Programación con

Visual Basic .NET

http://alarcos.inf-cr.uclm.es/per/fruiz/cur/vbn/vbn.htm

4 – Interfaz de Usuario. Entrada/Salida. Desarrollo de una aplicación

Francisco Ruiz

Manuel Serrano

Escuela Superior de Informática

Universidad de Castilla-La Mancha

Programación con **Visual Basic .NET**

Contenidos sesión 4

- Interfaz de Usuario
  - Formularios.
  - Controles.
  - Menús.
  - Formularios MDI.
- Archivos.
- Acceso a datos. ADO.NET
- Desarrollo de una aplicación.

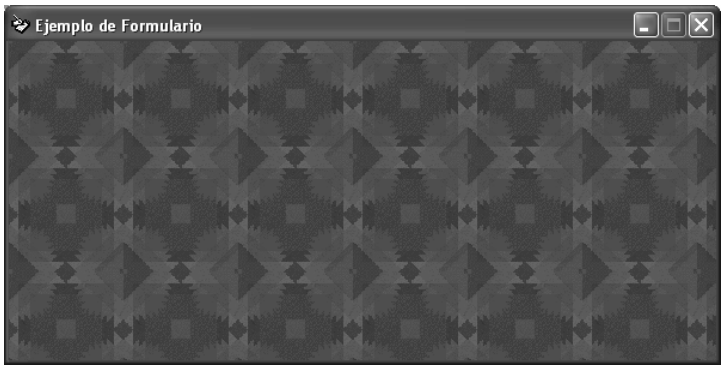
Formularios.

Creación de un formulario (i).

- System.Windows.Forms
- Clase Form
  - Size
  - Location / StartPosition(Manual)
  - BackgroundImage
  - Icon
  - MaximizeBox
  - FormBorderStyle
  - Text
  - (Name) --error

Formularios.

Creación de un formulario (ii).



- Ver->Código (F7)

Formularios.

Eventos del formulario.

```
Private Sub frmEjemplo_MouseMove(ByVal sender As Object, ByVal _
   e As System.Windows.Forms.MouseEventArgs) Handles _
   MyBase.MouseMove
        Me.Text = "Coordenadas ratón: X:" & e.X & " Y:" & e.Y
End Sub


Private Sub frmEjemplo_Closing(ByVal sender As Object, ByVal _
   e As System.ComponentModel.CancelEventArgs) Handles _
   MyBase.Closing
        If MessageBox.Show("¿Cerrar la ventana?", _
          "Atención", MessageBoxButtons.YesNo, _
           MessageBoxIcon.Hand) = DialogResult.No Then
             e.Cancel = True
        End If
End Sub
```


Controles.

Insertar controles en un formulario.

- Cuadro de herramientas.
- Clic sobre el control (selección).
- Doble clic / <Intro> (inserción).
- GridSize / SnapToGrid / DrawGrid
- Herramientas->Opciones->Diseñador de Windows Forms
- Formato (Alinear, Igualar tamaño, Bloquear controles...)
- Anchor
- Dock

Controles.

Controles más habituales.

- Button
- Label
- TextBox
- ListBox
- ComboBox
- CheckBox
- RadioButton
- GroupBox

Controles.

Controles más habituales. Button.

- Text
- TextAlign
- BackColor
- Cursor
- Image
- ImageAlign
- BackgroundImage
- FlatStyle
- Font

Name: btnMensaje

Text: Mostrar Mensaje

btnMensaje_Click

```
MessageBox.Show("Has pulsado
    el botón del formulario")
```

btnMensaje_MouseEnter

```
Me.btnMensaje.BackColor = Color.Cyan
```

btnMensaje_MouseLeave

```
Me.btnMensaje.ResetBackColor()
```

Controles.

Controles más habituales. Label.

- BorderStyle

Controles.

Controles más habituales. TextBox (i).

- Text
- Multiline
- WordWrap
- Enabled
- ReadOnly
- CharacterCasing
- MaxLength
- PasswordChar
- AutoSize

Controles.

