

PROGRAMANDO EN SERIO CON VISUAL BASIC 6.0

Segunda Edición

DAED
S·O·F·T·W·A·R·E

DAED Software y todas las obras, diseños y software bajo este nombre son propiedad intelectual de Andrés Escobar. Esta obra es LIBRE y esta totalmente permitida su copia, publicación y distribución.

Andres Escobar Duque
DAED Software - daedsoftware@yahoo.es

Programando en Serio con Visual Basic 6.0 (Segunda Edición)

2008

Visual Studio ® Visual Basic ® Microsoft Office ®
Son marcas registradas de Microsoft Corporation

Lo nuevo de la Segunda Edición

La segunda edición de este curso se ha diseñado con el fin de pasar a un nivel mas avanzado en el diseño de aplicaciones y manejo de VB 6.0. El formato del libro se ha cambiado a PDF para garantizar la integridad del curso y su distribución.

El curso conserva los temas antiguos pero con algunas modificaciones que incluyen ampliación de la información y nuevos ejemplos. Como nuevos temas el curso incluye:

- Manipulación de registros y campos de una base de datos
- Consultas con SQL
- Diseño de la IGU (Interfaz Grafica de Usuario)
- Sonidos para la interfaz

Contenido

- Controles intrínsecos (definición, uso)
- Controles extrínsecos (definición, uso)
- Propiedades de un proyecto
- Elementos de un proyecto
- Eventos, procedimientos
- Algunas sentencias

- Manejo de bases de datos
- **Manipulación de registros y campos de una base de datos**
- **Consultas con SQL**

- La API de Windows
- Funciones

- Activex - Creación de un control Activex

- **Diseño de la IGU (Interfaz Grafica de Usuario)**
- **Sonidos para la interfaz**

- Optimizar la Programación

Prologo

Todos tenemos una gran capacidad de auto - enseñanza, este y muchos otros manuales y cursos estimulan ese sentido de autoaprendizaje que tienes tu (lector), y de esto estoy seguro, sino, no te hubieras tomado la molestia de descargar este curso.

Entrando en materia, este no es exactamente un curso para "sabiondos" es un introductorio a la programación avanzada, es por así decirlo un empujoncito para aquellos que ya saben lo básico y quieren saber mas, así que por esta razón algunas explicaciones y ejemplos no van a excederse en "complicadas" y mantendré un nivel básico en algunos temas.

Como es costumbre mía, no hay mucho aquí para los expertos, ni para todos aquellos que me escriben diciéndome que mis tutoriales y cursos son muy sencillos, así son, porque así me lo propongo. Además dar todo el material "complicado" no tiene mucho sentido, ya que un verdadero programador se hace a si mismo y se desvela mucho tratando de encontrar soluciones. No pierdas las ganas y no esperes que todo te lo den listo.

Controles Intrínsecos

Los controles intrínsecos son aquellos que aparecen "cargados por defecto" en el cuadro de herramientas con abres VB. Estos se diferencian de los extrínsecos, por que una aplicación creada únicamente con estos controles (y sin agregar referencias o librerías) se puede distribuir sin instalador (algunas veces).

TextBox (Caja de texto) :

Se utiliza para el ingreso y validación de datos o información.

Algunas Propiedades

Text: Almacena la cadena de caracteres que contenga el control.

DataFiled: Enlaza en control con un campo de base de datos.

DataSource: Enlaza el control con un origen de manipulación de base de datos.

MultiLine: Permite escribir varias líneas en el control.

ScrollBars: Agrega barras de desplazamiento al control si esta MultiLine = True

PasswordChar: Es el carácter que ocultara el texto que se escribe en el control, por ejemplo PasswordChar = * (Asterisco).

Locked = Permite bloquear la escritura sobre el Textbox.

Hacer que un Textbox solo reciba números:

Podemos utilizar el siguiente condicional en el evento Validate que no nos permitirá salir del control si lo escrito en él no son números:

```
Private Sub Text1_Validate(Cancel As Boolean)
```

```
If Not IsNumeric(Text1.Text) Then
```

```
Cancel = True
```

```
Else
```

```
Text2.SetFocus
```

```
End If
```

```
End Sub
```

Pero para que las teclas "de letras" o "símbolos" no se puedan escribir es necesario:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
```

```
If KeyAscii <> Asc("9") Then
```

```
If KeyAscii <> Asc("8") Then
```

```
If KeyAscii <> Asc("7") Then
```

```
If KeyAscii <> Asc("6") Then
```

```
If KeyAscii <> Asc("5") Then
If KeyAscii <> Asc("4") Then
If KeyAscii <> Asc("3") Then
If KeyAscii <> Asc("2") Then
If KeyAscii <> Asc("1") Then
If KeyAscii <> Asc("0") Then
If KeyAscii <> 8 Then
If KeyAscii <> 32 Then
KeyAscii = 0
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End Sub
```

Formatear un Textbox

Esto se utiliza para que un Textbox muestre un numero separados por unidades y decimales.

```
Private Sub Text1_LostFocus()
If Text1.Text = "" Then
    Text1.Text = "0,00"
End If
Text1 = Format(CCur(Text1.Text), "###,###,###,##0.00")
End Sub
```

Para realizar operaciones matemáticas con un Textbox formateado se debes cambiar la función **Val** por **Cdbl**, ejemplo:

```
txResultado = Cdbl(Text1) + Cdbl(Text2)
```

Hacer que un TextBox solo escriba en Mayúsculas

Independientemente de que este activado el BloqMayus, el textbox siempre mostrara texto en mayúsculas.

```
Private Sub Text1_Change()
Text1.Text = UCase(Text1.Text)
```

```
Text1.SelStart = Len(Text1)
End Sub
```

Autoseleccionar el texto de un textbox al recibir el foco

```
Private Sub Text1_GotFocus()
Text1.SelStart = 0
Text1.SelLength = Len(Text1)
End Sub
```

CommandButton (Botón de Comando):

Este control permite encapsular procedimientos para que sean utilizados por el usuario con solo hacer click sobre él.

Algunas Propiedades

Cancel: Permite ejecutar el código escrito en el control presionando "Esc"

Default: Permite ejecutar el código escrito en el control presionando "Enter"

Style: Puede ser Graphical o Standard. Si es Graphical permite cambio de color entre otras.

Picture: Carga una imagen para el CommandButton si su propiedad Style es Graphical.

Label (Etiqueta):

Muestra la información, los títulos o los subtítulos que deseemos.

Algunas Propiedades

Alignment: Permite alinear el texto hacia la izquierda, derecha o centro.

BackStyle: Permite hacer que el fondo de la etiqueta sea transparente o solidó.

AutoSize: Ajusta el tamaño del control a la longitud del texto.

WordWrap: Muestra el texto (cuando es mucho) en varias líneas.

Hacer que una etiqueta cambie de color al pasar el mouse

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Label1.ForeColor = vbBlack
End Sub
```



```
Private Sub Label1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Label1.ForeColor = vbBlue
End Sub
```

CheckBox (Casilla de verificación)

Permite que se ejecute un procedimiento dependiendo de la verificación del Checkbox

Algunas Propiedades

Style: Si es Graphical convierte el control en un botón de chequeo.

Value: Activa o desactiva el valor de verificación. Cheked (Verifica), Unchecked (Desverifica) o Grayed (Inhabilitado)

Cambiar una etiqueta dependiendo del valor del ChekBox

```
Private Sub Check1_Click()
If Check1.Value = 1 Then
Label1.Font.Bold = True
Else
Label1.Font.Bold = False
End If
End Sub
```

OptionButton (Botón de opción)

Permite que un código se ejecute teniendo en cuenta el valor de este control.

Algunas Propiedades

Value: Puede ser (True) mostrando un punto negro o (False) quitándolo.

Mostrar la opción elegida

```
If Option1.Value Then
Text1.Text = " 01 Seleccionada Opcion 1"
Elseif Option2.Value Then
Text1.Text = " 02 Seleccionada Opcion 2"
End If
```

ComboBox (Caja Combo)

Permite escoger de una lista de opciones creadas un elemento.

Algunas Propiedades

List: Permite crear uno a uno los elementos de la lista del ComboBox

Style: Permite cambiar a un combo simple o lista.

Sorted: Si es (True) organiza los elementos de su lista alfabéticamente.

Agregar elementos mediante código

```
Combo1.AddItem ("Elemento1")
```

```
Combo1.AddItem ("Elemento2")
```

```
Combo1.AddItem ("Elemento3")
```

Realizar procedimientos dependiendo el elemento seleccionado

```
Private Sub Combo1_Click()
```

```
Select Case Combo1.Text
```

```
Case ("Elemento1")
```

```
' Procedimiento
```

```
Case ("Elemento2")
```

```
' Procedimiento
```

```
Case ("Elemento3")
```

```
' Procedimiento
```

```
End Select
```

```
End Sub
```

Cargar datos a un ComboBox desde una tabla

```
Dim X As Integer
```

```
If Tabla.EOF And Tabla.BOF Then Exit Sub
```

```
Tabla.MoveFirst
```

```
For X = 1 To Tabla.RecordCount
```

```
On Error Resume Next
```

```
Combo1.AddItem Tabla("Campo")
```

```
Tabla.MoveNext
```

```
Next X
```

ListBox (Lista)

Muestra datos o campos en forma de lista.

Algunas Propiedades

Style: Standard (Normal) o CheckBox (Una lista con casillas de verificación)

Agregar campos de base de datos a un ListBox

```
Private Sub Form_Load()  
    Tabla.MoveFirst  
    Dim N As Integer  
    For N = 1 To Tabla.RecordCount  
        Lista.AddItem Tabla ("nombrecampo")  
        Tabla.MoveNext  
        If Lista.ListIndex = Tabla.RecordCount Then  
            Exit For  
        End If  
    Next N  
End Sub
```

Quitar un elemento

```
List1.RemoveItem List1.ListIndex
```

Timer (Temporizador)

Realiza un procedimiento en un intervalo o intervalos de tiempo determinados.

Algunas Propiedades

Interval: Permite ingresar el valor del intervalo de tiempo. Un intervalo de 1000 equivale a 1 segundo.

Enabled: Si es (False) pausa al temporizador, si es (True) lo activa.

Hacer una etiqueta parpadeante

Colocar 50 como valor en la propiedad Interval.

```
Private Sub Timer1_Timer()  
    Static C As Integer  
    C = C + 1  
    If C = 1 Then
```

```
Label1.ForeColor = vbBlue  
Elseif C = 2 Then  
Label1.ForeColor = vbRed  
Elseif C >= 3 Then  
C = 0  
End If  
End Sub
```

OLE

Permite enlazar un archivo externo a VB a un programa creado por nosotros.

DriveListBox (Lista de unidades)

Muestra un combo con la lista de unidades del ordenador.

DirListBox (Lista de carpetas)

Muestra las carpetas del ordenador.

FileListBox (Lista de archivos)

Muestra los archivos del ordenador.

Los demás controles no serán explicados porque si ya sabes lo básico, ya sabes eso.

Controles extrínsecos

Son controles que se agregan a una aplicación por la vía "Componentes", estos son controles Activex, creados por el usuario o los que se instalan con Visual Studio. El uso de estos controles en una aplicación requiere la creación de un instalador para su distribución.

CommonDialog (Cuadro de dialogo)

Este control permite agregar a una aplicación los típicos cuadros de dialogo como: Abrir, Guardar Como, Imprimir, Paleta...

Para agregar un CommonDialog, ve a "Proyecto > "Componentes" y agregas:

Microsoft Common Dialog Control 6.0 (SP3)

Abrir un archivo para MediaPlayer 8

```
Private Sub Command1_Click()  
CommonDialog1.Filter = "Video (*.mpeg) *.mpeg|Musica (*.mp3) *.mp3|Todos los archivos (*.*)  
*.*)" "  
CommonDialog1.FilterIndex = 1  
CommonDialog1.ShowOpen  
lbNom.Caption = CommonDialog1.FileName  
MediaPlayer1.FileName = lbNom  
End Sub
```

Guardar como archivo de texto el contenido de un TextBox "txtEditor"

```
Private Sub mnuArchivoGuardarComo_Click()  
' Gestionar errores  
On Error GoTo GuardarComoProblema  
CommonDialog1.Filter = "Archivos de texto|*.TXT|Archivos de lotes|*.BAT|Archivos  
INI|*.INI"  
CommonDialog1.FilterIndex = 1  
CommonDialog1.Action = 2  
Open CommonDialog1.FileName For Output As 1  
Print #1, txtEditor.Text  
Close 1  
Exit Sub  
GuardarComoProblema:  
MsgBox Err.Description  
End Sub
```

Imprimir el contenido de un TextBox

```
Private Sub Command1_Click()  
On Error Resume Next  
With CommonDialog1  
.DialogTitle = "Imprimir"  
.Flags = cdlPDHidePrintToFile Or cdlPDNoPageNums Or cdlPDUseDevModeCopies  
.CancelError = True  
.ShowPrinter  
End With  
  
If Err = 0 Then  
Printer.Print Text1.Text  
Printer.EndDoc  
End If  
Err = 0  
End Sub
```

DTPicker (Recogedor de Fecha)

Este control es como un Combo Calendario, que permite elegir una fecha específica buscando el día, mes y año.

