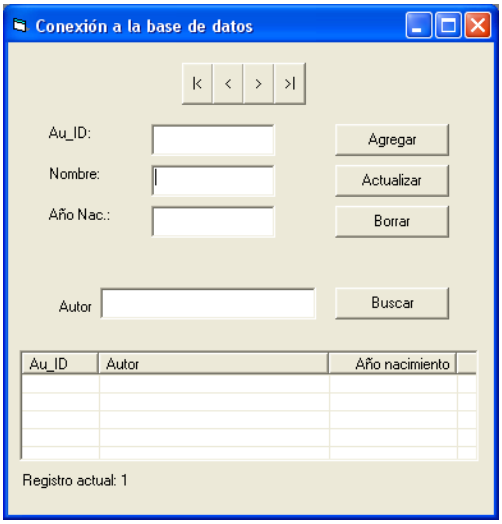
¿Cómo acceder a bases de datos?

Para facilitar la explicación de la conexión a una base de datos utilizaremos la base de datos Access ya que en el capitulo anterior se explica cómo crearla. Usaremos una base existente dentro de la instalación del Visual Basic llamada Biblio.mdb

Para comenzar crear un nuevo proyecto con un formulario como el que se muestra a continuación.



Los controles usados son:
cmdMover, un array de 0 a 3
Label1, un array de 0 a 2, Text1, un array de 0 a 2
cmdAdd, cmdActualizar, cmdBorrar
Label2, Text2, cmdBuscar
Un Listview1 para mostrar los resultados de la búsqueda
Una etiqueta (LblData) para mostrar algunos mensajes
Los controles necesarios son los utilizados hasta el momento en clase, excepto el Listview.

Para agregar un Listview previamente hay que agregarlo a la barra de herramientas de la siguiente forma: Ir al menú “Proyecto”, seleccionar “Componentes” y de la solapa “Controles” tildar la opción “Microsoft Windows Common Controls 5.0” (puede haber otras versiones).

Referencias necesarias:

En el menú Proyecto/Referencias... selecciona Microsoft ActiveX Data Objects 2.6 Library o cualquiera se las versiones, la única que no deberías seleccionar es la 2.0 que ya está obsoleta

Una vez añadida la referencia a los objetos ADO, puedes usar los objetos expuestos por esta librería. En este ejemplo usaremos dos de esos objetos, que serán los que en la mayoría de los casos usemos: el objeto **Connection** y el objeto **Recordset**.

El primero es el que permite acceder a la base de datos y el segundo será el que acceda a los datos propiamente dicho.

Los objetos ADO más comunes:

Normalmente, el objeto Connection suele declararse de forma que sea visible en todo el formulario, salvo en el caso de que añadieses algún, en cuyo caso, deberías declararlo Público o Global.

El objeto Recordset de ADO produce eventos, por tanto, si necesitamos acceder a esos eventos, declararemos la variable con WithEvents, de esa forma podemos interceptar los eventos que produzca de la misma manera que lo hacemos con el resto de controles.
En el procedimiento Buscar, veremos cómo usar otro recordset, pero de la forma tradicional: sin eventos.

Escribe este código en las declaraciones generales del formulario:

```
Option Explicit

' En ADO, se usa el objeto Connection para abrir las bases de datos
Private cnn As ADODB.Connection
' Necesitamos los eventos para controlar algunas cosas
Private WithEvents rst As ADODB.Recordset
```

Al cargar el formulario, creamos los objetos y asignamos la información correspondiente para abrir la base de datos y crear o llenar el Recordset.