Controles más habituales. TextBox (ii).

btnFoco_Click:

```
Me.txtNombre.Focus()
```

btnSoloLectura_Click:

```
If (Me.txtNombre.ReadOnly) Then
    Me.txtNombre.ReadOnly = False
Else
    Me.txtNombre.ReadOnly = True
End If
```

btnActivar_Click:

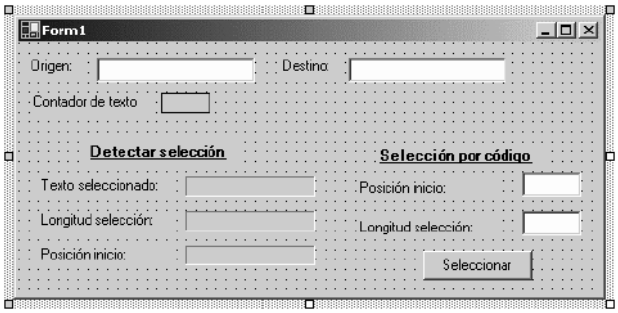
```
Me.txtApellidos.Enabled =
    Not (Me.txtApellidos.Enabled)
```

- TabIndex (Ver->Orden de tabulación) / TabStop

Controles.

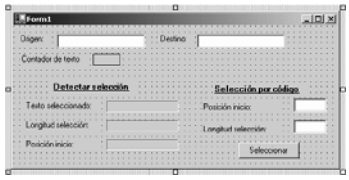
Controles más habituales. TextBox (iii).

- Copiar/Cortar/Pegar -> transparente al usuario
  - SelectionStart
  - SelectionLength
  - SelectedText



Controles.

Controles más habituales. TextBox (iv).



txtOrigen_TextChanged:

```
Me.lblContador.Text =
   Me.txtOrigen.TextLength
```

txtOrigen_MouseMove:

```
If e.Button.Left Then
   Me.lblTextoSelec.Text = Me.txtOrigen.SelectedText
   Me.lblLongitud.Text = Me.txtOrigen.SelectionLength
   Me.lblPosicion.Text = Me.txtOrigen.SelectionStart
End If
```

txtOrigen_KeyDown:

```
If e.Shift Then    'mayúsculas
   'flecha derecha
   If e.KeyCode.Right Then
        …
   End If
End If
```

btnSeleccionar_Click:

```
Me.txtPosicion.Text =
   Me.txtOrigen.SelectionStart
Me.txtLongitud.Text =
   Me.txtOrigen.SelectionLength
Me.txtDestino.Text =
   Me.txtOrigen.SelectedText
```


Controles.

Controles más habituales. RadioButton y GroupBox.

rbtTahoma_CheckedChanged:

```
Me.txtNombre.Font = New Font("Tahoma", 12)
```

rbtGaramond_CheckedChanged:

```
Me.txtNombre.Font = New Font("Garamond", 8)
```

rbtComic_CheckedChanged:

```
Me.txtNombre.Font = New Font("Comic Sans MS", 12)
```

rbtVerde_CheckedChanged:

```
Me.txtNombre.BackColor = Color.Green
```

rbtAzul_CheckedChanged:

```
Me.txtNombre.BackColor = Color.Blue
```

rbtAmarillo_CheckedChanged:

```
Me.txtNombre.BackColor = Color.Yellow
```

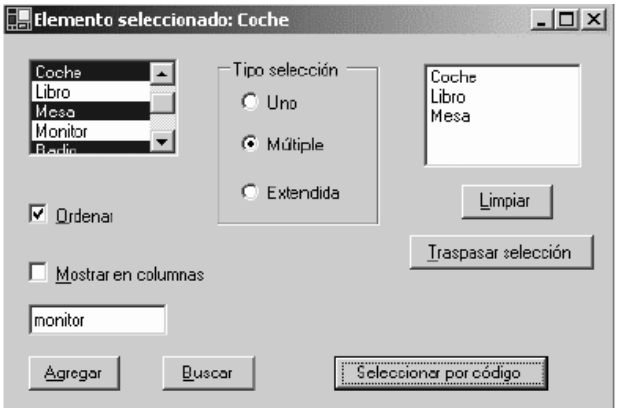
Controles.

Controles más habituales. ListBox (i).

- Items
- Sorted
- IntegralHeight
- MultiColumn
- SelectionMode
- SelectedItem
- SelectedItems
- SelectedIndex

Controles.

Controles más habituales. ListBox (ii).



Controles.