Para agregar DTPicker elige en "Componentes":

Microsoft Windows Common Controls-2 6.0 (SP4)

Para guardar una fecha elegida con DTPicker en una variable o en una etiqueta, se utiliza la propiedad Value de este así: **DTPicker1.Value**

ToolBar (Barra de Herramientas) & ImageList (Lista de Imágenes)

El control ToolBar permite crear eso; una barra de herramientas con la ayuda del control ImageList.

Para agregar ToolBar e ImageList seleccionamos de "Componentes":

Microsoft Windows Common Controls 6.0 (SP6)

Cargar imágenes al ImageList

Click derecho sobre el ImageList1 > "Propiedades"

En "General" elegimos el tamaño en que se verán las imágenes o iconos.

En la pestaña "Imágenes" hay un botón llamado "Insertar Imagen", este nos abre un cuadro de diálogo de "Buscar en" el cual nos permite cargar las imágenes una a una.

Luego de cargar las imágenes...

Crear la barra de herramientas

Click derecho sobre ToolBar1 > "Propiedades"

En "General" elegimos [ImageList1](#) para ImageList.

Y elegimos [1- tbrFlat](#) en Style.

En la pestaña "Botones" con el botón "Insertar" agregamos botones.

Llenar el "Caption" no es obligatorio ([preferiblemente no llenar](#))

Llenar "Key" SI es obligatorio, (con lo que se escriba aquí se programa el toolbar)

ToolTipText (no es obligatorio) muestra lo que escribamos aquí cuando pongamos el cursor sobre el botón del Toolbar.

Description (no es obligatorio llenar)

Image: aparece con un 0 (cero), para agregar una imagen se pone allí el numero de la imagen, es decir si quiero poner a ese botón la primera imagen del ImageList ponemos el numero 1. y así sucesivamente para otros botones.

Programar la barra de herramientas

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
```

```
Select Case Button.Key
```

```
Case "nuevo"
```

```
' Procedimiento
```

```
Case "guardar"
```

```
' Procedimiento
```

```
Case "borrar"
```

```
' Procedimiento
```

```
End Select
```

```
End Sub
```

"nuevo", "guardar" y "borrar" son los "Keys" de los botones.

MMControl (Control MultiMedia)

Permite reproducir varios archivos de audio.

Para agregar un MMControl en "Componentes" eliges:

Microsoft Multimedia Control 6.0

Reproducir un CD de Audio

```
Private Sub Form_Load()  
Form1.Show  
MMControl1.Notify = False  
MMControl1.Wait = True  
MMControl1.Shareable = False  
MMControl1.Command = "Close"  
MMControl1.DeviceType = "CDAudio"  
MMControl1.Command = "Open"  
End Sub  
  
Private Sub Form_Unload(Cancel As Integer)  
MMControl1.Command = "Stop"  
MMControl1.Command = "Close"  
End Sub  
  
Private Sub MMControl1_StopCompleted(Errorcode As Long)  
MMControl1.From = 1  
End Sub
```


Propiedades de un proyecto

Un proyecto tiene una serie de características u opciones que se pueden configurar entre las cuales se encuentran: Nombre del proyecto, versión, tipo de proyecto, título de la aplicación, icono, tipo de compilación, objeto inicial, entre otros.

Para ver y configurar las propiedades de un proyecto... click en "Proyecto" > "Propiedades de [NombreProyecto]"

Entre las opciones de "Propiedades" cabe destacar:

Ficha "General"

- **Tipo de Proyecto:** Permite cambiar el tipo de proyecto elegido, puede ser EXE Estándar, EXE Activex, DLL Activex o Control Activex.
- **Objeto Inicial:** Permite escoger el formulario u objeto que se mostrara primero cuando se ejecute el programa.
- **Nombre de Proyecto:** Es el nombre del proyecto creado, este es el nombre que sale en la ventana "Complementos" cuando creamos un componente Activex.

Ficha "Generar"

- **Aplicación:**

Título: Es el nombre que tendrá el programa en el administrador de tareas y en "Agregar o Quitar Programas" del panel de control.

Icono: Pone como icono de la aplicación el del formulario seleccionado.

- **Numero de Versión:** Es eso, el numero de la versión del programa.

Elementos de un proyecto

Para estudiar los elementos de un proyecto, tomare como ejemplo un EXE Standard.

Un proyecto Standard puede tener:

● Formularios

Es una ventana en blanco en la cual “dibujamos” los controles. Los formularios o “ventanas” pueden ser hijas o madres. Cuando una ventana es hija significa que se mostrara dentro de una ventana y si es madre será la ventana que contenga a las demás, por ejemplo el programa “Word” la hoja que lees es una ventana hija de la ventana madre que es donde están los menús, barras de herramientas, etc.

Para crear un programa que utilice una ventana o formulario madre e hija...

Agrega un formulario MDI (del menú Proyecto), luego agrega un formulario standard y coloca su propiedad MDIChild = True

Visual Basic trae la opción de agregar formularios “prediseñados”, al elegir “Agregar formulario” en esta venta aparecen opciones de formularios como, “Cuadro de dialogo Acerca de...”, “Cuadro de dialogo Inicio de sesión (el de la contraseña☺)”, “Explorador web”, “Pantalla de inicio (o Splash)” entre otros... Estos formularios ya traen una parte del código hecho, pero para que funcionen de verdad se necesita agregar los procedimientos requeridos.

Algunas Propiedades

WindowState = Permite elegir el estado del formulario o ventana cuando se ejecute, puede ser minimizado, maximizado o normal.

StarUpPosition = Aquí se elige la ubicación en la pantalla del formulario.

ShowInTaskBar = Si es False no aparece un botón de la ventana en la barra de tareas cuando se ejecuta el programa.

Picture = Sirve para agregar un fondo de archivo (ya sea una imagen por ejemplo) al formulario.

ControlBox = Si es False se quitan los botones “Minimizar - Maximizar/Restaurar - Cerrar” de la barra de titulo.

BorderStyle = Nos permite elegir el tipo de borde o mas bien el tipo de ventana o formulario que queremos.

Como crear un formulario de inicio (Splash)

Colocar la propiedad ControlBox = False; BorderStyle = 3 - FixedDialog y borra el "Caption". Luego colocamos las etiquetas o imágenes y agregamos un temporizador el cual tendra su propiedad Interval = 1000. y en su código ...

```
Private Sub Timer1_Timer()  
Static C As Integer  
C = C + 1  
If C = 3 Then  
Unload Me  
Form2.Show  
End If  
End Sub
```

En donde Form2 es el formulario que se mostrara después del Splash.

Mover un formulario sin la barra de titulo (Tomado de "El Guille")

```
Const WM_SYSCOMMAND As Long = &H112&  
Const MOUSE_MOVE As Long = &HF012&
```

```
Private Declare Function ReleaseCapture Lib "user32" () As Long  
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd  
As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
```

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)  
If Button = vbLeftButton Then  
moverForm  
End If  
End Sub
```

```
Private Sub moverForm()  
Dim res As Long  
,  
ReleaseCapture  
res = SendMessage(Me.hWnd, WM_SYSCOMMAND, MOUSE_MOVE, 0)  
End Sub
```

El procedimiento "moverForm" se puede aplicar a cualquier control en su evento MouseDown escribiendo el código como aparece en "Form_MouseDown"

● Módulos

Son archivos que se anexan al los programas, son como un especie de bloque compacto de código que agiliza la ejecución de los programas ya que por ejemplo si estamos creando un programa muy complejo que tiene muchos formularios y que varios de estos comparten código seria mas fácil escribir este en un modulo y así hacerlo disponible para cualquier formulario. Existen 2 clases de módulos:

Módulos estándar: Los módulos estándar (extensión de nombre de archivo .bas) son contenedores de los procedimientos y declaraciones a los que tienen acceso otros módulos (Formularios) de la aplicación.;

Módulos de clase: Los módulos de clase (extensión de nombre de archivo .cls) son la base de la programación basada a objetos en Visual Basic. Puede escribir código en módulos de clase para crear nuevos objetos. Estos objetos nuevos pueden incluir propiedades y métodos personalizados. En realidad, los formularios sólo son módulos de clase que pueden tener controles y que pueden mostrar ventanas de formulario.

● Controles de usuario

El elemento “Control de usuario” permite crear nuestros propios controles, ya sea una caja de texto personalizada, un botón o cualquier clase de control, como un reproductor de audio o de video en fin lo que uno quiera, para crear un control es necesario entender de tecnología “Activex” que explicare mas adelante. Así que el ejemplo practico lo veras en el tema “Activex”

● Data Enviroment

Este es una “Conexión” a datos (es decir Base de Datos), permite crear una conexión o enlace a las bases de datos para por ejemplo hacer un reporte, aunque su principal función sea la del enlace de datos a través de Internet.

● Data Report

Es un elaborado reporte de una base de datos que permite además de visualizar los datos, imprimirlos o exportarlos a otro tipo de archivo, permite gran flexibilidad en la manera en que se muestran los datos, ya que prácticamente se muestran en el lugar que “se te de la gana” ponerlos. Este

elemento también permite agregar etiquetas, cajas de texto, imágenes, figuras e incluso funciones.

Estos son los principales elementos de un EXE Standard, y no menciono a el “Web Class” ni al “DHTML Page” porque estos no están disponibles para un EXE Standard.

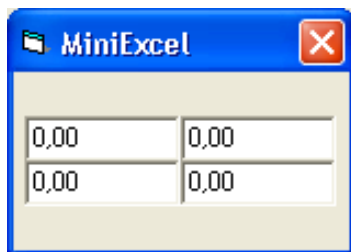
Eventos, procedimientos

Como se supone que esto es un curso avanzado, no voy a explicar que son los eventos y procedimientos, vamos al grano con ejemplos prácticos ☺

Imitación de una hoja Excel

Para este ejemplo vamos a utilizar los eventos del teclado “KeyPress”, “KeyDown”, Además el evento “GotFocus” y “LostFocus” y el “FormUnload”

Agrega 4 cajas de texto a un formulario así:



La idea es:

- ☺ Desplazarnos por las cajas de texto con las teclas direccionales.
- ☺ Cuando una caja reciba el cursor, su contenido se seleccione.
- ☺ Que las cajas solo reciban números.
- ☺ Que los números se muestren en formato decimal
- ☺ Y que al cerrar el programa nos pregunte si estamos seguros de salir.

Pon la propiedad “Text” de todas las cajas de texto a “0,00”

Para desplazarnos con las direccionales...

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = vbKeyRight Then
Text2.SetFocus
ElseIf KeyCode = vbKeyDown Then
Text3.SetFocus
End If
End Sub
```

```
Private Sub Text2_KeyDown(KeyCode As Integer, Shift As Integer)
```

```
If KeyCode = vbKeyLeft Then  
Text1.SetFocus  
ElseIf KeyCode = vbKeyDown Then  
Text4.SetFocus  
End If  
End Sub
```

```
Private Sub Text3_KeyDown(KeyCode As Integer, Shift As Integer)  
If KeyCode = vbKeyRight Then  
Text4.SetFocus  
ElseIf KeyCode = vbKeyUp Then  
Text1.SetFocus  
End If  
End Sub
```

```
Private Sub Text4_KeyDown(KeyCode As Integer, Shift As Integer)  
If KeyCode = vbKeyLeft Then  
Text3.SetFocus  
ElseIf KeyCode = vbKeyUp Then  
Text2.SetFocus  
End If  
End Sub
```

KeyCode nos permite capturar la tecla pulsada en el evento KeyDown, si la tecla pulsada es “vbKeyLeft” (Direccional Izquierda) por ejemplo; se lleva a cabo el procedimiento “Text#.SetFocus” y así el cursor pasa a otra caja.

Para capturar la tecla pulsada podemos escribir su respectiva constante. Algunas constantes son:

Direccional Arriba	= vbKeyUp
Direccional Abajo	= vbKeyDown
Direccional Izquierda	= vbKeyLeft
Direccional Derecha	= vbKeyRight
Enter	= vbKeyReturn
Escape	= vbKeyEscape
Control	= vbKeyControl
Barra espaciadora	= vbKeySpace

Las letras...

A ... Z = vbKeyA ... vbKeyZ

Los números...

1 ... 0 = vbKey1 ... vbKey0

Las de funciones...

F1 ... F12 = vbKeyF1 ... vbKeyF12

(Para conocer más constantes consulta la ayuda de Visual Studio)

Para que solo reciban números...

Copia este código dentro de cada Evento **KeyPress** de los 4 **TextBox**

```
If KeyAscii <> Asc("9") Then
If KeyAscii <> Asc("8") Then
If KeyAscii <> Asc("7") Then
If KeyAscii <> Asc("6") Then
If KeyAscii <> Asc("5") Then
If KeyAscii <> Asc("4") Then
If KeyAscii <> Asc("3") Then
If KeyAscii <> Asc("2") Then
If KeyAscii <> Asc("1") Then
If KeyAscii <> Asc("0") Then
If KeyAscii <> 8 Then
If KeyAscii <> 32 Then
KeyAscii = 0
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
```

Para que el contenido se seleccione al recibir el cursor...

