

**VISUAL BASIC – Diseño de Formularios y manipulación de Datos - Proyecto Cargos**

Este trabajo está asociado a un proyecto de Visual Basic llamado Cargos. Se intenta con ello mostrar cómo funciona el control Data y los demás controles gráficos habituales para captura de datos.

En todos los proyectos y formularios se trata de utilizar una terminología y sintaxis coherente. Esta información está presente en el Anexo que cuenta con dicha información.

Formulario Cargos1

Este formulario, cuyo nombre es cargos.frm, ha sido diseñado para poder ver cómo funciona el control Data en combinación con los controles de texto.

Primero que nada al diseñar un formulario hay que estar consciente de lo que queremos hacer con él. En este caso queremos poder recorrer la tabla, para posteriormente cuando hayamos avanzado poder realizar las Altas, Bajas y Modificaciones en la tabla Cargos.

Es recomendable utilizar para todos los controles, y como política coherente de diseño, nombres que indiquen el tipo de control utilizado y la finalidad del mismo. Ello ayuda conceptualmente y ahorra tiempo cuando hay que referenciar, más cuando unos eventos son dependientes de otros.

Así pues se han seguido los siguientes criterios:

Captura de datos para	Tipo de objeto	Propiedad (Nombre)
IdCargo	Cuadro de texto	txtIdCargo
NombreCargo	Cuadro de texto	txtNbreCargo
Salario Mínimo	Cuadro de texto	txtMinSal
Salario Máximo	Cuadro de texto	txtMaxSal
Datos de la tabla	Control Data	Cargos

Respecto al control Data, que hemos llamado Cargos, se debe establecer:

1. La conexión (connect) que en este caso fue access
2. El nombre de la Base de datos: hay que establecer la ruta de la base de datos Access que vamos a utilizar
3. DefaultCursorType: se refiere al tipo de conexión que vamos a realizar, en este caso se eligió la opción 0 –Default Cursor que es una colección por omisión
4. DefaultType: determina el tipo de datos de origen del cursor que puede ser Jet (si se usa Access) u ODBC para conexiones remotas.
5. RecordSource: es el origen de datos: una tabla, una consulta, etc.
6. RecordSourceType: es el tipo de objeto de dicho origen.

En cuanto a los controles de texto se los ha llamado indicando el tipo de control y su función. Además se deben establecer las propiedades relativas a los datos que van a manejar. En todos se hace en forma semejante, lo que queremos hacer es mostrar en dichos cuadros los valores de los distintos atributos para cada registro de la tabla.

Para ello hay que establecer las siguientes propiedades, que en el caso del txtIdCargo son:

DataSource : es el origen de datos para el control, en este caso la tabla Cargos

DataField: es el campo de datos que se desea mostrar; se selecciona de una lista desplegable por lo tanto el atributo correspondiente, es decir idcargo

Si se ejecuta el formulario se verá que podemos ir de un registro a otro.

Formulario Cargos2

Se basa también en la Tabla cargos. Es más elaborado que el anterior. En primer lugar hemos hecho que en el Data Cargos se muestre el registro absoluto y que además los datos del registro seleccionado se muestren en el área de texto con color azul.

Para ello hemos seguido los mismos criterios, sólo que ahora tenemos un nuevo control al que hemos llamado txtMostrar y que es del tipo cuadro de Texto sólo que se muestra un funciona como un área de edición.

Seguidamente se lista el código de este formulario

Option ExplicitPrivate Sub Position()

'calcula el número de registro del cargo que es distinto al IdRegistro'

'Es equivalente a decir el IdCargo 30 es el registro número...

Cargos.Caption = "Tabla Cargos: " & Str(Cargos.Recordset.AbsolutePosition + 1) & "/" & Str(Cargos.Recordset.RecordCount)

End Sub

Private Sub MostrarReg()

txtMostrar.ForeColor = RGB(0, 0, 255)

'txtMostrar.ForeColor = RGB(255, 0, 0)

txtMostrar.Text = "Base de datos: " & Cargos.DatabaseName & "Tabla: " & Cargos.Recordset.Name & vbNewLine

txtMostrar.Text = txtMostrar.Text & "IdCargo: " & Cargos.Recordset.Fields(0) & vbNewLine

txtMostrar.Text = txtMostrar.Text & "Nombre Cargo: " & Cargos.Recordset.Fields(1) & vbNewLine

txtMostrar.Text = txtMostrar.Text & "Salario Minimo: " & Cargos.Recordset.Fields(2) & vbNewLine

txtMostrar.Text = txtMostrar.Text & "Salario Máximo: " & Cargos.Recordset.Fields(3) & vbNewLine

End Sub

Private Sub Form\_Activate()

Cargos.Caption = "Tabla Cargos: Inicio de Sesión"

End Sub

Private Sub Cargos\_Reposition()

Call Position

Call MostrarReg

End Sub

Formulario Cargos3

Este formulario diseña la primera implementación consistente en poder manipular la tabla cargos. Ello implica poder agregar nuevos registros, modificar y/o eliminar registros existentes.