Controles más habituales. ListBox (iii).

```
Public Const TITULO As String = "Elemento seleccionado: "

lstValores_SelectedIndexChanged:
Me.Text = TITULO & Me.lstValores.SelectedItem

rbtUno_CheckedChanged:
Me.lstValores.SelectionMode = Selection.One
rbtMultiple_CheckedChanged:
Me.lstValores.SelectionMode = Selection.MultiSimple
rbtExtendida_CheckedChanged:
Me.lstValores.SelectionMode = Selection.MultiExtended

chkOrdenar_CheckedChanged:
Me.lstValores.Sorted = Me.chkOrdenar.Checked
chkColumnas_CheckedChanged:
Me.lstValores.MultiColumn = Me.chkColumnas.Checked
```


Controles.

Controles más habituales. ListBox (iv).

```
btnAgregar_Click:
   Me.lstValores.Items.Add(Me.txtValor.Text)
btnBuscar_Click:
   Dim iPosicion As Integer
   ' el método FindString() de la lista busca un valor
   iPosicion = Me.lstValores.FindString(Me.txtValor.Text)
   ' el campo NoMatches indica si no existe el valor buscado
   If iPosicion = Me.lstValores.NoMatches Then
        MessageBox.Show("No existe el valor")
   Else
        ' si encontramos el valor en la lista,
        ' lo seleccionamos por código
        Me.lstValores.SelectedIndex = iPosicion
   End If
```

Controles.

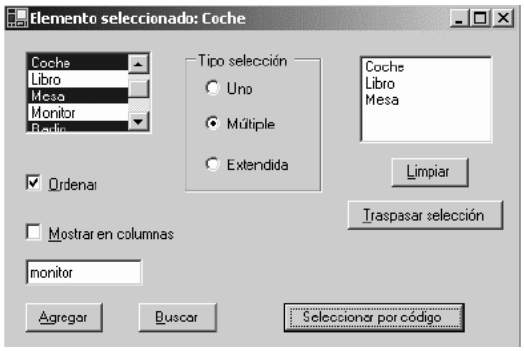
Controles más habituales. ListBox (v).

```
btnSelecCod_Click:
   Me.rbtMultiple.Checked = True
   Me.lstValores.SetSelected(1, True)
   Me.lstValores.SetSelected(3, True)
   Me.lstValores.SetSelected(5, True)
btnTraspasarSelec_Click:
   Dim oSeleccion As ListBox.SelectedObjectCollection
   ' obtenemos con los elementos seleccionados del ListBox
   oSeleccion = Me.lstValores.SelectedItems
   ' si hay elementos seleccionados, los traspasamos a otro ListBox
   If oSeleccion.Count > 0 Then
        Dim oEnumerador As IEnumerator
        oEnumerador = oSeleccion.GetEnumerator()
        While oEnumerador.MoveNext()
               Me.lstTraspaso.Items.Add(oEnumerador.Current)
        End While
   End If
```


Controles.

Controles más habituales. ListBox (vi).

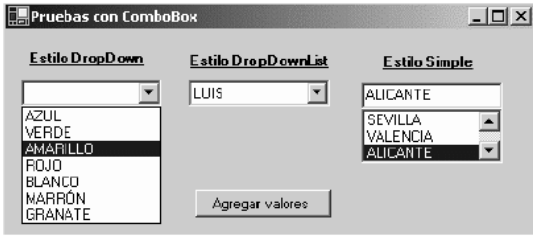
```
btnLimpiar_Click:
   Me.lstTraspaso.Items.Clear()
```



Controles.

Controles más habituales. ComboBox.

- TextBox + ListBox
- DropDownStyle
- DropDownList
- MaxDropDownItems




```
Me.cboColores.Items.AddRange(New String() {"AZUL", "VERDE", _
   "AMARILLO", "ROJO", "BLANCO", "MARRÓN", "GRANATE“)
```


Menús.

- MainMenu, ContextMenu (contenedores)
- MenuItem

Menús.

MainMenu (i).

- MainMenu, ContextMenu (contenedores)
- MenuItem
  - Text
    - &
    - -
  - Enabled
  - DefaultItem
  - Checked
  - RadioCheck
  - ShortCut
  - ShowShortCut
  - Visible
  - MdiList

Menús.

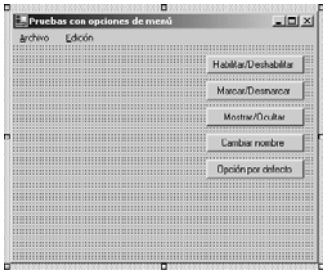
MainMenu (ii).

```
mnuAbrir_Click: MessageBox.Show("Opción Abrir")
mnuSalir_Click: Me.Close()
```

Menús.

