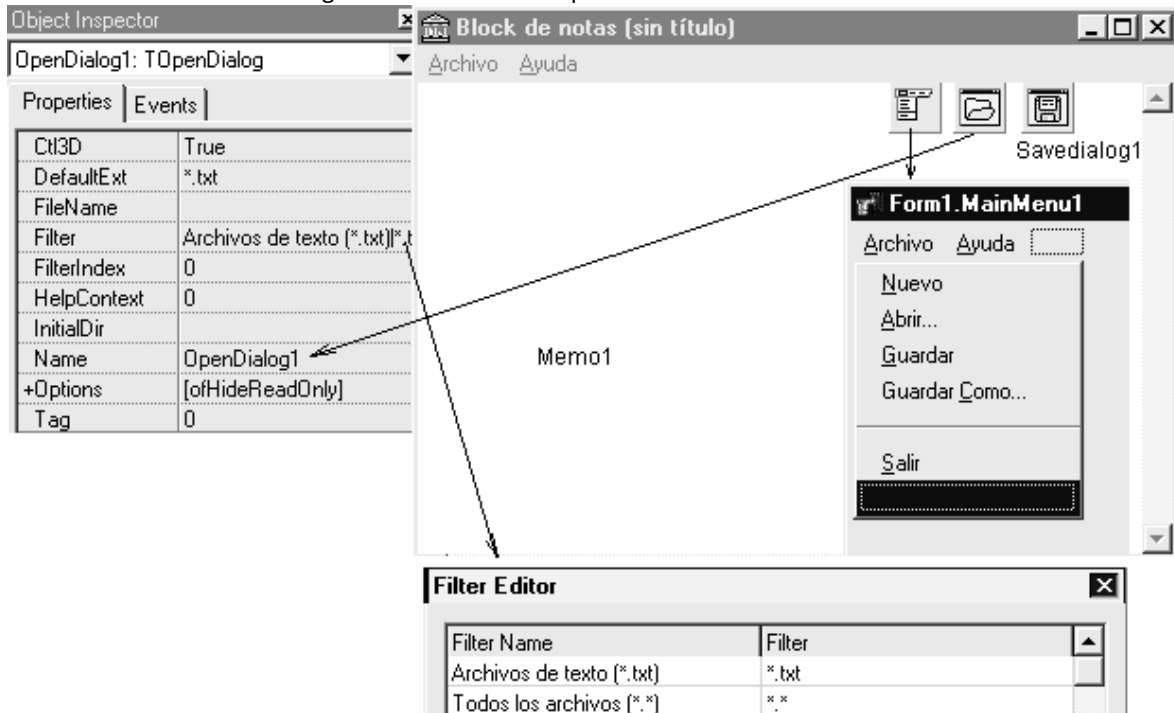
### Notas:

- Hemos creado un procedimiento independiente para comprobar si se sale del borde, por lo tanto hay que declararlo en la lista de declaración de procedimientos: `procedure ComprobarBordes();`
- La acción para el evento *ScrollHChange* es la misma que para *ScrollVChange*
- La acción para el evento *DialHChange* es la misma que para *DialVChange*

**Guarda** el proyecto con el nombre: **Dron.dpr** y la unidad **Dron1.pas** en la carpeta: **Dron**

## **Bloc de notas. Editor de textos.**

Uso de los cuadros de di3logo Abrir - Guardar – imprimir...



```

procedure TForm1.Nuevo1Click(Sender: TObject);
begin
  Memo1.Clear;
  OpenDialog1.FileName := '';
  Caption := 'Text Demo - [Untitled]';
end;

```

```

procedure TForm1.Abrir1Click(Sender: TObject);
begin
  with OpenDialog1 do
    if Execute then
      begin
        Memo1.Lines.LoadFromFile(FileName);
        Caption := 'Text Demo - ' + ExtractFilename(FileName);
      end;
end;

```

```

procedure TForm1.Guardar1Click(Sender: TObject);
begin
  if OpenDialog1.FileName <> '' then
    begin
      Memo1.Lines.SaveToFile(OpenDialog1.FileName);
    end
  else Guardarcomo1Click(Sender);
end;

```

```

procedure TForm1.Guardarcomo1Click(Sender: TObject);
begin
  with SaveDialog1 do
    if Execute then
      begin
        Memo1.Lines.SaveToFile(FileName);
        Caption := 'Text Demo - ' + ExtractFilename(FileName);
        OpenDialog1.FileName := FileName;
      end;
end;

```

```

procedure TForm1.Salir1Click(Sender: TObject);
begin
  Close;
end;

```

```

procedure TForm1.FormCloseQuery(Sender: TObject; var
  CanClose: Boolean);
var
  MsgResult: Word;
begin
  if Memo1.Modified then
    MsgResult := MessageDlg(Format('Archivo %s
      modificado. ¿Guardar?',
      [OpenDialog1.FileName]), mtWarning, mbYesNoCancel,
      0);
    case MsgResult of
      mrYes:
        begin
          Guardarcomo1Click(Sender);
          CanClose := True;
        end;
      mrNo: CanClose := True;
      mrCancel: CanClose := False;
    end;

```

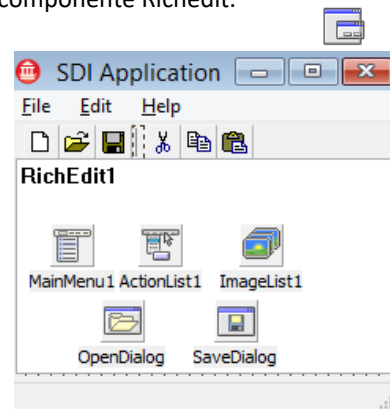
## USO DE COMPONENTES DE TEXTO. RICHEDIT y ActionList

Vamos a generar un pequeño procesador de textos utilizando la plantilla SDI y el componente Richedit.

1. Elije del menú principal: File ► New ► Other...
2. Escoge de *Delphi Projects: SDI Application*. Indicar la carpeta para el proyecto: Crear una nueva carpeta llamada *Editor*. El asistente nos generará un formulario con los componentes básicos. Un menú, una barra de herramientas y una barra de estado. Añade al formulario un objeto Richedit (Paleta Win32) alineado al cliente.

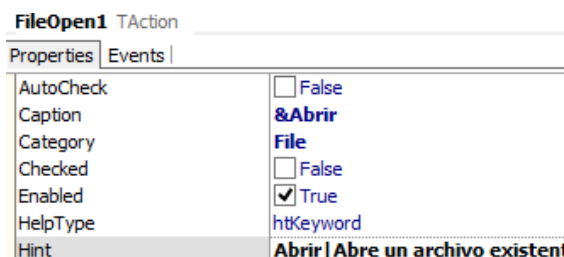
Cambia su propiedad Align: alCliente y su borra el texto de la propiedad Lines

La plantilla nos ha generado un formulario con una barra de herramientas y un menú que contienen llamadas a las mismas acciones, por ejemplo, la acción de Abrir puede ser llamada desde el botón de abrir o desde el menú: Archivo - Abrir. Cada objeto de ambas, hace referencia a una acción que está centralizada en la lista de acciones: ActionList1 y es llamado desde el evento **Action**.



Para completar el evento Abrir, pulsamos doble click sobre **Acciónlist1** y seleccionamos la acción: **FileOpen1**.

En sus propiedades: cambia sus propiedades *Caption* y *Hint* como en la figura.

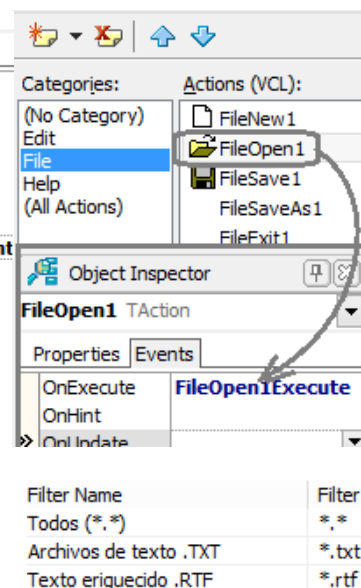


En el evento **OnExecute**, cambia el código: OpenFileDialog->Execute(); por:

```
if OpenFileDialog.Execute() Then
  RichEdit1.Lines.LoadFromFile(OpenDialog.FileName);
```

Del mismo modo selecciona la Acción **FileSave1** y cambia el código por:

```
if SaveDialog.Execute() Then
  RichEdit1.Lines.SaveToFile(SaveDialog.FileName);
```



En ambos objetos, cambiar la propiedad filter, para poder especificar e tipo de documento al abrir o guardar, como en la figura y las propiedades: Filterindex: 3  
DefaultExt: \*.rtf

Otras acciones:

- Acciones **Cortar**, **copiar** y **pegar**: Estas acciones funcionan sin código escrito porque estás asociadas a una combinación del teclado que se designa en la propiedad: **Shortcut**: Ctrl+X, Ctrl+C y Ctrl+V.
- Acción **Nuevo**: FileNew1 - FileNew1Execute: añade el código: `RichEdit1.Lines.Clear();`
- Acción **Salir**: FileExit1 - FileExit1Execute. Comprueba el código: `Close();` ó `Application.Terminate();`
- Acción **Acrescade** (Créditos): HelpAbout1 - HelpAbout1Execute. Comprueba el código: `AboutBox->ShowModal();`

### Problemas y mejoras:

#### Nombre de archivo al guardar:

Al escoger **Guardar** la aplicación **no** nos debería solicitar el nombre si ya se ha guardado o recuperado antes:

- Creamos una variable pública que guardará el nombre del archivo: `nombreambrearchivo: String;`
- Cambiamos el código del procedimiento FileSave1Execute:

```
if Sender=FileSave1 then
begin
  if nombreambrearchivo!=' ' then RichEdit1.Lines.SaveToFile(nombreambrearchivo);
  RichEdit1.Modified := False;
end
else
  if SaveDialog.Execute() then
begin
  RichEdit1.Lines.SaveToFile(SaveDialog.FileName);
  Nombreambrearchivo:=SaveDialog.FileName;
end;
```

- Añadimos el código al Abrir: `nombreambrearchivo:=OpenDialog.FileName;`

**Preguntar al cerrar o nuevo:**

Tanto si creamos un archivo nuevo o salimos del programa, la aplicación nos debería preguntar si deseamos conservar o guardar el documento editado. Para ello utilizamos la propiedad Modified del componente Richedit.

```
FileNew1Execute...
begin
if RichEdit1.Modified=True then
if MessageDlg("¿Guardar documento anterior?",
mtInformation, TMsgDlgButtons() << mbYes << mbNo << mbCancel, 0)= mrYes) then
FileSave1Execute(Sender);
end
RichEdit1.Lines.Clear();
RichEdit1.Modified:=False;
end
```

Al cerrar el formulario ocurre antes el evento **FormCloseQuery** donde se consulta antes de cerrar la variable CanClose:

```
FormCloseQuery...(Sender, CanClose)
{
if (RichEdit1.Modified=True) then
if (MessageDlg("¿Guardar documento?", mtInformation, TMsgDlgButtons() << mbYes << mbNo << mbCancel, 0)= mrYes)
then FileSave1Execute(Sender);
CanClose:=True;
end
```

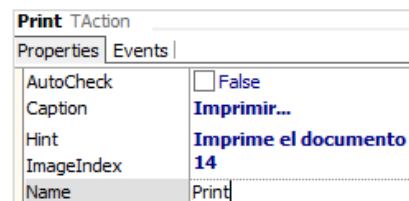
**Imprimir:**

- Añade un objeto **PrintDialog1** de la paleta Dialogs.

Añade una nueva acción a la lista **ActionList1** para imprimir, con las propiedades de la imagen al que le añadiremos la siguiente acción en el evento OnExecute:

```
if PrintDialog1.Execute() then RichEdit1.Print(nombreadarchivo);
```

- Añade un elemento en la sección file del MainMenu y un nuevo botón en la ToolBar1 de modo que su acción **Action** se vincule a: Print.
- Para generar la acción Configurar Impresora, puedes crear una Standard Action del tipo: FilePrintSetup. Luego puedes añadir el elemento en el menú para abrir el cuadro de diálogo: Configurar impresora:



**Ejecuta y prueba (Run ▶).** Si todo es correcto, Guarda todo (File ▶ Save All): Unit: Editor1 - Proyecto: **Editor**

**Procesar textos en Multidevice (Firemonkey)**

Utilizamos el componente Memo1 en modo móvil/Firemonkey para editar e imprimir texto:

Para Imprimir en modo Firemonkey añadir la librería FMX.Printer y utilizar las funciones begindoc y enddoc

Ejercicio para imprimir:

```
procedure TForm2.Impri1Execute(Sender: TObject);
```

```
var
```

```
MyRect: TRectF; //recuadro area de impresión
i:integer;
```

```
begin
```

```
if PrintDialog1.Execute then
```

```
with Printer do
```

```
try //Intenta si puede
```

```
BeginDoc; {iniciamos documento a imprimir
Printer.ActivePrinter.SelectDPI(1200, 1200);
```

```
MyRect := TRectF.Create(0, 0, Pagesetupdialog1.PageWidth, Pagesetupdialog1.PageHeight); //area de impresión
```

```
Printer.Canvas.Font.Size := 12; //tamaño del texto
```

```
//Printer.Canvas.textout() para usar con las VCL o Lazarus
```

```
for i:=1 to memo1.Lines.Count-1 do
```

```
// fills and draws the text in the specified rectangle area of the canvas:
```

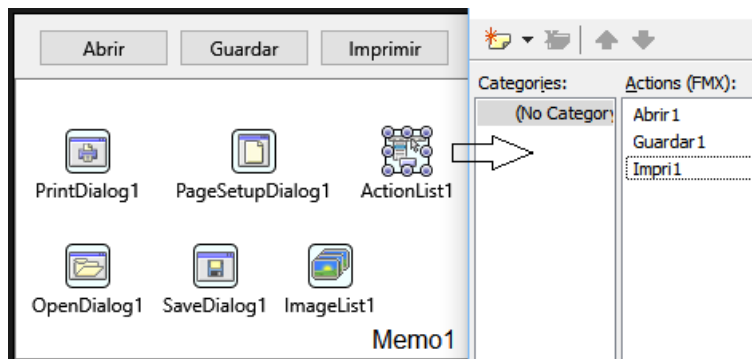
```
Printer.Canvas.FillText(MyRect, Memo1.Lines[i], false, 1, [], TTextAlign.Center, TTextAlign.Center);
```

```
finally
```

```
EndDoc; //Como ya tenemos el lienzo dibujado con nuestro texto, con EndDoc lo enviamos a imprimir
```

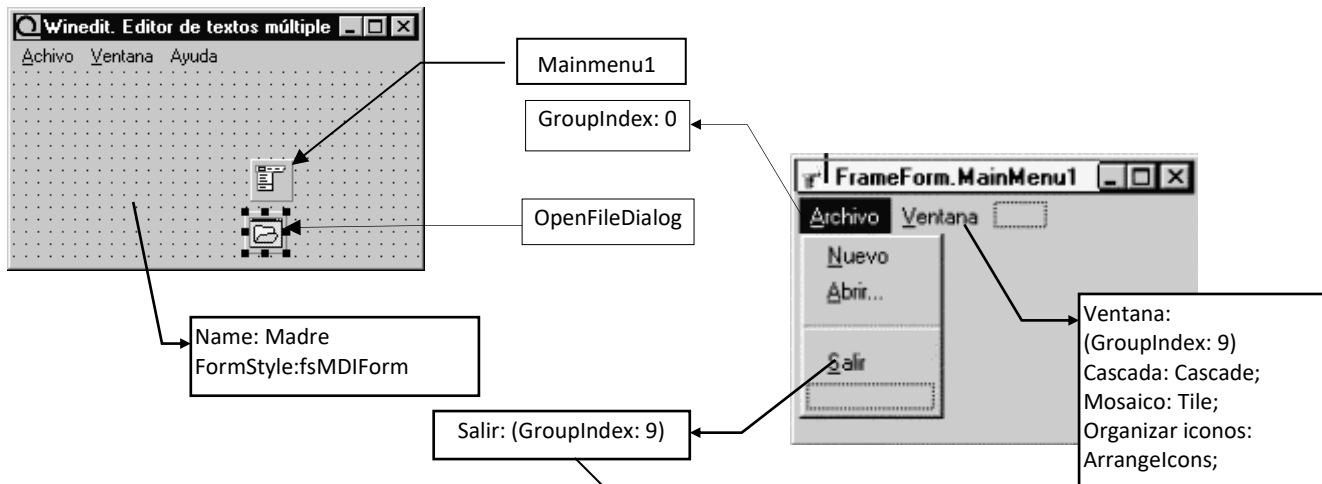
```
end;
```

```
end;
```



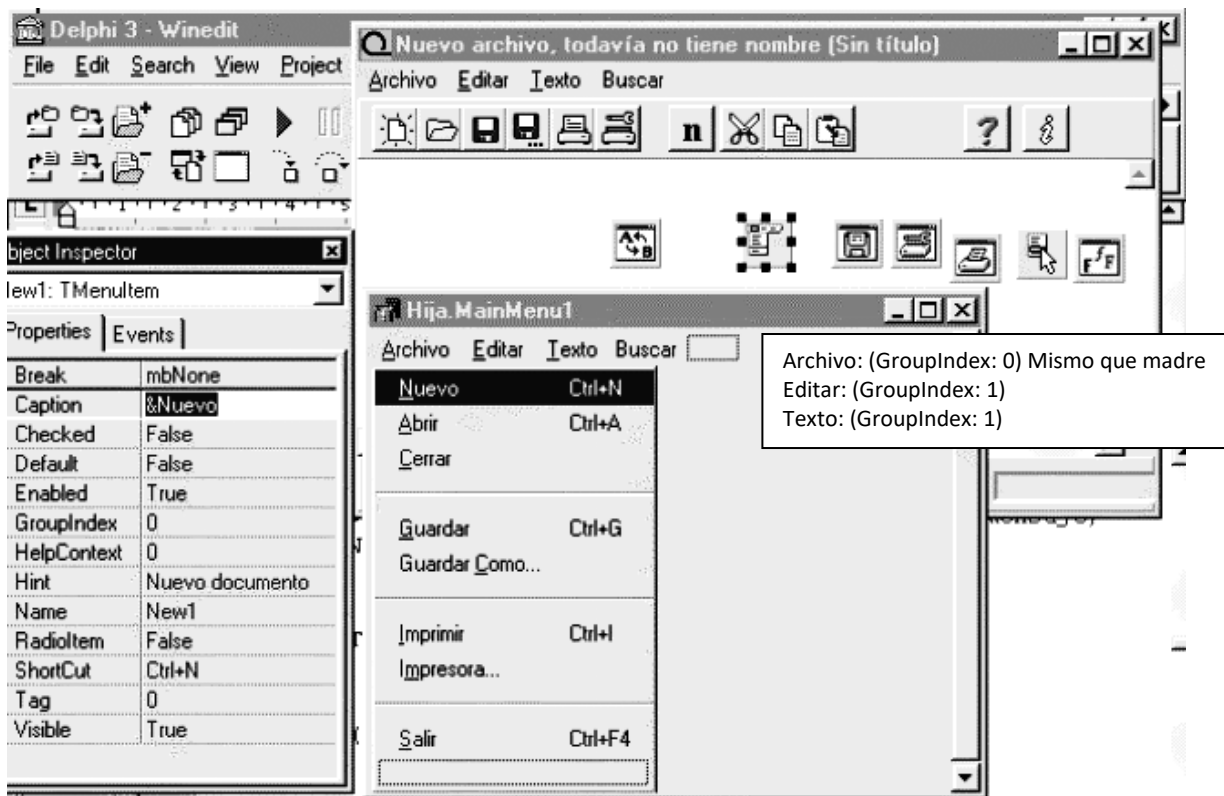
## Ejercicio de aplicación Delphi. Procesador de textos. Ventanas madre e hija

Una ventana Madre (MDI) contendrá varias ventanas hijas (Child). Para ello debemos cambiar la propiedad *FormStyle* a: *fsMDIform* para la ventana *madre* y *fsMDIChild* para la ventana *hija*. Luego, en *Project – Options*, poner la ventana *Hija* en la lista de available. Esta es la ventana *Madre*:



<p><b>Archivo – nuevo:</b>          procedure TMadre.Nuevo          THija.Create(Self);          end;</p>	<p><b>Archivo – salir (madre)</b>          procedure TMadre.Salir1          Close;          end;</p>	<p><b>Procedures ventana:</b>          CascadaClick: Cascade;          MosaicoClick: Tile;          OrganizarClick: ArrangeIcons;</p>
<p><b>Archivo – abrir:</b>          procedure TMadre.Abrir1;          if OpenFileDialog.Execute then          with THija.Create(Self) do          Open(OpenFileDialog.FileName);          end;</p>	<p><b>Archivo – salir (hija) {groupindex:0}</b>          procedure THija.Salir1          MadreSalir1(Sender);          end;</p>	

Aquí aparece la ventana Hija:



Aquí están los procedures de la ventana *Hija*:

```
procedure THija.SetFont
begin
  FontDialog1.Font := Memo1.Font;
  if FontDialog1.Execute then
    Memo1.Font := FontDialog1.Font;
  SetEditRect;
end;
```

```
procedure THija.Open(const AFilename:
string);
begin
  Filename := AFilename;
  Memo1.Lines.LoadFromFile(Filename);
  Memo1.SelStart := 0;
  Caption := ExtractFileName(Filename);
  Position := poScreenCenter;
  Memo1.Modified := False;
end;
```

```
procedure THija.Abrir1Click
begin
  madre.Abri1(Sender);
end;

procedure Hija.Salir1Click;
begin
  madre.Salir1Click(Sender);
end;
```

```
procedure THija.Imprimir1Click;
var
  if PrintDialog.Execute then
    memo1.Print(Filename);
end;

procedure THija.Impresora1Click(Sender: TObject);
begin
  PrinterSetupDialog1.Execute;
end;

Procedure Guardar
  if (Filename = '') or IsReadOnly(Filename) then
    GuardarcomoClick(Sender)
  else
    begin
      Memo1.Lines.SaveToFile(Filename);
      Memo1.Modified := False;
    end;

procedure THija.Guardarcomo1Click(Sender:
TObject);
begin
  SaveFileDialog.Filename := Filename;
  if SaveFileDialog.Execute then
    begin
      Filename := SaveFileDialog.Filename;
      Caption := ExtractFileName(Filename);
      Modifica.Caption := ' ';{retira mensaje}
      Save1Click(Sender);
    end;
end;

procedure THija.Save1Click(Sender: TObject);
begin
  if PathName = DefaultFileName then
    Guardarcomo1Click(Sender)
  else
    begin
      Editor.Lines.SaveToFile(PathName);
      Editor.Modified := False;
    end;
end;
```

```
procedure THija.ALinearClick
begin
  Left1.Checked := False;
  Right1.Checked := False;
  Center1.Checked := False;
  with Sender as TMenuItem do Checked :=
True;
  with Memo1 do
    if Left1.Checked then
      Alignment := taLeftJustify
    else if Right1.Checked then
      Alignment := taRightJustify
    else if Center1.Checked then
      Alignment := taCenter;
end;
```

```
procedure THija.FormClose(al cerrar la ventana hija)
begin
  Action := caFree; (para que pueda ser cerrada por
la madre)
end;

procedure THija.FormCloseQuery(acción antes de
cerrar)
var
  DialogValue: Integer;
  FName: string;
begin
  if Memo1.Modified then
    begin
      Confirma.showmodal; ->Muestra ventana confirma
      FName := Caption;
      Confirma.Label2.Caption := 'Grabar '+FName;
      case Confirma.Modalresult of
        mrOK: Save1Click(Self);
        mrCancel: CanClose := False;
      end;
    end;
end;
```

```
procedure THija.Nuevo1Click
begin
  Modifica.Caption := 'Otro documento nuevo';
  madre.Nuevo(Sender);
end;

procedure THija.Cerrar1Click
begin
  Close;
end;
```

```
procedure THija.NegritaClick(Sender: TObject);
begin
  if Negrita.Down then MEMO1.Font.Style :=
MEMO1.Font.Style + [fsBold]
  else MEMO1.Font.Style := MEMO1.Font.Style -
[fsBold];
end;
```

## ActionList y ActionManager: Procesador de textos utilizando el componente ActionList

Escoge: Archivo – Nuevo proyecto VCL (o *VCL Forms application*)

Cambia la propiedad: Form1.caption = Editor de textos

### Añadir componentes al formulario:

- Añadiremos un componente **RichEdit** de la paleta win32.

Una vez sobre el formulario cambiar la propiedad align a client.

- Pulsa doblo clic en el componete **StatusBar** de la paleta win32 y este se añadirá abajo del formulario como barra de estado.

Para crear un panel en la barra de estado:

En la propiedad **SimpleText** -> escribir: Sin título.txt.

En la propiedad **Panels** abrir el cuadro de Panels

Clic en el botón New para añadir un panel a la barra de estado statusBar1. Cierra el cuadro.

- Añadir menú y barra de herramientas:

De la paleta estandar escoge el componente: **Mainmenú1**

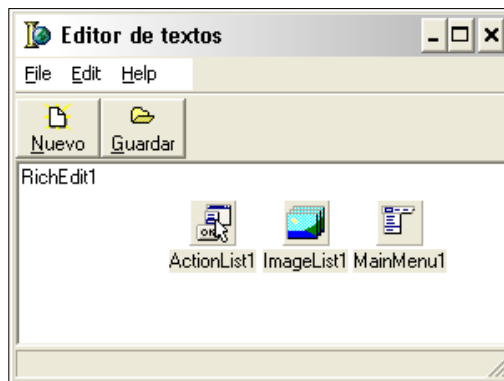
De la paleta Win32 escoge el componente: **ToolBar**

### Agrupar acciones mediante Action list:

Para agrupar las acciones del menú y de la barra de herramientas Delphi proporciona un "action manager" para centralizar ambos códigos e imágenes.

En la paleta adicional, haga doble clic en el componente: ActionManager o en la paleta estandar: action list. Para añadirlo al formulario. Como es un componente no visual, se puede poner en cualquier sitio.

Para ver los captions de componetes no visuales escoga: Tools - Environment Options, clic en Designer page, y seleccionar: Show component captions, OK..



Menu	Comando	Herramientas?	Descripción
Archivo	Nuevo	Sí	Creates a new file.
Archivo	Abrir	Sí	Opens an existing file for editing.
Archivo	Guardar	Sí	Saves the current file to disk.
Archivo	Guardar como...	No	Saves a file using a new name (also lets you save a new file using a specified name).
Archivo	Salir	Sí	Quits the editor program.
Edición	Cortar	Sí	Deletes text and stores it in the clipboard.
Edición	Copiar	Sí	Copies text and stores it in the clipboard.
Edición	Pegar	Sí	Inserts text from the clipboard.
Ayuda	Contenido	No	Displays the Help contents screen from which you can access Help topics.
Ayuda	Indice	No	Displays the Help index screen.
Ayuda	Acerca..	No	Displays information about the application in a box.

Doble-clic en Action list para abrirlo

En Editing Form1.ActionManager1 pulsar el botón: New Action. En sus propiedades escribir:

- › Caption = &New
- › Category = Archivo
- › Hint = Crea un archivo nuevo
- › ImageIndex = 6 (imagen 6 de ImageList)
- › Name = FileNew

Pulsar el botón: New Action. En sus propiedades escribir:

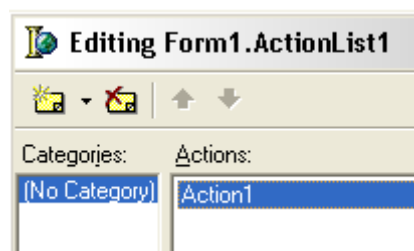
- › Caption = &Guardar
- › Category = Archivo
- › Hint = Guarda el archivo
- › ImageIndex = 8 (imagen 8 de ImageList)
- › Name = FileSave

Pulsar el botón: New Action. En sus propiedades escribir:

- › Caption = &Indice
- › Category = Ayuda
- › Hint = Indice ayuda
- › Name = HelpIndex

Pulsar el botón: New Action. En sus propiedades escribir:

- › Caption = &Acerca...
- › Category = Ayuda
- › Hint = Acerca de...      Name = HelpAbout

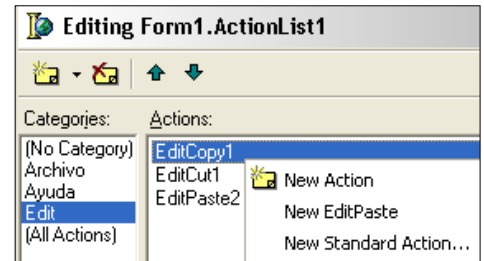


### Variante: ActionManager

Similar a Actionlist, tenemos el componente: **ActionManager** en la paleta Adicional. Va en combinación con Actionmainmenú y Actiontoolbar. Permite la apariencia Office XP y permite personalizar las barras de herramientas y el menú en tiempo real, con el componente: CustomizeDlg.

**Añadir acciones estandar al action list** (open, save as, exit, cut, copy, paste, help)

Pulsar el botón derecho del mouse a escoger del menú contextual:  
añadir New estándar Action: Añadir las acciones: TEditCut, TEditCopy, y TEditPaste. Se creará la categoría **Edit**



En la categoría archivo o File añade: TFileOpen, TFileSaveAs, and TFileExit.

En la categoría Help añade THelpContents.

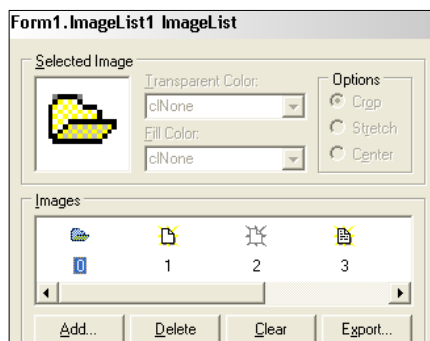
Guarda el proyecto: File - Save All

**Añadir imagenes al image list**

Truco: Abre el archivo (Archivo abrir) : C:\Archivos de programa\Borland\DelphiX\Source\Vcl\ActnRes.pas.

Aparece un ejemplo con las imágenes estándar ya listas.

También puedes Pulsar **Add** y buscar las imágenes en: C:\Archivos de programa\Archivos comunes\Borland Shared\Images\Buttons



Edit   Cut	0
Edit   Copy	1
Edit   Paste	2
File   New	6
File   Open	7
File   Save	8
File   SaveAs	30
File   Exit	43
Help   Contents	40

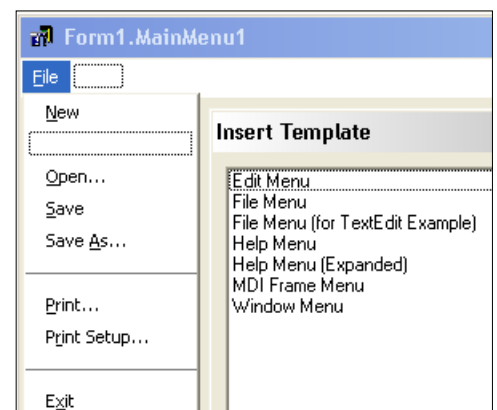
**Añadir un menú:**

De la paleta estandar escoge el componente: **Mainmenú1** y pulsa doble clic, para añadirlo al formulario.

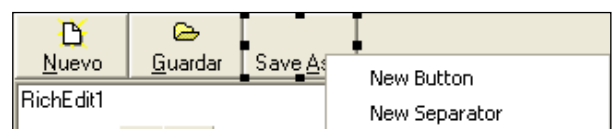
Pulsa luego doble clic sobre el componente del formulario mainmenu1 para editarlo.

En la propiedad: imagelist escoge: imagelist1

Puedes usar el botón derecho del mouse sobre el menú para añadir elementos al menú desde la plantilla: Insert from template. O añadir los elementos manualmente usando el botón derecho del mouse sobre el menú y escoger Insert.

**Añadir una barra de herramientas:**

- De la paleta Win32 escoge el componente: **ToolBar** y pulsa doble clic, para añadirlo al formulario.
- En la propiedad *imagelist*, escoge: imagelist1
- Pulsa el botón derecho del mouse para añadir botones a la barra de herramientas
- En la propiedad de cada botón de *Action*: Busca la *actionlist* que corresponda.