```
Private Sub Text1_GotFocus()
Text1.SelStart = 0
Text1.SelLength = Len(Text1.Text)
End Sub
```

```
Private Sub Text2_GotFocus()
Text2.SelStart = 0
Text2.SelLength = Len(Text2.Text)
```



```
End Sub
```

```
Private Sub Text3_GotFocus()  
Text3.SelStart = 0  
Text3.SelLength = Len(Text3.Text)  
End Sub
```

```
Private Sub Text4_GotFocus()  
Text4.SelStart = 0  
Text4.SelLength = Len(Text4.Text)  
End Sub
```

Mostrar los números en formato decimal...

```
Private Sub Text1_LostFocus()  
If Text1.Text = "" Then  
Text1.Text = "0,00"  
End If  
Text1 = Format(CCur(Text1.Text), "###,###,###,##0.00")  
End Sub
```

```
Private Sub Text2_LostFocus()  
If Text2.Text = "" Then  
Text2.Text = "0,00"  
End If  
Text2 = Format(CCur(Text2.Text), "###,###,###,##0.00")  
End Sub
```

```
Private Sub Text3_LostFocus()  
If Text3.Text = "" Then  
Text3.Text = "0,00"  
End If  
Text3 = Format(CCur(Text3.Text), "###,###,###,##0.00")  
End Sub
```

```
Private Sub Text4_LostFocus()  
If Text4.Text = "" Then  
Text4.Text = "0,00"  
End If  
Text4 = Format(CCur(Text4.Text), "###,###,###,##0.00")  
End Sub
```

Y finalmente para comprobar la salida del programa...

```
Private Sub Form_Unload(Cancel As Integer)
Dim M
M = MsgBox("¿Esta seguro de salir?", vbYesNo, "MiniExcel")
If M = vbYes Then
    Unload Me
End
Else
    Cancel = True
End If
End Sub
```

Con esto has visto algunos eventos y los procedimientos que se “adaptan” a ellos. Ten en cuenta que NO todos los procedimientos se pueden ejecutar en todos los eventos.

Algunas sentencias

Por sentencias podemos entender que son aquellas líneas de código formadas por palabras reservadas, entre mas sentencias conozcas mas maneras se te ocurrirán de resolver un algoritmo. En este capítulo voy a explicar y a poner en funcionamiento algunas sentencias.

● If ... Then ... Elseif ... Else ... End If

Su sintaxis es:

```
If <Condicion> Then
    Acciones
Elseif <Condicion> Then
    Acciones
Else
    Acciones
End If
```

Utilizaremos “Elseif” para escribir más de una condición usando un solo If. Así trataremos datos que se relacionan entre si o son similares. ¡No es lo mismo que hacer condicionales anidados!

Ejemplo:

```
If Month(Now) = 1 Then
    Mes = “Enero”
Elseif Month(Now) = 2 Then
    Mes = “Febrero”
'[...]
Elseif Month(Now) = 12 then
    Mes = “Diciembre”
End If
```

Si las acciones a realizar son una (1) sola podemos escribir un If de una sola línea.

```
If Mes = “Marzo” Then MsgBox ”Marzo”
```

La Sentencia IF también cuenta con los siguientes operadores logicos:

Not, Or, And ...

Puedes usar **Not** para negar una condición y se cumpla una acción; ejemplo:

```
If Not Textn.Text = "" Then  
    MsgBox "TextBox Contiene Texto"  
End If
```

Para evaluar varias condiciones y que reciban la misma acción se utiliza **Or**.

```
If Day(Now) = 1 Or Day(Now) = 2 Or Day(Now) = 3 Then  
    MsgBox "Días entre 1 y 3"  
End If
```

Puedes usar todos los Or que sean necesarios.

Para que se cumplan 2 o mas condiciones para realizar una acción usaras **And**.

```
If VarNum = 1 And VarText = "Texto" Then  
    MsgBox "Numero 1 y Mensaje Texto"  
End If
```

También puedes usar los And que necesites

Select Case ... Case ... End Select

Puedes utilizar el Select Case cuando creas “de creer” por ejemplo que estas haciendo un If muy complejo o largo, esto te simplifica en gran medida esta labor.

Los Select se deben usar en el evento de un control y se deben aplicar a una propiedad de este.

Su sintaxis es:

```
Select Case (Control u Objeto).Propiedad  
Case "xxx"  
    Acciones  
Case Else  
    Acciones  
End Select
```

Los **Select** también utilizan los operadores lógicos **Not**, **Or** y **And** casi de igual manera que los **If**.

En la lectura de este curso ya has visto 2 ejemplos del **Select** por eso no voy a poner mas.

With ... End With

Con **With** resumes la manera de asignar el valor de las propiedades de un control u objeto; por ejemplo:

```
With Text1  
.Text = "Texto"  
.SelStart = 0  
.SelLenght = 10  
.Locked = False  
End With
```

Manejo de bases de datos

Debido a la calidad de “avanzado” de este curso, explicare de manera breve como crear la base de datos desde el VisData.

Creación de la base de datos

Clic en los menús... → “Complementos” → “Administrador visual de datos” // cuando abra el Visdata, haces clic en “Archivo” → “Nuevo...” → “Microsoft Access” → “MDB de la Version 7.0” y a continuación Guarda la Base de Datos.

Has clic derecho sobre el espacio en blanco de la ventana “Ventana de base de datos” y del menú contextual has clic en “Nueva Tabla”.

Dale nombre a la tabla; ingresa los campos; elige el campo indice y “Generar tabla”

Una vez hecho todo este proceso, agrega la **Referencia DAO**, en agregar referencias.

“Microsoft DAO 2.5/3.51 Compatibily Library”.

Si quieres conocer mas a fondo este proceso de creación de bases de datos, descarga el manual “**Como crear programas con bases de datos (usando la referencia DAO)**” de www.lawebdelprogramador.com allí esta todo explicado con pelos y señales.

Para hacer funcionar el programa, tenemos que enlazar la base con nuestra aplicación, de la siguiente manera:

** Para que este código funcione, la base debe estar guardada en la misma carpeta de nuestro proyecto o programa.*

Option Explicit

‘Esta variable guardara la ruta de ejecución de nuestro programa y el nombre de la base

Dim Ruta As String

Public MiBase As Database

Public MiTabla As Recordset

```
Public Sub EnlazarBase()  
Ruta = "" & App.Path & "\Base.mdb"  
Set MiBase = OpenDatabase(Ruta)  
Set MiTabla = MiBase.OpenRecordset("Tabla")  
End Sub
```

Debido al carácter publico de este procedimiento, debe estar escrito en un modulo.

Y para ejecutarlo escribes **Call EnlazarBase** en el Form_Load del primer formulario del proyecto.

Una base de datos puede tener en muchos casos mas de una tabla, en este procedimiento puedes agregar todos los “Set Tabla...” que sean necesarios, al igual que si tu aplicación tiene varias bases de datos las puedes enlazar con este mismo procedimiento usando distintas variables de tipo DataBase y variables de Rutas; o lo puedes hacer en procedimientos distintos (ojo, de distinto nombre sino el VB te dirá que tienes un procedimiento Ambiguo, es decir repetido)

Manejo de bases de datos

- **Guardar los datos o registros**

Creas un procedimiento de guardar, por ejemplo Private Sub Guardar()

```
Private Sub Guardar()  
Tabla.Index = ("indice")  
Tabla.Seek "=", txCodigo ' aqui el control donde ingresas el indice a la tabla  
If Not Tabla.NoMatch Then  
MsgBox "Ya existe un registro con este codigo", vbInformation, "Guardar"  
Exit Sub  
Else  
Tabla.AddNew  
Tabla ("codigo") = txCodigo  
'[...]  
Tabla.Update  
End If  
End Sub
```

Este procedimiento verifica si el código o índice escritos ya han sido usados y si es así, te avisa de que ya existe sin generar error alguno.

- **Buscar datos**

Con este procedimiento puedes realizar consultas en los datos guardados.

```
Private Sub Buscar()  
Dim Buscar  
Buscar = InputBox("Escriba el Código a buscar")  
Tabla.Index = ("indice")  
Tabla.Seek "=", Buscar  
If Not Tabla.NoMatch Then  
VerCampos ' Este es el procedimiento que visualiza los campos  
Else  
MsgBox "El registro no existe", vbInformation, "Buscar"  
End If  
End Sub
```

Puedes crear la consulta directamente en el formulario sin el InputBox, cambiando en la línea “Tabla.Seek “=”, Buscar” la variable “Buscar” por el nombre del control (textbox) donde ingresas el código o índice a consultar y eliminas las líneas

```
Dim Buscar  
Buscar = InputBox("Escriba el Codigo a buscar")
```

- **Visualizar los datos**

Para ver los resultados de una consulta o desplazarte por los registros es necesario este procedimiento:

```
Private Sub VerCampos()  
Text1.Text = Tabla ("campo1")  
Text2.Text = Tabla ("campo2")  
Text3.Text = Tabla ("campo3")  
'[...]  
End Sub
```


- **Desplazarse por los registros**

Con este procedimiento podemos movernos al último registro, al primero, a un registro anterior o uno siguiente. Para hacer esto debes crear una matriz de 4 controles de un botón de comando, desde el Boton(0) hasta Boton(3), esto lo puedes hacer al copiar un boton, VB te preguntara si quieres crear un matriz de controles, le dices que Si y lo pegas el resto de veces necesario.

```
Private Sub bMover_Click(Index As Integer)
On Error Resume Next
If MiTabla.EOF = True And MiTabla.BOF = True Then
Exit Sub
End If
```

```
If Index = 0 Then
    MiTabla.MoveFirst
ElseIf Index = 1 Then
    MiTabla.MovePrevious
ElseIf Index = 2 Then
    MiTabla.MoveNext
ElseIf Index = 3 Then
    MiTabla.MoveLast
End If
```

```
If MiTabla.BOF Then
MiTabla.MoveFirst
ElseIf MiTabla.EOF Then
MiTabla.MoveLast
End If
If Err = 0 Then
VerCampos
End If
Err = 0
End Sub
```

- **Eliminar datos**

Adivina, con este borras o eliminas registros ☺

```
Private Sub bEliminar_Click()
Dim P
P = MsgBox("¿Esta seguro de eliminar este registro?", vbYesNo, "Eliminar")
```

```
If P = vbYes Then  
Tabla.Delete  
btMover_Click 0  
End If  
End Sub
```

Mas información que tengo sobre bases de datos publicada en www.lawebdelprogramador.com. Si no los conoces los títulos son:

- Como crear programas con bases de datos (usando la referencia DAO)
- Como crear programas con bases de datos y evitar errores (con DataControl)
- Como agregar un reporte a un programa con bases de datos.

Todos los encuentras en “Biblioteca de Temas” >> “Visual Basic ()”

Y para los que ya han leído estos manuales y en especial el del “Reporte” y no les actualiza el reporte sino hasta que salen de todo el programa, ahí les va la solución... (Que pena no haberla puesto en ese manual☺)

```
Private Sub DataReport_Terminate()  
    DataEnvironment1.rsNombreComando.Close  
End Sub
```

Cuando creas un comando el mismo VB crea un propiedad de este llamada rsNombreComando ¿si?, la identificas por el prefijo rs; lo que pasaba es que al cerrar el reporte NO se cerraba el enlace que este hacia con la base y al quedar “abierta” la conexión; no mostraba los cambios hechos en el formulario cuando se volvía a mostrar el reporte.

Manipulación de registros y campos de una Base de Datos

Partiendo del hecho de que ya sabemos que es un campo, podemos decir que un campo se puede guardar (update) y se puede modificar (edit), existen varias maneras para hacer esto y todo dependerá de cuando es eficiente hacerlo o no.

Maneras de Guardar

Antes que todo, debemos considerar algunos parámetros para guardar:

- ¿Se guardaran campos en blanco?
- ¿Se permitirá duplicar la clave principal?
- ¿Se guardara más de un registro a la vez?
- ¿Se deben convertir algunos datos antes de guardar?

Si se considera que si se pueden guardar campos en blanco hay que tener la precaución de identificar si hay campos creados como requeridos y que van a contener datos y no se guardaran vacíos (Null). Si no se quieren guardar datos en blanco entonces debemos crear un condicional en el cual se busque algún textbox u otro control vacío y en caso de hallarlo pedir al usuario que digite todos los campos.

Ejemplo:

```
If Text1.Text = "" Or Text2.Text = "" Then
    MsgBox "Debe digitar todos los campos"
    Exit Sub
Else
    Call Procedimiento_Guardar
End If
```

También podemos verificar si alguna variable o un control esta vacío así:
`Text1.Text = Empty` ó `vDias = Empty`

Después de analizar el tipo de información que se esta guardando podrás determinar que campos podrían quedar vacíos o no, por ejemplo, si guardas datos personales de alguien y tienes el campo "Correo Electrónico" este

podría quedar vacío en algunos casos cuando la persona no tiene correo electrónico ☺

Lo que si es imprescindible es que si diseñaste una tabla que requiere clave principal esta NUNCA podrá guardarse vacía, y aunque tal vez el programa no muestre error y la guarde así de todas maneras, debes pensar primero que esa clave es necesaria para consultas o para relaciones entre tablas.

¿Se permitirá duplicar la clave principal? En algunos casos no necesitaras guardar esta clave ya que puede tratarse de un “registro temporal” que nunca va a ser consultado después de la misma manera. Cuando digo: “Registro Temporal” me refiero a una tabla diseñada para guardar registros cuando se ejecuta un procedimiento, y al reiniciar el procedimiento se borrara y volverá a guardar otro registro. *Ejemplo:* De una tabla llamada Inventario necesitamos imprimir en un DataReport solo los artículos o mercancías con referencia X1, para este caso haremos una búsqueda con un filtro pero primero borramos la tabla temporal para eliminar datos anteriores.

On Error Resume Next

TablaTemporal.MoveFirst

For X = 1 To 10000

 TablaTemporal.Delete

 TablaTemporal.MoveNext

Next X

Ref = TablaInventario(“Referencia”)

If Ref = “X1” Then

 Articulo = TablaInventario(“Aritculo”)

 TablaTemporal.AddNew

 TablaTemporal(“CampoTemporal”) = Articulo

 TablaTemoral.Update

End If

De esta manera tenemos en una tabla solo los artículos de referencia X1 y podemos enlazar esta tabla al DataReport que mostrara esta lista.

Claro esta que el verdadero código para filtrar y guardar no es como en el ejemplo, esto es solo una representación muy simple.

Ahora si, volviendo a la pregunta ¿Se permitirá duplicar la clave principal? Si el registro NO es temporal es preciso (en la mayoría de los casos) no permitir duplicidad. Para hacer esto debes consultar la tabla antes de guardar y si no existe la clave que se va a grabar, proceder; sino, avisar al usuario que la clave ya existe.

Ejemplo: En una tabla que guarda datos personales la clave podría ser Número de Identificación y este no debería repetirse ya que NO hay dos personas con el mismo número de identificación.

`Tabla.Index = "Clave"`

`Tabla.Seek "=", Text_Numero_de_Identificacion.Text`

`If Not Tabla.NoMatch Then`

`MsgBox "Ya existe una persona registrada con ese numero de identificación"`

`Else`

`Call Procedimiento_Guardar`

`End If`

Para consultar o buscar usamos la propiedad Seek de la variable que manipula la tabla, esta solo buscara dentro de las claves de la tabla.

El parámetro también puede variar si es necesario para otros casos:

`Tabla.Seek "=", varClave` (Busca clave igual a varClave)

`Tabla.Seek "<", varClave` (Busca clave menor que varClave) *

`Tabla.Seek ">", varClave` (Busca clave mayor que varClave) *

`Tabla.Seek "<=", varClave` (Busca clave menor o igual que varClave) *

`Tabla.Seek ">=", varClave` (Busca clave mayor o igual que varClave) *

* Estos parámetros NO permiten que la clave sea alfanumérica. Solo el parámetro "=" permite cualquier tipo de datos como clave.

La instrucción `Tabla.NoMatch` determina (si es verdadero) que NO se encontró el registro, entonces negamos la instrucción: `Not Tabla.NoMatch` para que sea igual a `Tabla.Match`, es decir "encontrado" de ser verdadero.

¿Y si me equivoco o es necesario actualizar los datos?

Como no es posible repetir la clave y no es lógico crear otro registro para cambiar una fecha de nacimiento, una dirección o un teléfono por ejemplo, la solución posible seria eliminar el registro y hacerlo de nuevo, pero esa no es la

idea. Abrimos un nuevo registro y primero que todo escribimos el campo Clave y en ese momento debemos verificar si ya existe un registro con esta Clave para dos cosas: Primero; no perder tiempo en volver a escribir un registro que ya se digito, y segundo; para poder actualizar los datos. El programa debe mostrar un mensaje indicando que la clave escrita ya esta asignada y que si se desea actualizar los datos.

Esta como enredado esto ¿cierto? ☺ llevemos toda esta carreta a un ejemplo para que se entienda.

Ejemplo: La manera correcta de diseñar el algoritmo para guardar en BD.



Identificacion	92644357
Nombre	ANDRES ESCOBAR DUQUE
Direccion	CALLE 15 No 84-98
Telefono	300-3589151

Guardar

Vamos a suponer los siguientes nombres para los TextBox de la imagen que son los que nos interesan.

Identificacion:	txID
Nombre:	txNom
Dirección:	txDir
Teléfono:	txTel

El diseño del código no permitirá duplicar ni guardar vacía la clave principal: **Identificación (ID)**.

1. 'Presionamos Enter en **txID** para verificar la existencia de la clave en algún registro y declaramos una **variable** llamada **Editar** en la sección general de tipo Boolean.

Dim Editar As Boolean

```
Private Sub txID_KeyPress(Index As Integer, KeyAscii As Integer)
If KeyAscii = 13 Then    Tabla.Index = "Clave"
    Tabla.Seek "=", txID.Text
    If Not Tabla.NoMatch Then
        M = MsgBox("Ya existe un registro con este codigo. ¿Desea actualizar los datos?",
vbYesNo + vbQuestion, "Guardar")
        If M = vbYes Then
            Editar = True
            Call VerCampos
        End If
    End If
End Sub
```

Este es el procedimiento VerCampos:

```
Private Sub VerCampos()
txID.Text = Tabla("ID")
txNom.Text = Tabla("Nom")
txDir.Text = Tabla("Dir")
txTel.Text = Tabla("Tel")
End Sub
```

2. El procedimiento para guardar quedara así:

```
Private Sub btGuardar_Click()
If Editar = True Then
    Tabla.Edit
    Tabla.Fields("ID") = txID.Text
    Tabla.Fields("Nom") = txNom.Text
    Tabla.Fields("Dir") = txDir.Text
    Tabla.Fields("Tel") = txTel.Text
    Tabla.Update
    MsgBox "Datos guardados", vbInformation, "Guardar"
Else
    Tabla.AddNew
    Tabla.Fields("ID") = txID.Text
    Tabla.Fields("Nom") = txNom.Text
    Tabla.Fields("Dir") = txDir.Text
    Tabla.Fields("Tel") = txTel.Text
```

```
        Tabla.Update
        MsgBox "Datos guardados", vbInformation, "Guardar"
    End If
    Editar = False
End Sub
```

Las instrucciones Edit (Editar) permitirán actualizar los datos, los datos que no se cambien o no se asignen se conservaran igual.

Contar Registros

En algún momento te darás cuenta que es necesario saber cuantos registros tiene una tabla, para esto existe una propiedad llamada RecordCount, la forma de utilizarla es esta:

```
Variable = Tabla.RecordCount
```

Pero esta propiedad en la referencia DAO (que es la que utilizo en este libro) NO funciona mas de la mitad de las veces y arroja valores que no concuerdan. Para solucionar esto he diseñado este procedimiento (debe colocarse en un modulo):

En la sección General:

```
Public TotalRegistros As Double
```

```
Public Sub ContarRegistros()
    TotalRegistros = 0
    Dim X As Double
    X = 0
    On Error Resume Next
    Tabla.MoveFirst
    For X = 1 To 5000 'Suponiendo que nunca se alcanzaran 5000 registros
        On Error GoTo Fin
        Tabla.MoveNext
    Next X
    Fin:
    TotalRegistros = Val(X) - 1
End Sub
```


Debes crear un procedimiento como este para cada tabla que manejes, luego para conocer cuantos registros tiene llama a este procedimiento: [Call ContarRegistros](#)

Una vez llamado el procedimiento la variable publica TotalRegistros contiene el total de registros de la tabla especifica y la puedes utilizar así:

En lugar de esto:

```
For X = 1 To Tabla.RecordCount  
[...]
```

Has esto:

```
Call ContarRegistros  
For X = 1 To TotalRegistros  
[...]
```

Este procedimiento NO falla y siempre te da el numero exacto de registros.

Consultas con SQL

Para explicar este tema (y no ponernos a leer definiciones) vamos a desarrollar un pequeño programa.

Este programa deberá:

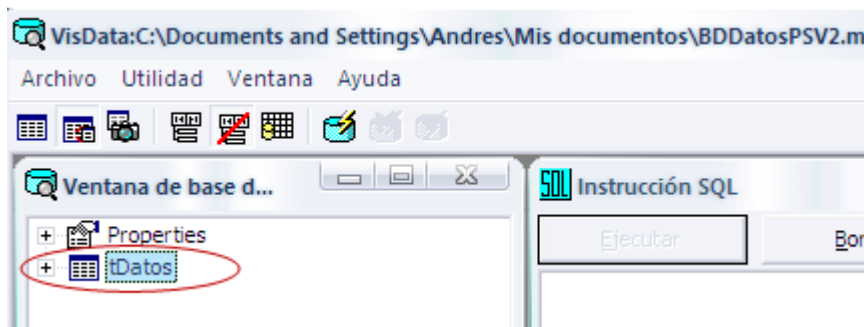
1. Permitirme buscar los datos “similares” según su cadena de texto al ir digitando y listarlos todos en un ListBox.
2. Al hacer click sobre cualquier de los elementos listados se deben mostrar otros datos relativos a él.

¿Cómo así? ☹ ya veras. Primero diseñamos una pequeña base de datos con la siguiente estructura usando VisData de VB:

Nombre de la Tabla: tDatos

Campos:	Tipo	Tamaño
- ID	Text	Por defecto
- Nombre	Text	250
- Teléfono	Double	Por defecto
- Ciudad	Text	Por defecto

Aquí mismo, en VisData, vamos a llenar varios registros a manera de ejemplo. Has doble click en el icono de la tabla que creamos en la ventana de base de datos:



Te muestra la siguiente ventana:



Para agregar registros debes hacer click en “Agregar”, cuando hayas terminado le das click a “Actualizar”. Repite estos pasos varias veces.

Ahora abrimos un nuevo proyecto “EXE Estándar” en VB y agregamos la referencia **Microsoft DAO 2.5/3.51 Compatibility Library**.

Agregamos un Modulo y pegas el siguiente código en el modulo:

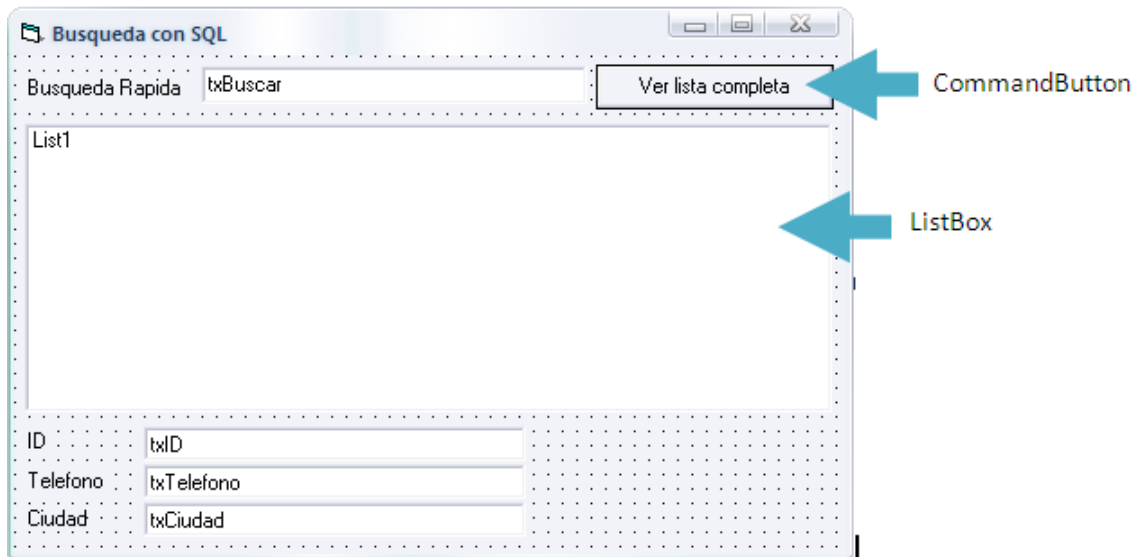
```
Option Explicit
Public Base As Database
Public Tabla As Recordset

Public Sub EnlazarBase()
Dim Ruta As String
Ruta = App.Path & "\BDDatosPSV2.mdb"
Set Base = OpenDatabase(Ruta)
Set Tabla = Base.OpenRecordset("tDatos")
End Sub
```

Luego pasamos al Form1, y pega este código:

```
Private Sub Form_Load()
Call EnlazarBase
txBuscar.Text = ""
txID.Text = ""
txTelefono.Text = ""
txCiudad.Text = ""
End Sub
```

Ya es hora, diseña la interfaz del Form1, trata que te quede igual:



Coloca a los TextBox los nombres que aparecen en la imagen. (Recuerda: también debes agregar un ListBox y un CommandButton; a estos deja los nombres por defecto.

Uuuff !!! pega este código. Corresponde al evento Change del Textbox txBuscar...