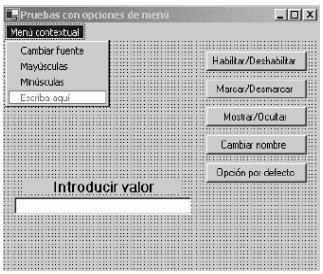
MainMenu (iii).

```
btnHabilitar_Click:
Me.mnuGuardar.Enabled = Not Me.mnuGuardar.Enabled
btnMarcar_Click:
Me.mnuPegar.Checked = Not Me.mnuPegar.Checked
btnMostrar_Click:
Me.mnuElipse.Visible = Not Me.mnuElipse.Visible
btnNombre_Click:
If Me.mnuAbrir.Text = "A&brir" Then
   Me.mnuAbrir.Text = "HO&LA"
Else
   Me.mnuAbrir.Text = "A&brir"
End If
btnDefecto_Click:
Me.mnuCopiar.DefaultItem = Not Me.mnuCopiar.DefaultItem
```



Menús.

ContextMenu.




```
mnuFuente_Click:
Dim oFuente As New Font("Comic", 15)
Me.txtValor.Font = oFuente
mnuMayusculas_Click:
Me.txtValor.Text =  Me.txtValor.Text.ToUpper()
mnuMinusculas_Click:
Me.txtValor.Text =  Me.txtValor.Text.ToLower()
```

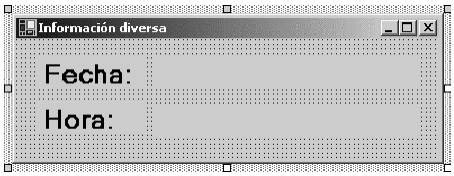
Formularios MDI (Multiple Document Interface) (i).

- 1 Formulario MDI + n Formularios Documento.
- IsMDIContainer / MdiParent


```
Imports System.IO

btnGrabar_Click:
Dim oEscritor as StreamWriter
oEscritor =
   File.CreateText(Me.txtArchivo.Text)
oEscritor.Write(Me.txtCarta.Text)
oEscritor.Close()
```


Formularios MDI (ii).

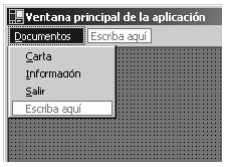
```
tmrTiempo_Tick:
Dim dtFecha As Date
dtFecha = DateTime.Today
Dim dtHora As Date
dtHora = DateTime.Now

Me.lblFecha.Text = dtFecha.ToString("d/MMM/yyyy")
Me.lblHora.Text = dtHora.ToString("h:m:s")
```



Formularios MDI (iii).

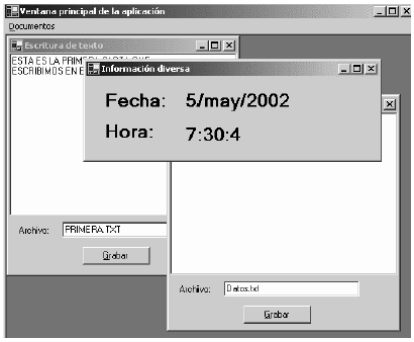
mnuCarta_Click:

```
Dim ofrmCarta As New frmCarta()
ofrmCarta.MdiParent = Me
ofrmCarta.Show()
```



mnuInformacion_Click:

```
Dim ofrmInfo As New frmInfo()
ofrmInfo.MdiParent = Me
ofrmInfo.Show()
```



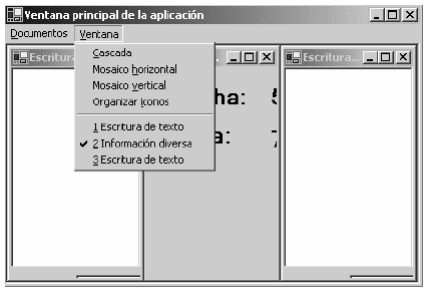
Formularios MDI (iv).

mnuCascada_Click:

```
Me.LayoutMdi(MdiLayout.Cascade)
```

mnuHorizontal_Click:

```
Me.LayoutMdi(MdiLayout.TileHorizontal)
```



Archivos.

Gestión del Sistema de Archivos.

- System.IO
- Objetos Stream
- TextReader / TextWriter
  - StreamReader, StreamWriter

Archivos.