## Códigos de ejemplo típicos de eventos en Procesadores de texto

### Trabajar con ventanas

#### Salir. Cerrar ventana.

```
begin
  Close;
end;
```

#### Pregunta para cerrar.

```
procedure TForm1.FormClose(Sender: TObject; var
Action: TCloseAction);
begin
  if MessageDlg('Salir del programa ?',
mtConfirmation,
  [mbYes, mbNo], 0) = mrYes then
    Action := caFree
  else
    Action := caNone;
end;
```

#### Evento al cerrar hija (on close)

```
procedure THija.FormClose(Sender: TObject; var
Action: TCloseAction);
begin
  Action := caFree; //Close Action Permitido
end;
```

#### Abrir ventana Acerca de...

```
AboutClick
begin
  AboutBox.ShowModal;
end;
```

#### Poner nombre del archivo a una ventana.

```
Ventana.Caption := ExtractFileName(FileName);
```

#### Ventana mosaico.

```
procedure TFrameForm.Tile1Click(Sender: TObject);
begin
  Tile;
end;
```

#### Ventana cascada.

```
procedure TFrameForm.Cascade1Click(Sender:
TObject);
begin
  Cascade;
end;
```

#### Ventana organizar.

```
procedure TFrameForm.ArrangeIcons1Click(Sender:
TObject);
begin
  ArrangeIcons;
end;
```

#### Nueva ventana hija. (desde la madre)

```
procedure TPrincipal.Nuevo1Click(Sender: TObject);
var
  Hija: THija;
begin
  Hija:= THija.Create(Application);
  Hija.Show;
end;
```

#### Nueva ventana hija. (desde la hija)

```
procedure THijaNew1Click(Sender: TObject);
begin
  Ventana.Caption:='nuevo';{retira mensaje}
  PrincipalForm. Nuevo1Click(Sender);
end;
```

#### Abrir ventana hija.

```
procedure TFrameForm.OpenChild(Sender: TObject);
var
  EditForm: TEditForm;
begin
  if OpenFileDialog.Execute then
  begin
    EditForm := TEditForm.Create(Self);
    EditForm.Open(OpenFileDialog.FileName);
    EditForm.Show;
  end;
end;
```

### Trabajar con textos

#### Abrir texto en un componente memo.

```
procedure TForm1.FileOpenClick(Sender: TObject);
begin
  with OpenFileDialog do
    if Execute then
      begin
        Memo1.Lines.LoadFromFile(FileName);
        Caption := 'Text Demo - ' +
ExtractFileName(FileName);
      end;
end;
```

#### Guardar texto memo.

```
procedure TForm1.Save1Click(Sender: TObject);
begin
  if OpenFileDialog1.FileName <> '' then
  begin
    Memo1.Lines.SaveToFile(OpenDialog1.FileName);
  end
  else SaveAs1Click(Sender);
end;
```

#### Guardar como... texto memo.

```
procedure TForm1.SaveAs1Click(Sender: TObject);
begin
  with SaveDialog1 do
    if Execute then
      begin
        Memo1.Lines.SaveToFile(FileName);
        Caption := 'Text Demo - ' +
ExtractFileName(FileName);
        OpenFileDialog1.FileName := FileName;
      end;
end;
```

#### Diálogo Buscar texto.

```
procedure TForm1.Find(Sender: TObject);
begin
  with Sender as TFindDialog do
    if not SearchMemo(Memo1, FindText, Options)
then
  ShowMessage('Cannot find "' + FindText +
'".');
end;
```

#### Diálogo Reemplazar texto.

```
var
  Found: Boolean;
begin
  with ReplaceDialog1 do
    begin
      if AnsiCompareText(Memo1.SelText, FindText) =
0 then
        Memo1.SelText := ReplaceText;
      Found := SearchMemo(Memo1, FindText, Options);
      while Found and (frReplaceAll in Options) do
        begin ...
```

<b>Trabajar con archivos (vc1)</b>
------------------------------------

**Detectar directorio.**

```
procedure TFMForm.DirectoryChange(Sender:..);
begin
  FileList.Directory := Directory.Directory;
  DirectoryPanel.Caption := Directory.Directory;
end;
```

**Detectar lista de archivos.**

```
procedure TFMForm.FileListChange(Sender: TObject);
var
  TheFileName: string;
begin
  with FileList do
    begin
      if ItemIndex >= 0 then
        begin
          TheFileName := FileName;
          FilePanel.Caption := Format('%s, %d bytes',
[TheFileName, GetFileSize(TheFileName)]);
        end
      else FilePanel.Caption := '';
    end;
end;
```

**Ejecutar archivo.**

```
procedure TFMForm.Open1Click(Sender: TObject);
begin
  with FileList do
    ExecuteFile(FileName, '', Directory, SW_SHOW);
end;
```

**Copiar, renombrar, mover archivos.**

```
procedure TFMForm.FileChange(Sender: TObject);
begin
  with ChangeForm do
    begin
      if Sender = Move1 then Caption := 'Move'
      else if Sender = Copy1 then Caption := 'Copy'
      else if Sender = Rename1 then Caption := 'Rename'
      else Exit;
      CurrentDir.Caption := FileList.Directory;
      FromFileName.Text := FileList.FileName;
      ToFileName.Text := '';
      if (ShowModal <> idCancel) and (ToFileName.Text <>
'') then
        ConfirmChange(Caption, FromFileName.Text,
ToFileName.Text);
    end;
end;
```

**Borrar archivos.**

```
procedure TFMForm.Delete1Click(Sender: TObject);
begin
  with FileList do
    if MessageDlg('Delete ' + FileName + '?',
mtConfirmation,
[mbYes, mbNo], 0) = idYes then
      if DeleteFile(FileName) then Update;
end;
```

**Llamadas a procedimientos**

En el menú: Archivo - Mover y Copiar llaman al mismo evento Filechange

```
procedure TFMForm.FileChange(Sender: TObject);
begin
  with Changedlg do
    begin
      if Sender = Move1 then Caption := 'Mover'
      else if (Sender = Copy1) or (Sender = copiara) or
(Sender = Copiara1) then Caption := 'Copiar'
      else if Sender = Rename1 then Caption := 'Renombrar'
      else Exit;
      CurrentDir.Caption := Statusbar.panels[2].text;
      FromFileName.Text := FileList.FileName;
      ToFileName.Text := '';
      if (ShowModal <> mrCancel) and (ToFileName.Text <>
'') then ConfirmChange(Caption, FromFileName.Text,
ToFileName.Text);
      {Llama al procedimiento confirmchange pasandole tres
variables de texto}
    end;
end;
```

```
Procedure TFMForm.ConfirmChange(const ACaption,
FromFile, ToFile: string);
begin
  if MessageDlg(Format('%s %s to %s?', [ACaption,
FromFile, ToFile]),
mtConfirmation, [mbYes, mbNo], 0) = mrYes then
    begin
      if ACaption = 'Mover' then
        MoveFile(FromFile, ToFile)
      else if ACaption = 'Copiar' then
        CopyFile(FromFile, ToFile)
      else if ACaption = 'Renombrar' then
        RenameFile(FromFile, ToFile);
      FileList.Update;
    end;
end;
```

## Usar archivos INI para configuración

Para grabar y leer datos de configuración de una aplicación, utiliza la clase *TIniFiles* para generar archivos del tipo INI que almacenan información de texto con la INicialización de la aplicación.

Es necesario añadir la librería INIFILES en la sección USES de la unidad.

Utiliza el proyecto anterior para añadir los siguientes procedimientos:

### **Grabación del archivo INI:**

---

```
procedure onclose
var
  datosini:TIniFile;
begin
  datosini:= TIniFile.Create('DATOS.INI'); //-->crea el archivo datos.ini
  datosini.WriteInteger('Formato', 'Posicionizquierda',form1.left);
  datosini.WriteString('Formato', 'nombre',form1.caption); //guarda dato del tipo texto string
  DATOSINI.FREE; --> cierra el archivo datos.ini
end;
```

### **Lectura del archivo INI:**

---

```
Procedure on create
var
  datosini: TIniFile;
begin
  datosini:= TIniFile.Create('DATOS.INI');
  form1.left :=datosini.readinteger('Formato', 'Posicionizquierda',100);
```

### **Ejemplo válido para multiplataforma**

#### **Como grabar y leer todos los caption del formulario según el idioma escogido:**

- Buscaremos todos los componentes de nuestro formulario usando la clase: TCOMPONENT que me devuelve el nombre del componente según el número de orden de creación en el formulario. Por lo tanto:
- **Components[3]** es una variable *array* o *matriz* (conjunto de variables ordenadas) que me devuelve el nombre del componente número 3
- **Component is TLABEL:** Evalua si el componente es de la clase o tipo TLabel

```
{Al cargar el formulario lee el archivo de idioma.lng que le ha dicho en el archivo datosini y que antes lo ha guardado el usuario al cambiar la configuración de idioma en el archivo Datos.ini}
```

```
var
  Ini: TIniFile;
  component:tcomponent;
  i:integer;
  dire:string;
begin
  Dire:= Application.GetNamePath; //dire es la ruta por defecto
  Ini := TIniFile.Create(dire+'\langs\'+idioma+'.lng'); //crea el archivo ini idioma.lng
  //--> Ruta para ios: TIniFile.Create(GetHomePath + PathDelim + 'Documents' + PathDelim + 'ex.ini');
  for i := 0 to ComponentCount - 1 do //--> Agrupa por componente seccion [CAPTIONS]
  begin
    Component := Components[ i ];
    if (Component is TLABEL) and ((Component as TLABEL).Caption <>'') then
      (component as TLABEL).Caption:=Ini.ReadString( 'CAPTIONS', (component as
      TLABEL).name,(component as TLABEL).Caption)
  end;
end;
```

**Creación de objetos** (Extracto canal *It Tools Ltda*)

- ▶ En una aplicación nueva VCL, abrimos en Code al inicio de la unidad.
- ▶ En la definición de tipos, crearemos la clase estudiante y el array MisNotas.

Luego crearemos el objeto estudiante del tipo Testudiante.

```

type //definición de tipos
  MisNotas= Array [1..4] of double;
  TForm1 = class(TForm)
  private
  public
  end;
  Testudiante = class //defino los datos de la clase estudiante
  Nombre: string;
  Apellido: string;
  Asignatura: string;
  Suma: double;
  end;
var
  Form1: TForm1;
  estudiante: Testudiante; //creo estudiante del tipo Testudiante

```

- ▶ En el formulario, añadidos los objetos de la imagen.
- ▶ Declaramos y añadimos los procedimientos y la función:

```

procedure TForm1.Button1Click(Sender: TObject);
var
  prom: MisNotas;
begin
  ConsultarNombre;
  CalcularPromedio(prom);
end;

procedure TForm1.ConsultarNombre;
begin
  estudiante:=Testudiante.create;
  estudiante.Nombre:=Edit1.Text;
  estudiante.Apellido:=Edit2.Text;
  estudiante.Asignatura:=Edit3.Text;
  Showmessage('Estudiante: ' + Estudiante.nombre + ' ' + Estudiante.Apellido);
  estudiante.Destroy; // para liberar memoria
end;

function TForm1.CalcularPromedio(Notes: MisNotas):double;
var f:string;
begin
  Notes[1]:=StrToFloat(EditN1.text);
  Notes[2]:=StrToFloat(EditN2.text);
  Notes[3]:=StrToFloat(EditN3.text);
  Notes[4]:=StrToFloat(EditN4.text);
  Result:=(Notes[1]+Notes[2]+Notes[3]+Notes[4])/4;
  f:=FloattoStr(Result);
  Showmessage('Promedio = '+f);
end;

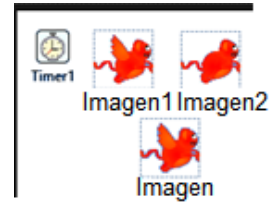
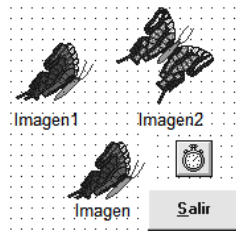
```

## Ejercicio base de animación. Flappy.

A cada impulso del timer la *imagen* se intercambia con las otras dos y avanza. Las otras dos imágenes deben estar ocultas y el timer activado (enabled). Ajustamos la propiedad interval del timer1 a 100 o 200.

El truco reside en utilizar una variable a modo de permiso o bandera *flag* que puede ser del tipo integer o mejor boolean (2 estados).

```
Visual Basic: Private Sub Timer1_Timer()  
    Static Flag As Integer  
    Imagen.Move Imagen1.Left + 20, Imagen.Top - 5  
    If Flag = 1 Then  
        Imagen.Picture = Imagen1.Picture  
        Flag = 0  
    Else  
        Imagen.Picture = Imagen2.Picture  
        Flag = 1  
    End If  
End Sub
```



```
C++ Builder:  
void __fastcall TForm1::Timer1Timer  
{  
    boolean Flag = True; //variable flag  
    Imagen->Left=Imagen->Left+10;  
    if (Flag == true)  
    {  
        Imagen->Picture= Imagen1->Picture;  
        Imagen->Top=Imagen->Top+10;  
        Flag= false;  
    }  
    else  
    {  
        Imagen->Picture= Imagen2->Picture;  
        Imagen->Top=Imagen->Top-10;  
        Flag= true;  
    }  
}}  
;
```

```
Pascal - Delphi:  
procedure TForm1.Timer1Timer(Sender: TObject);  
var  
    Flag: boolean; //variable flag  
begin  
    Imagen.Left:=Imagen.Left+10; //mueve a la derecha  
    If Flag = true Then  
    begin  
        Imagen.Picture:= Imagen1.Picture;  
        Imagen.Top:=Imagen.top+10;  
        Flag:= false;  
    end  
    else  
    begin  
        Imagen.Picture:= Imagen2.Picture;  
        Imagen.Top:=Imagen.top-10;  
        Flag:= true;  
    end;  
end;
```

### Firemonkey - Delphi

```
PajaroFlag: Boolean;  
RectangleMouseDown:  
begin  
    if Timer1.Enabled=true then  
    begin  
        if pajaro.Position.Y<75 then pajaro.Position.Y:=0  
        else pajaro.Position.Y:= pajaro.Position.Y-75;  
    end;  
    pajaro.RotationAngle:=0;  
Timer1Timer:  
if PajaroFlag=False then  
    begin  
        Pajaro.Bitmap.Assign(PajaroB.Bitmap);  
        PajaroFlag := True;  
    end  
    else  
    begin  
        Pajaro.Bitmap.Assign(PajaroA.Bitmap);  
        PajaroFlag := False;  
    end;  
end;
```

Guarda el proyecto con el nombre: Flappy1

## Bases de Datos con Delphi - Teoría

---

Delphi y C++ Builder tienen métodos de acceso a bases de datos que se basan en **TDataSet**. que, como su nombre lo indica, es un "set de datos". El set de datos abstracto no especifica el origen de la base de datos. Esto lo hace flexible e independiente.

Los componentes **Table** y **Query** son los componentes más comúnmente usados para acceso básico a bases de datos (pero no los únicos). Estos componentes representan una tabla y una consulta SQL, respectivamente.

### Conectores:

- **BDE**: Librerías para acceder a tablas de dBase con el motor DataBaseEngine (obsoleto)
- **ODBC**: Controlador creado por Microsoft para acceder a datos, integrado en Windows.
- **LiveBindings**. (*Versiones Xe*). Acceso desde controles de FireMonkey y dispositivos móviles.

### Driver o librerías de acceso a datos:

- **DbExpress**: Sustituto del BDE que ha quedado obsoleto.
- **ADO**: ActiveX Data Objects. Permite acceder a datos, sin necesidad de conocer la base de datos.
- **XML Client Dataset**: Acceso a tablas de datos XML Necesita **MIDAS.dll**
- **FireDac**. (*Versiones Xe*) Acceso directo a bases de datos

### Componentes No Visuales de Acceso a Datos:

**Conexión**: Configura el driver y el modo de acceso a la base de datos.

**Database**: Es la representación de todo el conjunto de datos, conjunto de tablas de datos o lugar donde se guardan.

**Table** (Tabla): Representa una tabla entera. TTable contiene información acerca del tipo de tabla, del índice, así como filtros. TTable tiene entre sus métodos, First, Next, Last, etc.

**Query** (Consulta): Se comporta como una tabla, pero al mismo tiempo es un comando de SQL.

¿Porqué **SQL**? La respuesta es sencilla: **COMPATIBILIDAD**. La especificación SQL-92 es soportada por todos los lenguajes generales.

**DataSource**: Es el "origen de los datos". Un componente intermedio que permite tener varias representaciones de los mismos datos

**UpdateSQL**: Permite editar un query SQL. Todos los queries tienen un Edit, Insert y Delete. **TStoredProc**: Permite ejecutar un comando SQL sin esperar ninguna respuesta, como un procedimiento.

### Componentes Visuales de Acceso a Datos

Los componentes de este tipo son llamados "Data-aware", y tienen el prefijo "DB". De este modo, un componente de edición "Edit" en su versión "Data Aware" es llamado "**DBEdit**". ComboBox se convierte en "**DBComboBox**".

**DBgrid** es una tabla de datos y **DBControlGrid** equivale a formularios continuos.

Además, existe un componente llamado "**DBNavigator**", que es una serie de botones "mapeados" a los comandos Next, Prior, Last, Edit, Delete, etc., de cualquier DataSource.

## Ejercicio 8. Bases de datos. Agenda de teléfonos.

### • Tabla de datos DBF - Modo DBE: (obsoleto)

En un gestor de base de datos, crear el archivo que contendrá los datos *Data Base File*.

Crear la estructura de la derecha y guardar como **Agenda.dbf**

En el formulario principal de Delphi, añadir los componentes: un

Table1 (Paleta BDE) y Datasource1 (Paleta Data Access)

En la *Table1*: La propiedad TableName es el nombre del archivo de

datos: *Agenda.dbf*; Name es el nombre de la Tabla: *Table1* y

Databasename es la ruta donde se encuentra el archivo de datos

Create dBASE for Windows Table: (Untitled)			
Field roster:			
	Field Name	Type	Size
1	APELLIDOS	C	50
2	NOMBRE	C	50
3	TELEFONO	C	20
4	EMAIL	C	50
5	DIRECCION	C	80
6	NOTAS	C	80
7	WEB	C	80

Create dBASE for Windows Table: (Untitled)			
Field roster:			
	Field Name	Type	Size
1	APELLIDOS	C	50
2	NOMBRE	C	50
3	TELEFONO	C	20
4	EMAIL	C	50
5	DIRECCION	C	80
6	NOTAS	C	80
7	WEB	C	80

### • Tabla de datos Access FireDAC

En Access y crea una base de datos con una tabla que contenga los campos de la imagen y guardar como *Agenda.adb* En el

formulario principal de Delphi, añadir los componentes: un

TFDTable y un TFDConecton (Paleta FireDAC) un Datasource1 (Paleta Data Access)

En TFDConecton, DriverName *MsAccess* y Database: la ruta de

archivo de base de datos de Access.

En la TFDTable1: La propiedad TableName poner el nombre de la

de Access; Name es el nombre de la Tabla: *Table1* y Conecton es

el TFDConecton creado.

### • Tabla de datos XML - Modo ClientDataset

- En el formulario principal de Delphi, añadir los componentes ClientDataset y un Datasource (Paleta Data Access).

- Pulsar el botón derecho sobre el componente ClientDataset y escoger de *Fields editor... New field*.

- Crear los campos de la imagen derecha: Nombre – Apellidos – Telefono – Email – etc... del tipo String y ancho de unos 50 a 100 caracteres.

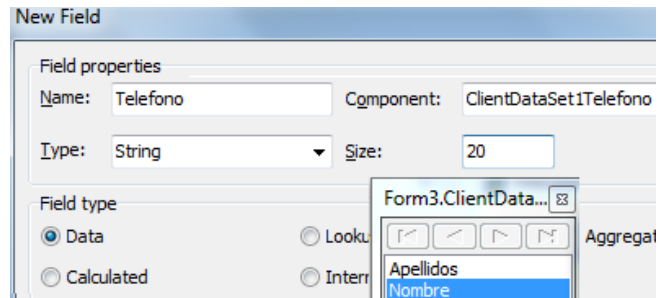
- Una vez finalizado, cierra la lista de campos y pulsa el botón derecho sobre el ClientDataset y escoge: *Create Dataset* y luego: *Save to MyBasexml table*. Guarda la tabla con el nombre: **Agenda.xml**

- Llámale al clientdataset con el nombre: **Tabla1**

- A la Datasource1 le pones en su propiedad Dataset: el nombre **Tabla1**.

- Añadir al formulario los componentes DbNavigator y un DBGrid de la paleta de *Datacontrols* y asignales la Datasource1 Asegúrate que la *Tabla1* esta en Active=True.

Ya puedes probar la aplicación (Run ▶ ó F9)



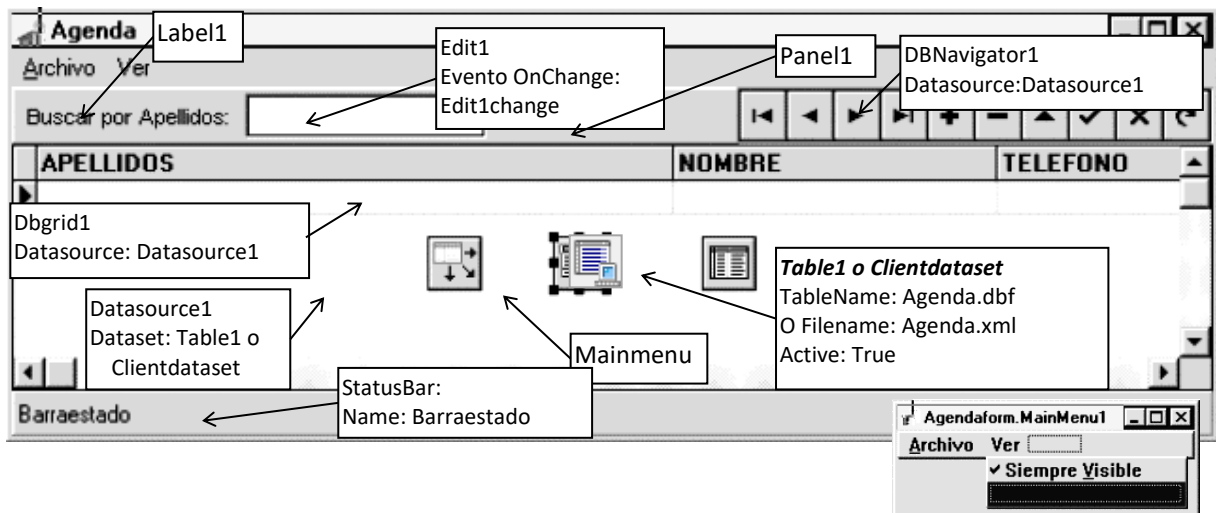
Comprueba que que permite añadir y borra datos en la tabla, pero estos se pierden al volver a ejecutar la aplicación.

Cierra prueba y uelve al diseño y selecciona la *tabla1* asigna la propiedad a la *Tabla1*: filename: *Agenda.xml* (también pedes asignarla en tiempo de ejecución)

Añade los otros componetes de la imagen: Un menú y un cuadro para buscar dentro de un panel alineado al Top.

El menú consta de los ítems: Archivo -> Salir y Ver -> Siempre visible y transparente

Guarda el proyecto: *Save Project as...* Crea una carpeta llamada *Agenda* y guarda la unidad como: *Agenda1.pas* y el proyecto como *Agenda.dpr*



- Vuelve a ejecutar el proyecto y comprueba que ahora sí se quedan guardados los datos en el archivo Agenda.XML.
- Para buscar el nombre:  
Es necesario indexar la tabla por el campo a buscar: Para ello escoge de la tabla1: IndexDef y crea un nuevo índice para el campo nombre y otro para el campo apellidos. Luego asigna el indexname: **Apellidos** a la tabla. Pula doble clic en el cuadro Edit1 para abrir el evento Edit1Change:  
Escribe el código:

Edit1Change

```
Tabla1.setRangeStart;
Tabla1.FieldName('Apellidos').AsString:=Edit1.Text;
Tabla1.Applyrange;
```

**Procedures:**CreateForm (asigna la table en tiempo de ejecución)

```
Table1.tablename:= 'Agenda.dbf' //→ para DBF
o Tabla1.filename:='Agenda.xml' // → para XML
```

```
Application.OnHint :=muestrahints;
```

Edit1Change

```
Tabla1.SetrangeStart;
Tabla1.FieldName('Apellidos').AsString:=Edit1.Text;
Tabla1.Applyrange;
```

Salir1Click

```
Close;
```

visibleClick:

```
if visible.Checked then Form1.FormStyle :=
fsStayOnTop
else Form1.FormStyle := fsNormal;
end;
```

TranparenteClick:

```
if transparente.Checked then
Form1.alphaBlend:=true;
```

```
//=>Mostrar infomación en barra de estado:
```

Procedure muestrahints

```
Barraestado.Caption := Application.Hint;
```

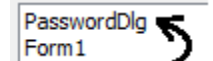
**Variantes al ejercicio:****Ordenar la tabla:**

- Si tienes instalados los componentes JVCL puedes cambiar el dbgrid por un JVultimdbgrid que te permite ordenar la tabla al pulsar en el encabezado del campo.

**Crear acceso con password.**

- Añade una ventana tipo **Password** que pida la clave antes de entrar:  
File – New – Other –(Delphi files o Dialogs) – Password dialog
- En (Project - Options - Autocreate forms): Cambiar el orden para que se muestre primero la ventana de Password antes que la principal (*main*).

Auto-create forms:



Si la respuesta es correcta, habilitar el botón OK que mostrará la ventana principal Form1.

Opción1: Añadir la unit1 de la ventana principal a la lista de uses del password y llamarla con form1.show

Opción2:

PasswordChange:

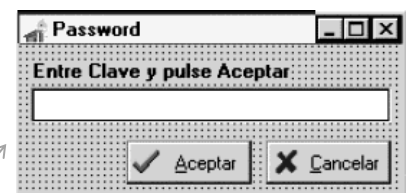
```
If Password.text= 'ofimega' then Okbtn.enabled:=true;
```

Okbtnclick:

```
// -> Opción 1
Application.CreateForm(Tform1, form1);
```

//-> Opción 2:

```
form1.Show; // -> muestra la ventana form1
hide; // -> oculta la ventana atual
```



## Acceso login usuario Acceso a tablas Access con Firedac

**Crear la Tabla:** Entra en Access y crea una base de datos con una tabla llamada Acceso que contenga los campos de la imagen:

Nombre del campo	Tipo de datos
Id	Autonumeración
Usuario	Texto
Clave	Texto
Permiso	Número

**Componentes:** Crea en Delphi un nuevo proyecto y añade los componentes de la imagen.

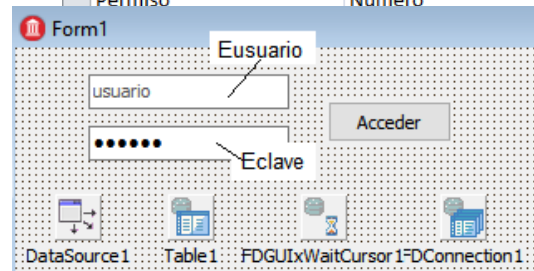
En *FDConeccion1* pulsa doble clic y especifica el driver: *MsAcces* y la ruta donde has guardado la base de datos que debe ser la misma del ejecutable. Luego cambia Conected: True

Driver ID:

Database:

El *DataSource1* asignar su dataset a: Table1

El *Table1*: Asigna su TableName = acceso y Active=True



**Complementos:** Para realizar las pruebas y comparaciones puedes añadir al formulario un DBNavigator y un DBGRID ambos conectados con el Datasource1

### Método de búsqueda de usuario por el índice:

```
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  encontrado:string;
begin
  Table1.IndexFieldNames := 'Usuario';
  Table1.SetKey;
  Table1.FieldByName('Usuario').AsString := EUsuario.Text;
  if Table1.GotoKey then
    begin
      ShowMessage('Encontrado a '+TUsuario.Value);
      if TClave.value=Eclave.Text then ShowMessage('Clave
correcta')
      else ShowMessage('Clave Incorrecta');
    end
  else
    ShowMessage('Usuario no encontrado');
end;
```

### Método de búsqueda secuencial:

```
procedure TForm1.BitBtn1Click (Sender: TObject);
var
  encontrado: Boolean;
  opciones:TLocateOptions;
begin
  opciones:= [LOCaseInsensitive];
  encontrado:= FDTTable2.Locate('Usuario', EUsuario.Text,
opciones);
  if encontrado=false then
    showmessage('Usuario incorrecto o inexistente')
  else
    begin
      showmessage('Muy bien! has encontrado a '+
FDTTable2Usuario.value + ' y su clave es la '
+ FDTTable2Clave.value);
    end;
end;
```

Método para crear nuevo usuario:

```
procedure TForm1.BNuevoUsuClick(Sender: TObject);
var
  encontrado: Boolean;
  opciones:TLocateOptions;
begin
  opciones:= [LOCaseInsensitive];
  if NUsuario.Text='' then showmessage('No puede estar vacio') else
  begin
    if NClave1.Text=NClave2.Text then
      begin
        encontrado:= TableAcceso.Locate('Usuario', NUsuario.Text,
opciones);
        if encontrado=true then
          showmessage('Ya existe un uuario con el mismo nombre')
      end
    else
      begin
        TableAcceso.Append;
        TableAccesoUsuario.value:=NUsuario.text;
        TableAccesoClave.value:=NClave1.Text;
      end;
    else
      begin
        showmessage('No coinciden las claves');
      end;
    end;
  end;
end;
```

### **Ejercicio nivel de acceso.**

Seleccionar el tipo de acceso según el nivel de permiso:

Una vez encontrado el usuario y contraseña leerá el permiso y si es nivel 0 muestre un label en la ventana con la infomación: "Modo Usuario + nombre" pero si es el nivel de permiso 1 muestre: "Has entrado como adminitrador" + nombre.

## Aplicación tetst de preguntas y respuestas TestWin - Acceso a tablas Access con **Firedac**

Utiliza el apartado anterior para iniciar sesión en la aplicación en modo usuario o administrador.

En el formulario pondremos una imagen alineada *al top* como encabezado de ventana y un *pagecontrol*, alineado *alClient*, al que le añadiremos tres pestañas o tabsheets con los nombres: sesión, frontal y trasera.

### Acceso a datos.

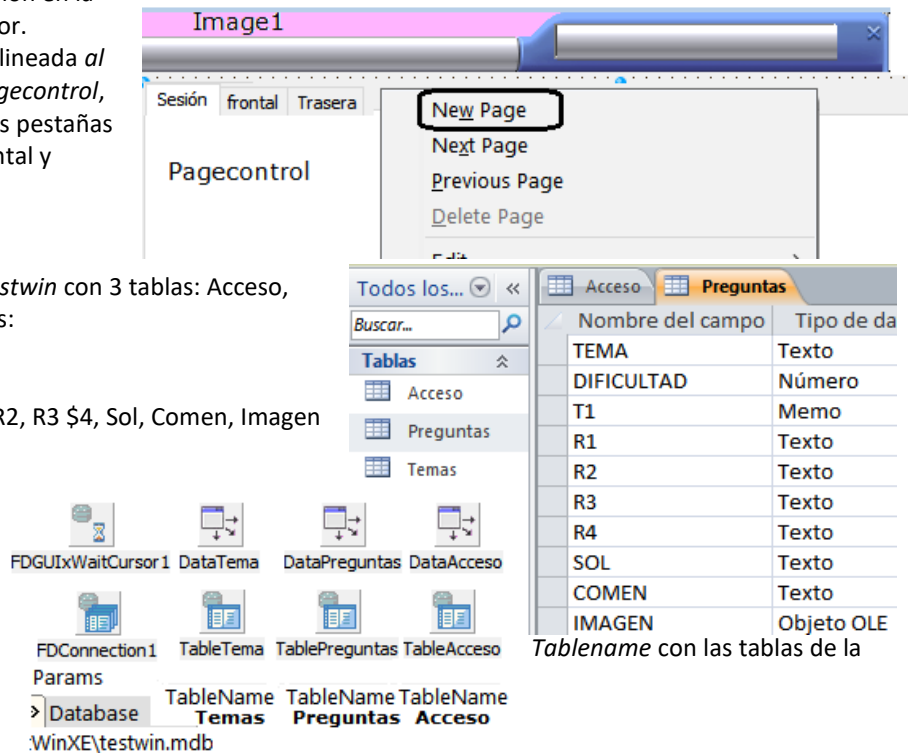
Desde Access creamos la base de datos *Testwin* con 3 tablas: Acceso, Preguntas y Temas y los siguientes campos:

- Acceso: Usuario – Clave – Permiso
- Temas: ID – Tema
- Preguntas: Tema, Dificultad, T1, R1, R2, R3, R4, Sol, Comen, Imagen

### Página Sesión

En el formulario, añadimos los objetos de **datos** de la imagen.