Se siguen los mismos criterios vistos anteriormente y por ello al botón Agregar se le va a denominar cmdAgr. Además se podrá activar mediante el teclado pulsando la combinación de teclas **Ctrl+ A. Idéntico criterio se sigue con los restantes botones, por ejemplo el boton Modificar se denomina cmdMod.**

Asimismo para poder revertir los cambios originados en inserciones o actualizaciones se Revertir y para poder confirmar dichas actualizaciones el botón Grabar.

El código utilizado es el mismo que el formulario anterior, sólo que se agrega el código pertinente a cada botón:

Private Sub cmdAgr\_Click()

'Se utiliza para agregar un registro nuevo

```
Cargos.Recordset.AddNew
Cargos.Caption = "Proceso de inserción"
cmdAgr.Enabled = False
End Sub
Private Sub cmdAgr_LostFocus()
    txtIdCargo.SetFocus
End Sub

Private Sub cmdMod_Click()
    Cargos.Recordset.Edit
    Cargos.Caption = "Proceso de actualización"
    cmdMod.Enabled = False
End Sub

Private Sub cmdRev_Click()
    'Se utiliza para deshacer una inserción o modificación
    If (cmdMod.Enabled = False Or cmdAgr.Enabled = False) Then
        Cargos.Recordset.CancelUpdate
        Cargos.Caption = "Proceso de reversión de inserción o actualización"
        cmdMod.Enabled = True
        cmdAgr.Enabled = True
    End If
End Sub

Private Sub cmdGrab_Click()
    'Se utiliza para confirmar una inserción o modificación
    If (cmdMod.Enabled = False Or cmdAgr.Enabled = False) Then
        Cargos.Recordset.Update
        Cargos.Caption = "Proceso de confirmación de inserción o actualización"
        cmdMod.Enabled = True
        cmdAgr.Enabled = True
    End If
End Sub

Private Sub Cargos_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    cmdAgr.Enabled = True
    cmdMod.Enabled = True
End Sub

Private Sub cmdElim_Click()
    If (Cargos.Recordset.EOF = False) Then
        Cargos.Recordset.Delete
        Cargos.Caption = "Proceso de eliminación de registro"
    End If
End Sub
```

Con el control Data se pueden editar y modificar los registros directamente. Ocurre que el DML permite actualizar un registro y confirmarlo o no, por ello se han mostrado los botones y el código correspondiente. Las eliminaciones no pueden revertirse ni confirmarse.

Si se ejecuta el formulario verán que no existen controles a la entrada de datos por parte del usuario. Por ejemplo:

1. dos campos son obligatorios: idCargo y nbrecargo y esta validación existen a nivel de la base de datos pero no del formulario.
2. Si se desea agregar un nuevo registro automáticamente VB lo crea con valores por omisión: IdCargo se inicializa en cero lo cual viola la regla de integridad de clave primaria motivo por el cual cuando se intente confirmar el alta la base de datos emitirá un mensaje de error y se puede terminar la aplicación. Esta situación no es conveniente, es decir que termina una aplicación por un motivo como este. Lo que hay que hacer es, además de tener el control desde la base de datos, es

- crear un método que valide que el usuario no deja los campos obligatorios en blanco y que además verifique que son del tipo correcto y de la longitud correcta.
3. por una cuestión de diseño de software no está mal considerar que se pueden manejar los errores o prevenirlos y que además se pueden diseñar ventanas de error o advertencia.

Esto es lo que vamos a agregar en el formulario siguiente y utilizaremos un Array de controles para tener un mayor control de diseño del formulario.

#### Formulario Cargos4

En este formulario hemos utilizado Arrays de Controles (ver ejemplos pertinentes y teoría). Tienen una gran ventaja en cuanto a la manipulación de eventos.

Hemos agregado los controles de validación necesarios para que el usuario no reciba mensajes en forma directa de la base de datos sino que la primera validación en cuenta a la corrección de los datos esté a nivel de la aplicación.

En este formulario se utiliza el control MsgBox que sirve para crear mediante programación cuadros de diálogo. En el Apéndice podrán encontrar las distintas alternativas para generar diferentes tipos.