La clase StreamWriter

```
Imports System.IO
Module Module1
    Sub Main()
        Dim swEscritor As StreamWriter
        ' creamos un stream de escritura, y al mismo tiempo un
        ' nuevo archivo para escribir texto sobre él
        swEscritor = New StreamWriter("\pruebas\NOTAS.txt")
        ' escribir líneas
        swEscritor.WriteLine("esta es la primera línea")
        swEscritor.WriteLine("segunda línea de texto")
        ' ahora escribimos texto pero sin provocar un salto de línea
        swEscritor.Write("Juan y Luna ")
        swEscritor.Write("van de paseo")
        swEscritor.Write(swEscritor.NewLine) ' esto introduce el salto de línea
        swEscritor.WriteLine("con esta línea cerramos")
        ' cerrar el stream y el archivo asociado
        swEscritor.Close()
    End Sub
End Module
```

Archivos.

La clase StreamReader (i)

```
Dim srLector As StreamReader = New StreamReader("\pruebas\NOTAS.txt")
Console.WriteLine("**Leer una primera línea**")
Dim Linea As String
Linea = srLector.ReadLine()
Console.WriteLine("La línea contiene --> {0}", Linea)
Console.WriteLine()
Console.WriteLine("**Ahora leemos el resto del archivo**")
Dim Texto As String
Texto = srLector.ReadToEnd()
Console.WriteLine("El texto restante contiene --> {0}", Texto)
srLector.Close()

' leer línea a línea mediante un bucle
Dim srLector As StreamReader = New StreamReader("\pruebas\Datos.txt")
Dim Linea As String
Dim ContadorLin As Integer = 1
Linea = srLector.ReadLine()
Do While Not (Linea Is Nothing)
     Console.WriteLine("Línea: {0} - Contenido: {1}", ContadorLin, Linea)
     ContadorLin += 1
     Linea = srLector.ReadLine()
Loop
```


Archivos.

La clase StreamReader (ii)

```
Imports System.IO
Imports System.Text
Module Module1
Sub Main()
     Dim srLector As StreamReader = New StreamReader("\pruebas\NOTAS.txt")
     ' obtener valores del stream con el método Read()
     Dim Valor As Integer : Dim Codigos() As Byte
     ' volcamos en un bucle los códigos de carácter leidos desde el archivo a un array Byte
     Valor = srLector.Read()
     While (Valor <> -1) ' cuando lleguemos al final, obtendremos -1
         If Codigos Is Nothing Then : ReDim Codigos(0)
         Else : ReDim Preserve Codigos(Codigos.GetUpperBound(0) + 1)
         End If
         Codigos(Codigos.GetUpperBound(0)) = Valor
         Valor = srLector.Read()
     End While
     Dim Codificador As New ASCIIEncoding() : Dim Parte2 As String
     ' con un ASCIIEncoding, y el método GetString(), obtenemos una cadena, pasando un array Bytes
     Parte2 = Codificador.GetString(Codigos)
     Console.WriteLine("Resultado:") : Console.WriteLine(Parte2)
     Console.ReadLine()
End Sub
End Module
```

Archivos.

La clase StreamReader (ii)

```
        ' obtener valores del stream con el método Read()
        Dim Valor As Integer : Dim Codigos() As Byte
        ' vamos a ir volcando en un bucle los códigos de carácter
        ' leidos desde el archivo a un array Byte
        Valor = srLector.Read()
        While (Valor <> -1) ' cuando lleguemos al final, obtendremos -1
            If Codigos Is Nothing Then
                    ReDim Codigos(0)
            Else : ReDim Preserve Codigos(Codigos.GetUpperBound(0) + 1)
            End If
            Codigos(Codigos.GetUpperBound(0)) = Valor
            Valor = srLector.Read()
        End While
        Dim Codificador As New ASCIIEncoding() : Dim Parte2 As String
        ' con el objeto ASCIIEncoding, método GetString(),
        ' obtenemos una cadena, pasando como parámetro un array de tipos Byte
        Parte2 = Codificador.GetString(Codigos)
        Console.WriteLine("Resultado de la lectura con ReadBlock()")
        Console.WriteLine(Parte2)
        Console.ReadLine()
    End Sub
    End Module
```


Archivos.