Cada *datasource* enlaza con su respectiva *TFDtable* y en ellas, cada base de datos.



Pestaña de acceso:

Añadimos dos *groupbox*

Destinado a iniciar sesión de usuario o crear nuevo usuario.

En cada uno de ellos insertamos dos cuadros de texto (edits) y dos botones (buttons) como en la figura.

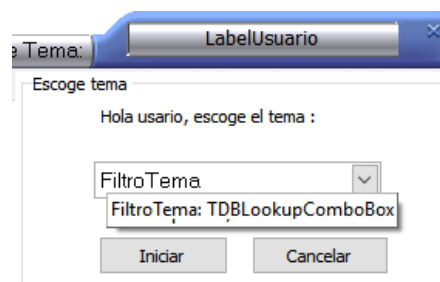
El segundo *groupbox*, se establece *visible = false*.

Al pulsar el botón *BNuevo* del primer *groupbox1* se mostrará el segundo y ocultará el primero:

```
procedure TForm1.BNuevoClick(...);
begin
  GroupBox2.Visible:=True;
  GroupBox1.Visible:=False;
end;
```

*Opcional:* Puedes añadir al formulario un *DBGrid* y un *DBNavigator* no visuales y vinculados a la tabla de *Acceso* para crear y comprobar a los usuarios.

Añadimos un nuevo *GroupBox3* que contendrá un *LookupComboBox* llamado *Filtrotema*, para la selección del tema y un *LabelUsuario* que informará del nombre del usuario. →

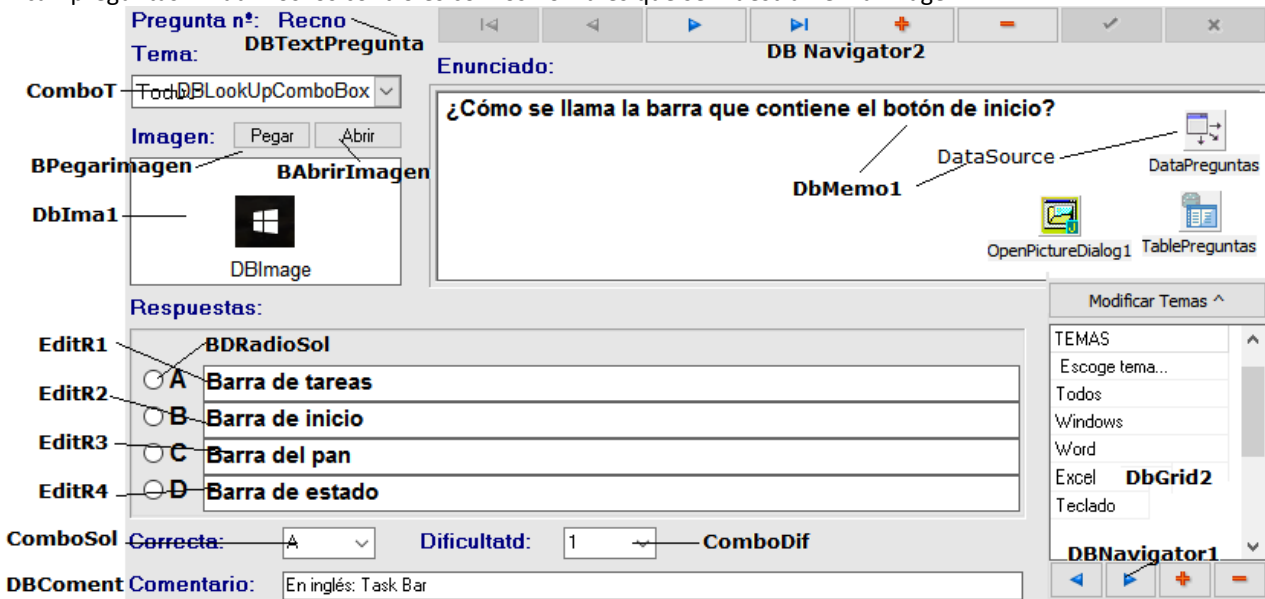


Al pulsar el botón para acceder *Bacceso*, comprueba con la función *Locate* si el usuario existe en la tabla. En caso afirmativo, muestra el *GroupBox3* para escoger el tema. Una vez escogido el tema, al pulsar el botón de *Iniciar*, activaremos la segunda página del *pagecontrol*:

```
procedure TForm1.BAcesoClick(Sender: TObject);
var
  encontrado: Boolean;
  opciones:TLocateOptions;
begin
  opciones:= [LOCaseInsensitive];
  encontrado:= TableAcceso.Locate('Usuario',
  EUsuario.Text, opciones);
  if encontrado=false then
    showMessage('Usuario incorrecto o inexistente')
  else
    begin
      GroupBox1.Visible:=False;
      GroupBox2.Visible:=False;
      GroupBox3.Visible:=True;
      LabelUsuario.caption:= TableAccesoUsuario.value;
    end;
```

**Página trasera**

Es la página de administración de preguntas y se accede a ella con privilegios de administrador. En ella podemos crear y modificar preguntas. Añadimos los controles con los nombres que se muestran en la imagen:



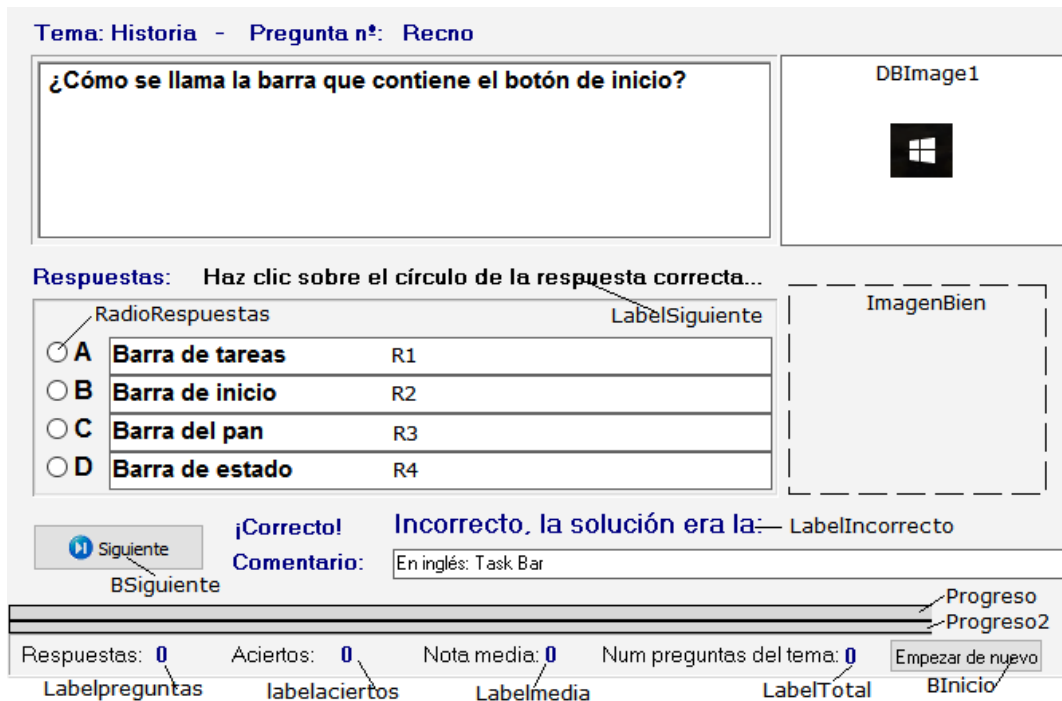
**Para cargar la imagen:** Añadir antes un elemento de diálogo: OpenPictureDialog

- Al pulsar el el botón pegar: DBIma1.PasteFromClipboard;
- Al pulsar en el botón Abrir: if OpenPictureDialog1.Execute() then dbIma1.Picture.LoadFromFile(OpenPictureDialog1.FileName);

**Para Modificar Temas:** Al pulsar el Botón *Modificar Temas*: GroupboxTemas.visible:=true;

**Página frontal**

En la pestaña frontal, añadir los controles con los nombres que se muestran en la imagen:



**Creamos las variables públicas:**

aciertos, preguntas: integer; //-> numéricas enteras  
 media: single; //-> decimales de precisión sencilla

**Añadimos los eventos siguientes:**

**Al pulsar (On Click) en el botón Bsiguiente:**

```

R1.font.color:=clblack;
R2.font.color:=clblack;
R3.font.color:=clblack;
R4.font.color:=clblack;
labelcorrecto.visible:=false;
labelincorrecto.visible:=false;
DBTextcomen.visible:=false;
Labelpregunta.caption:='Pregunta número ' +
inttostr(preguntas+1) + ' de ' +
inttostr(TablePreguntas.RecordCount);
labelsiguiente.caption:='Haz clic sobre el círculo de la respuesta
correcta...';
labelelige.visible:=true;
RadioRespuestas.itemindex:=-1;
Bsiguiente.enabled:=false;
//StatusBar1.Panels[0].Text:='Nº de pregunta: '
+inttostr(TablePreguntas.recno);
RadioRespuestas.Enabled:=True;
RadioRespuestas.Hint:='Elige';
TablePreguntas.Next;
StatusBar1.Panels[0].Text:='Nº de pregunta: '
+inttostr(TablePreguntas.recno);
imagenBien.picture.loadfromfile('DRAC.BMP');
StatusBar1.Panels[0].Text:='Nº de pregunta: '
+inttostr(TablePreguntas.recno);

```

**Al pulsar (On Click) en RadioRespuestas:**

```

var
letra:string;
begin
Inicio.Enabled:=true;
if Bsiguiente.enabled=false then
begin
case RadioRespuestas.itemindex of
0: letra:= 'A';
1: letra:= 'B';
2: letra:= 'C';
3: letra:= 'D';
end;
RadioRespuestas.Enabled:=false;
RadioRespuestas.Hint:='';
if letra=TablePreguntasSQL.value then
begin
labelcorrecto.caption:='!Correcto!, es la
'+TablePreguntasSQL.value;
labelcorrecto.visible:=true;
labelincorrecto.visible:=false;
aciertos:=aciertos+1;
labelaciertos.caption:=inttostr(aciertos);
imagenBien.picture.loadfromfile('DRACBIEN.BMP');
//<---cambio colores--->
If letra='A' THEN R1.font.color:=clblue
else if letra='B' THEN R2.font.color:=clblue
else if letra='C' THEN R3.font.color:=clblue
else if letra='D' THEN R4.font.color:=clblue;
end
else
begin
labelincorrecto.caption:='Incorrecto, la solución no era la '+
letra+' sino la '+TablePreguntasSQL.value;
labelcorrecto.visible:=false;
labelincorrecto.visible:=true;
imagenBien.picture.loadfromfile('DRACMAL.BMP');
letra:=TablePreguntasSQL.value;

```

```

//<---cambio colores--->
If letra='A' THEN R1.font.color:=clred
else if letra='B' THEN R2.font.color:=clred
else if letra='C' THEN R3.font.color:=clred
else if letra='D' THEN R4.font.color:=clred;
end;
DBTextcomen.visible:=true;
labelsiguiente.Caption:='Pulsa siguiente pregunta';
labelsiguiente.visible:=true;
labelelige.visible:=false;
Bsiguiente.enabled:=true;
preguntas:=preguntas+1;
labelpreguntas.caption:=inttostr(preguntas);
media:=round(aciertos*10/preguntas);
labelmedia.caption:=floattostr(media);
progreso.progress:=preguntas;
progreso2.progress:=aciertos;
If letra='A' THEN R1.color:=clwhite
else if letra='B' THEN R2.color:=clwhite
else if letra='C' THEN R3.color:=clwhite
else if letra='D' THEN R4.color:=clwhite;
if (preguntas=TablePreguntas.RecordCount) then //fin de
preguntas
begin
if aciertos=preguntas then Labelsiguiente.Caption:='¡¡Enhora
buena, has obtenido la máxima puntuación!! '+
Labelmedia.caption else
Labelsiguiente.Caption:='Fin de preguntas del tema. Has obtenido
un '+ Labelmedia.caption;
MessageDlg(Labelsiguiente.Caption, mtInformation, [mbOk], 0);
Bsiguiente.enabled:=false;
Inicio.Enabled:=true;
end;
Labeltotal.caption:=inttostr(TablePreguntas.RecordCount);
end;

```

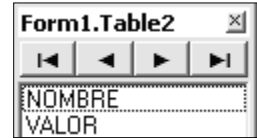
**Al pulsar (On Click) en el botón BInicio:**

```

TablePreguntas.First;
RadioRespuestas.Enabled:=true;
Bsiguiente.Enabled:=false;
progreso.progress:=0;
progreso2.progress:=0;
progreso.MaxValue:=TablePreguntas.RecordCount;
progreso2.MaxValue:=TablePreguntas.RecordCount;
aciertos:=0;
preguntas:=0;
Labelmedia.Caption:='0';
labelaciertos.Caption:='0';
Labelpreguntas.Caption:='0';
Labelsiguiente.visible:=true;
Labelsiguiente.Caption:='Inicio del tema '+ filtro.text + '. Clic sobre
el círculo de la respuesta correcta...';
RadioRespuestas.itemindex:=-1;
StatusBar1.Panels[0].Text:='Nº de pregunta: '
+inttostr(Table1.recno);
R1.font.color:=clblack;
R2.font.color:=clblack;
R3.font.color:=clblack;
R4.font.color:=clblack;
DBTextcomen.Visible:=false;
Labelincorrecto.Visible:=false;
Labelcorrecto.Visible:=false;
imagenBien.picture.loadfromfile('DRAC.BMP');
Labelpregunta.caption:='Pregunta: ';

```

# Gráfico de datos (probado en versiones Delphi 7 a XE7)



## Ingredientes:

### Crear una tabla:

a. **DBF (obsoleto):** Delphi 7: Escoge del menú: *Tools* ► *Data base desktop* con los campos: **Nombre** (de carácter o texto) y **Valor** (numérico entero) guardar la tabla de datos con el nombre: **Datos.dbf** Luego Añadir los componentes Table y datasource al formulario y asignar a la table el tablename: Datos.dbf y al Datasource el dataset: table1

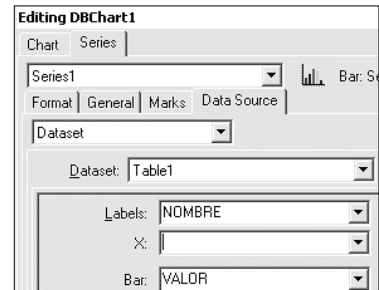
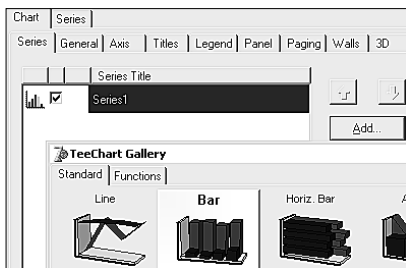
b. **XML:** En el formulario principal de Delphi, añadir los componentes: ClientDataset y un Datasource (Paleta Data Access). Pulsar el botón derecho sobre el componente *ClientDataset* y escoger de *Fields editor... New field.* con los campos: **Nombre** (de carácter o texto) y **Valor** (numérico entero). Botón derecho sobre el componente *ClientDataset* y escoger Create dataset - Save to my base xml table...datos.xml

Nombre o identificación	Valor (x)
Juan	5
Andrés	5
José	4
Pedro	7

Formulario: En la paleta *DataControls*, añadir al formulario un Dbgrid, y en su propiedad datasource: poner datasource1.

### Añadir el gráfico:

- Delphi 7: De la paleta *Datacontrols*, añadir un componente: **DBChart**.
  - Delphi XE7: De la paleta: TeeChartLit, añadir un **TBChart**
- Pulsar doble clic y en la solapa *series* poner como en la figura.  
En la solapa *Chart*, Añadir la serie1 (Add) tipo Bar como en figura.  
En la pestaña Datasource: Seleccionar Dataset y elegir el Dataset de nuestra tabla de datos.



### Añadir un botón para calcular

Añadir un botón al formulario para calcular la suma de valores. Al pulsar se genera el evento:

### Cocción:

```

procedure TForm1.calcula1Click(Sender: TObject);
var
suma, xmax, xmin, media: single;
antes: single;
begin
suma:=0;
dessa:=0;
varsuma:=0;
table1.First;
antes:=table1valor.value;
while not table1.eof do
begin
xmax:=max(table1valor.value, antes);
suma:=suma+table2valor.value;
table1.Next;
antes:=table1valor.value;
end;

```

```

if Table1.RecordCount<1 then showMessage('Introduzca valores en la tabla para calcular') else
begin
media:=suma/Table1.RecordCount;
table1.refresh;
table1.Edit;
table1.First;
table1.DisableControls; // -> ojo
while not table1.eof do
begin
table2.edit;
table2porcentaje.Value:=roundto((table2valor.value/suma)*100,-2);
table2.Next;
end;

```

- **Ejercicio propuesto1:** Añadir un campo **calculado** a la tabla Decil y cuartil . En el evento: OnCalcFields de la tabla escribir: `Table1Decil.value:=table1porcentaje.Value/10;`
- `Table1Cuartil.value:=table1porcentaje.Value/25;`
- **Ejercicio propuesto2:** 2: Añadir un componente **WebBrowser**: En el Form1.Oncreate: `dire:=ExtractFilePath(ParamStr(0)); WebBrowser1.Navigate('file://'+dire+'estadistica.htm');`

**PROGRAMA DE RECETAS DE COCINA PARA DELPHI. (Uso de Datos y los Tabsheets)**

Prepara una carpeta llamada *Recetas*.

- **Para DBF-MDB:** Crear un archivo DBF (*Data Base File*) o MDB (Access) con la estructura de la imagen, con un índice indexado por el campo *Titulo* y guarda la tabla con el nombre: *RECETAS* en la carpeta *Recetas*.

• **Para XML:**

Añade el componente **ClientDataset** y con el botón derecho agrega los campos de la imagen.  
 Crea el Dataset y guarda la Database con el nombre: *RECETAS.XML* en la carpeta *Recetas*  
 Crea los índices en la propiedad: *Indexdef*

Recetas.DBF				
	Field Name	Type	Size	Dec
1	NUMERO	N	3	0
2	TIPO	C	15	
3	TITULO	C	30	
4	DIFICULTAD	C	15	
5	TIEMPO_COC	N	3	0
6	TIEMPO_PRE	N	3	0
7	INGREDIENT	M		
8	PREPARAC	M		

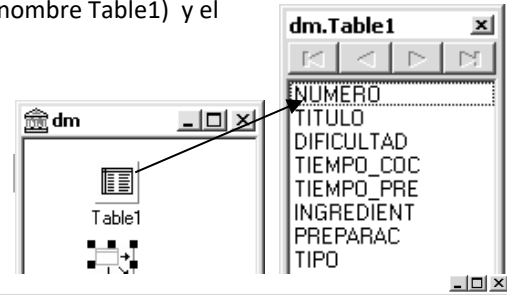
Table properties:

Indexes

Define... Modify...

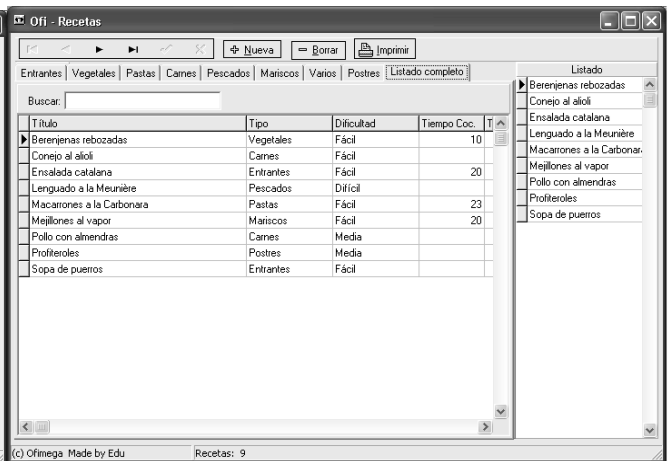
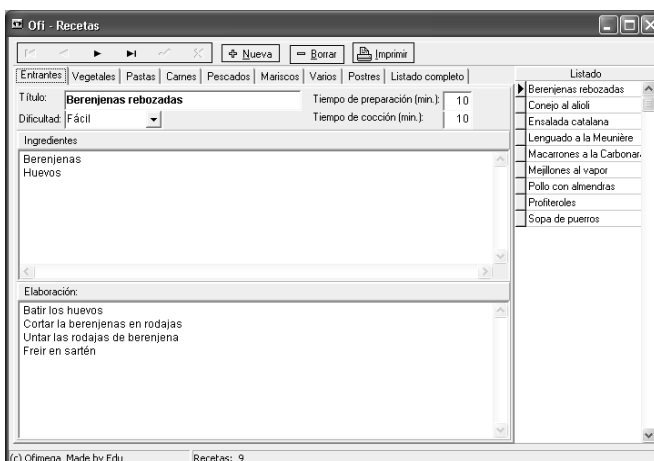
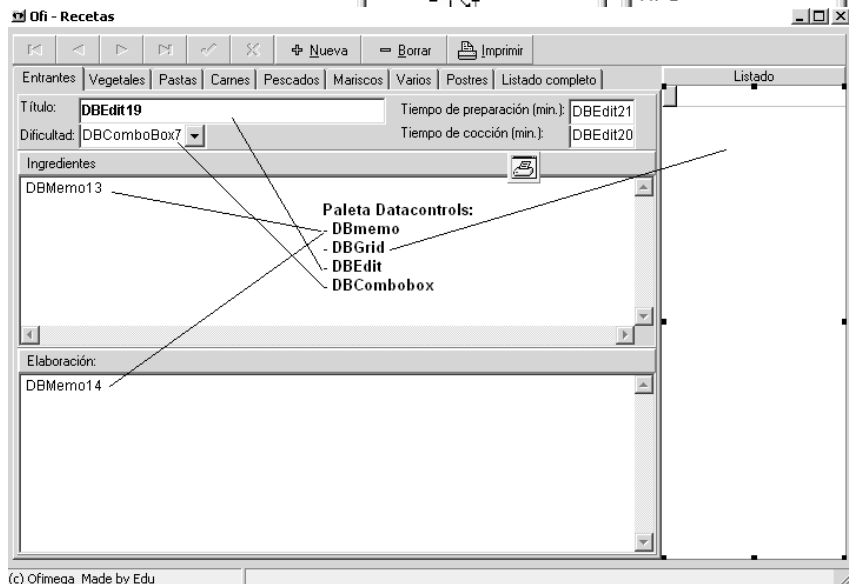
TITULO

- Añadir un nuevo formulario sólo para datos (Módulo de datos) del menú: *New – Form – Datamodule*. Llamado **DM**
- Añadir al datamodule DM, los componentes: **Table1** (o ClientDataset con el nombre *Table1*) y el componente **Datasource1** cambiando su propiedad *Dataset: Table1*
- Asignar a la table1 la propiedad (*TableName* o *FileName*) nombre del archivo *Recetasdbf* o *recetas.xml* según la base de datos.
- En modo DBF: añadir a la *Table1* todos los campos (botón derecho)
- En el formulario principal (*File – Project manager – Form1*) asignar la propiedad *Name: Receta*.
- Modificar el formulario como en la figura inferior:



- En él añadir:
  - Una *Toolbar* y una *Statusbar* (Win32)
  - Un panel alineado a la derecha con un *Splitter* (Additional) a su izquierda.
  - Un *Pagecontrol* (win32) a la izquierda alineado al cliente.

Sobre el **Pagecontrol1**, pulsar el botón **derecho** del mouse y escoger: **New page** y crear 9 solapas (TabSheets). Asignar a la propiedad *caption* de cada TabSheet los nombres que se muestran:  
 Entrantes, Vegetales, Pastas, carnes, etc...  
 En cada pestaña, añadir los componentes: 3 dbedits, 1 DBcombobox y 2 dbmemos como en la figura izquierda.  
 En el panel derecho añadir un componente *DBgrid* Align al cliente.  
 Asignar las propiedades para cada componente DB, el *Datasource: Datasource1* y el *Datafiled* el campo que le corresponda.  
 En la barra de herramientas toolbar con el botón derecho crear 3 nuevos botones y añadir un *DBNavigator*.



## CODIGO FUENTE del programa de recetas:

### var

Receta: TReceta; //->nombre de la ventana o formulario  
var **filtro**: string; //-> variable de texto

### procedure TReceta.PageControlChange(Sender: TObject);

```
begin
DM.table1.filtered:=true;
Edit1.text:='';
DM.Table1.CancelRange;
if PageControl1.activepage =tabsheet1 then
Filtro:=''Entrantes'';
if PageControl1.activepage =tabsheet2 then
Filtro:=''Vegetales'';
if PageControl1.activepage =tabsheet3 then
Filtro:=''Carnes'';
if PageControl1.activepage =tabsheet4 then
Filtro:=''Pescados'';
if PageControl1.activepage =tabsheet5 then
Filtro:=''Mariscos'';
if PageControl1.activepage =tabsheet6 then Filtro:=''Varios'';
if PageControl1.activepage =tabsheet7 then Filtro:=''Postres'';
if PageControl1.activepage =tabsheet8 then Filtro:=''Pastas'';
if PageControl1.activepage =tabsheet9 then
Begin
DM.table1.filtered:=False;
Filtro:=''Completo'';
end;
DM.table1.filter:='Tipo'+Filtro;
Panellistado.caption:='Listado '+filtro;
contar('Recetas');
end;
```

### procedure contar(Sender: TObject);

```
begin
Receta.StatusBar1.panels[1].text:=inttostr(dm.Table1.RecordCount);
end;
```

### procedure TReceta.SpeedButton1Click(Sender: TObject);

```
begin
if MessageDlg('Nueva receta?',mtConfirmation,[mbYes, mbNo], 0)= id_Yes then
begin
DM.table1.insert;
if PageControl1.activepage =tabsheet1 then DM.Table1TIPO.value:='Entrantes';
if PageControl1.activepage =tabsheet2 then DM.Table1tipo.value:='Vegetales';
if PageControl1.activepage =tabsheet3 then DM.Table1tipo.value:='Carnes';
if PageControl1.activepage =tabsheet4 then DM.Table1tipo.value:='Pescados';
if PageControl1.activepage =tabsheet5 then DM.Table1tipo.value:='Mariscos';
if PageControl1.activepage =tabsheet6 then DM.Table1tipo.value:='Varios';
if PageControl1.activepage =tabsheet7 then DM.Table1tipo.value:='Postres';
if PageControl1.activepage =tabsheet8 then DM.Table1tipo.value:='Pastas';
DM.Table1TITULO.value:='';
end;
end;
```

### procedure TReceta.SpeedButton2Click(Sender: TObject);

```
begin
if MessageDlg('Borrar receta?',mtConfirmation,[mbYes, mbNo], 0)= id_Yes then DM.table1.delete;
end;
```

### Impresión:

- Imprimir el formulario (ventana) : Añadir componente: Printdialog de la paleta Dialogs y en el botón imprimir escribir el código: **if PrintDialog1.execute then print;**
- Imprimir informe de datos: Utilizar el diseñador de informes Rave report o Qreport explicados en el siguiente tema.

Guarda el proyecto con el nombre: **Recetas** (también la ventana/unidad recetas1 y la ventana/datamodule dm)

```
procedure TReceta.TabSheet9Enter(Sender...
begin
contar('recetas');
end;
```

```
procedure TReceta.PageControl1Enter(Sender...
begin
contar('Recetas');
end;
```

```
procedure TReceta.contar(Sender:string);
begin
Receta.StatusBar1.panels[1].text:=sender+ ' :
' + inttostr(dm.Table1.RecordCount);
end;
```

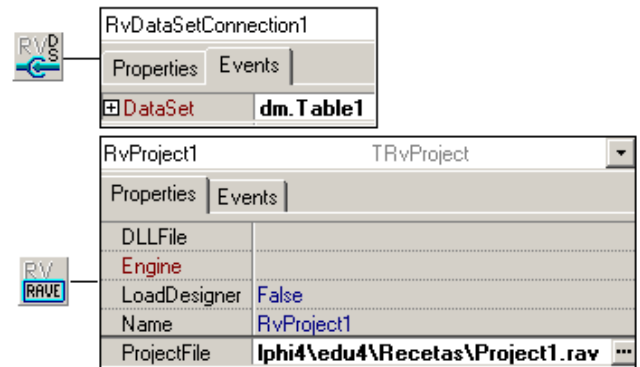
```
procedure TReceta.Edit1Change(Sender...
Var
Campobusca: string;
begin
Campobusca := 'TITULO';
DM.table1.SetRangeStart;
DM.Table1.FieldName(Campobusca).AsString:=Edit1.Text;
DM.Table1.ApplyRange;
end;
```

Ofimega

## INFORMES de impresión de datos (reports)

### a) Crear un informe con RaveReports

- Añadir al programa de rectas, los componetes: RvDataSetConnection y RvProject  
Con las propiedades de la figura...->
- En el botón para imprimir, añadir:  
DM.TABLE1.refresh;  
RvProject1.execute;
- Pulsa doble clic sobre el objeto **RvProject** para abrir el programa generador de informes: Rave Report.

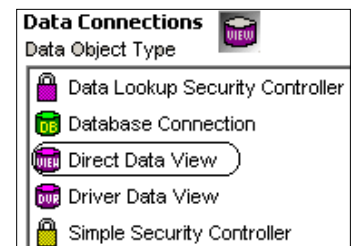


Pulsa el botón: New data Object, escoge: **Direct data View**

Puedes añadir las regiones, bandas y componentes Datatext de la figura, y que se encuentran en la paleta: Report.

Recuerda especificar la propiedad Datafield a cada DataText.

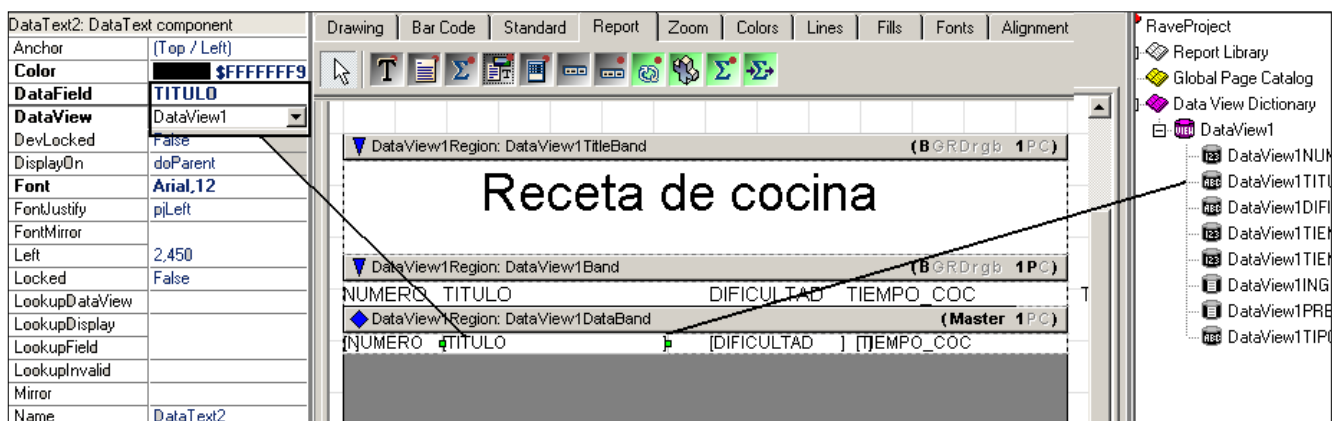
Guarda al finalizar el informe con el nombre: **Project1.rav** en la misma carpeta donde tienes tu proyecto de Delphi.



### b) Crear un Informe con FastReport .

Incorporado en la versión XE7. Tan sencillo como soltar en el formulario el componente *fxreport1* y un *DataReport*.

Pulsar doble clic encima del *Fxreport1* para diseñar el informe. Elegir del menú: Report – Data para escoger el dataset de datos.



**Truco: Cambiar el formulario de vista previa a español:** Busca en la carpeta: Rave5\Lib los formularios:

RPFFormPreview.FRM y RPFFormSetup.FRM y ábrelos desde Delphi.