```
Private Sub txBuscar_Change()  
On Error Resume Next  
Dim RecArt As Recordset  
Dim SQL As String  
  
If IsNull(txBuscar.Text) Or (txBuscar.Text = "") Then Exit Sub  
SQL = "SELECT ID, NOMBRE, TELEFONO, CIUDAD FROM TDATOS WHERE NOMBRE like '*'"  
& txBuscar.Text & "*"  
Set RecArt = Base.OpenRecordset(SQL, dbOpenSnapshot)  
  
List1.Clear  
  
With RecArt  
If (.BOF And .EOF) Then  
  
Else  
.MoveFirst  

```

```
Do While Not .EOF
Dim N As Integer
For N = 1 To Tabla.RecordCount
    List1.AddItem .Fields("nombre")
    .MoveNext
    If List1.ListIndex = RecArt.RecordCount Then
        Exit For
    End If
Next N
.MoveNext
Loop
End If
End With
End Sub
```

Ya podemos probar. Lo que hace este código es buscar las “coincidencias” de lo que digitas en el txBuscar con el campo “Nombre” en todos los registros. Si te das cuenta al digitar una sola letra, el programa te lista TODOS los nombres que contengan esa letra, a medida que digitas mas letras la lista se va filtrando automáticamente.

Agrega el código del botón que mostrara la lista completa de todos los nombres:

```
Private Sub Command1_Click()
If Tabla.BOF And Tabla.EOF Then
    MsgBox "No hay registro de Articulos", vbExclamation, "Error de Carga"
    Exit Sub
End If
List1.Clear
Tabla.MoveFirst
Dim N As Integer
For N = 1 To Tabla.RecordCount
List1.AddItem Tabla("nombre")
Tabla.MoveNext
    If List1.ListIndex = Tabla.RecordCount Then
        Exit For
    End If
Next N
txBuscar.Text = ""
txBuscar.SetFocus
End Sub
```

Para terminar, debemos hacer que al hacer click sobre cualquier nombre de la lista nos muestre su respectivo ID, teléfono y ciudad.

Pega este código:

```
Private Sub List1_Click()  
Dim sBuscar As String  
    Dim tRs As Recordset  
    sBuscar = "SELECT * FROM TDATOS WHERE nombre LIKE '" & List1.Text & "' ORDER  
BY nombre"  
    Set tRs = Base.OpenRecordset(sBuscar, dbOpenSnapshot)  
    With tRs  
        If (.BOF And .EOF) Then  
  
        Else  
            .MoveFirst  
            Do While Not .EOF  
                txID.Text = .Fields("ID")  
                txTelefono.Text = .Fields("Telefono")  
                txCiudad.Text = .Fields("Ciudad")  
                .MoveNext  
            Loop  
        End If  
    End With  
End Sub
```

Pruébalo !!! Ejecuta, has click en el botón “Ver Lista Completa” y luego has click sobre cualquier nombre, veras sus respectivos datos.

En este ejemplo las variables SQL y sBuscar las he utilizado para almacenar una instrucción **SQL** que organiza los campos de acuerdo a un criterio y luego se graban en una variable de tabla temporal, en este caso esa tabla temporal esta representada por la variable “tRs”

Ten MUY en cuenta este ejemplo, ya que las soluciones que puedes derivar de este son infinitas. Personalmente siempre me baso en este ejemplo para hacer consultas con criterios, incluso lo he usado para guardar y eliminar bajo algunas condiciones y parámetros personalizados.

Algunas APIs

Te preguntaste alguna vez si se podía hacer un programa que manipulara muchas funciones propias del sistema operativo como por ejemplo, copiar archivos, eliminar o mover, reproducir sonidos, registrar programas, bloquear o desbloquear los periféricos de entrada como teclado y mouse, como deshacer cambios, poner tus formularios y programas por encima de todas las ventanas y ser siempre visibles, abrir programas; pues todo esto y aproximadamente 80.000 cosas mas son posibles gracias a las APIs de Windows, incluso un virus hecho en VB utiliza como minimo de 2 a 3 APIs (claro que aquí no les voy a enseñar como darles ese uso Je, je ☺)

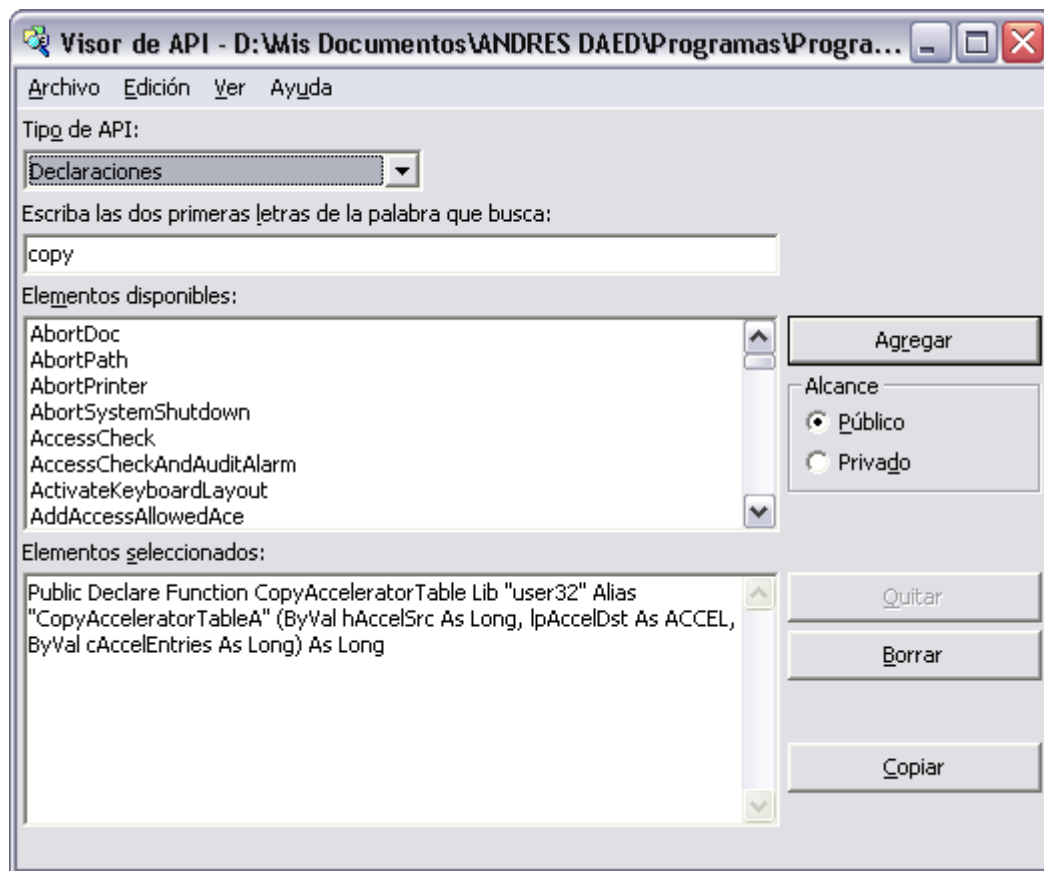
Las APIs (Interfaz de programación de aplicaciones) son funciones del propio Windows y son las que le permiten hacer lo que hace, cada proceso que realiza este sistema operativo, esta manipulado por una API, y es en serio son como 80.000 funciones API que tiene el Windows.

Las APIs están organizadas en librerías de Windows como Kernel32.dll, Advapi32.dll y User32.dll

Necesitan de una declaración ya sea privada o pública en nuestro programa para que podamos manipularlas, además algunas necesitan de un reguero de constantes para funcionar.

Pero ¿Dónde consigo esas declaraciones y constantes? Es más o menos fácil si te sabes el nombre de la API que buscas. Visual Studio tiene en sus herramientas (si las instalaste obviamente) una llamada API TextView o Visor de Texto API, ya te voy a enseñar a manejarla.

Este es el API TextView o Visor de API...



Para usarlo abres el menú archivo y le das clic en “Cargar archivo de texto” y cargas el que dice “WIN32API.txt”. En “Tipo de API” puedes seleccionar si lo que buscas es una declaración, constantes o tipos, escribes el nombre o parte del nombre de la API que buscas y le das doble clic a esta API en Elementos disponibles. De ahí la puedes copiar a tu programa.

Pero lamentablemente no es así no mas que ya manipulas al sistema operativo, necesitas de un código que manipule a la API (es que no basta con declararlas y poner sus constantes) entonces aquí voy a poner unos ejemplos de APIs con sus declaraciones, constantes y códigos para manipularlas.

● Abrir un programa previamente instalado.

Esta API solo funciona con programas instalados y registrados en Windows (en el Regedit) no creas que con esto abres cualquier .EXE, en este ejemplo vamos a abrir cualquiera de Microsoft Office.

En un modulo pega lo siguiente:

```
Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

```
Public Const SW_SHOWNORMAL = 1
```

Y para el botón o menú, o que se yo, que abra “MS-Word” lo siguiente:

```
ShellExecute hwnd, "open", "winword.exe", vbNullString, vbNullString, SW_SHOWNORMAL
```

Puedes buscar en C:\Archivos de programa\Microsoft Office\... los .EXE de Excel, Power Point, y también los accesorios de Windows, el único que me se es calc.EXE (la calculadora)

Reproducir un sonido (.wav)

```
Private Declare Function mciExecute Lib "winmm.dll" (ByVal lpstrCommand As String) As Long  
Dim Rep
```

```
Private Sub btPlay_Click()  
Rep = mciExecute("Play C:\Sonido.wav")  
End Sub
```

```
Private Sub btStop2_Click()  
Rep = mciExecute("Stop C:\Sonido.wav")  
End Sub
```

Una ventana siempre visible

Esto en un modulo:

```
Public Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
```

Y esto en el formulario:

```
Private Const HWND_TOPMOST = -1
```

```
Private Sub Form_Load()  
Me.ScaleMode = vbPixels
```

```
SetWindowPos Me.hwnd, HWND_TOPMOST, Me.ScaleLeft, Me.ScaleTop, Me.ScaleWidth,  
Me.ScaleHeight, 0  
End Sub
```

Copiar un archivo

Todo esto va en el formulario:

```
Private Declare Function CopyFile Lib "kernel32" Alias "CopyFileA" (ByVal lpExistingFileName As  
String, ByVal lpNewFileName As String, ByVal bFailIfExists As Long) As Long  
Dim Copiar As Long
```

```
Copiar = CopyFile(App.Path & "\BaseExtras.mdb", "C:\BaseExtras.mdb", 1)
```

En esta API, el ultimo parámetro ",1" puede ser cambiado por cero "0" para que sobrescriba al archivo de destino, si ya existe.

Funciones

Con Visual Basic es muy fácil hacer ciertas cosas que no son muy obvias gracias a sus funciones; las funciones son bloques de código que siempre devuelven uno o varios valores, ¿entiendes esto?, pues yo hace mucho no lo entendía, y si estas en las mismas aquí te va:

Cuando en programación se habla de devolver, significa que el programa o algoritmo va a generar un resultado, ya sea, un número, un texto o un valor lógico, por ejemplo: si hacemos una función que suma 2 números, la función debe devolver el resultado de la suma de esos 2 números.

Existen 2 clases de funciones, las internas y externas.

Funciones Internas: son las que vienen incluidas en el propio lenguaje de programación.

Funciones Externas: son las que el mismo programador crea en su código.

Ya no más carreta, aquí van los ejemplos y usos de algunas funciones internas de VB:

● Val

Convierte un dato cualquiera a numérico, si no se puede (por decir si aplicamos Val a "ABC123") generara un error. Utiliza Val para hacer operaciones con cajas de texto.

Ejemplo:

```
txR = Val(Tx1.Text) + Val(Tx2.Text)
```

● CDbI

Permite convertir y manipular números reales (casi igual que Val)

Ejemplo:

```
txR= CDbI(Tx1.Text) * CDbI(Tx2.Text)
```

● Str

Hace todo lo contrario de Val, pero no genera error.

Ejemplo:

```
txID.Text = Str(Num)
```

● Len

Devuelve la longitud de un texto

Ejemplo:

```
LongText = Len(Text1.Text)
```

● Int

Devuelve un valor entero redondeado de un valor

Ejemplo:

Numero = 15,8

NumEntero = Int(Numero) 'Devuelve 16

● Fix

Devuelve la parte entera de un número

Ejemplo:

Numero = 15,8

NumEntero = Fix(Numero) 'Devuelve 15

● Rnd

Devuelve un numero aleatorio entre un rango de números.