La clase FileStream (i)

```
' escrituras con Filestream
Dim oFileStream As FileStream
oFileStream = New FileStream("\pruebas\apuntes.dtt", FileMode.CreateNew)
oFileStream.Write(New Byte() {15, 160, 88, 40, 67, 24, 37, 50, 21}, 0, 6)
oFileStream.WriteByte(75)
Console.WriteLine("Opciones en el FileStream")
Console.WriteLine("Podemos leer: {0}", IIf(oFileStream.CanRead, "SI", "NO"))
Console.WriteLine("Podemos escribir: {0}", IIf(oFileStream.CanWrite, "SI", "NO"))
Console.WriteLine("Podemos movernos: {0}", IIf(oFileStream.CanSeek, "SI", "NO"))
oFileStream.Close()
oFileStream = Nothing
```

Archivos.

La clase FileStream (ii)

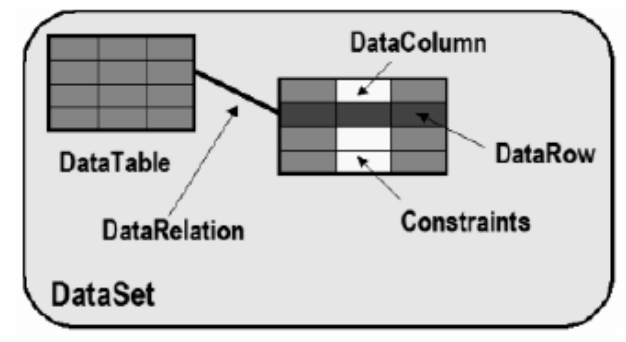
```
' lectura con FileStream
Dim oFileStream As FileStream
oFileStream = New FileStream("\pruebas\apuntes.dtt", FileMode.Open)
Dim Valor As Byte
Valor = oFileStream.ReadByte() ' obtener un valor
Console.WriteLine("Se ha leido el valor: {0}", Valor)
Console.WriteLine("Nos desplazamos dos bytes en el stream")
oFileStream.Seek(2, SeekOrigin.Begin)
Valor = oFileStream.ReadByte()
Console.WriteLine("Se ha leido el valor: {0}", Valor)
Console.WriteLine("La posición actual del stream es: {0}", _
oFileStream.Position)
' leer varios valores, pasándolos a un array previamente dimensionado
Dim VariosValores(3) As Byte
oFileStream.Read(VariosValores, 0, 4)
Console.WriteLine("Leer bloque de valores del stream")
Dim Enumerador As IEnumerator
Enumerador = VariosValores.GetEnumerator()
While Enumerador.MoveNext
     Console.WriteLine("Valor: {0}", Enumerador.Current)
End While
Console.ReadLine()
```


Acceso a datos. ADO.NET

- Conjunto de interfaces, clases, estructuras y enumeraciones que permiten el acceso a datos desde la plataforma .NET
- Permite un modo desconectado a los datos.
- Los datos pueden provenir de múltiples fuentes de datos y de diferentes arquitecturas de almacenamiento.
- Basado en XML.

ADO.NET

La clase DataSet



ADO.NET

Espacios de Nombres

- System.Data
- System.Data.SqlClient
- System.Data.OleDb
- System.Data.SqlTypes
- System.Data.Common
- System.Data.Internal

ADO.NET

System.Data

- DataSet
- DataTable
- DataRow
- DataColumn
- DataRelation
- Constraint
- DataColumnMapping
- DataTableMapping

ADO.NET

System.Data.SqlClient / System.Data.OleDb

- SqlCommand / OleDbCommand
- SqlConnection / OleDbConnection
- SqlCommandBuilder / OleDbCommandBuilder
- SqlDataReader / OleDbDataReader
- SqlDataAdapter / OleDbDataAdapter
- SqlParameter / OleDbParameter
- SqlTransaction / OleDbTransaction

ADO.NET

Las clases Connection

```
Imports System.Data.SqlClient
'....
Try
' crear el objeto de conexión
Dim oConexion As New SqlConnection()
' pasar la cadena de conexión
oConexion.ConnectionString = "server=(local);" & _
    "database=Xnorthwind;uid=sa;pwd=;"
' abrir conexión
oConexion.Open()
MessageBox.Show("Conectado")
' cerrar conexión
oConexion.Close()
MessageBox.Show("Desconectado")
Catch oExcep As SqlException
' si se produce algún error, lo capturamos mediante el objeto
' de excepciones particular para el proveedor de SQL Server
MessageBox.Show("Error al conectar con datos" & ControlChars.CrLf & _
   oExcep.Message & ControlChars.CrLf & oExcep.Server)
End Try
```