**Opcional:** Puedes añadir también un componente: **RVSystem** al formulario de recetas y que enlazará con la propiedad **Engine** del objeto RaveReport. Este componente te permitirá especificar los parámetros de impresión y vista previa deseados.



### c) Crear un Informe con QReport.

El paquete Qreport (quickreport). Se incorporó en la versión Delphi 7: **DCLQRT70.BPL**

En otras versiones: Descargar de <http://www.quickreport.co.uk/>

Crea un formulario nuevo (New form) y sueltas en el el componente de la paleta Qreport: **QuickRep**

Dentro de este, pon dos QRBand y cambia el tipo de banda a: Page header y Detail respectivamente.

Suelta en estas bandas los objetos: QRLabel y QrDBText.

Para ejecutarlo escribe el código:

```
if(Application.MessageBox('¿Imprimir esta receta?', 'Imprimir registro - Print', MB_YESNO) = IDYES)
then Informe.QuickReport1.Print; //-> ó Informe.QuickReport1.Preview //-> para vista previa
```

**Truco:** Para cambiar la ventana a español, abre el formulario: QRPrev.DFM en la carpeta: DELPHI\LIB

## LENGUAJE SQL EN DELPHI

Manipulación de Datos mediante lenguaje SQL: SELECT, INSERT, UPDATE, DELETE

**SELECT:** Extrae datos de una o más tablas indicando los campos (o columnas) separados por comas.

El asterisco (\*) sustituye a todos los campos

### Sintaxis (fórmula):

```
SELECT {Campo1 AS NombreCampo1,
Campo2, ...CampoN }
FROM {Tabla/Vista}
[ WHERE {Condición} ]
[ LEFT JOIN OtraTabla ]
```

### Ejemplo:

```
SELECT * FROM CLIENTES
WHERE CIUDAD = "MEXICO" AND ESTADO = "DF"
```

**FROM:** Especifica la tabla o tablas (separadas por comas) de la cual se extraen los datos.

**JOIN:** Junta varias tablas a través del campo índice, que existe en las dos tablas.

**WHERE:** Filtra los registros que cumplan el criterio de condición.

Estos pueden utilizar los operadores de comparación (=, <, > ...) o predicados adicionales IN, ANY, ALL, EXISTS.

**ORDER BY:** La cláusula Order by especifica el orden de extracción.

```
SELECT * FROM PARTS ORDER BY PART_NAME ASC
```

```
SELECT * FROM PARTS WHERE PART_NO > 543
```

**GROUP BY:** Especifica cómo se agrupan las filas o registros.

**UNION:** Combina el resultado de dos o más SELECTS para convertirla en una sola tabla. (JOIN)

**INSERT:** Añade un registro a la tabla APPEND Inserta el registro al final de la tabla:

Insert permite añadir datos al mismo tiempo mediante dos métodos:

1.- Añade una fila (registro) a la tabla asignando valores a dos columnas:

```
INSERT INTO FACTURAS
( FACNO, NOMBRECLIENTE, CIUDAD, ESTADO )
VALUES ( 99, 'Winston Churchill', 'Londres', 'La
Mera Capital' )
```

2.- Especifica valores a insertar mediante el estado SELECT: este INSERT se combina con otro SELECT para copiar los renglones de la factura número dos y ponerlos en la factura número 99.

```
INSERT INTO FACTURARENGLON
(FACNO, NOMBRECLIENTE, CIUDAD, ESTADO)
SELECT 99, ITEMNO, DESCRIPTION, PRECIO_USD FROM
FACTURARENGLON WHERE FACNO = 2
```

**UPDATE:** Modifica datos de la tabla. Ej: UPDATE GOODS SET CITY = 'SANTA CRUZ' WHERE GOODS.CITY = 'SCOTTS VALLEY'

**DELETE:** Borra datos de la tabla. Ej: DELETE FROM GOODS WHERE PART\_NO = 'AA0093'

### Funciones agregadas a SQL

- SUM(), Total
- AVG(), Promedio de valores no nulos
- MIN(), mínimo valor
- MAX(), máximo valor
- COUNT(), cuenta el nº de valores según un criterio.
- De fecha: Permite extraer de una fecha: YEAR, MONTH, DAY, HOUR, MINUTE y SECOND

```
SUM( Field * 10 )
SUM( Field ) * 10
SUM( Field1 + Field2 )
```

### Substitución de variables:

En Delphi, cuando se añade una palabra precedida por un signo de dos puntos (:), la palabra es automáticamente convertida en un parámetro, el cual es agregado a la propiedad: Params

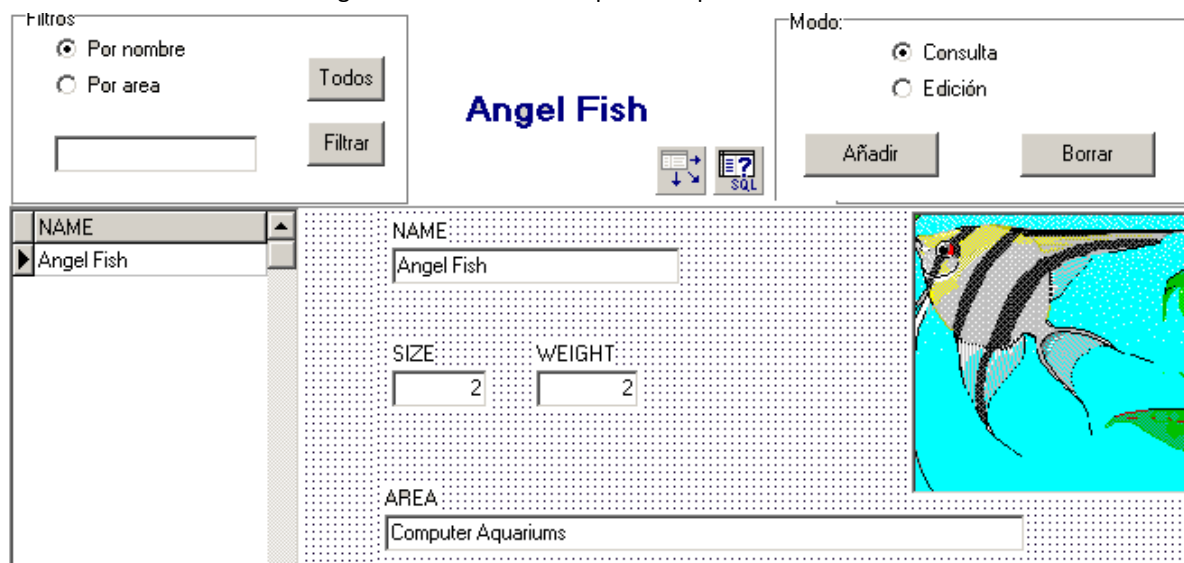
```
SELECT * FROM "CLIENTS.DBF" CLIENTES WHERE Last_Name LIKE :Apellido
```

**LIKE:** devuelve todos los registros que se parezcan a la cadena. La línea que asigna el parámetro añade un signo de porcentaje (%) al final:

```
qryClientes.ParamByName('Apellido').AsString := dlgConsulta.edtApellido.Text+'%';
```

## Ejercicio de consulta de datos con SQL

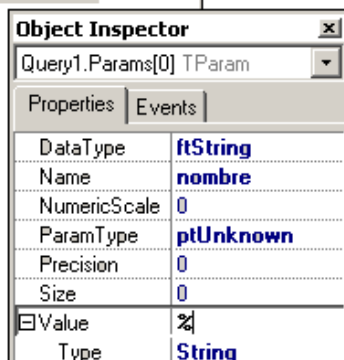
- Crea una aplicación nueva en Delphi. File ▶ New ▶ *Application*.
- De la pestaña Data Access y DBE, inserta los componentes: **Query** y **DataSource** que representan las partes "no visuales" del acceso a base de datos.
- Ahora, vaya al TQuery y cambie la propiedad "Databasename" a Dbdemos, que es un **alias** local que apunta al directorio de Demos de Delphi.
- Selecciona el **Datasource**. En la propiedad Dataset, selecciona: **Query1**
- Selecciona el Query1, haz doble-click en su propiedad: **SQL** Escribe lo siguiente:  
SELECT \* from animals where upper(NAME) like [:nombre] order by NAME
- Observe que :nombre es un parámetro. En la propiedad **Params** pulsa doble clic y pon su Datatype a string
- A continuación, vaya a la propiedad "**RequestLive**" de su Query y póngala en "**True**". Esto es importante porque SQL siempre devuelve resultados "Read-Only", o "muertos" (dead) a menos que especifiquemos que nuestro query debe ser "vivo" (live)
- Haz doble-click en el query1. A continuación verás una ventana vacía. Haz click con el boton derecho del mouse y selecciona: "**Add Fields**". Nos aparece la lista de las "columnas" (campos) disponibles. Así que simplemente presiona {Enter} para aceptar el añadir todos los campos.
- "Arrastra" desde la lista de campos del query1, hasta el formulario. Cuando "sueltes" cada campo, Delphi creará un componente TLabel y un DbEdit, listos para recibir los datos adecuados. Conecta todos los DBEdits con la propiedad Datasource al Datasource1. A asignar su datafield al campo correspondiente.



- Para ver los datos vamos a abrir el query1 (active = true). Completamos el formulario con los objetos que se muestran en la figura. En la propiedad Params del Query 1 le añadimos el valor: % para que por defecto aparezcan todos.

- En el botón **Filtrar**, añadimos el código:

```
begin
  query1.active:=false;
  Query1.Params[0].Value :=
    UpperCase(edit1.Text)+'%';
  query1.active:=true;
end;
```



- En el botón **Todos**, escribimos:

```
query1.active:=false;
Query1.Params[0].Value := '%';
query1.active:=true;
```

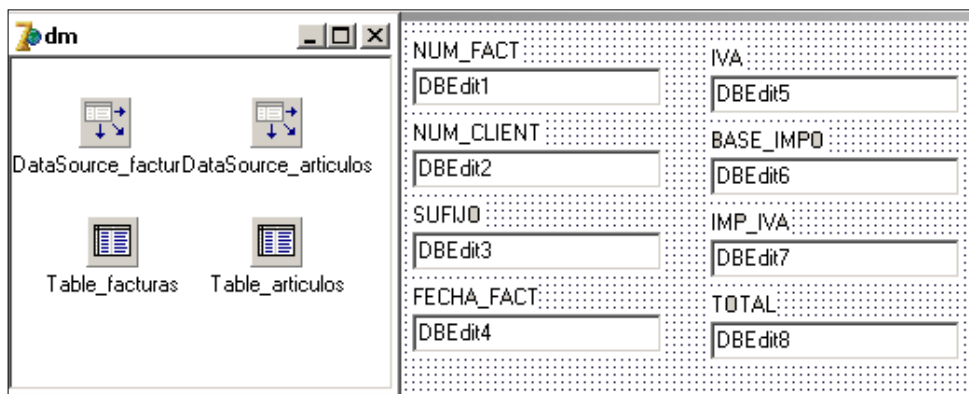
### Notas:

- ▶ **Upper** en el SQL y **Uppercase** en el procedure, convierten ambos a mayúsculas para que no haya distinción entre mayúsculas y minúsculas.
  - ▶ El botón de radio: Edición, cambia la propiedad del query1: **RequestLive** a true.
  - ▶ Los botones añadir y borrar añaden el código: **Query1.Append**; y **Query1.Delete**; respectivamente.
- Para consultar por área, añadir otro query2 con este campo como parámetro en el SQL. Con el mismo datasource, cambiar de uno a otro, según se active uno u otro botón de radio.

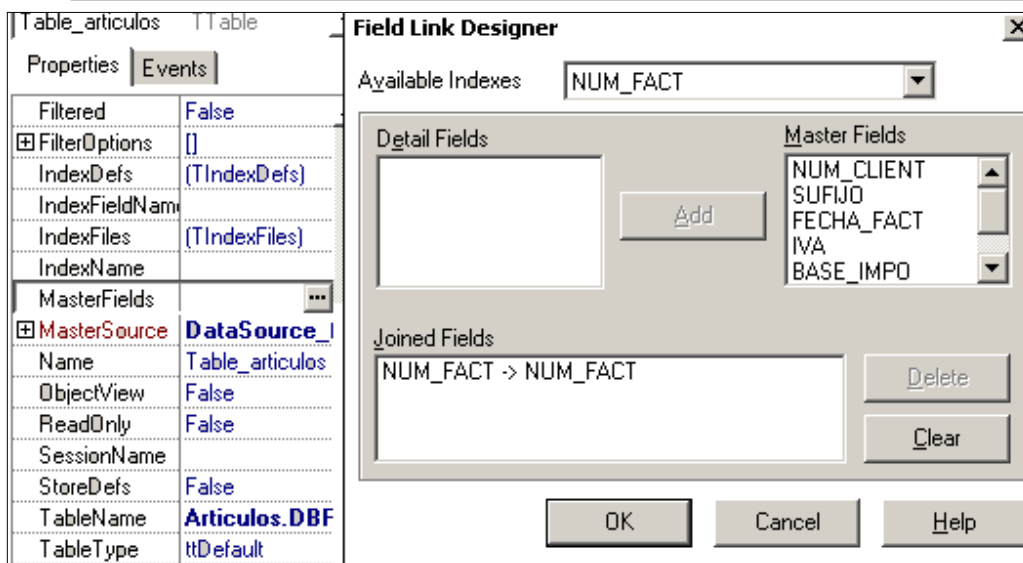
## Programa facturación Pasos básicos.

1. Crear las tablas:
2. Crear un nuevo form (New ►Form) Datamodule1, con el nombre: DM.
3. Insertar en el DM las tablas para Facturas y artículos: Table\_facturas y table\_articulos, asignarles los indexnames creados y añadir dos datasources: Datasource\_facturas y Datasource\_articulos.
4. Añadir: All fields al la table facturas y artículos.

5. Seleccionar todos los campos de tablefacturas y soltar sobre el form principal llamado: Facturación



6. Crear un DBgrid con Align al bottom y asignarle la datasource de articulos. Luego añadir todos los campos al DBGrid de articulos.



7. Para relacionar facturas con artículos debemos volver al DM y elegir en mastersource de artículos el Datasource de facturas. Luego, en el MasterFields, relacionar los campos NUM\_Fact de cada tabla. Activar ambas tablas (active=true) para ver el funcionamiento.

8. Añadir en el form principal un DBNavigator para las facturas.

### 9. Crear el campos calculados:

Asignar a un campo existente un calculado: Para el importe: En la table\_articulos poner en el evento: **onCalcfileds:**

**Table\_articulosimporte.Value:=Table\_articuloscant.Value\*Table\_articulosprecio.Value;**

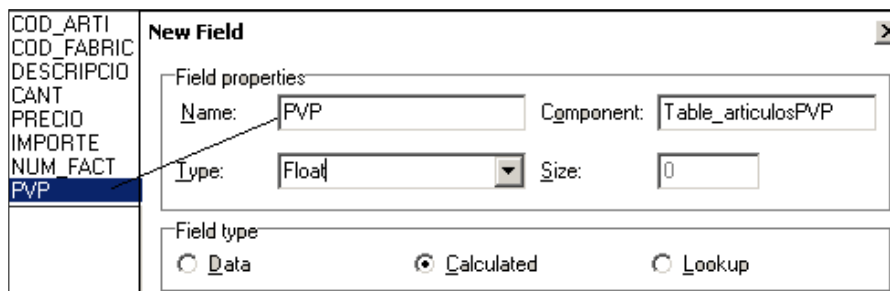
Luego poner en el campo Importe su propiedad: **FieldKind: fkCalculated**

Crear un campo calculado nuevo:

llamado PVP (como referencia):

En la table\_articulos anadir un New field... del tipo: Flota y calculado.

Añadir al evento: On CalcFileds:



**Table\_articulosPVP.Value:=Table\_articulosprecio.Value\*(1+Table\_facturasiva.Value/100);**

10. Crear un botón: Calcular para calcular la base imponible.

**Procedure TForm1.Calcular1Click(Sender: TObject);**

```

var
baseimpo:double;           //-> variable temporal de baseimponible
num_rec:pointer;          //-> puntero de registro
begin
with dm do
begin
table_articulos.disablecontrols;           //-> evita cambiar registro el usuario
num_rec:=table_articulos.GetBookmark;      //-> coge num de registro
table_articulos.first; //->sitúa el puntero de registros al principio de la tabla
while not table_articulos.eof do {mientras no llegue al End Of File...(final de la tabla) haz el bucle}
begin
baseimpo := baseimpo + table_articulosimporte.AsFloat; {suma la cantidad anterior }
table_articulos.next;           {salta al siguiente campo}
end;           {repite el bucle}
table_articulos.enablecontrols;
table_facturas.Edit;
Table_facturasBASE_IMPO.Value:=baseimpo;
Table_facturasIMP_IVA.Value:=baseimpo*table_facturasiva.value/100;           //->importe del IVA
Table_facturasTOTAL.Value:=baseimpo*(1+table_facturasiva.value/100);           //->Total factura
table_facturas.post;
table_articulos.GotoBookmark(num_rec);           //-> vuelve al registro que estaba
end;
end;

```

**Cálculo automático:** Por último, para calcular automático, añadir al evento del la table\_articulos: AfterPost:  
Table\_articulosAfterPost: **Form1.Calcular1Click(nil);** //→ llama al procedure sin sender

**Mejoras:**

- Añadir el componente DBCombobox al % de IVA con los items: 0,7,12,16. En el campo IVA del DataModule cambiar su DefaultExpression = 16.
- Para autoincrementar el número de factura: Como no existe del tipo Autonumerico en Dbase, no vale con cambiar el Autogeneratevalue a AutoInc sino escribir el código:

Var pública en el Data module: Numero:single:

```

Table_facturasBeforeInsert:           Table_facturas.last;
                                       Numero:=Table_facturasNUM_FACT.value;
Table_facturasAfterInsert:           Table_facturasNUM_FACT.value:=numero+1;

```

**Control de registros de una base de datos.Navegador manual.**

Añadir al formulario los botones: Adelante , atrás primero último, añadir y borrar.



```

procedure TForm1.primeroClick(Sender: TObject);
begin
with table1 do
begin
if sender=primero then first
else if sender=anterior then prior
else if sender=siguiente then next
else if sender=ultimo then last
else if sender=nueva then
begin
if(Application.MessageBox('¿Crear nuevo registro?', 'Agregar registro', MB_YESNO) = IDYES) then
begin
table1.append; //-> Inserta
table1.Post; //-> Guarda
end;
end
else if sender=borrar then
begin
if(Application.MessageBox('¿Confirma que desea eliminar este registro?', 'Confirmar eliminación de registro',
MB_YESNO) = IDYES) then delete;
end;
Table1.Refresh;
StatusBar1.panels[1].text:='Total registros: '+ inttostr(Table1.RecordCount);
end; Table_facturasFECHA_FACT.value:=date;

```

## Nivel Avanzado. Control de Base de datos. Relacionar dos tablas (facturación)

### Cómo crear una consulta (Query) SQL para relacionar dos tablas por un campo en común:

Véase lenguaje SQL en este manual

Pasos previos para el ejercicio:

- Crear un directorio C:\FACTU ( o similar)
- Utilizar la herramienta Data Base Desktop para crear previamente las tablas de bases de datos: FACTURA.DBF y ARTI.DBF cuyo campos mínimos sean:

FACTURA."NUM\_FACT",  
FACTURA."FECHA\_FAC",  
FACTURA."COD\_CLIENT",  
FACTURA."IVA",

ARTI."REFER",  
ARTI."DESCRIPCIO",  
ARTI."CANT",  
ARTI."DESC",  
ARTI."PRECIO",  
ARTI."NUM\_FACT",

Esta acción es automática con File - New - DataBase Form.

- Es conveniente usar un data module para las Datasources y las queries.
- Relacionamos las facturas con sus artículos por el campo num\_fact común a los dos.
- Recordar que: Query1NUM\_FACT: TFloatField;

### Ejercicio. Facturación

Nota: En esta aplicación es posible añadir y modificar datos en factura, pero no en artículos. Para ello es necesario crear un botón para añadir artículo, que utiliza la table 2.  
Las consultas SQL no permiten modificaciones en la tabla relacionada si no tenemos **RequestLive** a true. Por eso trabajaremos directamente sobre las tablas en página siguiente.

On Form create:  
Query1.Open;  
Query2.Open;

#### Líneas SQL para la Query1. :

```
Select
FACTURA."NUM_FACT",
FACTURA."NUM_PRES",
FACTURA."FECHA_FAC",
FACTURA."COD_CLIENT",
FACTURA."BASE_IMPON",
FACTURA."IVA",
FACTURA."TOT_FACT"
From "C:\FACTU\FACTURA.DBF"
As FACTURA
```

#### Líneas SQL para la Query2:

```
Select
ARTI."REFER",
ARTI."DESCRIPCIO",
ARTI."CANT",
ARTI."DESC",
ARTI."PRECIO",
ARTI."IMPORTE",
ARTI."PVP"
From "C:\FACTU\ARTI.DBF" As ARTI
Where "C:\PRG\FACTU\ARTI.DBF"."NUM_FACT" =:"NUM_FACT"
```

**Método 2:** En esta otra ventana se utilizan directamente tres tablas: Facturas – Artículos y Clientes.

Botones de nuevo y borrar artículos:

```
procedure TFormfactu.nuevoClick -> table2.insert;
procedure TFormfactu.borraartClick -> table2.delete;
```

Table1: Facturas  
Table2: Artículos  
Table3: Clientes  
Table4: Empresa

The screenshot shows the 'Win Factu' application window with the title 'Añadir factura'. The interface includes a menu bar (Datos, Consultas, Herramientas, Ayuda) and a toolbar with buttons for 'Nuevo art.' and 'Borra art.'. The main form contains input fields for 'Núm. factura' (1), 'Código cliente' (0), 'Nombre' (90), 'Fecha factura' (24/08/98), '% IVA' (16), 'Euro' (0), 'Dirección', 'Cod. Postal' (43840), and 'Población' (SALOU). Below the form is a table with columns: Código, Descripción, Cant., Precio, Importe, P.V.P., and Nu. The table contains three rows of data. At the bottom, there are summary fields for 'Base Imponible', '% IVA', 'Importe IVA', 'Total Factura', and 'Total Euros'. A 'Botón calcular' is located at the bottom right.

The 'Field Link Designer' dialog box is shown. It has a dropdown for 'Available Indexes' set to 'NUM\_FACT'. There are two lists: 'Detail Fields' and 'Master Fields'. 'NUM\_FACT' is in both. There is an 'Add' button between them. Below are 'Joined Fields' and buttons for 'Delete' and 'Clear'. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

Para realizar la unión necesitamos un campo índice común (NUM\_FACT) para las dos tablas, y en la tabla de artículos, la propiedad master fields (campo índice maestro) se realiza la unión de la figura.

Pulsar el botón Add y en Joined Fields (campos de unión) aparecerá:

NUM\_FACT->NUM\_FACT

Procedure para añadir una nueva factura:  
Antes De estos procedures crear la Variable numero:DOUBLE (del tipo flotante)  
1º saber cuál es la última factura

```
procedure TFormfactu.Table1BeforeInsert
begin
table1.last;
numero:=table1num_fact.value;
end;
```

2º sumar 1 al número de la última factura y ponérselo a la nueva, también preparar la fecha de la factura con el día de hoy:

```
procedure TFormfactu.Table1AfterInsert
begin
table1num_fact.value:=numero+1;
table1fecha_fac.value:=date;
table1iva.value:=16;
estado.caption:='Añadida la factura num. ' +
floattostr(numero);
end;
```

#### Cómo borrar la última factura

```
procedure TFormfactu.EliminarClick
begin
table1.last;
numero:=table1num_fact.value;
if MessageDlg('¿Eliminar la factura ' + floattostr(numero)+
?', mtInformation, [mbYes, mbNo], 0) = mrYes
then table1.delete;
end;
```

```
procedure TFormfactu.Table1AfterDelete;
begin
ShowMessage('Eliminada la factura num. ' +
floattostr(numero));
end;
```

## Ejercicio Facturación. Cómo crear un campo calculado para el IMPORTE:

Supongamos que tenemos una tabla cuya Table Name sea Articulos.dbf y que posee entre otro el campo CANT (cantidad) y el campo PRECIO:

- Pulsa doble clic encima de la tabla para activar el Fields Editor.
- Pulsa el botón derecho encima y elige añadir campos (coge todos). De esta manera se declaran en la unidad PAS todos los campos y sus tipos.
- A continuación elige New Field. Nuevo campo cuyo nombre será IMPORT, del tipo FLOAT y CALCULATED. Termina con OK.
- Pulsa doble clic en el evento ONCALCFIELDS de la tabla. Debemos introducir una fórmula para el campo.
- Presta atención al nombre que tiene los campos en el apartado TYPE de la unidad PAS y escribe el valor:  
Table2IMPORT.VALUE:=Table2CANT.VALUE\*Table2PRECIO.VALUE;
- Para probar su funcionamiento, puedes crear 3 componentes DBEDIT para la cantidad, precio e importe. Recuerda poner en la propiedad **Passwordchart #0** para obligar a que sea un dato numérico.

También es posible crear campos calculados uniendo campos de texto. Ejemplo:

CustomersCityStateZip.Value := CustomersCity.Value + ',' + CustomersState.Value + ' ' + CustomersZip.Value;

```
Campo calculado importe (En la tabla 2 On calcfields)
procedure TFormfactu.Table2CalcFields(DataSet: TDataSet);
begin
  table2importe.value:=table2cant.value*table2precio.value;
  table2pvp.value:= table2importe.value*(1+table1iva.value/100);
end;
```

**Cómo calcular la base imponible:** La base imponible no es un campo calculado, pues es el resultado de la suma de varios campos de varios registros mientras un campo calculado se obtiene de operar con campos de un mismo registro.

### Método 1. Base imponible (suma de varios registros) como consulta SQL

Para ello se debe crear una consulta SQL e incluir las siguientes líneas de comando Strings. (Este método no es aconsejado en este ejercicio pues las consultas SQL no permiten modificaciones sin RequestLive a true)

```
SELECT BASE_IMPON FROM FACTURAS
SUM(IMPORTE) FROM ARTICULOS WHERE NUM_FACT=:NUMERO
```

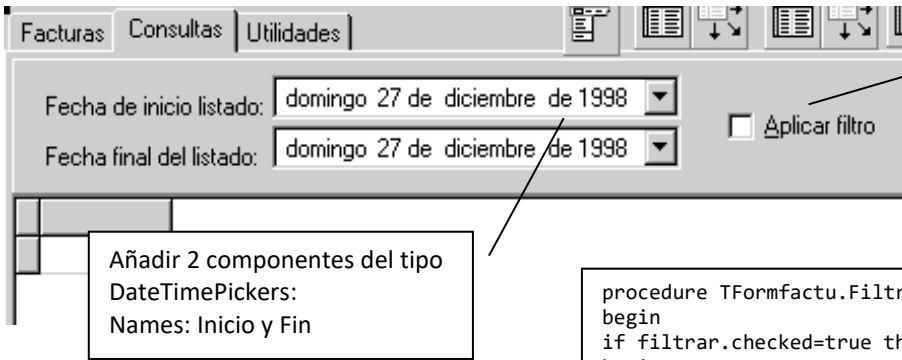
**Método 2: Base imponible con un bucle While** que suma consecutivamente y uno a uno cada registro y lo almacene en una variable numérica.

Como este método es más lento (pero más flexible) es aconsejable crear un botón llamado "Calcular" cuyo procedimiento al hacer clic sea el siguiente.

La visualización de la base imponible debe ser en un componente Label no Edit pues es más fácil y además no debe ser modificable.

```
procedure TFormfactu.CalcularClick(Sender: TObject);
var
  baseimpo: Real; {crea la variable numérica real donde se almacenará el resultado de la suma}
  impiva: Real;
begin
  baseimpo:=0.0;
  impiva:=0.0;
  table2.disablecontrols;
  table2.first; {sitúa el puntero de registros al principio de la tabla}
  while not table2.eof do {mientras no llegue al End Of File...(final de la tabla) haz el bucle}
  begin
    baseimpo := baseimpo + table2importe.AsFloat; {suma la cantidad anterior más la de este campo}
    table2.next; {salta al siguiente campo}
  end;{repite el bucle}
  table2.enablecontrols;
  impiva:=baseimpo*table1iva.value;
  labelbaseimpo.Caption :=Floattostr(baseimpo);
end;
```

**Ficha Listado de facturas. Crear un filtro entre dos fechas.**



CheckBox: Filtrar

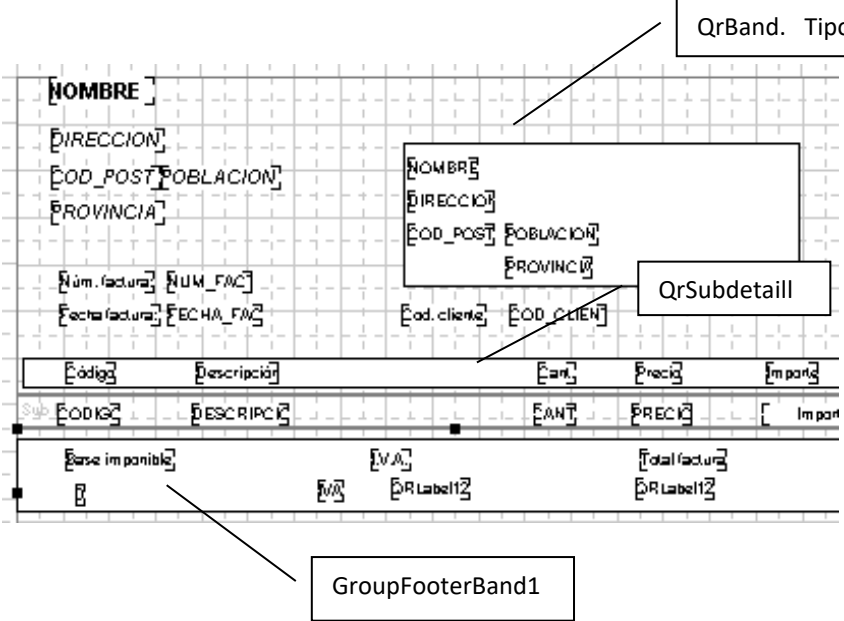
Añadir 2 componentes del tipo DateTimePickers:  
Names: Inicio y Fin

Crear un componente tabla y un datasource:  
  
Propiedades de la tabla:  
Name: Consulfactu  
Table name: Factura.dbf  
  
Propiedades del Datasource:  
Dataset: Consulfactu

```
procedure TFormfactu.FiltrarClick(Sender: TObject);
begin
if filtrar.checked=true then
begin
table1.active:=false;
consulfactu.active:=true;
consulfactu.IndexFieldNames := 'Fecha_fac';
with consulfactu do
begin
SetRangeStart; { inicio indice }
FieldByName('Fecha_fac').AsString :=
datetostr(inicio.date);
SetRangeEnd; { final indice }
FieldByName('Fecha_fac').AsString :=
datetostr(fin.date);
ApplyRange; { establece el rango al Dataset }
end;
end
else
begin
{no hay que poner filtrar.checked:=false;}
consulfactu.CancelRange;
consulfactu.active:=false;
table1.active:=true;
end;
end;
```

**Impresión de la factura:**

1. Crear una nueva ventana QuikReport1
2. Añadir los apartados: QrBand, QrSubdetaill y GroupFooterBand1
3. Añadir los componentes: QRLabel para etiquetas o títulos y QRDbtext para los campos
4. Aplicar un filtro para que sólo se imprima una factura



Al pulsar el botón Imprimir:

```
procedure TFormfactu.SpeedButton2Click(Sender: TObject);
var
textofiltro:string;
begin
textofiltro:='num_fact'+floattostr(table1num_fa
ct.value);
table1.filter:=textofiltro;
table1.filtered:=true;
quickreport1.preview;
table1.filtered:=false;
end;
```

Por último: Crear el programa de Instalación:  
Para ello utilizar el programa anexo InstallShlyd y seguir los pasos para crear el disco de instalación del programa. Recuerda que también debes añadir a los archivos de instalación el control de Base de datos DBE

## Código fuente base de ejemplo para programa de facturación:

```
unit factu1;

interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls, Buttons, ComCtrls, ExtCtrls, Menus,
DBTables, DB, DBGrids, Mask, DBCtrls, dialogs, math, Grids;

var
    Formfactu: TFormfactu;
    numero:double;
    baseimpo: Real;

uses empresa1, CLIENTES, Imprime1, Acerca;

procedure TFormfactu.FormCreate(Sender: TObject);
begin
    table1.open;
    table2.open;
    table3.open;
    table4.open;
    table1.last;

    procedure TFormfactu.Table1AfterInsert
    begin
        table1num_fact.value:=numero+1;
        table1fecha_fac.value:=date;
        table1iva.value:=16;
        estado.caption:='Añadida la factura num. ' +
        floattostr(numero);
    end;

    procedure TFormfactu.Table1BeforeInsert
    begin
        table1.last;
        numero:=table1num_fact.value;
    end;

    procedure TFormfactu.Table1AfterDelete
    begin
        estado.caption:='Eliminada la factura num. ' +
        floattostr(numero);
        ShowMessage('Eliminada la factura num. ' +
        floattostr(numero));
    end;

    procedure TFormfactu.EliminarClick(Sender...
    begin
        table1.last;
        numero:=table1num_fact.value;
        table2.disablecontrols;
        table1.disablecontrols;
        if MessageDlg('¿Eliminar la factura nº ' +
        floattostr(numero)+ '?', mtInformation, [mbYes,
        mbNo], 0) = mrYes
        then
        begin
            table1.delete;
            table2.first;
            while not table2.eof do
            begin
                table2.delete;
                table2.next;
            end;
        end;

        table2.last;
        table2.delete;
    end;
    table1.enablecontrols;
    table2.enablecontrols;
end;

procedure TFormfactu.nuevoClick(Sender: TObject);
begin
    table2.insert;
end;

procedure TFormfactu.borraartClick(Sender: TObject);
begin
    table2.delete;
end;

procedure TFormfactu.Table2CalcFields
begin
    table2importe.value:=table2cant.value*table2precio.v
    alue;
    table2pvp.value:=
    table2importe.value*(1+table1iva.value/100);
end;

procedure TFormfactu.calcula1Click(Sender: TObject);
var
    impiva: Real;
    totfact:Real;
    toteuros:Real;
begin
    baseimpo:=0.0;
    impiva:=0.0;
    totfact:=0.0;
    toteuros:=0.0;
    table2.disablecontrols;
    table2.open;
    table2.first;
    while not table2.eof do
    begin
        baseimpo := baseimpo + table2importe.AsFloat;
        table2.next;
    end;
    {TFormfactu.Table1CalcFields;}
    {table1base_impon.value:=baseimpo;}
    table2.enablecontrols;
    impiva:=baseimpo*table1iva.value/100;
    labelbaseimpo.caption:=Floattostr(baseimpo);
    labelimpiva.caption:=Floattostr(impiva);
    totfact:=baseimpo+impiva;
    {if totfact>0 then
    toteuros:=totfact/table1euro.value;}
    labeltotal.caption:=Floattostr(totfact);
    labeleuros.caption:=Floattostr(toteuros);
end;
```

## Filtros y búsquedas en tablas

### Búsquedas blandas desde un edit o Dbcombo.

```
{el Edit se llama busca y la variable Campobusca: String;}
procedure TFichero.buscaChange(Sender: TObject);
begin
  Campobusca := 'Apellidos';
  table1.SetRangeStart;
  Table1.FieldName(Campobusca).AsString:=busca.Text;
  Table1.ApplyRange;
end;
```

### Búsquedas duras

```
Table1.Open; // abrir la tabla
Table1.SetKey; // Ponemos el DataSet en situación dsSetKey
Table1.FieldName('Autor').AsString := 'Borges';// Damos valor al campo
Table1.GotoNearest; // Nos posicionamos en ese valor o el más cercano
showmessage(table1.FieldName('Autor').AsString);
Table1.close;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  temp:integer;
  opcion:TLocateOptions;
begin
  Table1.Locate('NOMBRE',Edit1.Text,[]);
end;
```

Los parámetros del método **Locate()**, son;

Locate("Columna de búsqueda" ,"Dato a buscar" , [loCaseInsensitive, loPartialKey]);

cuando opción = loCaseInsensitive β ignora diferencia entre mayúsculas y minúsculas

cuando opción = loPartialKey β resultado parcial, ejemplo si se busca "ju" se posesiona en el primer renglón donde ese campo empiece con "ju", ejemplo Juan, juvenil, etc.