Ejemplo:

Randomize

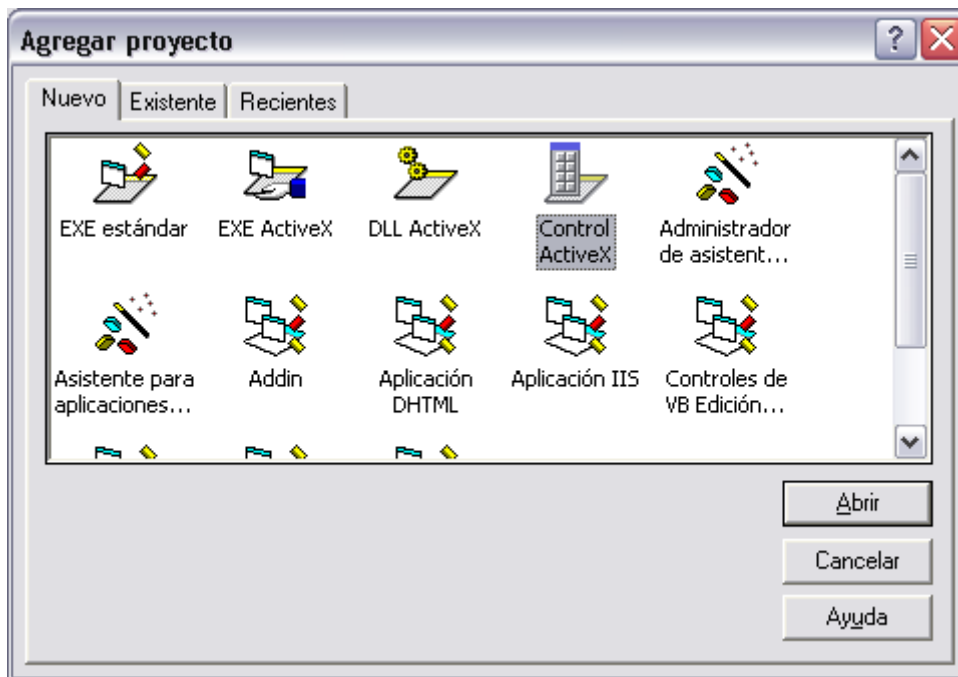
Valor = Int((100 * Rnd) + 1)

Activex - Creación de un control

Activex es la tecnología de programación que permite crear y distribuir nuestros propios controles u objetos, para ser usados por cualquier lenguaje que los soporte.

Manos a la obra...

- Crea un nuevo proyecto “EXE Estándar” en VB.
- Agrega a este un proyecto “Control Activex”

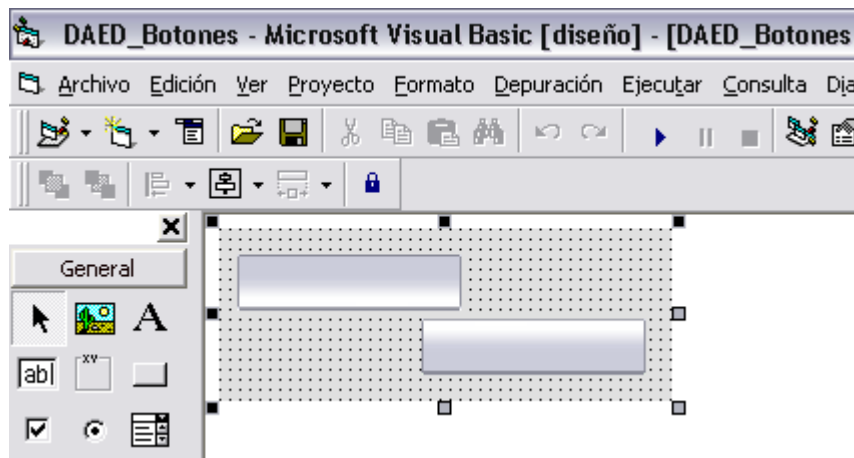


La idea es que crees un “Grupo de Proyectos”, el estándar lo usaremos para ir probando y depurando nuestro control.

Pero antes de empezar, necesitamos 2 imágenes de botones, una oprimido y otra no oprimido, puedes crear las tuyas dibujándolas y puliéndolas en algún editor de imágenes; para empezar te colaboro con estas:



Utilizando un control Image en el formulario del proyecto “Control Activex” carga estas imágenes (cada una en un Image) ponles como nombre “Face1” y “Face2”



Selecciona al mismo tiempo los 2 controles Image y pon sus propiedades Left = 0 y Top = 0. Quedaran una sobre la otra. Debe quedar “Face1” encima, si no es así, has click derecho sobre “Face2” y elige “Enviar al fondo”. Para poder ajustar el tamaño del control, coloca a True la propiedad “Stretch” de los Image

Ajusta el tamaño del formulario al tamaño de los Image (Face1 y Face2) de modo que no sobre salga.



Coloca una Etiqueta (Label) sobre los Image. Colócale por nombre “lbCaption”; pon su propiedad BackStyle = 0 - Transparent; escribe algo de texto en su propiedad “Caption” que será lo que salga por defecto cuando lo utilices, por ejemplo un botón de comando dice “Command1”. Puedes poner lo que quieras. Desde su propiedad “Font” pon su “Caption” en negrita.



Ajusta el tamaño de la etiqueta para que quede igual al tamaño de los Image.

Ahora copia este código, que permitirá que nuestro control se comporte como un botón de comando cuando se hace click sobre el.

```
Private Sub Face1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Face1.Visible = False
    Face2.Visible = True
    lbCaption.ForeColor = vbBlue
End Sub
```

```
Private Sub Face1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Face1.Visible = True
    Face2.Visible = False
    lbCaption.ForeColor = vbBlack
    RaiseEvent Click
End Sub
```

```
Private Sub lbCaption_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Face1.Visible = False
    Face2.Visible = True
    lbCaption.ForeColor = vbBlue
End Sub
```

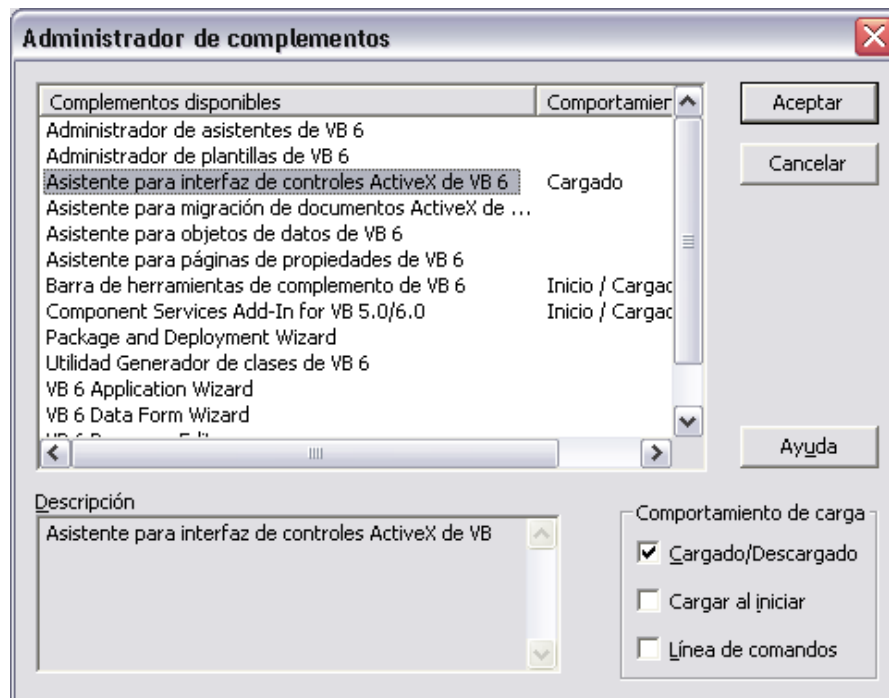
```
Private Sub lbCaption_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Face1.Visible = True
    Face2.Visible = False
    lbCaption.ForeColor = vbBlack
End Sub
```

```
Private Sub UserControl_Resize()
```

```
Face1.Top = 0
Face1.Left = 0
Face1.Height = UserControl.Height
Face1.Width = UserControl.Width
Face2.Top = 0
Face2.Left = 0
Face2.Height = UserControl.Height
Face2.Width = UserControl.Width
lbCaption.Top = 125
lbCaption.Left = 0
lbCaption.Height = UserControl.Height
lbCaption.Width = UserControl.Width
End Sub
```

Para crear las propiedades y eventos del control usaremos el “Asistente para interfaz de control Activex”.

Has click en el menú “Complementos” > “Administrador de Complementos” y vamos a elegir de allí "asistente para interfaz de controles activex"

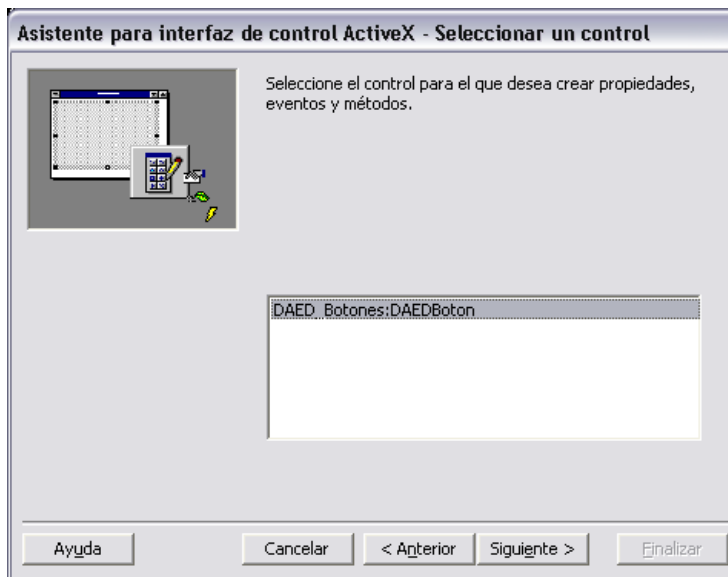


En “Comportamiento de carga” elige “Cargado/Descargado”.

Luego regresa al menú “Complementos” y elige la nueva opción que debe aparecer “Asistente para interfaz de control Activex”



Dale “Siguiente >”



Otra vez “Siguiente >”

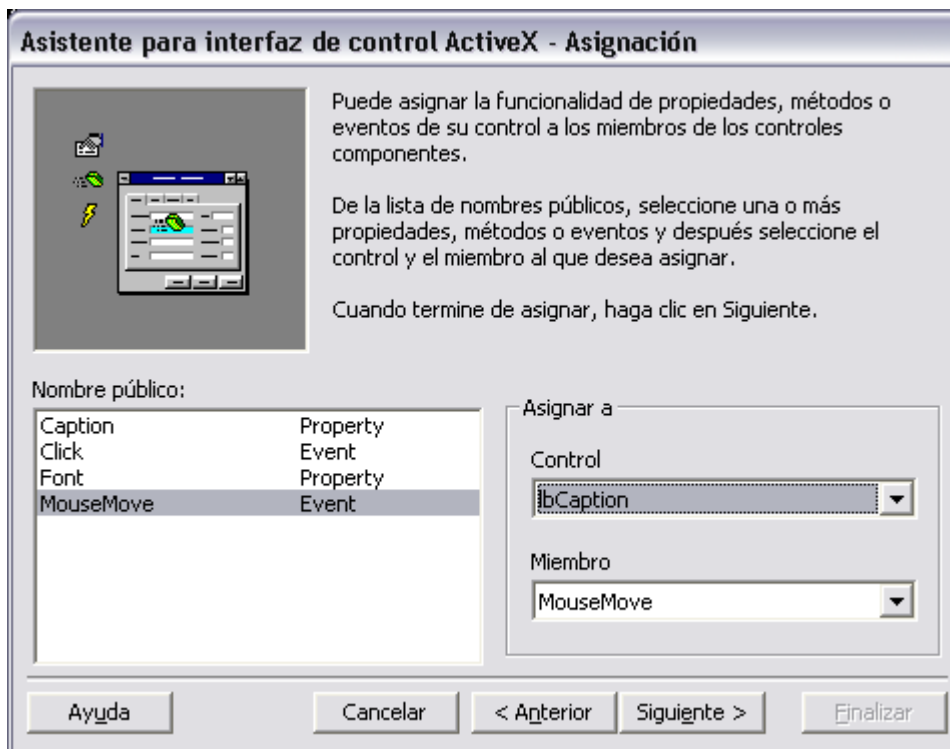


Busca en “Nombres Disponibles” las propiedades que quieras; para este ejemplo agrega Caption, Font, Click, MouseMove y le das “Siguiente >” a la que sigue también “Siguiente >”.

Los miembros personalizados te permiten crear propiedades que no estaban disponibles en la ventana anterior o utilizar las propiedades que ya existen pero en español o con el nombre que quieras, por ejemplo, si prefieres que la propiedad “Caption” de tu control se llame “Subtitulo” o “Font” “Fuente” etc, entonces utilizaras estos miembros personalizados. En este ejemplo no los usaremos.