La base de datos se abre usando el objeto Connection, del cual usaremos el método Open al cual hay que indicarle el "proveedor" y el nombre de la base de datos:

```
Private Sub Form_Load()
   '
   Text2 = ""
   '
   ' Indicar el path correcto de la base de datos
   ' Lugar donde se encuentra la base de datos
   Const sPathBase As String = "C:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB"
   '
   ' Crear los objetos
   Set cnn = New ADODB.Connection
   Set rst = New ADODB.Recordset
   '
   ' Crear la conexión manualmente
   ' Usar "Provider=Microsoft.Jet.OLEDB.3.51;" para bases de Access 97
   ' Usar "Provider=Microsoft.Jet.OLEDB.4.0;"  para bases de Access 2000
   With cnn
      .ConnectionString = _
         "Provider=Microsoft.Jet.OLEDB.3.51;" & _
         "Data Source=" & sPathBase & ";"
      .Open
   End With
   ' Indicarle de que tabla vamos a leer los datos
   rst.Open "SELECT * FROM Authors", cnn, adOpenDynamic, adLockOptimistic

   With ListView1
      ' El tipo de Listview que queremos es del tipo "reporte"
      .View = lvwReport
      ' Que muestre las líneas de separación entre datos
      .GridLines = True
      ' Que no se puedan modificar los datos del listview
      .LabelEdit = lvwManual
      ' Añadimos las cabeceras
      .ColumnHeaders.Add , , "Au_ID", 900
      .ColumnHeaders.Add , , "Autor", 2700
      .ColumnHeaders.Add , , "Año nacimiento", 1500, lvwColumnRight
   End With
   '
   ' Si hay datos, posicionarlo en el primer registro:
```

```
    cmdMover_Click 0
End Sub
```

Como el Recordset de ADO produce eventos, vamos a usar uno de esos eventos: MoveComplete, el cual se produce cada vez que se cambia el registro activo para actualizar la información del registro que está activo.

Las variables declaradas con WithEvents siguen la misma "nomenclatura" que los eventos de los controles, por tanto, ese evento estará en: rst_MoveComplete (selecciona el objeto rst de la lista desplegable derecha y el evento MoveComplete de la lista de la izquierda), éste es el código para mostrar los datos cada vez que se cambia el registro activo, aunque realmente no sería necesario si hubiésemos "ligado" los controles con el recordset... pero de esa forma no tendríamos el control total sobre los datos, así que vamos a seguir con esto de usar los eventos:

```
Private Sub rst_MoveComplete(ByVal adReason As ADODB.EventReasonEnum, _
                                                    ByVal pError As ADODB.Error, adStatus As
                                                    ADODB.EventStatusEnum, _
                                                     ByVal pRecordset As ADODB.Recordset)
    ' Cada vez que el registro actual cambie,
    ' se producirá este evento (igual que con el DataControl)
    With rst
            '--------------------------------------------------------------------
        ' Cuando en un Recordset no hay datos, tanto BOF como EOF devuelven True
            '--------------------------------------------------------------------
        If .EOF And .BOF Then
            lblData.Caption = "No hay ningún registro activo"
            .MoveFirst
        Else
            Text1(0) = .Fields("Au_ID")
            ' Si el dato es nulo, añadirle una cadena vacia
            Text1(1) = .Fields("Author") & ""
            Text1(2) = .Fields("Year Born") & ""
        End If
    End With
End Sub
```

En este evento podemos usar tanto el objeto rst como el que está en el parámetro: pRecordset, los dos se refieren al mismo objeto.

Hacemos una pequeña comprobación de que no nos encontremos con un recordset vacío o que esté fuera de los límites permitidos, cosa que ocurre cuando queremos pasar al siguiente registro cuando estamos al final (se produce EOF) o cuando pasamos al registro anterior y estamos al principio, (se produce BOF).

Si todo va bien, asignamos a las cajas de textos el contenido de los campos correspondientes.

En algunos de los campos añadimos una cadena vacía al contenido del campo, esto es para los casos en que esos campos contengan un valor nulo.