### Contar el número de registros:

Se puede utilizar este ejemplo en el ejercicio anterior para calcular el número de facturas totales.

```
numero.caption:=IntToStr(table1.recordcount)+' fichas';
```

## FILTROS EN TABLAS DELPHI

Existen dos maneras de poner filtros o condiciones para una tabla, ellas son:

1.- **Interna**, En el componente **Table**, modificar las propiedades: propiedad **Filter** = condición y propiedad **Filtered** = true

Ejemplos para la tabla de clientes:

```
ClaveCliente < 3 , Ciudad <> "Tarragona" , ClaveCliente > 2 and Ciudad = 'Tarragona'
```

2.- **Externa**, es el usuario quien construye el filtro o condición, usando un componente Edit para el filtro y un botón de ordenes (OK) para que se ejecute y despliegue la tabla filtrada, el código en dicho botón es;

```
onclickbutton1()
```

```
Table1.Filter = Edit1.Text;
```

```
Table1.Filtered = true;
```

Ejemplo:

```
procedure TForm1.botonfiltroClick(Sender: TObject);
begin
  if botonfiltro.Down=true then
  begin
    CalcularClick(Sender);
    table1.Filter:='PROVINCIA =' +StaticTextaProv.Caption;
    table1.Filtered:=true;
  end
  else table1.Filtered:=false;
```

## Base datos tipo cliente - servidor Delphi: Interbase – Firebird - Mysql

En lugar de usar tablas locales tipo BD o mdb, utilizaremos una base de datos del tipo servidor como **Interbase** (o su equivalente gratuita **Firebird**):

1. Instalar **Interbase** o **Firebird** en el PC: <https://downloads.embarcadero.com/free/interbase>
2. Crear la base de datos con IBconsole o con IBExpert

### Herramientas visuales de Interbase: IBConsole:

El primer paso para utilizar **IBConsole** es registrar el servidor Interbase y conectar a servidores locales y remotos.

**Para registrar un servidor:** seleccionar del menú **Server | Register ...** o **Create...** lo que mostrará la ventana para conectarse a un servidor local o remoto.

Indicar si el servidor es local o remoto. Por defecto es local. En el caso que sea remoto, indicar el nombre del servidor y el protocolo de red a utilizar. Si es TCP/IP se puede indicar la dirección IP en lugar del nombre.

Si se quiere registrar el servidor...

- o Indicar un nombre de alias. En el caso de servidor local, el nombre es automático.
- o Indicar una descripción para el servidor.
- o Indicar que la información de alias debe ser almacenada.

Si se quiere conectar al servidor...

- o Indicar el nombre de usuario (**SYSDBA**)
- o Indicar la clave (**masterkey**)

La primera vez que entras a Interbase, el sistema tiene predefinido un usuario

llamado **SYSDBA** que tiene todos los derechos de acceso en Interbase. El

password de este usuario es inicialmente **masterkey**

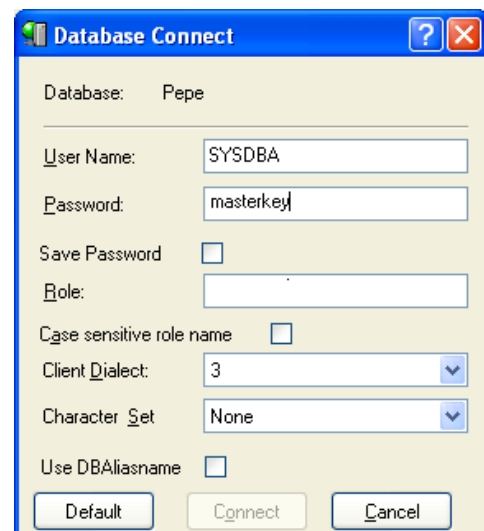
### Para crear una base de datos:

Indicar un nombre de archivo .GDB. Seleccionar del menú **Database | Create Database...** Escribe el nombre **AGENDA** (puedes ponerle extensión **GDB** o **IB**). Escoge el lugar donde guardar. Pulsa OK.

Para conectar con la base de datos:

Pulsa el botón derecho sobre la tabla y escoge: **Connect**.

Te pedirá user y contra. Rellenar como en la figura: →



### Para crear de una tabla

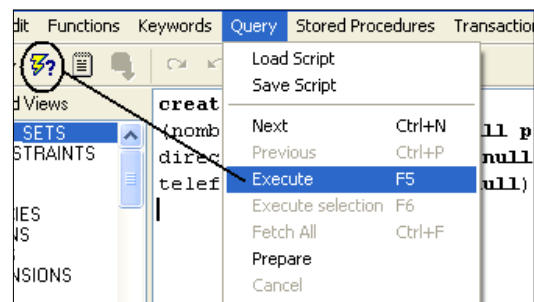
Vamos a crear una tabla desde **Interactive SQL**:

Escoge del menú de **IBConsole: Tools ▶ Interactive SQL**.

Copia este código en **Interactive SQL** y escoge del menú:

**Query ▶ Execute**

```
create table telefonos
(nombre varchar(24) not null primary key,
direccion varchar(24) not null,
telefono varchar(24) not null)
```



**Sintaxis create table:** En primer lugar indicamos el nombre de la tabla y luego, entre paréntesis y separados por comas, los campos que la componen. Para cada campo indicamos su nombre, tipo de dato e imposiciones. Todos los campos tienen las mismas imposiciones salvo el campo **nombre** que, además de la imposición **not null**, tiene la imposición **primary key**.

### Seleccionar los campos:

Escoge del menú de **Interactive SQL: Query ▶ Wizard** añade todos los campos y escribe un alias. Se generará una consulta como esta:

```
select A."NOMBRE",A."DIRECCION",A."TELEFONO" from "TELEFONOS" A
```

Escoge del menú: **Query ▶ Execute**. Se mostrará la tabla.

**Insertar datos en la tabla:**

```
Insert into telefonos(nombre , direccion, telefono)
values ('Ofimega', 'Calle Barcelona', '977350471')
```

Recuerda que el campo *nombre* no se puede duplicar por ser primary key.

**Modificar datos:**

Cambiamos el telefono con el siguiente código:

```
Update telefonos set telefono = '977351777'
```

**IBExpert**

Es una herramienta gratuita que se descarga desde la página de firebird y es un **interfaz grafico** para manipular bases de datos y tablas de internase y firebird.

1. Descargar versión personal desde: <http://ibexpert.net/ibe/index.php?n=Main.DownloadFree>

2. Registrarte para que te envíen el *password* y nombre de usuario. Introducirlos al ejecutar el programa.

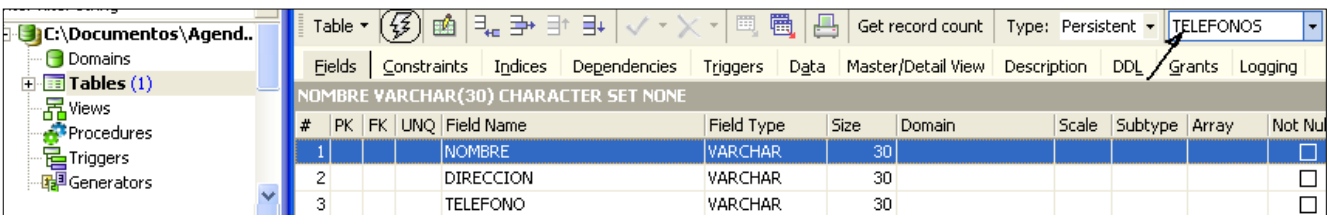
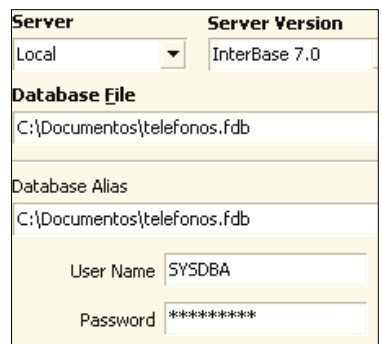
3. Crear la base de datos:  
Escoge del menú: **Database ▶ Create database...**

Introduce el nombre de usuario, password, versión de la base de datos y lugar donde se guarda. Ver figura →

4. Conectar la base de datos. Pulsa sobre el nombre de la base de datos de la lista y escoge: **Conect.**

5. Crear la tabla: Pulsa el botón dercho del mouse sobre el elemento **Tables** y escoge: **New table...**

Escribe el nombre de los campos, su tipo y tamaño en la pestaña: **Fields:**

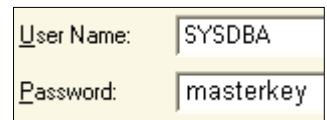


6. Escribe el nombre de la tabla TELEFONOS en el lugar indicado por la flecha de la imagen.

7. Rellenar datos: Selecciona la pestaña **Data** y escribe algunas filas.

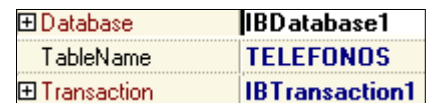
**Conexión Interbase con Delphi:**

- En una aplicación nueva, añadimos al formulario un **Datasource**, un **DBGrid** y un **DBNavigator**. Luego relacionamos el **DBnavigator** y el **DBgrid** con el **Datasource**.
- De la paleta de componentes **Internase**, añadir un **IBDatabase** y un **IBtransaction**.
- El componente **IBdatabase** se llamará **IBdatabase1** y en el componente **IBtransaction** cambia la propiedad **Defaultdatabase** por **IBDatabase1**.
- En el componente **IBDatabase1** cambia la propiedad **Database** por: C:\...\AGENDA.GDB ó AGENDA.IB. Si la base es de Firebird: AGENDA.FDB
- Cambia la propiedad **Connected = true**. Te solicitará en User name y el Password. Poner el de la imagen derecha: →



**Metodo1. Modo DBE:**

Añadir un componente **IBTable** y cambiar las propiedades como en la imagen derecha. →



**Metodo2. Sin tabla:**

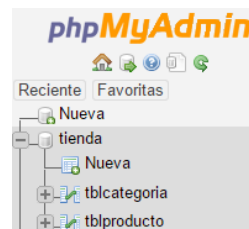
Añadir un componente **IBdataset** que provee un editor SQL. Cambia su propiedad **Database** por nuestra **Database1** y su propiedad **SelectSQL** por: `Select * from telefonos`.

Es aconsejable crear un datamodule llamada **DMconexion** con los componentes: **IBDatabase**, **IBtransaction** y **IBdataset**. Compila (F9) y guarda el proyecto con el nombre: **Agenda**.

## Conexión con Mysql

Vamos a hacer una sencilla aplicación que permita la conexión y manipulación de datos en una DB MySQL mediante ODBC o mediante driver directo por ejemplo ZeosDBO.

Utilizando **Xampp** o **Wampp** , y desde **phpMyAdmin**, debes haber creado antes la base de datos Mysql en tu pc usando: localhost como servidor de prueba.



### • Método 1: Mediante ODBC:

Es un controlador del sistema operativo que permite la conexión a datos de cualquier aplicación. MySQL posee un puente ODBC (MyODBC) que puede ser bajado del sitio en forma gratuita y viene con un instalador de fácil ejecución. Gracias a este driver podremos acceder desde un cliente Windows, con MS Access por ejemplo, a un servidor remoto MySQL

#### Instalar myodbc:

Descargamos de la página de MYSQL <http://dev.mysql.com/downloads> el conector **ODBC** para Windows.

Una vez instalado el conector ODBC debemos crear un nuevo origen de datos para la conexión:

#### Para crear un Origen de datos ODBC:

Desde el menú Inicio de Windows: Configuración ► Panel de Control ► Herramientas administrativas ► Orígenes de datos (ODBC)

Pulsa en Agregar

Indica el driver de Mysql ODBC

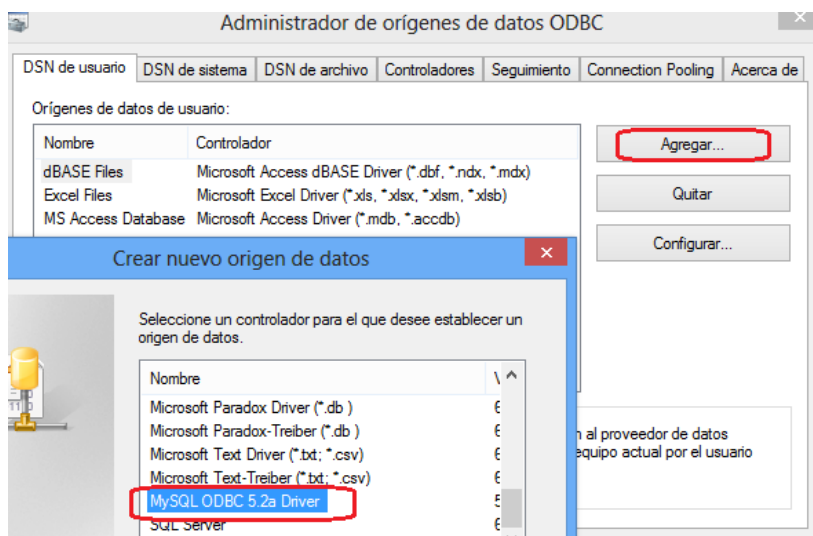
Usar como conexión local:

- Server: **Localhost**

- User: **root**

Sin contraseña

Puedes escoger la base de datos



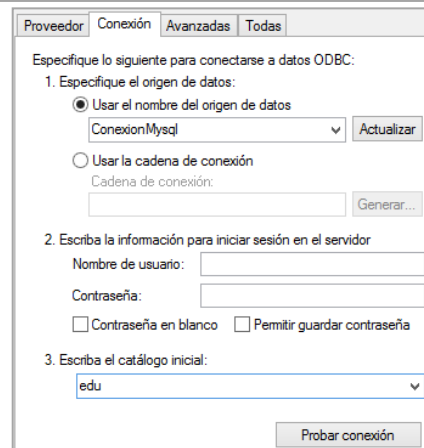
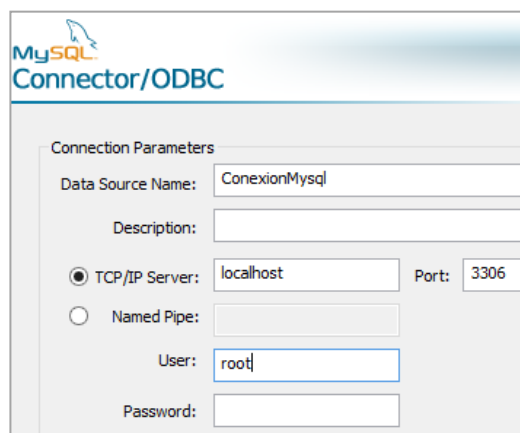
Entra en Delphi y añade un componente **ADO table**. y cambiamos su propiedad conexión string.

En la propiedad del ADO table, Table Name, indica la el nombre de la tabla: clientes.

### Crear el formulario en Delphi para usar ODBC.

En un formulario vacío coloca los siguientes componentes:

- Un **panel** con la alineación (Align) en Top.
- Dentro del panel un: **DBNavigator**
- Un botón: Name: Control, Caption: Abrir
- Un botón TbitBtn: Name:Cerrar, Kind: bkClose, Caption: Cerrar
- Un **DBGrid** con alineación (Align) AllClient
- De la pestaña ADO o dbGo, selecciona un **AdoTable** y modifica las siguientes propiedades
  - ConnectionString: Generamos la conexión con el driver ODBC a la conexión que hemos creado antes: ConexionMysql y pulsamos Probra conexión. Debe ser exitosa.En el catálogo inicial escoge la base de datos. (Edu) Pulsa aceptar. Se mostrará la cadena:  
Provider=MSDASQL.1;Persist Security Info=False;User ID=root;Data Source=telefonos
  - TableName: telefonos
- De la pestaña DataAccess seleccione el componente **DataSource**. Asocie el DataSet con la Tabla
- Actualice los DataSource del DBNavigator y DBGrid con el TDataSource.



Para comprobar que todo está correcto, modifica la propiedad Active del TTable a True y verás en el DBGrid la estructura de la agenda creada anteriormente.



Para que la misma sea funcional debes escribir algo de código:

Clica en el formulario el botón Abrir y en el editor de código escribe lo siguiente:

```
procedure TForm1.ControlClick(Sender: TObject);
begin
  If Control.Caption='Abrir' then
  begin
    Control.Caption := 'Cerrar';
    Tabla.Open;
  end
  Else
  begin
    Control.Caption := 'Abrir';
    Tabla.Close;
  end;
end;
```

```
procedure TForm1.CerrarClick(Sender: TObject);
begin
  Tabla.Close;
  Halt(0);
end;
```

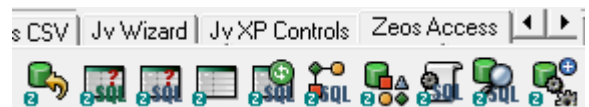
Compila el formulario y ejecuta la aplicación.

### • **Método 2: Usar componentes Zeos**

Es un conjunto de componentes de distribución gratuita y posibilitan la conectividad con distintas bases de datos: MySQL, PostgresSQL, etc.

**Instalación de Zeos:** Descarga el paquete desde el sitio de ZeosLib y descomprime el archivo .zip Copia dentro de la carpeta Archivos de programas\Borland-Embarcadero...\Source\Zeos la carpeta de los packages con la versión de delphi que corresponda. Ejecuta **ZComponentDesign.dpk**

- Copia la DLL correspondiente a su versión de MySQL que se encuentra en la carpeta Archivos de programas\Borland\Source\Zeos\lib\mysql a la carpeta de WINDOWS\SYSTEM32
- Dentro de Delphi (en environment options) agrega el directorio Zeos\packages\delphi7\build a la opción Library Path
- Desde Delphi, abre la carpeta: \...\Zeos\packages\delphi7 y compila el grupo **ZEOSDBO.BPG** o individualmente *ZCore.bpl, ZParseSql.bpl, ZPlain.bpl y ZDbc.bpl*
- Instala el componente: *ZComponentDesign.bpl* Aparecerá la paleta de componentes: *ZeosAccess* →

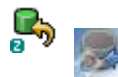


### Crear el formulario en Delphi para usar Zeos.

Como en el ejemplo anterior, inicie un nuevo proyecto en Delphi agregando al formulario todos los componentes del ejemplo ODBC menos la Ttable. (Panel, DBNavigator, botón Abrir y Cerrar, DBGrid)

Agregue un **Zconnection** modificando las siguientes propiedades:

- Database: *agenda*
- HostName: *localhost* o el nombre del equipo/dirección IP
- User: **root** (en caso de poner otro nombre proveer la clave en la propiedad Password)
- Protocol: Escoger MySQL .



Agregue un **ZTable** y modifique las siguientes propiedades:

- Connection: **ZConnection1**
- TableName: **myagenda**



#### **Para conectar directo a un PC servidor remoto:**

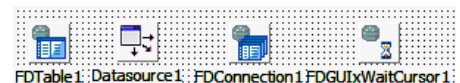
Es necesario detectar la IP del servidor (mediante una web como <http://www.hcidata.info>) el nombre de la base de datos y su password para agregarlo al controlador ODBC

Database	<b>Agenda</b>
DesignConnect	False
HostName	<b>localhost</b>
LoginPrompt	False
Name	<b>ZConnection1</b>
Password	
Port	0
Properties	(TStrings)
Protocol	<b>mssql</b>
ReadOnly	False
SQLHourGlass	False
Tag	0
TransactIsolation	tiNone
User	<b>root</b>

### • **Método 3: Mediante acceso FireDAC**

Para Delphi XE7 y superiores dipones acceso directo universal DAC:

Desde la paleta FireDAC insertar los componentes de la figura  
En FDconnection pulsa doble clic y especifica el driver: MySql



# Conexión con SQLite o Mysql a Partir de Delphi XE

## Método 1: Usar DBExpress actualizado

Las últimas versiones de RAD studio incorporan el driver MYSQL en la SQLConexion del DBExpress o de FireDAC. No obstante se deberá incorporar las librerías *libmysql.dll* que ya lo lleva el Xampp en `xampp\mysql\lib` y *dbxmys.dll* al path.

El acceso a las bases de datos MySQL o SQLite y las creación de las tablas es posible hacerlo muy fácilmente desde la pestaña **Data Explorer**.

Crear una nueva conexión y con el botón derecho encima seleccionar: **New table**.

Añadir los campos de texto: Nombre y teléfono de ancho 50 caracteres y guardar la tabla con

Arrastrar la tabla creada sobre el formulario. Aparecerán los componentes:

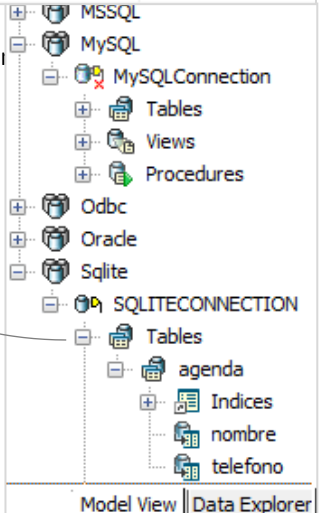
Agendatable y mSqlConection

Activa las casillas:

Conected y active.



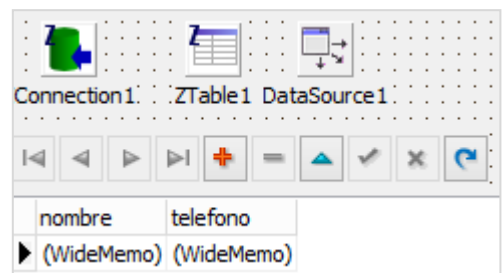
LibraryName	dbxmys.dll
LibraryNameOsx	libsqlmys.dylib
VendorLib	LIBMYSQL.dll
VendorLibWin64	libmysql.dll
VendorLibOsx	libmysqlclient.dylib
HostName	localhost
Database	tienda
User_Name	root



Añade unos componentes DBGrid, DBNavigator y Datasource al formulario

## Método2: Usar Zeos

1. Descargar la librería gratuita ZeosDBO Version 714.
2. Instalar el paquete para XE7
3. Añadir los componentes de la imagen al formulario:
4. En Zconexion asignar la database y activar conected.
5. En Ztable: Asignar a la Tablename: agenda y activar.
6. En Datasource: Poner el dataset: Ztable1
7. En el DBGrid y el DBNavegador asignar el Datasource1
8. Probar (Run ▶)

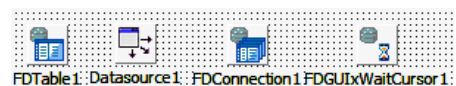


## Método3: Usar FireDAC

Para Delphi XE7 y superiores dispones acceso directo universal DAC:

Desde la paleta FireDAC insertar los componentes de la figura

En FDconexion pulsa doble clic y especifica el driver: MySQL



## Multi-Device Aplicaciones multiplataforma / entorno FireMonkey desde XE5

### Configurar un dispositivo ANDROID

Las aplicaciones Android se pueden probar en un emulador Android en el PC o en un dispositivo físico Android (teléfono o tablet).

#### Método 1:

Para probar la emulación de Android se debe instalar las **android SDK manager**. Desde la versión XE ya se incorpora un SDK manager que se puede actualizar desde: *Tools – Options – Environment – SDK Manager*.

Añadir las rutas para las SDK path. Normalmente, SDK en:

```
C:\Users\Public\Documents\RAD Studio\12.0\PlatformSDKs\adt-bundle-windows-x86-20130522\sdk
```

Y las NDK en:

```
C:\Users\Public\Documents\RAD Studio\12.0\PlatformSDKs\android-ndk-r8e
```

Desde el SDK manager, escoge del menú: **Tools – Manage Avds....**

Android Virtual Devices para crear diversos emuladores Android. (Estos emuladores son pesados).

#### Método 2:

Otro modo de probar las aplicaciones es compilar directamente sobre el dispositivo android conectado. Si es necesario, bajarse el **ABD driver universal USB** de la página: <http://adbdriver.com/downloads/> e instalar en el PC con el dispositivo conectado. Luego instalar el APK instalador (instalador de paquetes APK para Android) en tu dispositivo desde la Play Store de Google.

El teléfono o dispositivo android debemos conectarlo al PC y activar las opciones del desarrollador (pulsando varias veces en el número de compilación) y activar en el modo desarrollador: **Depuración USB** . Con esto nos aparecerá el dispositivo en el target de Android SDK.

Al compilar la aplicación en el dispositivo, en realidad, esta queda instalada. Otra manera manual de instalarla y probarla en el dispositivo sería coger el archivo APK generado de la carpeta de nuestro proyecto: *Android\Release\BIN* y luego copiarlo al dispositivo e instalarlo con el **APK installer**

En el modo de emulación (sin usar el dispositivo físicamente) escoger en el manager AVD virtual de arrancar (Start) el AVD deseado.

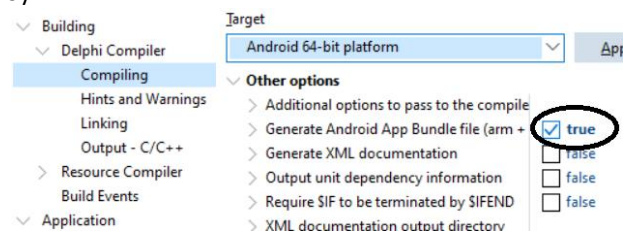
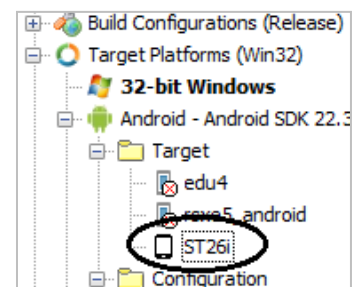
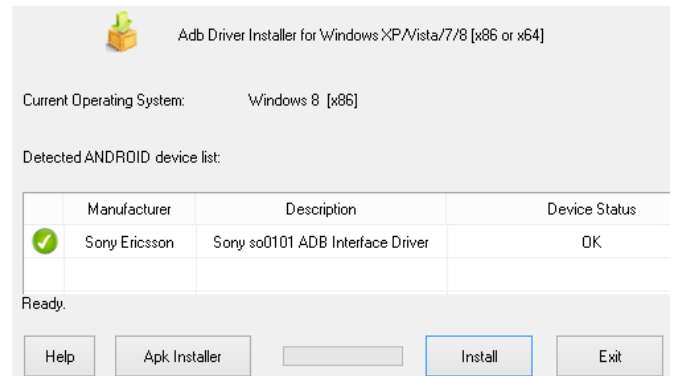
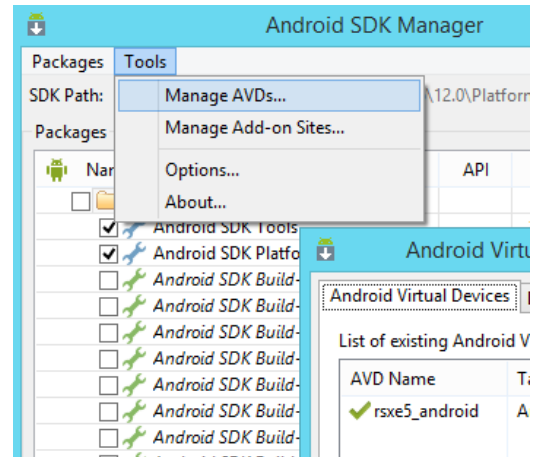
#### Configurar para subir a al PlayStore:

1. Activar en el *Project manager* el modo de Build: *Release* (versión final).
2. En project - Options: En pestaña *Uses permissions*: Activar o desactivar las funciones deseadas para el dispositivo (Acceso a cámaras o localización)
3. En project - Options: En pestaña *Provisioning*: Crear un certificado
4. En project – Deployment – pulsar el botón de la barra: Deploy.
5. Crear una cuenta de desarrollador de Google previo abono de 25 \$.

#### Para las nuevas versiones de Android 64 bits:

Google pide el formato AAB (Android App Bundle) en lugar del formato apk. Estas están disponibles a partir de las versiones de Delphi Rio 10.3.3 o Sidney 10.4.

En las opciones del proyecto se debe activar:  Generar el archivo App Bundle.