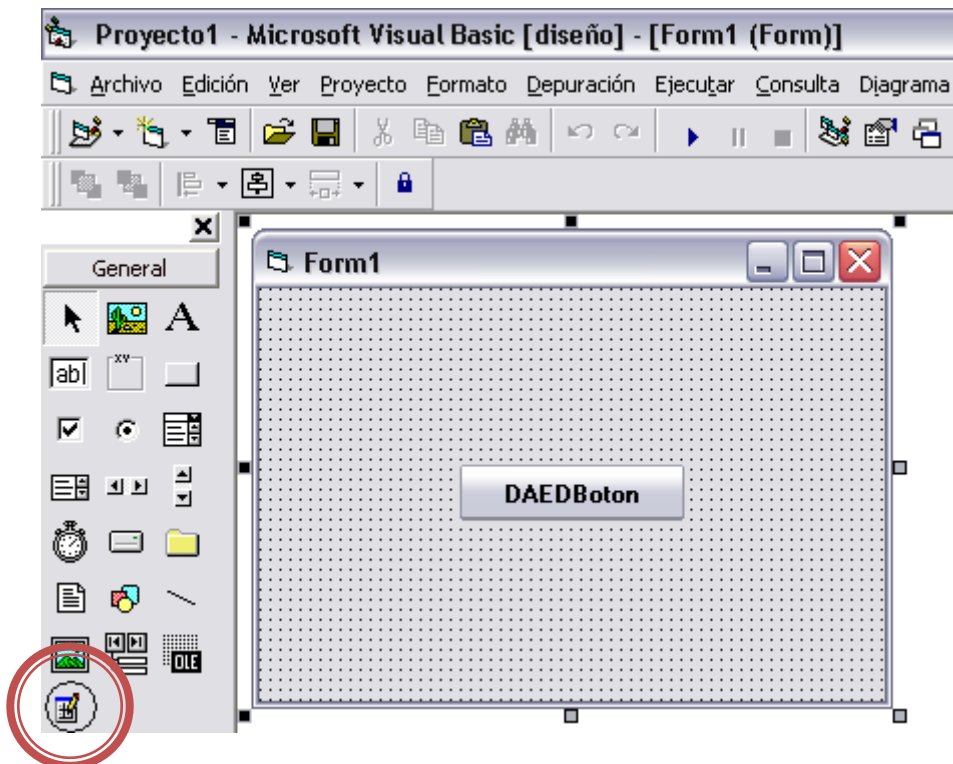
En Asignación, selecciona cada nombre público (propiedad) y asígnalas todas (en nuestro ejemplo) al control lbCaption.



Y para terminar click en “Finalizar”

Todo el código que vuelve a nuestro control un control de verdad se genera automáticamente.

Para probarlo cierra todas las ventanas del proyecto Control Activex (si están abiertas no se puede probar) y en el cuadro de herramientas del proyecto estándar del grupo de proyectos debe aparecer un nuevo control.



Para usar este control en otro proyecto o programa que diseñes, debes compilarlo, has click en el menú “Archivo” > “Generar NombreProyectoControlActivex.ocx”

Diseño de la IGU (Interfaz Grafica de Usuario)

¿Qué es esta vaina? Bueno, la IGU es como lo dice aquí arribita es la Interfaz Grafica de Usuario, es decir la IMAGEN del programa con la que interactúa el usuario, son: las ventanas, botones, listas, menus, imágenes y todo lo que se VE.

¿Qué tan importante es eso? Mucho, la IGU de un programa es como el empaque de un producto comercial cualquiera, te has fijado lo bien elaboradas que son las cajitas de las galletas navideñas ¿para que las hacen así? Facil, todo entra por los ojos, la presentación de un producto es FUNDAMENTAL, esto determina bastante que un producto sea comprado o no. Pero para el software, además de esto, es muy importante porque un buen diseño de IGU hace programas MAS fáciles de manejar y de aprender.

Y si no me crees hay varios ejemplos que lo demuestran: el nuevo Office 2007, mejor en varios aspectos que su versión anterior Office 2003 pero sobretodo por su NUEVA IGU constituida por la nueva Cinta de Opciones que reemplazo los menús tradicionales por pestañas y barras de herramientas muy sofisticadas. El nuevo Windows Vista y su famosa interfaz AeroGlass... crees que los locos de Microsoft se matan tanto por nada... la imagen vende.. y vende mucho. Para no extendernos tenemos Linux Ubuntu, WinAmp 5.5, NOD32 V.3, Mozilla FireFox y muchos otros que en su ultima versión cuentan con una asombrosa interfaz.

¿Y se pueden hacer esas interfaces en VB? ... pues lamentablemente NO se pueden hacer en VB, pero con un buen editor de imágenes podemos crear todos los elementos necesarios y luego colocarlos en VB, esta parte es fácil, lo complicado es el diseño grafico.

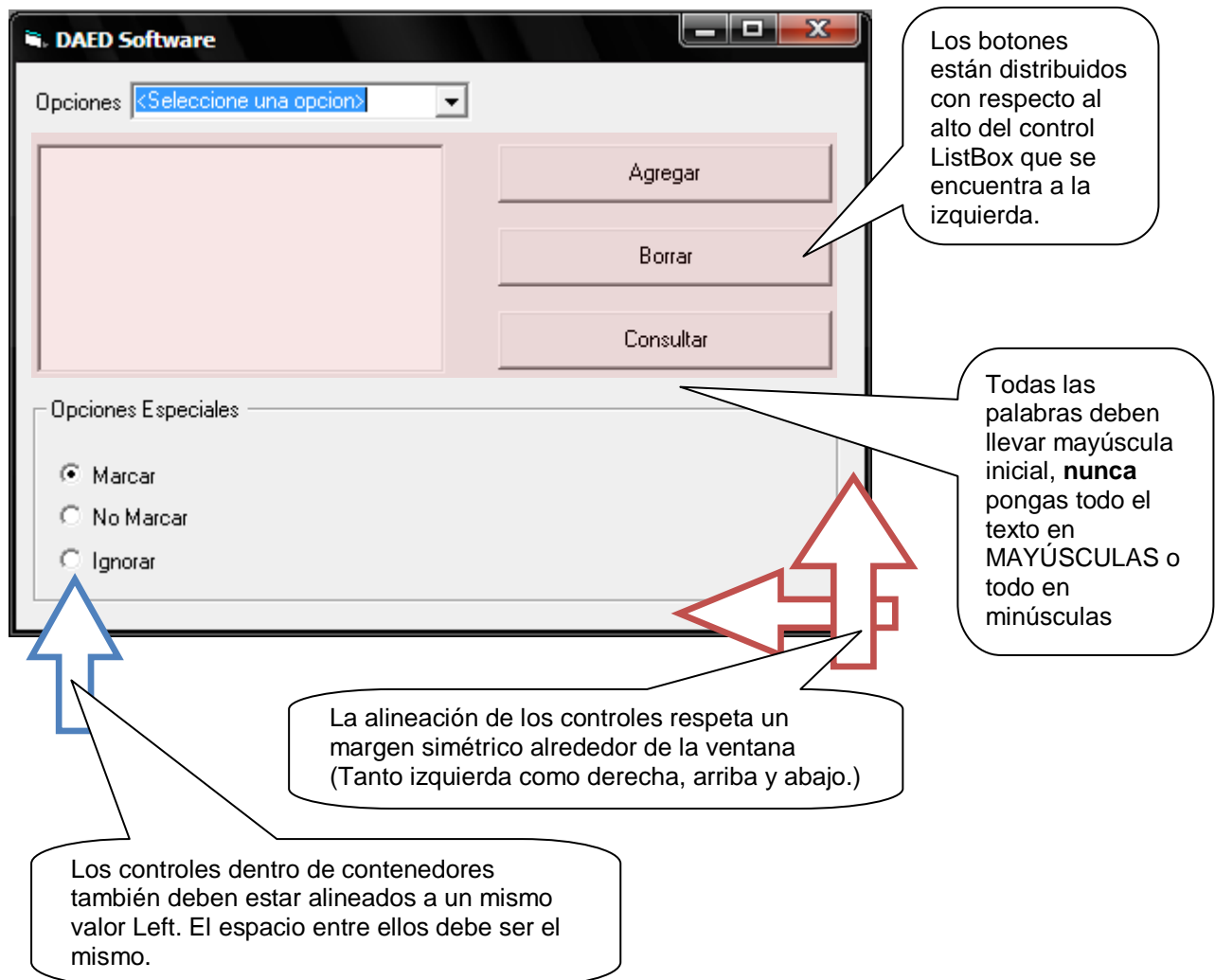
Pero lo mas importante es la distribución, tamaño y posición de todo.. fíjate:

¿Qué debo tener en cuenta?

1. Cuando vamos a sistematizar un proceso que se lleva de manera manual en formatos de papel especial NO debemos hacer una fotocopia del papel en una ventana. A veces lo mas apropiado será hacerlo en varias ventanas NUNCA amontones una gran cantidad de textbox, botones, combos y listas en una sola ventana. Fijate en los asistentes de instalación, todo lo que te preguntan podrían ponerlo en una sola ventana, pero eso seria muy engorroso de diseñar y manejar, por eso el proceso completo de una instalación se muestra en diferentes ventanas consecutivas.

2. Ten muy en cuenta la alineación, distribución y tamaño de los controles:

Manera correcta:

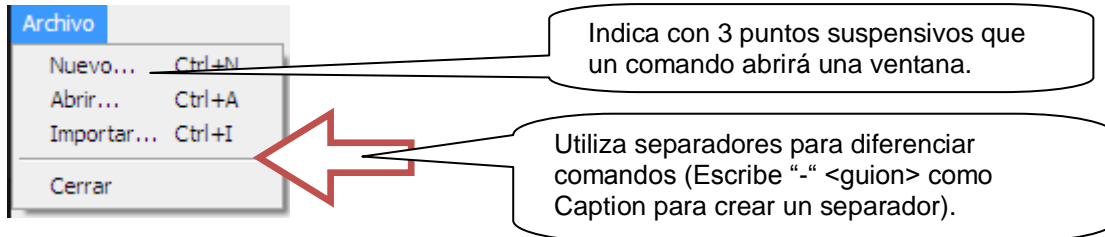




No hagas esto:



3. Los menús deben ser escritos con mayúscula inicial en todas las palabras menos en preposiciones como “y”, “de”, “a” Etc. No olvides los respectivos ShortCuts y atajos de Alt.



4. En las barras de herramientas utiliza iconos apropiados que indiquen fácilmente que comando representan.

5. Coloca títulos explícitos en los Labels de modo que el usuario comprenda la interfaz lo mas fácilmente posible.

6. Agrega ToolTipText a todo lo que puedas, son una buena ayuda.



Puedes mejorar mucho tus IGU haciendo tus propios botones, contenedores, textbox y otros con la ayuda de PhotoShop, Corel u otro editor de imágenes o usando OCX ya listos (que consigues en internet) con skins para aplicar a tus programas en VB

Sonidos para la interfaz

Tal vez el ejemplo mas representativo de esto es Encarta, cada vez que pasas el cursor por algún botón o elemento de la interfaz reproduce un sonido, esto también es muy común en juegos y otros.

Pues manos a la obra:

1. Abre un nuevo proyecto en VB. y agrega al Form1 un botón de comando.
2. Pega este código en la sección general del Form1

Option Explicit

```
Private Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal  
lpszSoundName As String, ByVal uFlags As Long) As Long  
Dim MoveCursor As Boolean
```

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As  
Single)  
MoveCursor = True  
End Sub
```

3. Crearemos un procedimiento que será el encargado de reproducir el sonido pero teniendo en cuenta que solo lo reproduzca una vez, ya que este procedimiento se ejecutara en el MouseMove si no haces esto el sonido se repetirá infinitamente mientras el cursor este sobre el control donde se originara el sonido.