ADO.NET

Las clases Command (i)

- CommandText
- CommandTimeout
- CommandType
- Connection
- Parameters
- CreateParameter
- ExecuteNonQuery
- ExecuteReader
- ExecuteScalar
- Prepare

ADO.NET

Las clases Command (ii)

```
' crear conexión
Dim oConexion As New SqlConnection()
oConexion.ConnectionString = "Server=(local);" & _
   "Database=Gestion;uid=sa;pwd=;"
' crear sentencia SQL
Dim sSQL As String
sSQL = "INSERT INTO Clientes (IDCliente,Nombre,FIngreso) " & _
   "VALUES(10,'Alfredo','18/7/2002')"
' crear comando
Dim oComando As New SqlCommand(sSQL, oConexion)
Dim iResultado As Integer
oConexion.Open() ' abrir conexión
iResultado = oComando.ExecuteNonQuery() ' ejecutar comando
oConexion.Close() ' cerrar conexión
MessageBox.Show("Registros añadidos:" & iResultado)
```


ADO.NET

Las clases DataReader (i)

- FieldCount
- IsClosed
- Item
- Close
- GetXXX
- NextResult
- Read

ADO.NET

Las clases DataReader (ii)

```
Private Sub btnEmpleados_Click(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles btnEmpleados.Click
' crear conexion
Dim oConexion As New SqlConnection()
oConexion.ConnectionString = "Server=(local);" & _
   "Database=Northwind;uid=sa;pwd=;"
' crear comando
Dim oComando As New SqlCommand("SELECT * FROM Employees", oConexion)
' crear DataReader
Dim oDataReader As SqlDataReader
oConexion.Open()
oDataReader = oComando.ExecuteReader() ' obtener DataReader
' recorrer filas
While oDataReader.Read()
   Me.lstEmpleados.Items.Add(oDataReader("LastName"))
End While
oDataReader.Close()
oConexion.Close()
End Sub
```


ADO.NET

La clase DataSet (i)

- Clear
- AcceptChanges
- GetChanges
- HasChanges
- RejectChanges
- Merge
- CaseSensitive
- DataSetName
- HasErrors
- Relations
- Tables

ADO.NET

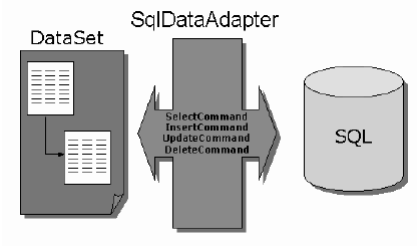
La clase DataSet (ii)

```
' crear conexión
Dim oConexion As New SqlConnection()
oConexion.ConnectionString = "Server=(local);Database=Northwind;uid=sa;pwd=;"
' crear adaptador
Dim oDataAdapter As New SqlDataAdapter("SELECT * FROM Customers ORDER BY
   ContactName", oConexion)
' crear conjunto de datos
Dim oDataSet As New DataSet()
oConexion.Open()
' utilizar el adaptador para llenar el dataset con una tabla
oDataAdapter.Fill(oDataSet, "Customers")
oConexion.Close()
' una vez desconectados, recorrer la tabla del dataset
Dim oTabla As DataTable
oTabla = oDataSet.Tables("Customers")
Dim oFila As DataRow
For Each oFila In oTabla.Rows
   ' mostrar los datos mediante un objeto fila
   Me.lstCustomers.Items.Add(oFila.Item("CompanyName")
        & " - " & oFila.Item("ContactName") & " - " & _
oFila.Item("Country"))
Next
```


ADO.NET

Las clases DataAdapter (i)

- InsertCommand
- SelectCommand
- UpdateCommand
- DeleteCommand



Desarrollo de una Aplicación

- ProTex v. 1.0
  - Sencillo procesador de textos
  - Aplicación MDI con 2 formularios
  - Formulario frmPrincipal (Menú):
    - Archivo: Nuevo, Abrir, Guardar, Salir
    - Formato: Fuente, Tamaño, Color de Texto, Color de Fondo
    - Ventana: Cascada, Horizontal, Vertical
  - Formularios frmDocumento (cuadro de texto)