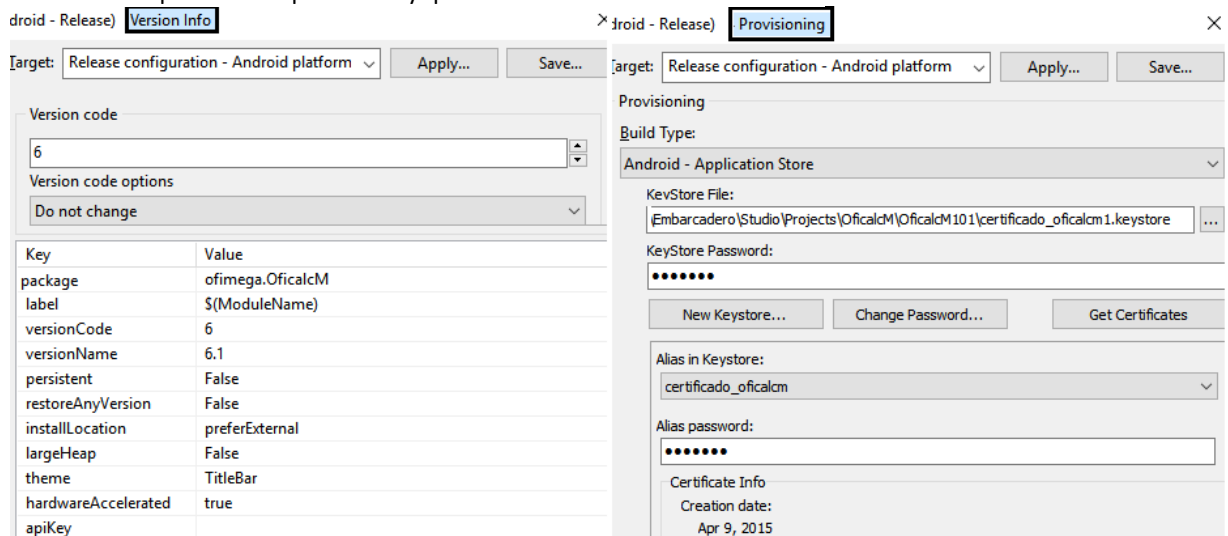
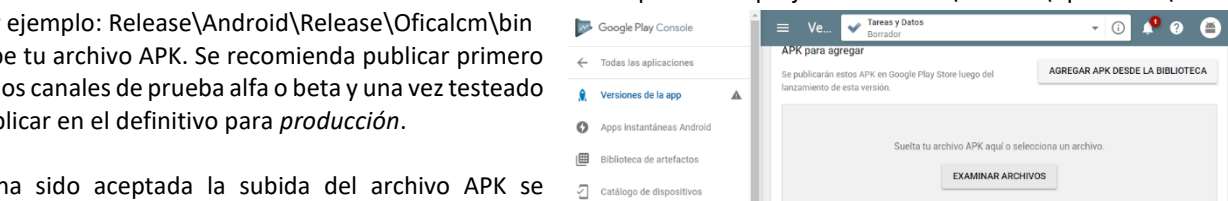
## Publicación en Google Play. Subir la APK o AAB (Android App Bundle)

### Configurar y generar el APK o AppBundles final:

- Desinstala la versión *debug* del dispositivo y activa, en el *Project manager*, el modo **Release** (versión final).
- En *Project - Options*: En pestaña *Uses permissions*: Activar o desactivar las funciones deseadas para el dispositivo (Acceso a cámaras o localización). Para la plataforma Android 64 bits Activa el  Generar el archivo App Bundle
- En *Project - Options*: Versión Info: Establece en número de versión y la versión de compilación *Version Code* que será la evaluada por Google.
- En *Project - Options*: En pestaña *Provisioning*: Crear un certificado.  
En este certificado va la huella digital del producto para firmar el APK en Google Play (SHA1).  
Ejemplo: SHA1: B9:D6:FD:B4:0D:9F:EE:57:67:C6:7D:49:DA:D1:CA:44:F4:F2:E9:C3  
Se recomienda conservar el archivo .keystore para incluirlo en nuevas versiones.
- En *Project - Deployment* – pulsar el botón de la barra: *Deploy*. Esto genera el archivo de instalación APK para Android

### Publicar y subir a Google Play:

- Tras Crear la cuenta de desarrollador de Google (previo abono de 25 \$ unos 20 €)
- Accede a [Google Play Developer Console](#).
- Para subir una aplicación nueva: Selecciona *Tus aplicaciones* o *Todas las aplicaciones*. Pulsa en *Crear una aplicación*.
- Escribe el Idioma y el nombre de la aplicación y pulsa en *Crear*.
- Selecciona el archivo APK o AAB normalmente desde la subcarpeta de tu proyecto: `Android\Release\Aplicación\bin`  
Por ejemplo: `Release\Android\Release\Oficalcm\bin`
- Sube tu archivo APK. Se recomienda publicar primero en los canales de prueba alfa o beta y una vez testeado publicar en el definitivo para *producción*.
- Si ha sido aceptada la subida del archivo APK se recomienda que guardes la configuración del proyecto pulsando en *Save...* tipo \*.opset.
- Antes de publicar, deberás especificar si es de pago o gratuita y el tipo de clasificación del contenido para obtener un certificado de la IARC.
- Esperar unos días para su comprobación y que se muestre el estado: **Publicado**.



### Para actualizar la versión de tu APK o AppBundle

- Selecciona en *Project - Options*: Release configuration – *Android platform*
- En la pestaña *Provisioning*: recupera el certificado anterior y las configuraciones guardadas en \*.opset.
- Genera el nuevo el APK o AAB, en modo Android Release, con una versión superior.
- Accede a *Google Play Developer Console* con tu cuenta. Selecciona el APK o AAB anterior y Pulsa en Subir.

**Nota:** Debes tener Activada la *Play App Signing*. Para ello debes descargar un script de Java que genera el archivo de claves para exportar y subir a la *Play Console* y obtener el certificado.

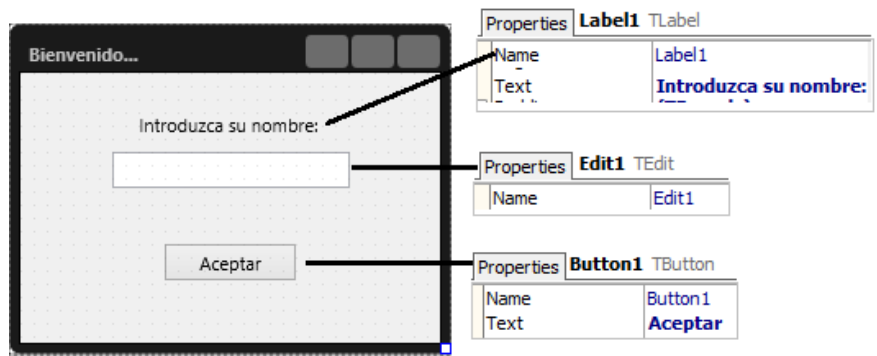


## Aplicación para Win 32 –win 64 e IOS - Mac

Para probar la emulación para IOS crearemos una aplicación básica para FireMonkey;

- Escoge: *New > FireMonkey Descropt Application* o *File > New > Other > Delphi Projects > FireMonkey HD Application*.
- Añade, de la paleta Estandar, estos tres elementos al formulario:

- Tedit – Edit1
- TButton que diga “Hola”.
- TLabel que diga: “Introduzca su nombre”



- Pulsa *doble clic* sobre el botón *Button1* para escribir en el procedure:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    ShowMessage('Hola,'+Edit1.Text);
end;
```

### Probar la aplicación:

Pulsa el botón **Run** o elige del menú: Run – run para comprobar su funcionamiento.

### Guardar el proyecto:

Crea una carpeta que se llame: *HolaFM*.

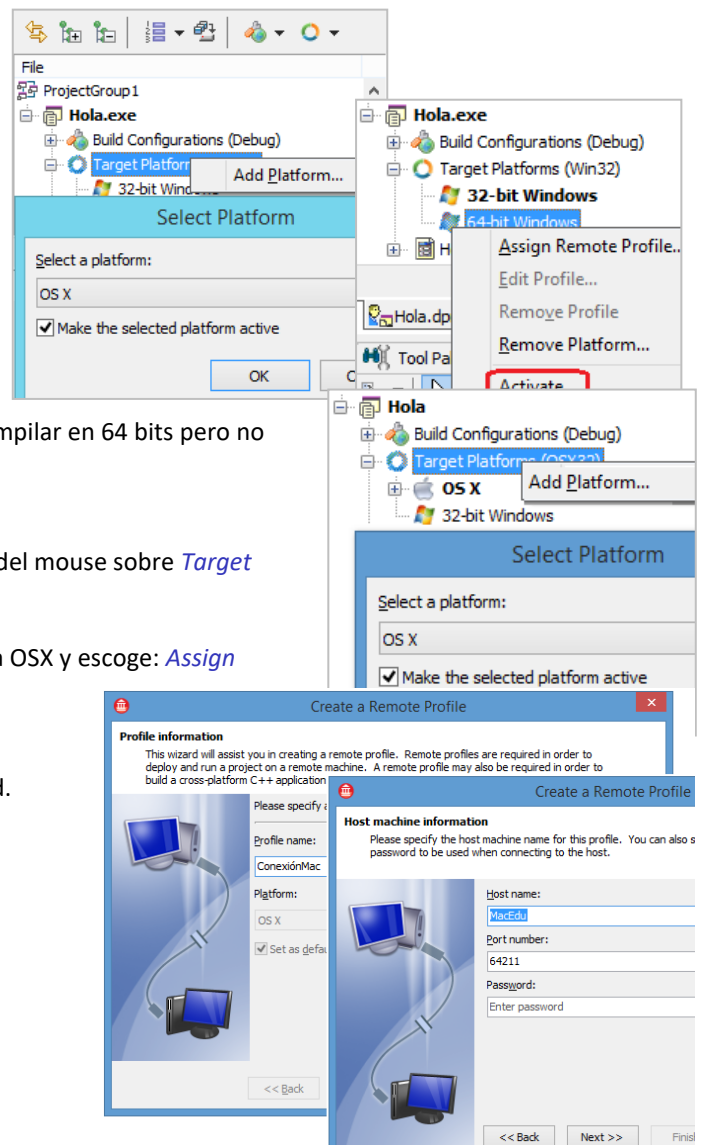
- Escoge: **File** ▶ **Save as...** Para guardar el archivo de código en la carpeta con el nombre: **HolaFM1.pas**
- Escoge: **File** ▶ **Save project as...** Para guarda el proyecto en la carpeta con el nombre **HolaFM.dpr**

### Activar la versión para Windows 64-bit:

1º En el panel *Project Manager*, pulsa con el botón *Derecho* del mouse sobre *Target Platforms*. Add Platform...

2º Pulsa con el botón *derecho* del mouse sobre la plataforma 64 bits y escoge: *Activate*.

3º Compila de nuevo la aplicación y comprueba que se muestra en la carpeta *Win64\Debug* la aplicación exe.



Nota: Si tu FM RAD Studio corre bajo x86 (Win32) podrás compilar en 64 bits pero no podrás ejecutarlo.

### Configurar para Mac OsX y ejecutar en un Mac:

1º En el panel *Project Manager*, pulsa con el botón *Derecho* del mouse sobre *Target Platforms*. Add Platform...Escoge: OSX.

2º Pulsa con el botón *derecho* del mouse sobre la plataforma OSX y escoge: *Assign Remote Profile*.

3º Selecciona Add... para abrir la ventana de conexión con un equipo mac por red.

4º Escribe el nombre de la conexión: ConexionMac

Pon el nombre de tu equipo Mac en la red o su IP

Entonces, el IDE estará conectado con tu Mac. Si pulsas F9 se ejecutará la aplicación en tu Mac.

## Multi-Device / FireMonkey - Gráfica con el componente PlotGrid.

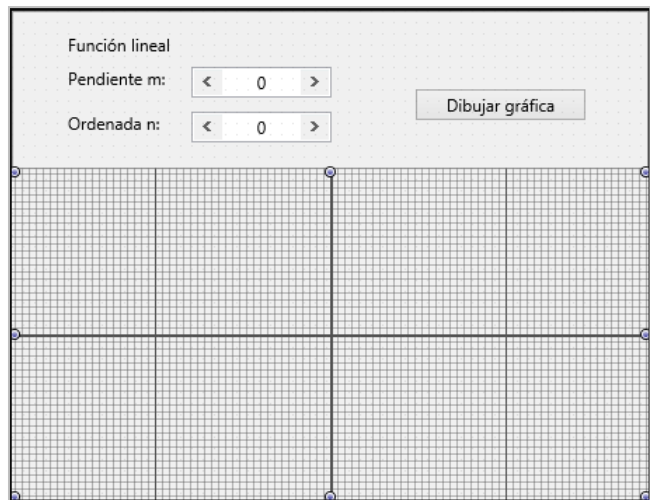
Para probar el uso del componente PlotGrid rejilla gráfica y los gestos del mouse:

- Escoge: [New > Multi-Device / FireMonkey Application](#) (según versión)
- Inserta en el formulario un panel y lo alineas al Top.
- Inserta en el formulario el componente **Tplotgrid** con name: PG1 y o alineas al cliente.
- Pon en el panel tres **Labels** y dos **Spinbox** con nombres: *Editm* y *Editn*  
Cambia sus valores max/min para que adopte valores entre -100 y +100
- Pon en el panel un **Button1** con la propiedad text: Dibujar gráfica.
- Pulsa doble clic en el botón *Button1* de *Dibujar gráfica* y añade al código On clic: Button1Click:

```
begin
    PG1.Repaint;
end;
```

- En el componente de rejilla gráfica plotgrid PG1 selecciona el evento *onPaint*: PG1.paint
- Escribe el código siguiente:

```
procedure TForm1.PG1Paint(Sender: TObject; Canvas: TCanvas; const ARect: TRectF);
var
    p1,p2,orig:TpointF; { Tpointf:punto de coordenadas decimales, tpoint: coordenadas enteras}
    x,x0,y0:integer;
    m,n:single;
begin
    m:=EditM.Value;
    n:=EditN.Value;
    //Punto origen en el medio de la rejilla:
    orig:=PointF(PG1.Width/2, PG1.Height/2);
    Canvas.BeginScene; //->inicia el lienzo
    //-> color de la línea azul:
    Canvas.Stroke.Color := TAlphaColorRec.Blue;
    //-> grosor de la línea a 2 :
    canvas.StrokeThickness := 2;
    //escalar a la resolución de la rejilla:
    n:=n*PG1.Frequency;
    //dibujo del lado derecho de la línea:
    p1:=PointF(orig.X,orig.y-n);
    //define el primer punto
    x:=round(PG1.Width/2); //posición x en la
    rejilla del origen x
    p2:=PointF(PG1.Width,orig.y-m*x-n); //define el punto2 a final de la rejilla
    canvas.DrawLine(p1, p2, 1); //dibuja la línea entre los 2 puntos con opacidad
    //dibujo del lado izdo
    p2:=PointF(0,orig.y+m*x-n);
    canvas.DrawLine(p1, p2, 1);
    canvas.EndScene;
```



Prueba y guarda en una carpeta nueva, la unidad con el nombre: grafica1 y el proyecto con el nombre: Grafica.

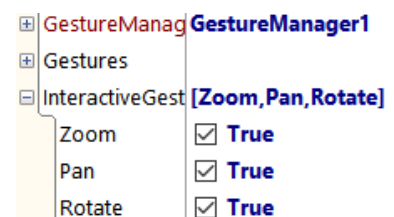
### Gestos para rotar - mover - zoom

- Añade un componente GestureManager:
- Al Formulario Form1, asigna a la propiedad GestureManager el GestureManager1



- Al Formulario Form1, asigna el evento OnGesture: FromGesture:
 

```
procedure TForm2.FormGesture(Sender: TObject;
    const [Ref] EventInfo: TGestureEventInfo; var Handled: Boolean);
var
    LObj: IControl;
begin
    if EventInfo.GestureID = igiRotate then
    begin
        LObj := Self.ObjectAtPoint(ClientToScreen(EventInfo.Location));
        if LObj is Tplotgrid then //si el cliente es el Plotgrid
        begin
            if (TInteractiveGestureFlag.gfBegin in EventInfo.Flags) then
                AnguloAntes := PG1.RotationAngle
            else if EventInfo.Angle <> 0 then
                PG1.RotationAngle := AnguloAntes - (EventInfo.Angle * 180) / Pi;
            end;
        end; end;
```



- Añade la variable pública AnguloAntes:double y comprueba en un dispositivo táctil.

**Firemonkey:****Uso de listas String Grid. Conversor**

1. Ideal para organizar listados en varias filas y columnas.
2. Escoge: *File > New > Multidevice*
3. Inserta en el formulario un Tgridpanel alineado al cliente
  - ▶ En la fila 0 columna 0 añadimos un spinBox con el nombre: **valor1**
  - ▶ En la fila 0 columna 1 añadimos un TEdit con el nombre: **resultado1**
  - ▶ En la fila 1 columna 0 añadimos un TStringGrid con el nombre: **magni1**
  - ▶ En la fila 1 columna 1 añadimos un TStringGrid con el nombre: **result1**

En el evento Oncreate del formulario añadimos los elementos a las listas en tiempo real:

```
var
  sufimetro:string;
  i:integer;
begin
  sufimetro:='metros'; //Longitud
  magni1.RowCount:=15;
  magni1.cells[0,0]:='Tera'+sufimetro+'- Tm';
  magni1.cells[1,0]:='1E12';
  magni1.cells[0,1]:='Giga'+sufimetro+'- Gm';
  magni1.cells[1,1]:='1E9';
  magni1.cells[0,2]:='Mega'+sufimetro+'- Mm';
  magni1.cells[1,2]:='1E6';
  magni1.cells[0,3]:='Miria'+sufimetro+'- Mm';
  magni1.cells[1,3]:='1E5';
  magni1.cells[0,4]:='Kilo'+sufimetro+'- Km';
  magni1.cells[1,4]:='1E3';
  magni1.cells[0,5]:='Hecto'+sufimetro+'- Hm';
  magni1.cells[1,5]:='1E2';
  magni1.cells[0,6]:='Deca'+sufimetro+'- Dam';
  magni1.cells[1,6]:='10';
  magni1.cells[0,7]:=''+sufimetro+'- m';
  magni1.cells[1,7]:='1';
  magni1.cells[0,8]:='Deci'+sufimetro+'- dm';
  magni1.cells[1,8]:='0,1';
  magni1.cells[0,9]:='Centi'+sufimetro+'- cm';
  magni1.cells[1,9]:='1E-2';
  magni1.cells[0,10]:='Mili'+sufimetro+'- mm';
  magni1.cells[1,10]:='1E-3';
  magni1.cells[0,11]:='Micra'+sufimetro+'- µm';
  magni1.cells[1,11]:='1E-6';
  magni1.cells[0,12]:='Nano'+sufimetro+'- nm';
  magni1.cells[1,12]:='1E-9';
  magni1.cells[0,13]:='Angstromio'+'- Å';
  magni1.cells[1,13]:='1E-12';
  magni1.cells[0,14]:='Pico'+sufimetro+'- pm';
  magni1.cells[1,14]:='1E-15';
  Magni1.EndUpdate;
  Result1.RowCount:=15;
  for i := 0 to Result1.RowCount-1 do
  Result1.Cells[0,i]:=magni1.cells[0,i];
```

En Magni1 on select cell;

```
var
  Result:Double;
  i:Integer;
  Op1,Op2:double;
begin
  try
    Op2:=StrToFloat(Magni1.Cells[1,Arow]);
    //-----> pasa a unidades base
    for i := 0 to Result1.RowCount-1 do
      Result1.Cells[1,i]:=FloatToStr(Op2*Valor1.Value/StrToFloat(Magni1.Cells[1,i]));
    //Actualizamos resultados...
    if Result1.Selected<0 then i:=0 else
      i:=Result1.Selected;
    if Result1.Selected<>0 then Result1.SelectRow(0)
    else Result1.SelectRow(1);
    Result1.SelectRow(i); //cambio de fila para
    activar el evento
  except
    Valor1.Text:='1'; //evita recursividad
  end
end

En Result1SelectCell:
begin
  Resultado1.Text:=Result1.cells[1,Arow];
end;
```

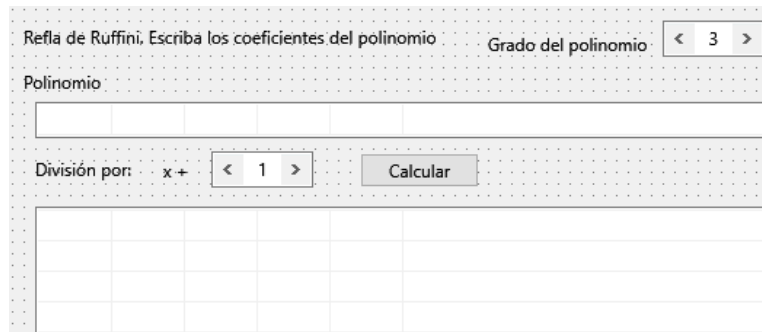
Prueba y guarda en una carpeta nueva, la unidad con el nombre: unit1 y el proyecto con el nombre: Conversor.

## Ejercicio Uso de StrinGrid y arrays con Firemonkey: División por Ruffini

- Crea un nuevo proyecto llamado Ruffini (File > New > Multidevice)
- Añade los componentes de la imagen adjunta:

- ✓ 1 spinbox
- ✓ 5 labels
- ✓ 2 StrinGrid

- Llama a las Stringgrids con el nombre: coef y tabla
- Llama al coef divisor a
- Añade el evento en el botón:



```
procedure TForm1.FormCreate(Sender);
```

```
var co:integer;
```

```
begin
```

```
  for co := 0 to round(grado.Value) do
```

```
    coef.Cells[co,0]:='0';
```

```
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var
```

```
d:real;
```

```
fi,co,I:integer; //fila,columna
```

```
begin
```

```
  d:=strtofloat(a.Text)*-1; //cambiamos el signo
```

```
  tabla.Cells[0,1]:=floattostr(d);
```

```
  for co := 1 to round(grado.Value) do //ponemos los coeficientes
```

```
    if coef.Cells[co-1,0]='' then tabla.Cells[co,0]:='0'
```

```
    else tabla.Cells[co,0]:=coef.Cells[co-1,0];
```

```
  //bajamos el primer coef;
```

```
  co:=1; //->celda[col,fil]
```

```
  fi:=2;
```

```
  tabla.Cells[co,fi]:=tabla.Cells[co,0];
```

```
  for co:=1 to round(grado.Value) do
```

```
    begin
```

```
      //multiplicamos y ponemos arriba izquierda
```

```
      tabla.Cells[co+1,fi-1]:=floattostr(d*strtofloat(tabla.Cells[co,fi]));
```

```
      //sumamos y ponemos abajo:
```

```
      if tabla.Cells[co+1,fi-2]='' then
```

```
        tabla.Cells[co+1,fi-2]:='0';
```

```
        tabla.Cells[co+1,fi]:=floattostr(strtof(float(tabla.Cells[co+1,fi-2])+strtof(float(tabla.Cells[co+1,fi-1]))));
```

```
      end;
```

```
end;
```

```
procedure TForm1.GradoChange(Sender: TObject);
```

```
begin
```

```
if grado.Value<coef.ColumnCount-1 then
```

```
  begin //quita columnas
```

```
    coef.Columns[tabla.ColumnCount-1].DisposeOf;
```

```
    tabla.Columns[tabla.ColumnCount-1].DisposeOf;
```

```
  end
```

```
else if grado.Value>coef.ColumnCount-1 then
```

```
  begin //añade columnas
```

```
    coef.AddObject(TStringColumn.Create(Self));
```

```
    tabla.AddObject(TStringColumn.Create(Self));
```

```
    coef.Cells[tabla.ColumnCount-1,0]:='0';
```

```
    coef.Columns[tabla.ColumnCount-1].Width:=50
```

```
    tabla.Columns[tabla.ColumnCount-1].Width:=50
```

```
  end;
```

```
  tabla.RealignContent;
```

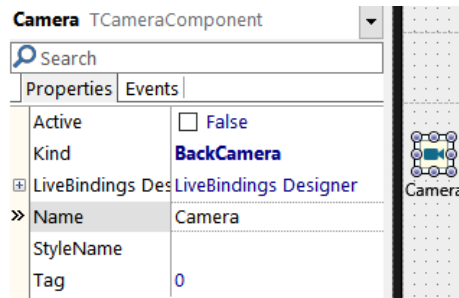
```
end;
```

## Sensores mobile. Antorcha android (versión adaptada Samples\FlashLight)



- Descarga de internet imagen de bombilla o similar y de un botón encendido y apagado:
- Crea un nuevo proyecto llamado **Linterna** (File > New > Multidevice application)
- Añade el componente **TCamera** y los componentes de la imagen derecha: →
- Alinea el **LayoutContenedor** al cliente y el **LayoutBotones** al centro.

- Cambia la propiedad **Kind** a **Back Camera**



- Añade el código:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  apagado.Enabled := Camera.HasFlash;
  Camera.Active := True;
end;
```

```
procedure TForm1.ApagadoClick(Sender: TObject);
begin
  apagado.Visible := False;
  encendido.Visible := True;
  Camera.TorchMode := TTorchMode.ModeOn; // En C++: Camera->TorchMode = TTorchMode::ModeOn;
end;
```

```
procedure TForm1.EncendidoClick(Sender: TObject);
begin
  apagado.Visible := True;
  encendido.Visible := False;
  Camera.TorchMode := TTorchMode.ModeOff; // En C++: Camera->TorchMode = TTorchMode::ModeOff;
end;
```

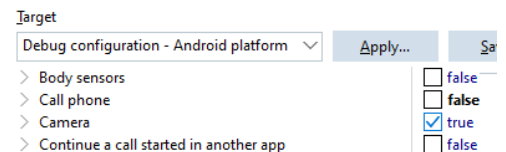


- Compila en dispositivo: En el Project Manager, selecciona la plataforma Android y conecta un móvil Android al Puerto USB. Recuerda activas las opciones del desarrollador (pulsando varias veces en el numero de compilación) en el móvil y tener instalado el driver ADB driver installer.
- Ejecutar y comprobar: Escoje **Run > Run Without Debugging**
- Guarda el Proyecto: **Linterna** y la unidad: **Linterna1**.

### Permisos:

- Activar en Project – Opciones: ✓ Camera
- En el dispositivo móvil, ir a **Ajustes – Aplicaciones** y habilitar el permiso de la plicación: ✓ Camera

### Uses Permissions



### Mejoras:

- 1.- Modo intermitente:

Añade un timer y un botón para modo intermitente. Pon el el timer el Código:

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if Camera.TorchMode = TTorchMode.ModeOn then Camera.TorchMode := TTorchMode.Modeoff
  else Camera.TorchMode := TTorchMode.ModeOn;
end;
```

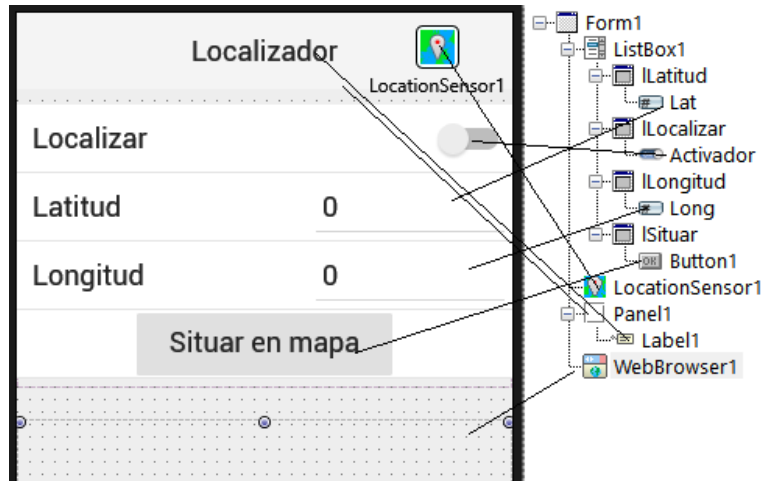
- 2.- Añade las sombras y extras que se muestran en el ejemplo:

C:\Users\Public\Documents\Embarcadero\Studio\16.0\Samples\Object Pascal\Mobile Samples\Device Sensors and Services\FlashLight

## Sensores mobile. Localizador sensor (versión adaptada Samples)

- Crea un nuevo proyecto llamado Localizador (File > New > Multidevice application)
- Añade los componentes de la imagen:→

- La propiedad del *numberbox Lat* será del tipo decimal *Float*, con 3 decimales y valores comprendidos entre max: 80 y min -80
- La propiedad del *numberbox Long* será del tipo decimal *Float*, con 3 decimales y valores comprendidos entre max: 180 y min -180



- Añade el código:

```
procedure TForm1.ActivadorSwitch(Sender: TObject);
begin
  LocationSensor1.Active := Activador.IsChecked; //-> activa el location sensor
end;

procedure TForm1.LocationSensor1LocationChanged(Sender: TObject;
  const OldLocation, NewLocation: TLocationCoord2D);
begin
  Lat.Value:=NewLocation.Latitude; //->devuelve posición latitud del GPS
  Long.Value:=NewLocation.Longitude; //->devuelve posición longitud del GPS
  Button1Click(Sender); //->situa en el mapa
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  formatsettings.DecimalSeparator='.'; //->formato decimal punto americano
  WebBrowser1.Navigate('https://maps.google.com/maps?q='+floattostr(Lat.Value)+'+', '+
floattostr(Long.Value));
end;
```

- Compilar en el dispositivo: Conecta un móvil Android al Puerto USB y en el *Project Manager*, selecciona la plataforma Android. Recuerda activas las opciones del desarrollador (pulsando varias veces en el numero de compilación) en el moóvil y tener instalado el driver ADB driver installer.
- Ejecutar y comprobar: Escoje *Run > Run Without Debugging*
- Guarda el Proyecto: **Localizador** y la unidad: **Localizador1**.

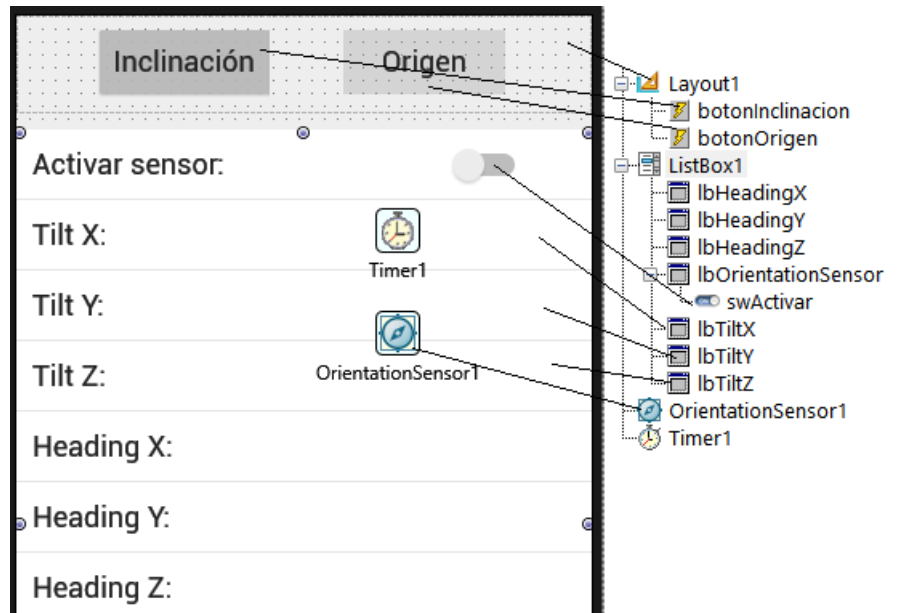
## Sensores mobile. Orientación sensor (versión adaptada Samples)

### Propiedades:

El Botón de Inclinación muestra el ángulo de rotación en grados en el eje X, Y y Z

El Botón de Origen muestra las coordenadas X, Y y Z del vector que apunta al norte magnético, medido en microteslas.

- Crea un nuevo proyecto llamado Localizador (File > New > Multidevice application)
- Añade los componentes de la imagen.
- Añade el código y guardal con el nombre sensor1



```

OrientationSensor1SensorChoosing(
  Sender: TObject; const Sensors: TSensorArray;
  var ChoseSensorIndex: Integer);
var
  I: Integer;
  Found: Integer;
begin
  Found := -1;
  for I := 0 to High(Sensors) do
  begin
    if botonInclinacion.IsPressed and
      (TCustomOrientationSensor.TProperty.TiltX in
      TCustomOrientationSensor(Sensors[I]).AvailablePr
      operties) then
      begin
        Found := I;
        Break;
      end
    else if botonOrigen.IsPressed and
      (TCustomOrientationSensor.TProperty.HeadingX in
      TCustomOrientationSensor(Sensors[I]).AvailablePr
      operties) then
      begin
        Found := I;
        Break;
      end;
  end;
  end;

  if Found < 0 then
  begin
    Found := 0;
    botonInclinacion.IsPressed := True;
    botonOrigen.IsPressed := False;
    ShowMessage('Brújula no disponible');
  end;