El resto del código para los botones de Mover, Nuevo, Actualizar y Eliminar los datos.

```
Private Sub cmdMover_Click(Index As Integer)
    ' Mover según el botón pulsado
    'On Error Resume Next
    '
    With rst
        If Index = 0 Then       ' Primero
            .MoveFirst
        ElseIf Index = 1 Then   ' Anterior
            .MovePrevious
        ElseIf Index = 2 Then   ' Siguiente
```

```
            .MoveNext
        ElseIf Index = 3 Then   ' Último
            .MoveLast
        End If
        '
        If .BOF Or .EOF Then
            .MoveFirst
            lblData.Caption = " No hay datos..."
        Else
            lblData.Caption = " Registro actual: " & rst("Au_ID")
        End If
    End With
    '
    Err = 0
End Sub

Private Sub cmdAdd_Click()
    ' Añadir un nuevo registro
    rst.AddNew
    ' Añadimos algún texto, para que no se pierda este registro
    Text1(1) = "Nuevo"
    ' Actualizamos los datos
    rst.Update
    ' Movemos al último registro para que los cambios se hagan permanentes
    ' y se muestre el nuevo registro
    rst.MoveLast
End Sub

Private Sub cmdActualizar_Click()
    ' Guardar el contenido de las cajas de texto
    With rst
        ' Este campo es auto numérico, así que no asignarlo
        '.Fields("Au_ID") = Text1(0) + 0
        ' Añadimos una cadena vacía al final
        ' ya que si Text1(1) está vacío, se asignará un valor NULL y dará error
        .Fields("Author") = Text1(1) & ""
        ' Idem con el año de nacimiento, pero como es numérico, se sumará 0
        .Fields("Year Born") = Text1(2) + 0
        ' Actualizar los datos en el recordset
        .Update
    End With
End Sub

Private Sub cmdBorrar_Click()
    ' Borrar el registro actual
    ' Se comprueba que haya algún registro activo,
    ' para ello se comprueba que no hayamos pasado del principio o el final del Recordset
    '
    ' Comprobar que hay registros, porque si no hay, dará error
    If (rst.EOF Or rst.BOF) Then
        ' Avisar de que no hay registros
        lblData.Caption = "Ningún registro activo"
    Else
        ' Eliminar el registro actual
        rst.Delete
        '
        ' Movemos al primer registro para que los cambios se hagan permanentes
        ' (también podríamos haberlo movido al último registro)
```

```
        rst.MoveFirst
    End If
End Sub
```

Para terminar, vamos a ver el código para Buscar datos:

```
Private Sub cmdBuscar_Click()
    ' Mostrar los datos en el listview
    Dim sBuscar As String
    Dim tRs As Recordset
    Dim tLi As ListItem
    '
    ' Comprobar si tiene caracteres "no válidos" para ADO:
    ' NOTA: Replace es una función de VB6
    sBuscar = Text2
    sBuscar = Replace(sBuscar, "*", "%")
    sBuscar = Replace(sBuscar, "?", "_")
    '
    Text2 = sBuscar
    ' Formar la cadena de la consulta:
    ' Se busca por el nombre del autor y se muestran clasificados por el nombre
    sBuscar = "SELECT * FROM Authors WHERE Author LIKE '" & sBuscar & "' ORDER BY Author"
    ' Creamos un recordset del tipo "estático", el cual no es modificable
    ' para poder modificarlo, tendría que ser del tipo dbOpenDynamic
    Set tRs = cnn.Execute(sBuscar)
    ' Comprobar que hay datos en el recordset
    With tRs
        ' Si no hay datos...
            '-------------------------------------------------------------------------
            ' Cuando en un Recordset no hay datos, tanto BOF como EOF devuelven True
            '-------------------------------------------------------------------------
        If (.BOF And .EOF) Then
            MsgBox "No se han encontrado los datos buscados"
        Else
            ' Mostrar los datos hallados
            ListView1.ListItems.Clear
            .MoveFirst
            Do While Not .EOF
                Set tLi = ListView1.ListItems.Add(, , .Fields("Au_ID") & "")
                tLi.SubItems(1) = .Fields("Author") & ""
                tLi.SubItems(2) = .Fields("Year Born") & ""
                .MoveNext
            Loop
        End If
    End With
End Sub
```

Caracteres comodines:

Hacemos una comprobación para cambiar los caracteres comodines que hubiese en el texto indicado para la búsqueda.

Según la ayuda de ADO, se usará el carácter % (tanto por ciento), cuando queramos indicar cualquier cosa que haya en el sitio en que se encuentra dicho carácter, por ejemplo: %jan% encontrará todos los registros que contenga "jan", esté en el sitio que esté: Janet, Alejandro, etc. Si pusiéramos jan%, sólo mostraría los que empezaran con jan, etc.

Por otro lado, el carácter _ (subrayado bajo) significa "cualquier carácter que esté en esa posición", por ejemplo: %r_us% encontrará cualquier palabra que contenga: una r seguida de cualquier cosa, seguida de us. Tal es el caso de Rouse, Marcus, etc.