El código de este formulario es el siguiente:

#### Option Explicit

'Estos métodos son los mismos que están en Cargos3

#### Private Sub Position()

'calcula el número de registro del cargo que es distinto al IdRegistro'

'Es equivalente a decir el IdCargo 30 es el registro número...

```
dataCargos.Caption = "Tabla Cargos: " & Str(dataCargos.Recordset.AbsolutePosition + 1) &  
"/" & Str(dataCargos.Recordset.RecordCount)
```

End Sub

#### Private Sub Form\_Activate()

dataCargos.Caption = "Tabla Cargos: Inicio de Sesión"

End Sub

#### Private Sub Cargos\_Reposition()

Call Position

End Sub

'Ahora vienen los métodos más optimizados utilizando los arrays

'de controles

'PARA EL ARRAY DE CONTROLES de los botones de comandos

#### Private Sub cmdComandos\_Click(Index As Integer)

'usamos un select case

Select Case (Index)

Case 0: 'para insertar

txtTexto(0).SetFocus

dataCargos.Recordset.AddNew

dataCargos.Caption = "Proceso de inserción"

cmdComandos(0).Enabled = False

cmdComandos(1).Enabled = False

cmdComandos(4).Enabled = False

Case 1: 'para modificar un registro existente

txtTexto(0).SetFocus

dataCargos.Recordset.Edit

```
dataCargos.Caption = "Proceso de actualización"
cmdComandos(0).Enabled = False
cmdComandos(1).Enabled = False
cmdComandos(4).Enabled = False
Case 2: 'Se utiliza para confirmar una inserción o modificación
If (cmdComandos(1).Enabled = False Or cmdComandos(0).Enabled = False) Then
dataCargos.Recordset.Update
dataCargos.Caption = "Proceso de confirmación de inserción o actualización"
cmdComandos(1).Enabled = True
cmdComandos(0).Enabled = True
cmdComandos(4).Enabled = True
End If
Case 3: 'Se utiliza para deshacer una inserción o modificación
If (cmdComandos(1).Enabled = False Or cmdComandos(0).Enabled = False) Then
dataCargos.Recordset.CancelUpdate
dataCargos.Caption = "Proceso de reversión de inserción o actualización"
cmdComandos(1).Enabled = True
cmdComandos(0).Enabled = True
cmdComandos(4).Enabled = True
End If
Case Else: 'corresponde al índice 4 que es para eliminar _
un registro existente
If (dataCargos.Recordset.EOF = False) Then
dataCargos.Recordset.Delete
dataCargos.Caption = "Proceso de eliminación de registro"
End If
End Select
End Sub
```

'para validar las entradas realizadas por el usuario en cuanto \_  
a los campos que son obligatorios utilizaremos el método Validate

Private Sub txtTexto\_GotFocus(Index As Integer)

'sirve para darle formato al ingreso de datos del usuario

```
Select Case (Index)
Case 2, 3:
txtTexto(Index).Text = Format(txtTexto(Index).Text, "$#,###,###.00")
End Select
End Sub
```

'el usuario en IdCargo sólo puede ingresar numeros

Private Sub txtTexto\_KeyPress(Index As Integer, KeyAscii As Integer)

Dim i As Integer

```
If (Index = 0) Then
If (KeyAscii >= 48 And KeyAscii <= 57) Then
KeyAscii = KeyAscii
Else
i = MsgBox("Debe Ingresar sólo valores numéricos", vbCritical, "Error")
KeyAscii = 0
End If
End If
```

End Sub

'así se valida que el usuario ingrese sólo 2 números

Private Sub txtTexto\_Validate(Index As Integer, Cancel As Boolean)

Dim i As Integer

Select Case (Index)

Case 0:

'primero se verifica que el campo no esté vacío

```
'luego se verifica para cada campo a considerar la longitud de la cadena
If txtTexto(Index).Text = "" Or txtTexto(Index).Text = 0 Then
    i = MsgBox("Debe ingresar los datos requeridos", vbInformation, "Alerta")
    Cancel = True
Else
    If (Len(txtTexto(Index)) > 2) Then
        i = MsgBox("Debe ingresar hasta dos números", vbInformation, "Alerta")
        Cancel = True
    Else
        Cancel = False
    End If
End If
Case 1:
If txtTexto(Index).Text = "" Then
    i = MsgBox("Debe ingresar los datos requeridos", vbInformation, "Alerta")
    Cancel = True
Else
    If (Len(txtTexto(Index).Text) > 12) Then
        i = MsgBox("Debe ingresar hasta doce cacteres", vbInformation, "Alerta")
        Cancel = True
    Else
        Cancel = False
        'sirve para cambiar toda la cadena a mayúsculas
        txtTexto(Index).Text = UCase(txtTexto(Index).Text)
    End If
End If
Case 2, 3:
    If (Len(txtTexto(Index).Text) > 10) Then
        i = MsgBox("Debe ingresar un valor no mayor a 9.999.999,99", vbInformation,
"Alerta")
        Cancel = True
    Else
        Cancel = False
    End If
End Select
End Sub
```