  ChoseSensorIndex := Found;
end;

botonInclinacionClick(Sender: TObject);
begin
  OrientationSensor1.Active := False;
  botonOrigen.IsPressed := False;
  botonInclinacion.IsPressed := True;
  OrientationSensor1.Active :=
swActivar.IsChecked;
end;

```

```

end;

Timer1Timer(Sender: TObject);
begin
  lbTiltX.Text := Format('Tilt X: %f',
  [OrientationSensor1.Sensor.TiltX]);
  lbTiltY.Text := Format('Tilt Y: %f',
  [OrientationSensor1.Sensor.TiltY]);
  lbTiltZ.Text := Format('Tilt Z: %f',
  [OrientationSensor1.Sensor.TiltZ]);
  lbHeadingX.Text := Format('Heading X: %f',
  [OrientationSensor1.Sensor.HeadingX]);
  lbHeadingY.Text := Format('Heading Y: %f',
  [OrientationSensor1.Sensor.HeadingY]);
  lbHeadingZ.Text := Format('Heading Z: %f',
  [OrientationSensor1.Sensor.HeadingZ]);
end;

FormActivate(Sender: TObject);
begin
  {$ifdef IOS}
  {$ifndef CPUARM}
  lbOrientationSensor.Text := 'Sin sensor';
  swOrientationSensorActive.Enabled := False;
  {$endif}
  {$endif}
end;

botonOrigenClick(Sender: TObject);
begin
  OrientationSensor1.Active := False;
  botonInclinacion.IsPressed := False;
  botonOrigen.IsPressed := True;
  OrientationSensor1.Active :=
swActivar.IsChecked;
end;
swActivarSwitch(
  Sender: TObject);
begin
  { activate or deactivate the orientation
  sensor }
  OrientationSensor1.Active :=
swActivar.IsChecked;
  Timer1.Enabled := swActivar.IsChecked;
end;

```

## Transmisión Bluetooth. Teoría

---

Los dispositivos BlueTooth pueden actuar como Masters o como Slaves (Amos o esclavos). La diferencia es que un BlueTooth Slave solo puede conectarse a un master y a nadie más, en cambio un master BlueTooth, puede conectarse a varios Slaves o permitir que ellos se conecten y recibir y solicitar información de todos ellos, arbitrando las transferencias de información ( Hasta un máximo de 7 Slaves).

Así pues un nodo BlueTooth puede ser Master o Slave y dispone de una dirección única, así como de un nombre para identificarse y muy habitualmente también incluye un PIN de conexión o número de identificación que debe teclearse para ganar acceso al mismo.

Como el BlueTooth lo desarrolló Nokia para conectar teléfonos móviles, a otros dispositivos como auriculares, micrófonos o conexiones al audio del coche, existe un procedimiento definido que se llama Pairing (o emparejamiento) que vincula a dos dispositivos Bluetooth.

Cuando vinculas dos dispositivos BT, se inicia un proceso en el que ellos se identifican por nombre y dirección interna y se solicitan la clave PIN para autorizar la conexión.

Si el emparejamiento se realiza con éxito, ambos nodos suelen guardar la identificación del otro y cuando se encuentran cerca se vuelven a vincular sin necesidad de intervención manual. Por eso el CD de tu coche reconoce el móvil de tu bolsillo en cuanto te subes y puedes reproducir la música que tienes en tu Smartphone.

Aunque para que tu BlueTooth pueda enviar o recibir música, debe aceptar otra norma posterior llamada Advanced Audio Distribution Profile (A2DP) y que en caso de ser algún sistema antiguo te impedirá la reproducción.

Naturalmente, a lo largo de los años la norma ha ido variando y existen varias versiones de la misma, con compatibilidad siempre con las versiones anteriores que se diferencian en la distancia que pueden alcanzar (entre 50 y 100 metros, teóricamente y sin obstáculos) además de la velocidad de transferencia.

### LOS COMANDOS AT

La transmission serie RS232 solo utilizaba dos hilos de comunicaciones, no había más remedio que incorporar una orden de atención que significara que a continuación venia una orden de programación del modem, que no debía ser transmitida al otro extremo. Es decir que las ordenes eran del tipo "AT+Orden", donde AT era el comando especificado de atención y al conjunto de ellas se llamó comandos AT. Lisa comandos para HC-06.

COMMAND	FUNCTION
AT	Test UART Connection
AT+RESET	Reset Device
AT+VERSION	Query firmware version
AT+ORGL	Restore settings to Factory Defaults
AT+ADDR	Query Device Bluetooth Address
AT+NAME	Query/Set Device Name
AT+RNAME	Query Remote Bluetooth Device's
AT+ROLE	Query/Set Device Role
AT+CLASS	Query/Set Class of Device CoD
AT+IAC	Query/Set Inquire Access Code
AT+INQM	Query/Set Inquire Access Mode
AT+PSWDAT+PIN	Query/Set Pairing Passkey
AT+UART	Query/Set UART parameter
AT+CMODE	Query/Set Connection Mode
AT+BIND	Query/Set Binding Bluetooth Address
AT+POLAR	Query/Set LED Output Polarity
AT+PIO	Set/Reset a User I/O pin

## Sensores mobile. Bluetooth (versión apaptada de Samples)

Añadir la librería a uses: **System.Bluetooth**

Y los objetos en la declaración *public*:

- FBluetoothManager: TBluetoothManager;
- FDiscoverDevices: TBluetoothDeviceList;
- FPairedDevices: TBluetoothDeviceList;
- FAdapter: TBluetoothAdapter;

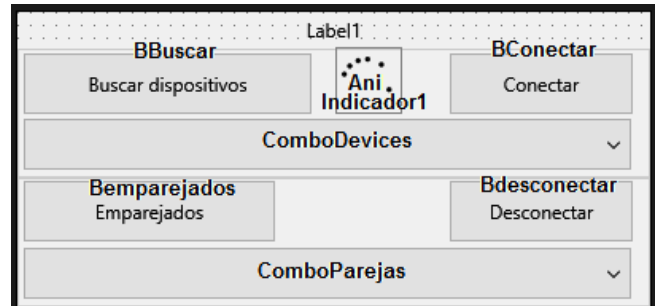
Añadir el procedure manualmente:

```
procedure ListaDispositivos(const Sender: TObject;
const ADevices: TBluetoothDeviceList);
```

```
procedure TForm1.BConectarClick(Sender:TObject);
begin
if FBluetoothManager.ConnectionState =
TBluetoothConnectionState.Connected then
begin
if ComboDevices.ItemIndex > -1 then
FAdapter.Pair(FDiscoverDevices[ComboDevices.Item
Index])
Else ShowMessage('Dispositivo no seleccionado');
end;
end;
```

```
procedure TForm1.BDesconectarClick(Sender:
TObject);
begin
if FBluetoothManager.ConnectionState =
TBluetoothConnectionState.Connected then
begin
if ComboParejas.ItemIndex > -1 then
FAdapter.UnPair(FPairedDevices[ComboParejas.Item
Index])
else ShowMessage('Dispositivo emparejado no
seleccionado');
end;
end;
```

```
procedure TForm1.BEmparejadosClick(Sender:
TObject);
var
I: Integer;
begin
if FBluetoothManager.ConnectionState =
TBluetoothConnectionState.Connected then
begin
ComboParejas.Clear;
FPairedDevices :=
FBluetoothManager.GetPairedDevices;
if FPairedDevices.Count > 0 then
for I:= 0 to FPairedDevices.Count - 1 do
ComboParejas.Items.Add(FPairedDevices[I].DeviceN
ame)
else
ComboParejas.Items.Add('No hay dispositivos
emparejados');
end;
end;
```



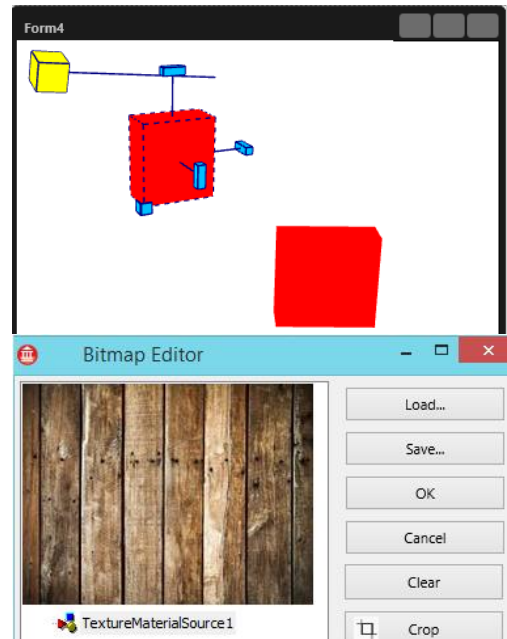
```
procedure TForm1.BuscarClick(Sender: TObject);
begin
AniIndicator1.Visible := True;
Buscar.Text := 'Buscando...';
Buscar.Enabled := False;
ComboDevices.Clear;
if FBluetoothManager.ConnectionState =
TBluetoothConnectionState.Connected then
begin
Label1.Text := 'Dispositivo Bluetooth
detectado
'+FBluetoothManager.CurrentAdapter.AdapterNa
me+'";
FBluetoothManager.StartDiscovery(10000);
FBluetoothManager.OnDiscoveryEnd :=
ListaDispositivos;
end;
end;
```

```
procedure TForm1.ListaDispositivos (const
Sender: TObject; const ADevices:
TBluetoothDeviceList);
var
I: Integer;
begin
Buscar.Text := 'Terminado';
Buscar.Enabled := True;
AniIndicator1.Visible := False;
FDiscoverDevices := ADevices;
ComboDevices.Clear;
if ComboDevices.Count=-1 then
ComboDevices.Items.Add('No hay dispositivos')
else
for I := 0 to ADevices.Count - 1 do
ComboDevices.Items.Add(ADevices[I].DeviceName
+ ' -> ' + ADevices[I].Address);
ComboDevices.ItemIndex := 0;
end;
```

```
procedure TForm1.FormShow(Sender: TObject);
begin
try
FBluetoothManager:=TBluetoothManager.Current;
FAdapter:=FBluetoothManager.CurrentAdapter;
except
ShowMessage('Bluetooth no detectado');
end;
end;
```

# FireMonkey: Manipular objetos3D (versión apaptada de docwiki.embarcadero.com)

- Escoge: *New > Multi-Device / FireMonkey Application > FireMonkey 3D Application*  
o *File > New > Other > Delphi Projects > FireMonkey 3D Application* (según la versión)
- Añade los siguientes componentes 3D de la **Tool Palette** al formulario:
  - 1 **TLight** de la paleta Scenes
  - 2 **TCubes** de la paleta Shapes.
- Ajusta su posición arrastrando el objeto, y su tamaño, desde la marca azul de su izquierda, como en la imagen.



- Para cambiar la propiedad *Material* del primer cubo es necesario en la versión XE3 añadir al formulario un componente: [TTextureMaterialSource](#): Selecciona en el Object Inspector, la *TextureMaterialSource*, la propiedad: *Texture* [...] >Edit.

Se abrirá la ventana editor. Pulsa en el botón *Load* para cargar la textura deseada dese un bitmap y clicla en *OK* al finalizar.

Asigna al primer cubo el materialsource: TextureMaterialSource1

- Para cambiar la iluminación del segundo cubo, es necesario en la versión Xe3 añadir al formulario un componente a [TLightMaterialSource](#). Para ajustar la propiedad: *Texture* [...] >Edit. Se abrirá la ventana editor. Pulsa en el botón *Load* para cargar la textura deseada dese un bitmap y clicla en *OK* al finalizar.

Asigna al segundo cubo el materialsource: LightMaterialSource1



- Compila la aplicación para observar la diferencia en la textura entre los dos cubos.

## Añadir y ajustar objetos 2D, como botones, en espacios 3D:

Para ello arastra desde la paleta 3dlayers, un component de capa: TLayer3D sobre el formulario y pon, sobre este, un botón: TButton.

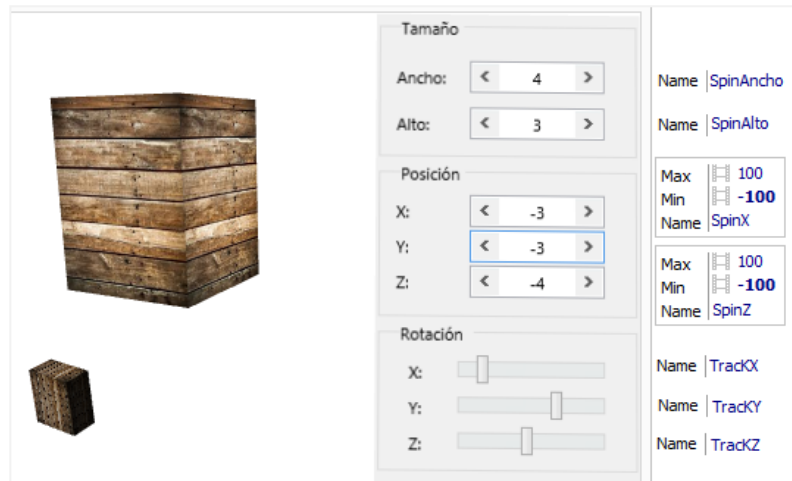
Para simlar una superficie 2D, cambia el tipo de proyección del TLayer3D: En el Object Inspector, cambia las propiedades:

Projection = pjScreen //vista plana  
Align = alMostRight  
Width = 180

Añade 3 TGroupBoxes sobre el objeto TLayer3D. Pon la propiedad de los 3, Align to alTop.

Añade los objetos y cambia el nombre de los objetos que se muestran en la imagen.

Añadimos los spinbox para el valor.



## Eventos:

- Utilizamos el mismo procedimiento para el ancho y alto: SpinAnchoChange
- Utilizamos el mismo procedimiento al cambiar los tres objetos SpinEdit de posición: SpinAnchoChange
- Utilizamos el mismo procedimiento al cambiar los tres objetos TrackBar de rotación: TrackXChange

```
procedure TForm4.Form1DCreate(Sender: TObject); // -> llamado en el evento Oncreate del Form
begin
//inicializa tamaño:
  SpinAncho.Value:=Cube1.Width; //pone el spinancho al ancho del cubo
  SpinAlto.Value:=Cube1.Height; //pone el spinalto al alto del cubo
//inicializa posición
  SpinX.Value:=Cube1.Position.X;
  SpinY.Value:=Cube1.Position.Y;
  SpinZ.Value:=Cube1.Position.Z;
//inicializa rotación
  TrackX.Value:=Cube1.RotationAngle.X;
  TrackY.Value:=Cube1.RotationAngle.Y;
  TrackZ.Value:=Cube1.RotationAngle.Z;
end;

procedure TForm1.SpinAnchoChange(Sender: TObject);
begin
  if sender=SpinAncho then Cube1.Width:=SpinAncho.Value // si es llamado desde el spinancho...
  else Cube1.Height:= SpinAlto.Value; //sino es que es llamado desde el spinalto
end;

procedure TForm1.SpinXChange(Sender: TObject);
begin
  Cube1.Position.X:=SpinX.Value; // posición x del cubo la que tenga el SpinX
  Cube1.Position.Y:=SpinY.Value; // posición y del cubo la que tenga el SpinY
  Cube1.Position.Z:=SpinZ.Value; // posición z del cubo la que tenga el SpinZ
end;

procedure TForm1.TracKXChange(Sender: TObject);
begin
  Cube1.RotationAngle.X:= TrackX.Value; // rotación x del cubo la que tenga el TrackX
  Cube1.RotationAngle.Y:= TrackY.Value; // rotación y del cubo la que tenga el TrackY
  Cube1.RotationAngle.Z:= TrackZ.Value; // rotación z del cubo la que tenga el TrackZ
end;
```

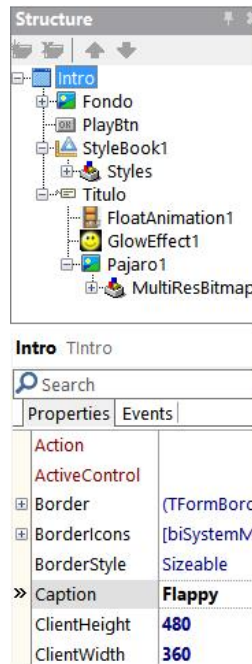
## Firemonkey: Introducción a juego Flappy

- Crea un nuevo proyecto llamado Flappy (File > New > Multidevice application)

### Creación de la pantalla de Introducción:

- Asigna el tamaño a la ventana: 360 width x 480 Height aprox.
- Añadiremos: 3 Imágenes, 1 label, 1 botón y 1 Stylebook.
- En el Stylebook, cargaremos el estilo **Air** de la carpeta *Samples* (Apply and close).
- Al Formulario Form1 le asignaremos ese Stylebook1

- ▶ Propiedades del botón: Name: PlayBtn – Text: Play – Textsettings Fontsize: 24 – StyleSettings Fontcolor: False
- ▶ Propiedades del label: Name: Titulo – Text: Flappy - FontColor: Purple – Size: 48 – Align: Top - Margin Top 50
- ▶ Propiedades de la Image1: Name: Fondo – Align: Contents – Cargar el MultiResBitmap: Fondo.png
- ▶ Propiedades de la Image2: Name: Pajaro – Align: Center – Cargar el MultiResBitmap: Pajaro1.png



- Integrar dentro del Label *Titulo*, un *GlowEffect* (Softness 0.4 color gold) y un *FloatAnimation* (*Position.Y* de *Startvalue* 30 a *stopvalue* 67 *Duration*: 0.5)

Corre el programa (Run ▶) para comprobar que el título oscila en vertical.

Guarda la unidad con el nombre: `uIntro`

Código al pulsar el botón Play:

```
procedure TMenuForm.PlayBtnClick(Sender: TObject);
begin
  Game.Iniciar;
  Intro.Hide;
  {$IFDEF MSWINDOWS}
  Game.ShowModal; //si es windows llama a la otra ventana en modo modal
  {$ELSE}
  Intro.Show; //si no es windows muestra la ventana
  {$ENDIF}
end;
```

### Creación de la pantalla del juego:

Escoge del menú: File – New – Multi-device Form Delphi. Tipo HDForm. Mismo tamaño que el form anterior.

Nombre del form: **Game**. Nombre de la unidad: **uGame**

La primera capa Layout: Nombre: **ReadyLayout**, dentro de esta añadimos los siguientes objetos:

- ▶ Un Label: ReadyLbl con texto Text: Readdy! - size 48 – GlowEffect Gold – Align Horizontal
- ▶ Un rectángulo: Fill: Orange – Stroke: White - Corner rounded con un label en su interior con el texto: “Pulsa para mantener el pájaro en vuelo”
- ▶ Poner esta capa ReadyLayout en Visible:False
- ▶ Evento la hacer clic en la capa:

```
procedure TGame.ReadyLayoutClick(Sender: TObject);
begin
  ReadyLayout.Visible := False;
  puntuacion.Visible := True;
  Timer1.Enabled:=True;
end;
```

La segunda capa Layout: Nombre: **GameOverLayout**, dentro de esta añadimos los siguientes objetos con la disposición de la figura

1 Label con el texto: Game Over

1 Rectángulo color crema con de esquinas redondeadas y dos labels dentro

2 Botones para **Salir** y **Volver** a jugar con el código:

Salir: `game.close; //Intro.show;`

Volver: `game.iniciar;`

Hacemos la capa *GameOverLayout* visible false.



### El formulario Game:

Propiedad Quality: HighPerformance

Evento al ocultar: TGame.FormHide(Sender: TObject): Timer1.Enabled := False;  
Intro.Show;

Objetos:

- Añadimos una imagen de fondo. LLamada fondo y cargamos el MultiresBitMap: Fondo.png
- Añadimos tres imágenes: pajaroA y pajaroB visible false y pajaro visible true, donde cargamos los bitmaps de pajaro.png y pajaro2.png. En el bitmap de pajaroA tendrá las alas subidas y en el bitmap del pajaro B tendrá las alas bajadas.  
*Nota: este sprite lo puedes descargar desde la entrada Google: "flappy bird sprite"*
- Añadimos un Timer: Timer1
- Añadimos una capa *Layout* de nombre tubo que contendrá dos rectángulos: *TopTubo* y *TopTuboCap* como en la imagen:
- Añadimos un rectangle con el nombre Suelo y align al Bottom, color tierra.



```
var
  Game: TGame;
  Desplazatubos: integer;
  factor: single;
  EnemyList: TStringList;
  Score: Integer; //puntuacion
  PajaroFlag: boolean; //variables permiso para mover
                          pajaro
  JuegoAcabado: boolean;
```

uses uIntro;

**procedure TGame.FormCreate**

```
begin
  EnemyList := TStringList.Create;
  factor := 1.5; //factor de escala
  GameOverLayout.Position.Y := Game.Height;
end;
```

**procedure TGame.FormDestroy(Sender: TObject);**

```
begin
  EnemyList.Free;
end;
```

**procedure TGame.FormHide(Sender: TObject);**

```
begin
  Timer1.Enabled := False;
  Intro.Show;
end;
```

**procedure TGame.FormMouseDown**

```
Shift: TShiftState; X, Y: Single);
begin
  if Timer1.enabled=True then
  begin
    if pajaro.Position.Y<75 then
    begin
      pajaro.Position.Y := 0;
    end
    else
    begin
      pajaro.Position.Y := pajaro.Position.Y-75;
      //sube 75 pixeles
    end;
    pajaro.RotationAngle := 0; //mira al frente
  end;
end;
```

**procedure TGame.ReadyLayoutClick**

```
begin
  ReadyLayout.Visible := False;
  puntuacion.Visible := True;
  Timer1.Enabled:=True;
end;
```

**procedure TGame.Timer1Timer**

```
const
  VelocidadTubos = 2;
//velocidad de paso de los tubos
var
  R: TLayout;
  I: Integer;
  POffset,WOffset: Integer;
  R1, R2: TRect; //rectangulos
                  para colisión
```

```
begin
  Game.BeginUpdate;
  if Timer1.Enabled=true then
  begin
    JuegoAcabado := False;
    //si toca suelo se acaba el
    juego...
    if (Pajaro.Position.Y +
    Pajaro.Height)>Suelo.Position.Y
    then
```

```
begin
  JuegoAcabado := True;
end;
```

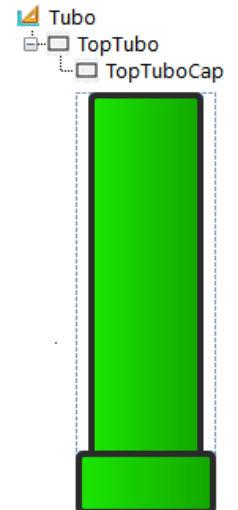
```
if Desplazatubos=0 then
begin
  if (Random<0.5) then
  begin
    WOffset := round((200*Factor));
  end
  else
  begin
    WOffset := round((250*Factor));
  end;
  if (Random<0.5) then
  begin
    POffset := round((-125*Factor));
  end
  else
  begin
    POffset := round((-25*Factor));
  end;
```

```
R := TLayout(Tubo.Clone(Self)); //clona el objeto tubo
EnemyList.AddObject('',R); //añade el objeto R a la
                             lista
```

```
R.Parent := Game;
R.Visible := True; //muestra el objeto tubo
R.Position.X := Game.Width+R.Width; //lo pone a la
                                     derecha
R.Position.Y := (0-WOffset)+POffset; //posiciona arriba
                                     + altura aleatoria
```

```
R := TLayout(Tubo.Clone(Self)); //clona otro objeto
tubo
EnemyList.AddObject('',R); //añade el objeto R a la
                             lista
```

```
R.Parent := Game;
R.Visible := True;
R.Position.X := Game.Width+R.Width;
R.Position.Y := (Game.Height-R.Height+WOffset)+POffset;
//posiciona abajo + altura aleatoria
```



```

R.RotationAngle := 180; //lo gira 180 grados
end;

if Desplazatubos>(VelocidadTubos*30) then
begin
  Desplazatubos := 0;
end
else
begin
  Inc(Desplazatubos); //aumenta el desplazamiento
end;
//Rectangulo R1 esta definido por el area del pajaro
R1 :=
Rect(Trunc(Pajaro.Position.X),Trunc(Pajaro.Position.Y),Trunc(Pajaro.Position.X+Pajaro.Width),Trunc(Pajaro.Position.Y+Pajaro.Height));

for I := EnemyList.Count-1 downto 0 do
begin
  if Assigned(EnemyList.Objects[I]) then
  begin
    R := TLayout(EnemyList.Objects[I]);
    R.Position.X := R.Position.X-5;
    //Rectangulo R2 esta definido por el area del tubo
    R2 :=
Rect(Trunc(R.Position.X),Trunc(R.Position.Y),Trunc(R.Position.X+R.Width),Trunc(R.Position.Y+R.Height));
    //si hay intersección entre las dos areas...
  if ChequeaColision(R1,R2)=True then
  begin
    JuegoAcabado := True;
  end
  else
  begin
    if (((R.Position.X+(R.Width/2))<Pajaro.Position.X)
AND (R.Tag=1) AND (R.TagFloat=0)) then
    begin
      R.TagFloat := 1;
      Puntua(Score+1);
    end;
  end;

  if (R.Position.X<((R.Width*-1)-10)) then
  begin
    R.DisposeOf;
    EnemyList.Delete(I);
  end;
end;

  if Pajaro.RotationAngle<90 then
    Pajaro.RotationAngle := Pajaro.RotationAngle+5;

Pajaro.Position.Y :=
Pajaro.Position.Y+(Max(Pajaro.RotationAngle,1)/5);
//(recuerda poner libreria math para usar Max)
  if PajaroFlag=False then
  begin
    Pajaro.Bitmap.Assign(PajaroB.Bitmap);
    PajaroFlag := True;
  end
  else
  begin
    Pajaro.Bitmap.Assign(PajaroA.Bitmap);
    PajaroFlag := False;
  end;
end;

end;

if JuegoAcabado=True then GameOver;
end;
Suelo.BringToFront;
puntuacion.BringToFront;
if JuegoAcabado=true then GameOverLayout.BringToFront;
Game.EndUpdate;
end;

procedure TGame.Puntua(I: Integer);
begin
  Score := I;
  puntuacion.Text := IntToStr(Score);
end;

procedure TGame.BtnSalirClick(Sender: TObject);
begin
  Game.Close;
  Intro.Show;
end;

procedure TGame.BtnVolverClick(Sender: TObject);
begin
  game.iniciar;
end;

function TGame.ChequeaColision( R1, R2 : TRect; OffSetY
: LongInt = 4; OffSetX : LongInt = 4): Boolean;
begin
  //Rectangulo R1 esta definido por el area del pajaro
  //Rectangulo R2 esta definido por el area del tubo
  //Detectar si hay interesección entre las dos areas..
  //Automático: -> Result:=IntersectRect(R1,R2);
  //Manual: ->
  {Result:=( NOT ((R1.Bottom - (OffsetY * 2) < R2.Top +
OffsetY) or(R1.Top + OffsetY > R2.Bottom - (OffsetY *
2)) or( R1.Right - (OffsetX * 2) < R2.Left +
OffsetX) or( R1.Left + OffsetX > R2.Right - (OffsetX
* 2)))));}
end;

procedure TGame.GameOver;
begin
  JuegoAcabado := True;
  Timer1.enabled:=false;
  GameOverLayout.BringToFront;
  GameOverLayout.Visible := True;
  Puntuacion.Visible := False;
end;

procedure Tgame.Iniciar;
var
  I: Integer;
  R: TLayout;
begin
  JuegoAcabado := False;
  GameOverLayout.Visible := False;
  pajaro.Position.X := 96;
  pajaro.Position.Y := 248;
  pajaro.RotationAngle := 0;
  ReadyLayout.Visible := True;
  puntua(0);
end;

```

## Conversor de monedas de FireMonkey/Multi-device con servicio web

- Selecciona desde el menú Add → New - Multidevice / FireMonkey Application.
- Elegimos una aplicación FireMonkey HD y comenzaremos por desarrollar una aplicación Windows 32 Bits.

### Diseño del formulario:

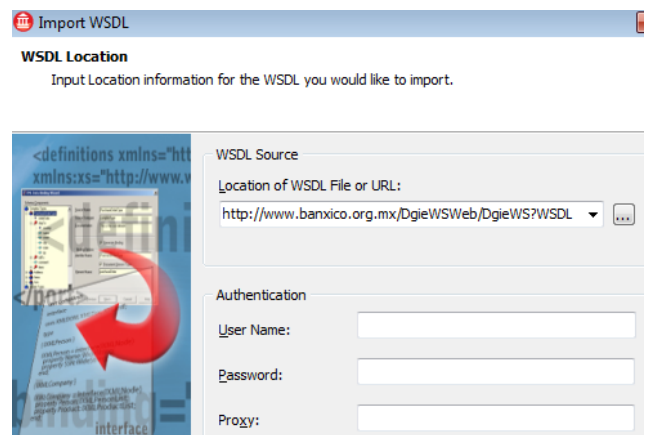
- Colocamos 2 Edit y 3 Labels como se muestra en la imagen:
  - Colocamos un *ListBox* y configuramos las siguientes propiedades: StyleLookup = transparentlistboxstyle XE4: GroupingKind = gsGrouped  
Clic derecho del mouse en el *ListBox* donde seleccionaremos la opción *AddItem* y agregamos: 5 *ListBoxItem*'s en XE 4: 2 *ListBoxGroupHeader*'s  
Cambiamos el texto de cada ítem en la propiedad *Text* por el nombre de cada moneda y dejamos la primera activada (Is selected).
- En Xe4 y XE5 puedes añadir las banderas de cada país en la propiedad *ItemData - Bitmap* También



### Servicio de datos web:

- El siguiente paso es generar nuestra clase que consumirá el Servicio Web del Banco de México y que nos proporcionará las paridades o tipos de cambio de las monedas que vamos a utilizar en nuestra aplicación, el WSDL de este servicio web es el siguiente: <http://www.banxico.org.mx/DgieWSWeb/DgieWS?WSDL>

- Escoge del menú: Component: Import – WSDL para importar el WSDL a nuestra aplicación con el asistente que se muestra. Escoger los valores por defecto. Al final nos generará un archivo pas.
- Este importador genera el código necesario para poder consumir el servicioWeb y ya sólo nos resta escribir el código para que nuestra aplicación se conecte y nos muestre las paridades para cada una de las monedas seleccionadas.
- Añadimos un añadir un componente THTTTPRIO de la carpeta webservices a nuestro formulario, y añadir la información necesaria a éste. Llamamos al método "tiposDeCambioBanxicoRequest" del servicio



- Al seleccionar la moneda deseada de la lista, pasamos el texto de cada moneda, al objeto de destino:

```
procedure TForm1.ListBox1Change(Sender:
TObject);
begin
    Label2.Text:=ListBox1.Items[ListBox1.ItemIndex];
    lbMonedaExt.ItemData.Bitmap := (Sender as TListBoxItem).ItemData.Bitmap;
end;
```

- Escribimos el código necesario en el evento OnExit del Edit2 donde capturamos la cantidad a convertir para que nos muestre el valor en euros de acuerdo a la moneda seleccionada.