```
Private Sub Sonido()  
Dim Son  
Static C As Integer  
If MovForm = True Then C = 0  
C = C + 1  
If C = 1 Then  
    Son = sndPlaySound(App.Path & "\BeepBoton.wav", &H1)  
Else  
    Son = Empty  
End If  
End Sub
```

El archivo “**BeepBoton**” puede ser cualquiera pero en formato WAV únicamente. Aunque este archivo NO exista el programa reproducirá un

sonido (Beep) al pasar el mouse por encima del botón que hemos agregado al Form1, pero antes copia y pega esto:

```
Private Sub Command1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Sonido
    MoveCursor = False
End Sub
```

Con la ayuda de un buen editor de Audio puedes crear los sonidos que quieras y usarlos con este código !!!

Ejecuta y pasa el mouse sobre el botón.

Optimizar la Programación

Este ultimo apartado trata mas de cómo hacer de la programación un oficio mas eficaz y exitoso, tal vez por que cuando era estudiante de Análisis y Programación de Sistemas, vi en mis compañeros y amigos la necesidad de estas técnicas [\(por así decirlo\)](#) para programar o porque me gusta escribir mucha carreta !!!

Consejos Teóricos

1. Como escribiría un hacker "Todo es posible", a la fecha no creo que hayan programas imposibles de hacer, inclúyete en la lista de los que podemos hacer lo que se nos pida y ocurra.
2. Desprográmate para programar. [\(haber que entiendes aquí\)](#)
3. El análisis previo antes de iniciar un proyecto te evitara reescribir una buena cantidad de código o reestructurar un programa, aunque algunas soluciones solo se encuentran en el camino.
4. Cuando realices un trabajo grande, trata (si te es posible) de alejarte de la programación por lo menos un día o dos cada semana.
5. Hacer programas es como hacer música, sin inspiración "*¡ pailas !*"

Consejos Prácticos

1. Requerir declaración de variables.

Esto es muy importante ya que, ¿quien no ha escrito mal el nombre de una variable y luego se le forma un lío porque no sabe como debería estar escrita?; al requerir la declaración de variables VB te mostrara un mensaje de error al no encontrar la variable mal escrita declarada. Esto realmente solo es necesario cuando se hacen grandes proyectos, con muchos formularios, módulos etc. Pero siempre es un buen hábito.

Para requerir la declaración de variables...

- Has click en el menú “Herramientas” > “Opciones” en el cuadro de dialogo que se muestra has click en la casilla de verificación “Requerir declaración de variables” que se encuentra en la pestaña “Editor”
- Al hacer esto aparece en el editor de código (En la parte superior) Option Explicit. [\(Cuando se abren nuevos proyectos\)](#)
- Para que VB detecte algún error antes de compilar, ejecuta el programa presionando Ctrl + F5 que es “Iniciar con compilación completa”
- Para que siempre que ejecutes un programa lo haga con compilación completa dale click a la pestaña “General” del cuadro “opciones” (del menú herramientas) y desactiva la casilla “Compilar a petición” con esto VB analizara el código antes de la ejecución y mostrara los errores. Una vez desactivada “Compilar a petición” se puede ejecutar el programa con F5 o haciendo click en “El play”.

2. Escribir la variables y nombre de controles con mayúsculas y minúsculas

Si vas ha usar una variable por ejemplo ValorUni escríbela así con la V mayúscula y la U mayúscula; además siempre que codifiques hazlo en minúsculas; si la variables esta bien escritas cuando salgas de la línea, valoruni se transformara a: ValorUni, al ver el cambio podrás estar seguro que la escribiste bien. Igual con los nombres de los controles (TextBox, CommandButtons, etc) y además para los controles utiliza los prefijos. Yo utilizo estos:

Prefijo	Control	Ejemplo
tx	Textbox	txCodigo
bt	Botón de comando	btAceptar
frm	Forms	frmPrincipal
lb	Label	lbNombre
opt	Botón de Opción	optClase
cb	Combo	cbLista

3. Definir el tipo y ámbito de una variable

Una variable tiene un tipo (String, Variant, Long, Integer, etc...) y un ámbito (Public, Dim, Static...) se recomienda analizar primero para que va ha ser utilizada la variable y donde se va usar. Ejemplo:

Si necesito una variable para contener el Resultado de una operación aritmética que me pueda arrojar resultados decimales, la declarare como Double, y si la quiero utilizar en todo un formulario escribo esto en “(General)” que se encuentra en la parte superior de editor (debajo de Option Explicit) o dale click al combo “Objeto” y busca “(General)”

Dim Resultado As Double

Si quiero utilizar la variable desde cualquier formulario agrego un modulo: “Proyecto” > “Agregar modulo” y escribo también en “(General)”

Public Resultado As Double

Si solo voy a utilizar la variable en un procedimiento la declaro dentro de él así:

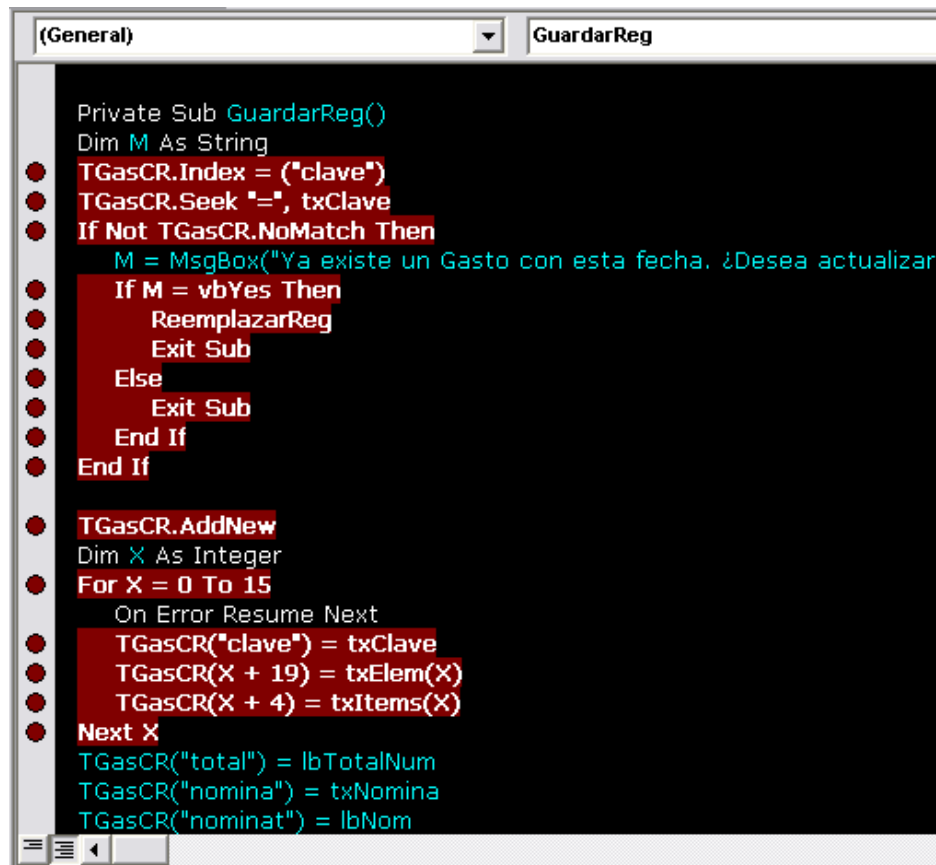
Dim Resultado As Double

Lista de tipos y ámbitos

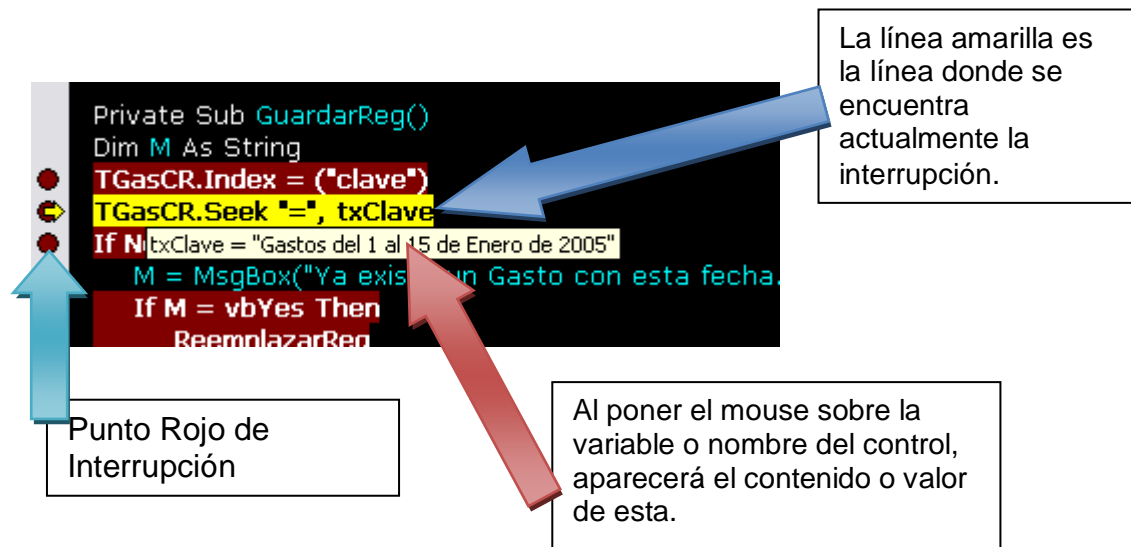
Ámbito	Definición	Tipo	Definición
Dim	Dimensionada. Puede ser a nivel de modulo o formulario o procedimiento	String	De tipo cadena de caracteres
Public	Publica. Se puede utilizar en cualquier parte del proyecto	Integer	Numérico, entero
Static	De valor fijo, estático	Long	Entero largo
Global	Igual que Public (Se solía usar en las versiones anteriores de VB)	Double	Decimal de doble precisión
		Boolean	De tipo lógico
		Variant	De cualquier tipo

4. Depurar el Código

Depurar el código consiste en corregir o eliminar los errores que se producen en este, no es recomendable acumular los errores, en el momento que se van presentando hay que ir depurándolos. Se pueden usar los Puntos de Interrupción para entender paso a paso como corre o se ejecuta un código o procedimiento. Ejemplo:



Los puntos de interrupción (los puntos rojos) indican la línea de código que se interrumpirá en su ejecución, al interrumpirse podemos conocer el estado de las variables con solo poner el mouse y ver el ToolTipText de una variable así:



Para colocar uno o varios puntos de interrupción solo hay que hacer click en el lugar donde aparece en la imagen el punto rojo.

Luego se ejecuta el programa normalmente y este se detendrá y mostrara el editor de código donde se haya señalado.

Para quitarlos, se puede hacer click sobre el punto rojo, o en el menú "Depuración" > "Borrar todos los puntos de interrupción"

Final final, no va mas !!!

Sinceramente creo que este el curso que mas tiempo me ha llevado realizar, la primera edición la publiqué creo que a principios de 2007 y desde entonces (tan solo 3 meses después) he estado escribiendo esta segunda y ultima edición (al menos por mi parte, programando en serio con VB llego a su final).
Que cruel... y tan solo en la segunda edición ☹

Ha sido una experiencia gratificante todos sus mensajes, su apoyo y aceptación de este curso, razón por la cual los invito a que lo distribuyan gratuitamente a todo el que quieran ☺

Ya el curso se acabo, lo que sigue es pura carreta...

DAED Software nació en Febrero de 2003, fecha en la que entre a la carrera técnica de Análisis y Programación de Sistemas, desde entonces programar ha sido una obsesión... ha sido mi vida. En resumen, horas y horas frente al PC diseñando software, interfaces y escribiendo cursos y tutoriales.

Mi interés por enseñar a otros lo que mas me gusta nació de una curiosidad, hace mucho tiempo encontré gracias a Google la Web del Programador, me di cuenta que la pagina era hecha por muchos, con todos sus aportes, tutoriales, cursos y códigos que están a disposición de cualquiera gratuitamente. Quise vincularme de alguna manera a esta web y para probar escribí un manual muy sencillo (y hasta estúpido) de cómo poner una ventana Splash a un programa (era tan sencillo que todo lo que contenía ocupaba tan solo una hoja).

Una semana después revise mi correo y vaya sorpresa: 3 mensajes de diferentes partes del mundo con sugerencias y comentarios sobre el manualcito ese del Splash... no lo dude mas... tenia que ser parte importante de esta Web. De ese modo escribí muchos manuales y tutoriales de diferentes temas. Pero fue solo con Programando en Serio con Visual Basic cuando realmente conocí la fama ☺

Para empezar al los pocos días de enviarlo a la web del programador recibí un mensaje de Xavi (uno de los duros de esta web) FELICITÁNDOME. Al principio

no me la creía. Pero después llegaron mas mensajes de felicitación y hasta llamadas a mi celular. Llegue a reunir 380 mensajes por leer en mi correo !!!

Pongo a continuación ahhh, se me olvidaba. Para el que un día me dijo: ¿Por qué tu le llamas a tus programas u obras: MUERTE Software? y si alguien también cree esto: mi marca es DAED y no DEAD y si, yo se que al revés dice muerte en ingles... es pura coincidencia, DAED son las siglas de mi nombre completo... nada mas.

Ahora si.. Pongo a continuación los nombres de todos aquellos con quienes de una u otra manera estoy agradecido por su contribución a mi formación como Programador:

Javier Miranda, Elkin Lopez, Eydy, Elkin Santis, Marcela Ardila, Gloria Duque, Diego Escobar, Luis Arrieta, Guillermo Som (mi maestro, aunque no se si él sepa), lawebdelprogramador.com, lawebdeprogramacion.com, vbexperto.com, islaprogramacion.com, elguruprogramador.com.ar, portalvb.com, elguille.info, Google.com

DAED Software - Febrero 2008

© Obra libre

The logo for DAED Software. The word "DAED" is written in a large, bold, black, sans-serif font. Below it, the word "SOFTWARE" is written in a smaller, black, sans-serif font, with dots separating each letter: "S.O.F.T.W.A.R.E". The entire logo is set against a white background with a subtle shadow effect.