```
procedure TForm1.Edit2Exit(Sender: TObject);
begin
    valorME := StringReplace(edit2.Text,'$',',',[rfReplaceAll]);
    valorME := StringReplace(valorME,',',',',[rfReplaceAll]);
    Edit1.Text:= format('%5.2m',[strtoCurr(ValorME)*ParidadSel])
end;
```

- Compilamos. Nuestro servidor de aplicaciones *PAServer* debe estar en ejecución para poder realizar la compilación de la aplicación.



## Aplicación Multi-device (Firemonkey) usando SQLite: Notas

Podemos usar **dbExpress** o **FireDAC** para conectarse a la Base de Datos. Para plataformas móviles, mejor usar las bases de datos **InterBase ToGo** o **SQLite**. Pueden ejecutarse sobre dispositivos iOS y Android.

› Escoge una nueva aplicación (File > New > Multi-Device Application)

### A Crear la base de Datos y la tabla en el Data Explorer

1. Ve al panel **Data Explorer**,

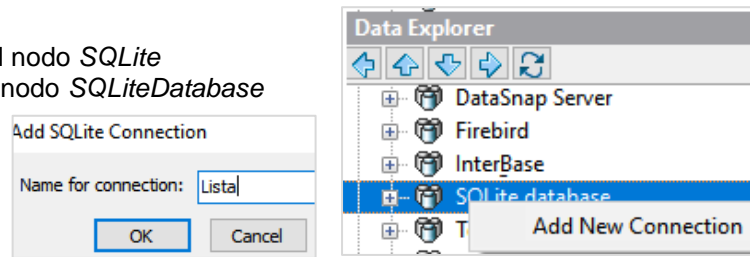
› En el modo DBExpress: Click-derecho en el nodo **SQLite**

› En el modo FireDAC: Click-derecho en el nodo **SQLiteDatabase**

Selección: **Add New Connection**:

2. Pon el nombre de la conexión,

Por ejemplo: **Lista**



3. Especifica la ubicación del archivo de base de datos:

Por ejemplo:

..\Documents\Embarcadero\Studio\Projects\Basesdatos\Lista

En el modo DBExpress (DBX)

En el cuadro de modificación, en avanzadas, busca la propiedad **FailIfMissing** y ponla a **False** (ordena crear un nuevo fichero de base de datos si no existe).

Nota: Si no existe la propiedad **FailIfMissing** puedes añadirla con el botón derecho: **add**

En el modo FireDAC: (FD)

De modo similar, pon la propiedad **OpenMode: Create UTF8**:

Si no existe la crea en formato UTF8.

5. Clica en el botón **Test** para probar la conexión y cierra la ventana con **OK**.

6. Arrastrar esta conexión sobre el formulario y te aparecerá el componente **tdfConexión: Listaconexión**.

7. En el Object Inspector, cambia las propiedades del **TFDConnection**:

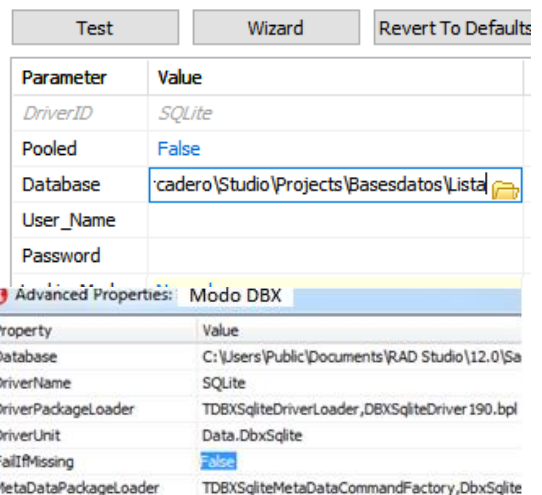
› **LoginPrompt** a **False** (sin loquearse).

› **Connected** a **True**.

Cambia el **Lockingmode** de **exclusive** a **normal**.

Nota1: Si se muestra el mensaje: **Failed: "sqlite3.dll not found"**

Asegúrate que el archivo: **sqlite3.dll** está en **system** o descargalo desde: <http://www.sqlite.org/download.html>.



### Crear la Tabla

a. En el modo DBX:

En el Data Explorer, click-derecho en **Tables**, y luego seleccione **New Table** desde el menú contextual

Escribe el nombre del campo: **Elemento** del tipo **TEXT** y guarda la tabla con el nombre: **Lista**.

b. En el modo FireDAC:

Añade al formulario un: **TFDQuery** y cambia las propiedades:

› **Name** property to **FDQueryCreateTable**.

› **SQL** property: **CREATE TABLE IF NOT EXISTS Item (Elemento TEXT NOT NULL)**

Click derecho en **FDQueryCreateTable** y escoge **Execute**.

c. Mediante un administrador sqlite..

La antigua versión del navegador Mozilla Firefox, permite instalar el complemento: **SQLite Manager**.

Para ejecutarlo: Pulsa la tecla **Alt** para activar el menú del Firefox y escoge: **Herramientas – SqliteManager**.

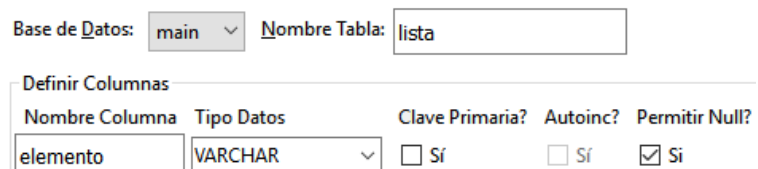
Otra opción es descargar e instalar la aplicación gratuita: **SQLite administrator** de [sqliteadmin.orbmu2k.de](http://sqliteadmin.orbmu2k.de)

Para crear la base de datos: **Base de datos – Nueva base de datos: Notas**, e indica la carpeta a guardar.

Para crear la tabla: Escoge del menú:

**Tabla – Crear tabla**. Utiliza los valores de la figura.

SQL: **CREATE TABLE "main"."lista"**  
(**"elemento"** VARCHAR)



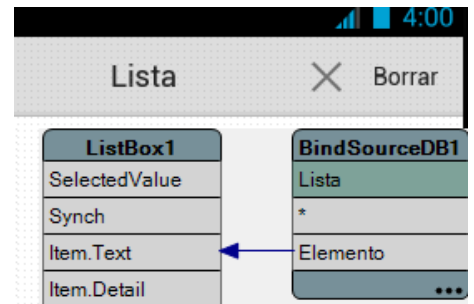
## B Crear el formulario para Android:

- 1º. Suelta un **TToolBar** sobre el Formulario.
- 2º. Suelta un **TButton** sobre el componente ToolBar. Cambia las propiedades:
  - Name: *ButtonAdd*.
  - StyleLookup: *addtoolbuttonbordered* o mejor: *addtoolbuttonbordered*. Align Left – Margin: 2
- 3º. Suelta un **TButton** sobre el componente ToolBar. Cambia las propiedades:
  - Name: *ButtonDelete*.
  - StyleLookup: *deletetoolbutton*.
  - Text: **Borrar**
  - Visible a False. Align Right – Margin: 2
- 4º. Suelte un **TLabel** sobre el componente ToolBar. Cambia las propiedades:
  - Align: *alClient*.
  - StyleLookup: *toollabel*.
  - Text: **Lista** o *Lista de tareas*
  - TextAlign o TextSettings - HorzAlign: *taCenter*.
- 5º. Arrastra un componente **TListBox** sobre el formulario.  
O un **TListView**. Cambia la propiedad: Align: *alClient*.



## Conectando a los Datos

- 1º. Suelta la tabla **Lista** si no lo has hecho ya, para crear el componente *TSQLConnection* y *TSQLDataSet* en el Formulario (en FireDAC: *ListaConection* y *ListaTable* o *NotasConection* y *NotasTable*)
- 2º. En el componente *Connection* del Formulario, tener la propiedad *Connected* en **True**.
- 3º. El componente **Lista** Table del Formulario y cambia la propiedad **Active** a True.
- 4º. Selecciona View > **LiveBindings Designer** o View > Tools o Tool Windows > *LiveBindings Designer* según versiones.  
O pulsa en la opción *Bind Visually* del *ListBox*
- 5º. Selecciona el elemento de la Lista y arrastra sobre *ItemText* del *ListBox1*. Como en la imagen.



Opcional: Usar el asistente *LiveBinding Wizard* porque además te añade un componente Navigator.

## Hacer Visible el Botón Delete

Seleccione *ListBox1* y defina el siguiente manejador de evento para el evento *OnItemClick*:

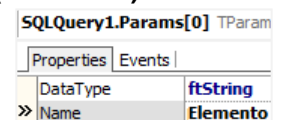
- › En Delphi: `ButtonDelete.Visible := ListBox1.Selected <> nil;`
- › En C++: `ButtonDelete->Visible = (ListBox1->Selected != NULL);`

## Agregar Entrada a la Lista con el botón + add

- › Suelta un componente **TSQLQuery** (Si usas DBExpress) o un **TFDQuery** (si usas FireDAC) al formulario.
- › En el *Object Inspector* cambia la Propiedad *Name* a *SQLQueryInsert* y conecta con *SQLConnection*
- › Poner la propiedad SQL como se muestra: **INSERT INTO lista (Elemento) VALUES (:Elemento)**
- › En la propiedad *Params*, expande (...).  
Selecciona el parámetro *Elemento* y cambia a tipo texto: *DataType* a *ftString*.
- › Pulsa en el formulario, doble-click sobre el componente *AddButton*.

Añade el siguiente código para el evento *OnClick*:

```
procedure TForm1.ButtonAddClick(Sender: TObject);
var
  Textoentrada: String;
begin
  try
    if InputQuery('Entre Nuevo elemento', 'Name', Textoentrada) and (Textoentrada.Trim <> '') then
      begin
        SQLQueryInsert.ParamByName('Elemento').AsString := Textoentrada;
        SQLQueryInsert.ExecSQL();
        lista.Refresh;
        ListBox1.Repaint;
        ButtonDelete.Visible:= ListBox1.Selected <> nil;
      end;
    except
      on e: Exception do
        begin
          ShowMessage(e.Message);
        end;
      end;
    end;
end;
```



En C++:

```
String caption = "Enter New Item";
String Prompts[1];
Prompts[0] = "Name:";
String Defaults[1];
Defaults[0] = "";
_di_TInputCloseQueryProc Met = new InputQueryMethod(&OnInputQuery_Close);
TDialogServiceAsync::InputQuery(caption, Prompts, 0, Defaults, 0, (TInputCloseQueryProc *)Met);
```

La función *InputQuery* muestra un cuadro de diálogo que pide al usuario final que introduzca un texto. Esta función devuelve *True* cuando el usuario selecciona OK, así que usted puede agregar datos a la base de datos sólo cuando el usuario selecciona OK y el texto contenga algunos datos

### Eliminar una Entrada de la Lista con el botón Delete

1. Suelte un componente TSQLQuery al formulario.
2. Establecer las siguientes propiedades en el Object Inspector:
  - Poner la propiedad Name a *SQLQueryDelete*.
  - Poner la propiedad SQLConnection a *SQLITECONNECTION*.
  - Poner la propiedad SQL como se muestra: **DELETE FROM LISTA WHERE elemento = :elemento**
  - Poner el botón Expand (...) sobre la propiedad Params.
  - Poner el parámetro Elelemnto a *DataType ftString*.
3. En el Form Designer, doble-click al componente *DeleteButton*.

Agregue el siguiente código a este manejador de evento:

```
procedure TForm1.ButtonDeleteClick(Sender: TObject);
var
    Textoentrada: String;
begin
    Textoentrada:= ListBox1.Selected.Text;
    try
        SQLQueryDelete.ParamByName('Elemento').AsString := Textoentrada;
        SQLQueryDelete.ExecSQL();
        Lista.Refresh;
        ButtonDelete.Visible := ListBox1.Selected <> nil;
    except
        on e: Exception do
            begin
                ShowMessage(e.Message);
            end;
        end;
    end;
end;
```

En C++:

```
{
    String TaskName = ((TListViewItem*)(ListView1->Selected))->Text;
    try {
        FDQueryDelete->ParamByName("Elemento")->AsString = TaskName;
        FDQueryDelete->ExecSQL();
        FDQuery1->Close();
        FDQuery1->Open();
        ButtonDelete->Visible = (ListView1->Selected != NULL);
    }
    catch (Exception &e) {
        ShowMessage(e.Message);
    }
}
```

Probar: Para realizar las pruebas de añadir y eliminar a la lista puedes realizarlas bajo la plataforma Win32

### Almacenar la base de datos local en el móvil:

Es necesario realizar las siguientes acciones:

Cambiar la configuración (para conectarse al archivo de base de datos) a un archivo local bajo el folder *Documents* (para Dispositivos iOS) o *internal storage* (para Dispositivos Android).

Antes de poder ejecutar la aplicación en el móvil, es necesario configurar el despliegue de su archivo de base de datos:

1. Abrir el **Deployment Manager** seleccionando *Project > Deployment*.
2. Seleccione *Add Files*, y seleccione el archivo de base de datos antes creado **Lista.db**
3. cambie Remote Path a **Startup\Documents\** (para iOS) o **assets\internal\** (para Android).
4. Seleccione la columna Platforms (doble-click en el ellipsis [...] en la línea de Lista.db):
  1. Asegúrese que iOS Simulator y iOS Device o Android están presentes para Lista.db
  2. Quita Win32 desde la lista si está presente (no hay que copiar archivos de base de datos para Win32).
  5. Cambia el combo desplegable superior a: *All-Configurations - iOS Device platform* o *All-Configurations - Android platform* y asegúrese que *Lista.db* está lista se inicia en *Startup\Documents\* o *assets\internal*.

### Especificando la Ubicación de la base de datos SQLite sobre el Dispositivo Móvil

1. En el Form Designer, seleccione el componente **Lista**.
2. En el Object Inspector, doble-click al evento BeforeConnect.
3. Añada el siguiente código al manejador de evento:

```
procedure TForm1.SQLConnectionSQLiteBeforeConnect(Sender: TObject);
begin
  {$IF DEFINED(iOS) or DEFINED(ANDROID)} //si la plataforma es IOS o Android ...
    Notas.Params.Values['Database'] := TPath.Combine(TPath.GetDocumentsPath, 'Notas.s3db');
  //-> 0 ...
  FDConnection1.Params.Values['Database'] :=
    TPath.Combine(TPath.GetDocumentsPath, 'shoplist.s3db');
  {$ENDIF}
end;
```

Agrega **System.IOUtils** en la **uses** clause de tu unidad para el registro *TPath*.

### Crear una Tabla si No Existe

Puedes crear una tabla cuando la tabla no existe, usando la sentencia CREATE TABLE IF NOT EXISTS. Puedes crear una tabla después que el componente SQLConnection se conecte a la base de datos y antes que el componente TSQLDataSet se conecte a la tabla. Usa los siguientes pasos:

1. En el *Form Designer*, selecciona el componente **ShoppingList** o **Notas**
2. En el *Object Inspector*, doble-click en el evento: *AfterConnect*.
3. Agrega el siguiente código para este evento:

```
procedure TForm1.ShoppingListAfterConnect(Sender: TObject);
begin
  ShoppingList.ExecuteDirect('CREATE TABLE IF NOT EXISTS Item (ShopItem TEXT NOT NULL)');
end;
```

### Comprueba la Aplicación en un móvil con Android:

- › Conecta el móvil por USB al PC, con permisos y de depuración USB activados y driver ADB android
- › En el *Project Manager*, selecciona la plataforma android.
- › Escoge alguno de los siguientes comandos: ▶ **Run > Run** o ▶ **Run > Run Without Debugging**

Comprueba su funcionamiento en el móvil.

Para ampliación, Véase:

[http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Mobile\\_Tutorial:\\_Using\\_FireDAC\\_and\\_SQLite\\_\(iOS\\_and\\_Android\)](http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Mobile_Tutorial:_Using_FireDAC_and_SQLite_(iOS_and_Android))

## Ejercicio 2 App mobile con SQLite y FireDAC Lista de tareas

### Crear la base de datos y la tabla:

- Si tienes instalado el complemento en Mozilla Firefox, escoge del menú(Alt): Herramientas – **SQLiteManager** o ejecuta **SQLite administrator** descargado de [sqliteadmin.orbmu2k.de](http://sqliteadmin.orbmu2k.de)
- Para crear la base de datos:  
Base de datos – Nueva base de datos: **Tareas**, e indica lugar a guardar: ...projects\Tareas\Tareas.sdb
- Creas la tabla **Tareas** con los campos que se muestran en la imagen.

Nombre Campo	Tipo Campo	Valor Por Omi...	Const
id	INTEGER	PRIMARY KEY AUTOINCREMENT	
titulo	VARCHAR(100)		NULL
tipo	VARCHAR(50)	Personal	NULL
fecha	DATE	CURRENT_DATE	NULL
detalle	VARCHAR(200)		NULL
finalizado	BOOLEAN	0	NULL

### Creas una aplicación Multi-device nueva en Delphi:

- Escoge una nueva aplicación (File-New-Multi-Device App)

### Crear la conexión con la base de datos

En el panel **Data Explorer, FireDAC**: Click-derecho en el nodo **SQLiteDatabase -** Selecciona: **Add New Connection.**

Pon el nombre de la conexión: **Tareas**

Abre la database: ...Projects\Tareas\Tareas.sdb

Establece: **Lockingmode: normal ; DateTimeFormat: DateTime**

- Arrastra la tabla **Tareas** sobre el formulario para que aparezcan los objetos: **TareasConnection** y **TareasTable**.

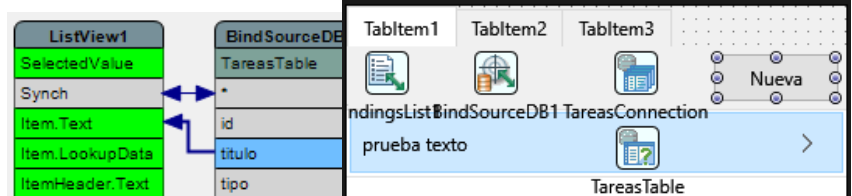
Activa sus propiedades: **Connected: True** y **Active: true.**

- Arrastra sobre el formulario un componente **tTabControl**. Añade, al menos, 3 items y alinea al cliente.

- En la primera pestaña, añade un **toolbar** con un botón a la derecha con el texto **Nueva**.

- Añade un **ListView** alineado al **cliente**. Pulsa el botón derecho encima y escoge: **Bind Visually.**

- Enlaza \* del **BindSourceDB** con **Synch** del **ListView1**, y el campo **titulo** con el elemento **Item.Text** como en la imagen. →



- Añade el código al Botón **Nueva**:  
`TabControl1.ActiveTab:=Tabitem1;`  
`tareasTable.Append;`

### Diseño del Tabitem2 para detalles.

Selecciona el segundo **tabTabitem2** y Activa el modo **Bind Visually** o **LiveBindings**

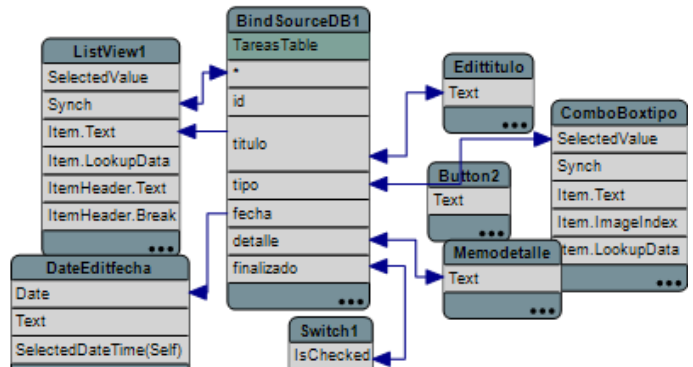
Para generar los controles enlazados:  
Pulsa el botón derecho sobre el **BindSource**: **titulo** y escoge: **Link to new Control.. TEdit.**

Sobre **tipo** →: **Link to new Control.. TComboBox**

Sobre **fecha** →: **Link to new Control.. TDateEdit.**

Sobre **detalle** →: **Link to new Control.. TDEdit.**

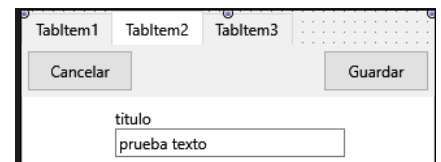
Sobre **finalizado** podríamos añadir un control **tCheckBox** o un **switch**.



Añadimos una **Toolbar** con dos botones para **Cancelar** y **Guardar**, como en la imagen. Con los siguientes códigos:

Para Guardar:  
`if tareasTable.State in [dsEdit,dsInsert] then`  
`tareasTable.Post;`  
`TabControl1.ActiveTab:=Tabitem1;`

Para Cancelar:  
`tareasTable.Cancel;`  
`TabControl1.ActiveTab:=Tabitem1;`



Para ocultar las pestañas, agregar al evento **OnFormShow**: `TabControl1.TabPosition:= TTabPosition.None;`

Para ir desde el **ListView** a la segunda pestaña. Agregar en el evento **OnItemClick**:

`TabControl1.ActiveTab:=Tabitem2;`  
`tareasTable.Edit;`

Botón para **Borrar**:  
`tareasTable.Delete;`  
`tareasTable.Refresh;`  
`TabControl1.ActiveTab:=Tabitem1;`

Componente **tSwitch** controlado de manera manual: En la segunda pestaña, añade y escribe en su evento **OnSwitch**:

`if Switch1.IsChecked then`  
`TareasTablefinalizado.Value:=True`  
`else TareasTablefinalizado.Value:=False;`

Opciones:

También puedes cambiar el código al pulsar en el botón para Guardar la tarea:

Luego se deberá actualizar el switch al cambiar de elemento en la lista listview on Change:

```
Switch1.IsChecked:=
TareasTablefinalizado.Value;
```

Otro modo más fácil es vincular el swithc visualmente con LiveBindings:

```
procedure TForm1.BGuardarClick(Sender: TObject);
begin
if tareasTable.State in [dsEdit,dsInsert] then
begin
if Switch1.IsChecked then
TareasTablefinalizado.Value:=True
else TareasTablefinalizado.Value:=False;
end;
tareasTable.Post;
TabControl1.ActiveTab:=Tabitem1;
end;
```

*Añadir imágenes en la lista Listview para control de finalizado:*

En el panel Structure, selecciona la lista listView1 – ItemAppearance. ImageListItemBottomDetail.

Añade una ImageList con los iconos: Listo y pendiente, vinculados al campo finalizado.

**Asignar conexión para dispositivo móvil:**

- › En el componente: TareasConnection, quitar la ruta del archivo Tareas.sdb y en el evento BeforeConect, añadir el código para la ruta:

```
{$IF DEFINED(iOS) or DEFINED(ANDROID)}
TareasConnection.Params.Values['Database'] := TPath.Combine(TPath.GetDocumentsPath, 'Tareas.sdb');
{$ENDIF}
```

- › Agrega **System.IOUtils** en la **uses** clause de tu unidad para el registro **TPath**.
- › Selecciona en el Project Manager, el target: Android.
- › Desde el explorador de Windows, arrastrar el archivo Tareas.sdb sobre el proyecto del Project manager,
- › Abrir el Deployment manager (Project – Deployment), busca el archivo Tareas.sdb y cambia la ruta remota por: *assets\internal\ para All configuration - Android Platform*. Deploy
- › Compila la aplicación en un dispositivo Android conectado. Ver: <https://youtu.be/K6yBTXwac7o>

Si existe error de DEF en la conexión FireDAC: Quitar ConectionDefName y poner una *ConectionName* en la *TareasConnection* y en la *Conectionname* de la *TFDquery* con el mismo nombre.

Es recomendable generar la conexión en ejecución:

```
with TareasConnection do
begin
Params.Clear;
DriverName:= 'SQLite';
Params.Values['Database']:=System.Ioutils.TPath.Combine(System.Ioutils.TPath.GetHomePath(), 'Tareas.sdb');
Connected:= True;
end;
TareasTable.SQL.Text:='SELECT * FROM Tareas';
TareasTable.Active:=true;
```

## Procedimientos y funciones básicas - Delphi (alfabético)

Abs	Returns the absolute value of the argument.	FloatToDecimal	Converts a floating-point value to a decimal representation.
AnsiCompareStr	Performs a case sensitive compare of two strings.	FloatToStrF	Converts the floating-point value to its string representation.
AnsiCompareText	Performs a non-case sensitive compare of two strings.	FloatToStr	Converts the floating-point value to its string representation.
AnsiLowerCase	Converts characters to lowercase.	FloatToText	Converts the given floating-point value to its decimal representation.
AnsiUpperCase	Converts characters to uppercase.	FloatToTextFmt	Converts the given floating-point value to its decimal representation.
Append	Opens an existing file for appending.	FmtLoadStr	Loads a string from a program's resource string table
AppendStr	Adds a given string to an existing string	FmtStr	Formats a series of arguments and the results are returned in the parameter Result.
ArcTan	Returns the arc tangent of the argument.	Format	Formats a series of arguments and returns the result as a Pascal string.
AssignCrt	Associates a text file with a CRT window.	FormatBuf	Formats a series of arguments
Assigned	Tests to determine if a procedural, pointer, or Delphi object variable is nil.	FormatDateTime	Formats a date and time using the specified format.
AssignFile	Assigns the name of an external file to a file variable.	FormatFloat	Formats the floating-point value using the format string given by Format.
BlockRead	Reads one or more records into a variable.	Frac	Returns the fractional part of the argument.
BlockWrite	Writes one or more records from a variable.	Free	Destroys an object instance.
Break	Terminates a for, while, or repeat statement.	FreeMem	Disposes a dynamic variable of a given size.
ChangeFileExt	Changes a file's extension.	GetDir	Returns the current directory of specified drive.
ChDir	Changes the current directory.	GetMem	Creates a dynamic variable of the specified size and puts the address of the block in a pointer variable.
Chr	Returns a character with a specified ordinal number.	GotoXY	Moves the cursor to the given coordinates within the virtual screen.
CloseFile	Closes an open file.	Halt	Stops program execution and returns to the operating system.
ClrEol	Clears all characters from the cursor position to the end of the line without moving the cursor.	Hi	Returns the high-order byte of the argument.
ClrScr	Clears the screen and returns the cursor to the upper left corner.	High	Returns the highest value in the range of the argument.
Concat	Concatenates a sequence of strings.	Inc	Increments a variable.
Continue	Continues a for, while, or repeat statement.	Include	Includes an element in a set.
Copy	Returns a substring of a string.	InitWinCrt	Creates the CRT window if it has not already been created.
Cos	Returns the cosine of the argument (x is an angle, in radians).	Insert	Inserts a substring into a string.
Date	Returns the current date.	Int	Returns the Integer part of the argument.
DateTimeToFileDate	Converts the Delphi date format to the DOS date format.	IntToHex	Converts an integer to a hexadecimal
DateTimeToStr	Converts a value from time format to a string.	IntToStr	Converts an integer to a string
DateTimeToString	Converts a value from time format to a string.	IOResult	Returns the status of the last I/O operation performed.
DateToStr	Converts a value from date format to a string.	IsValidIdent	Returns true if the given string is a valid identifier.
DayOfWeek	Returns the current day of the week	KeyPressed	Determines if a key has been pressed on the keyboard.
Dec	Decrements a variable.	Length	Returns the dynamic length of a string.
DecodeDate	Decodes the specified date	Ln	Returns the natural logarithm of the argument.
DecodeTime	Decodes the specifies time.	Lo	Returns the low-order Byte of the argument.
Delete	Deletes a substring from a string.	LoadStr	Loads the string resource from the application's executable file.
DeleteFile	Deletes the given file.	Low	Returns the lowest value in the range of the argument.
DiskFree	Returns the amount of free disk space.	LowerCase	Lowercases the given string
DiskSize	Returns the size of the specified disk.	MaxAvail	Returns the size of the largest contiguous free block in the heap.
EncodeDate	Returns values specified in date format.	MemAvail	Returns the amount of all free memory in the heap.
EncodeTime	Returns values specified in time format.	MkDir	Creates a subdirectory.
Eof	Returns the end-of-file status.	Move	Copies bytes from source to dest.
Eoln	Returns the end-of-line status of a text file.	New	Creates a new dynamic variable and sets a pointer variable to point to it.
Erase	Erases an external file.	NewStr	Allocates a new string on the heap.
Exit	Exits immediately from the current block.	Now	Returns the current date and time.
Exp	Returns the exponential of the argument.	Odd	Tests if the argument is an odd number.
ExpandFileName	Expands the given file name.	Ofs	Returns the offset of a specified object.
ExtractFileExt	Returns the file extension.	Ord	Returns the ordinal number of an ordinal-type value.
ExtractFileName	Returns the file name.	ParamCount	Returns the number of parameters passed to the program on the command line.
ExtractFilePath	Returns the complete file path.	ParamStr	Returns a specified command-line parameter.
FileAge	Returns the age of the file.	Pi	Returns the value of Pi.
FileClose	Closes a file.	Pos	Searches for a substring in a string.
FileDateToDateTime	Converts a DOS date format to the Delphi date format.	Pred	Returns the predecessor of the argument.
FileExists	Returns True if file exists.	Ptr	Converts a segment base and an offset address to a pointer-type value.
FileGetAttr	Returns file attributes	Random	Returns a random number.
FileGetDate	Returns the DOS date-and-time stamp of the file.	Randomize	Initializes the built-in random number generator with a random value.
FileOpen	Opens a specific file.		
FilePos	Returns the current file position of a file.		
FileRead	Reads from a specific file.		
FileSearch	Searches through the directories for a specific file.		
FileSeek	Changes the current position of the file.		
FileSetAttr	Sets file attributes		
FileSetDate	Sets the DOS date-and-time stamp of the file.		
FileSize	Returns the current size of a file.		
FillChar	Fills a specified number (count) of contiguous bytes with a specified value.		
FindClose	Terminates a FindFirst/FindNext sequence.		
FindFirst	Searches a directory for a specified file name and set of attributes.		
FindNext	Returns the next entry that matches the name and attributes.		

Read	For typed files, reads a file component into a variable. For text files, reads one or more values .	StrLIComp	Compares two strings, up to a maximum length, without case sensitivity.
ReadBuf	Inputs a line from the CRT window.	StrLower	Converts a string to lower case.
ReadKey	Reads a character from the keyboard.	StrMove	Copies characters from one string to another.
ReadLn	Executes the Read procedure, then skips to the next line of the file.	StrNew	Allocates a string on a heap.
ReAllocMem	Deallocates a block from the heap.	StrPas	Converts a null-terminated string to a Pascal-style string.
Rename	Renames an external file.	StrPCopy	Copies a Pascal-style string to a null-terminated string.
RenameFile	Renames a file.	StrPos	Returns a pointer to the first occurrence of a string in another string.
Reset	Opens an existing file.	StrScan	Returns a pointer to the first occurrence of a character in a string.
Rewrite	Creates and opens a new file.	StrRScan	Returns a pointer to the last occurrence of a character in a string.
RmDir	Removes an empty subdirectory.	StrToDate	Converts a string to a date format.
Round	Rounds a real-type value to an Integer-type value.	StrToDateTime	Converts a string to a date/time format.
RunError	Stops program execution.	StrToFloat	Converts the given string to a floating-point value.
ScrollTo	Scrolls the CRT window to show a screen location.	StrToInt	Converts a string to an integer
Seek	Moves the current position of a file to a specified component.	StrToIntDef	Converts a string to an integer or the default
SeekEof	Returns the end-of-file status of a file.	StrToTime	Converts a string to a time format.
SeekEoln	Returns the end-of-line status of a file.	StrUpper	Converts a string to upper case.
Seg	Returns the segment of a specified object.	Succ	Returns the successor of the argument.
SetTextBuf	Assigns an I/O buffer to a text file.	Swap	Swaps the high- and low-order bytes of the argument.
Sin	Returns the sine of the argument.	TextToFloat	Converts the null-terminated string to a floating-point value.
SizeOf	Returns the number of bytes occupied by the argument.	Time	Returns the current time.
SPtr	Returns the current value of the SP register.	TimeToStr	Converts a time format to a string.
Sqr	Returns the square of the argument.	TrackCursor	Scrolls the CRT window if necessary to ensure that the cursor is visible.
Sqrt	Returns the square root of the argument.	Trunc	Truncates a real-type value to an Integer-type value.
SSeg	Returns the current value of the SS register.	Truncate	Truncates the file at the current file position.
Str	Converts a numeric value to a string.	TypeOf	Returns a pointer to an object type's virtual method table.
StrCat	Appends a copy of one string to the end of another and returns the concatenated string.	UpCase	Converts a character to uppercase.
StrComp	Compares two strings.	UpperCase	Uppercases the given string.
StrCopy	Copies one string to another.	Val	Converts a string value to its numeric representation.
StrDispose	Disposes a string on a heap.	WhereX	Returns the X coordinate of the current cursor location.
StrECopy	Copies one string to another and returns a pointer to the end of the resulting string.	WhereY	Returns the Y coordinate of the current cursor location.
StrEnd	Returns a pointer to the end of a string.	Write	For typed files, writes a variable into a file component. For text files, writes one or more values to the file
StrFmt	Formats a series of arguments.	WriteBuf	Writes a block of characters to the CRT window.
StrLCat	Appends characters from one string to the end of another and returns the concatenated string.	WriteChar	Writes a single character to the CRT window.
StrLComp	Compares two strings without case sensitivity.	Writeln	
StrLComp	Compares two strings, up to a maximum length.		
StrLCopy	Copies characters from one string to another.		
StrLen	Returns the number of characters in Str.		
StrLFmt	Formats a series of arguments and the result contains a pointer to the destination buffer.		