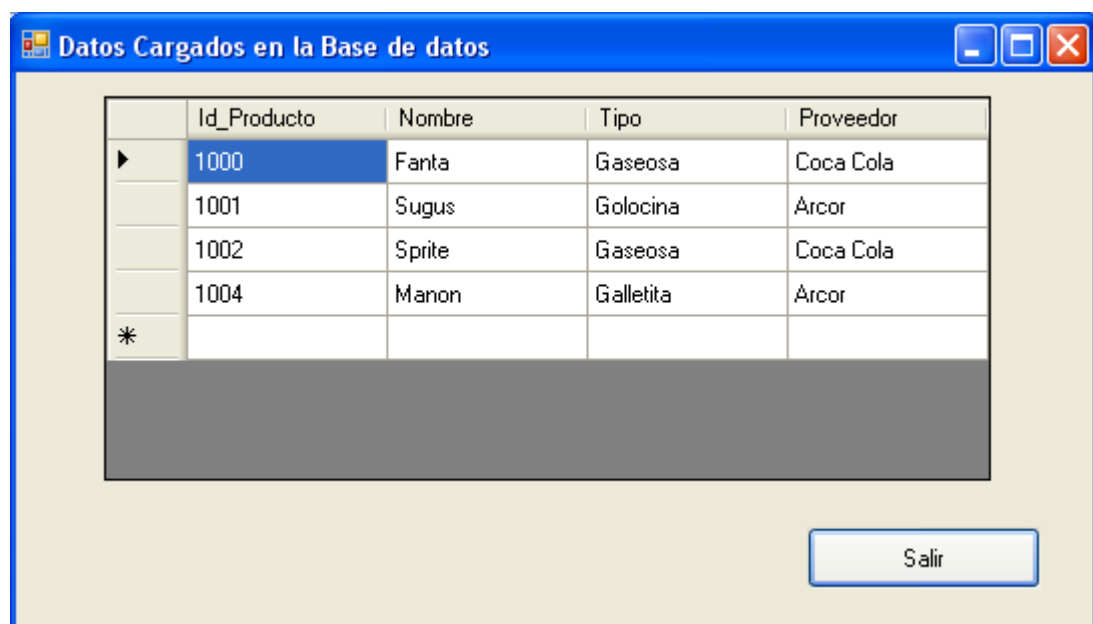


Introducción

Vamos a trabajar sobre la base de datos UTN_Negocio en forma desconectada, realizando todas las actualizaciones en un DataSet (representación de una base de datos en memoria) y luego actualizaremos la base de datos utilizando un objeto de tipo SqlDataAdapter.



Primero debemos declarar las variables que utilizaremos en el proyecto.

En el DataSet guardaremos la tabla que traemos a partir del comando Select del DataAdapter.

Se modificarán y se actualizarán estos cambios en la base de datos a través del objeto SqlDataAdapter, con la conexión que configuramos en el objeto SqlConnection.

CÓDIGO:

```
Private _dataSet As DataSet  
Private _dataAdapter As SqlDataAdapter  
Private _Select As SqlCommand  
Private _Insert As SqlCommand  
Private _Update As SqlCommand  
Private _Delete As SqlCommand  
Private _Connection As SqlConnection
```



Configurar el DataAdapter

Para configurar el objeto SqlDataAdapter debemos, en primer lugar, instanciar el objeto SqlConnection, pasándole al constructor la cadena de conexión (Connection String), para que pueda conectarse a la base de datos.

Aquí se muestran dos maneras para pasar este parámetro.

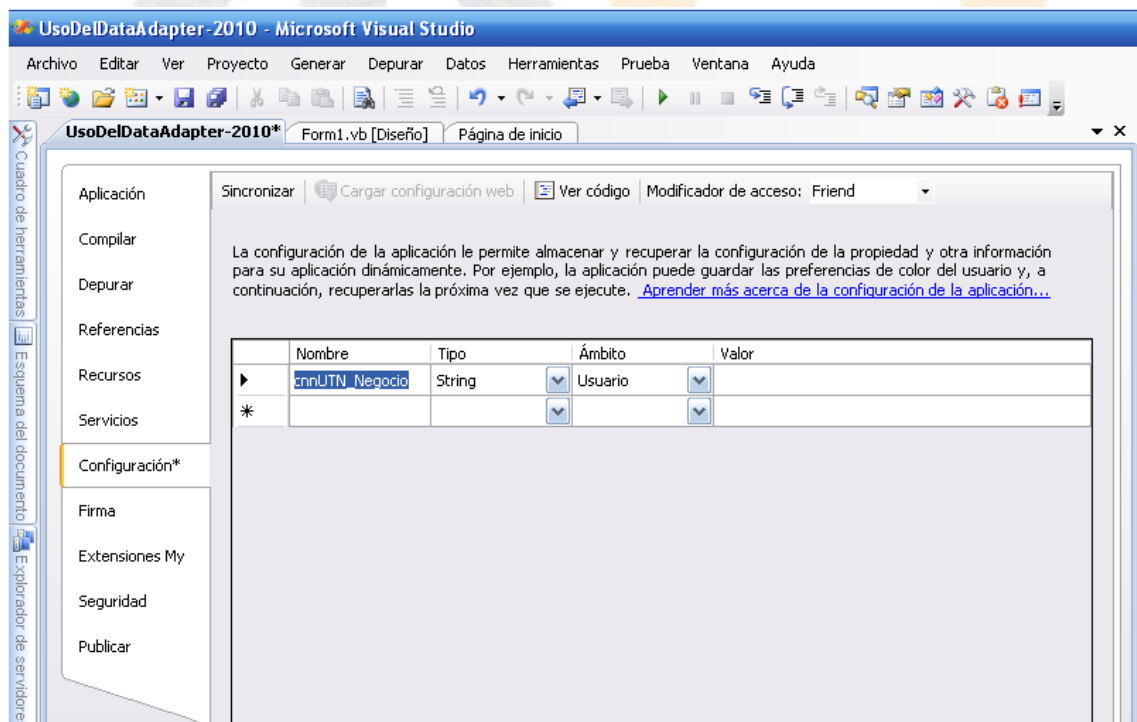
1. Podemos escribir la cadena de conexión en el parámetro del constructor.

CÓDIGO:

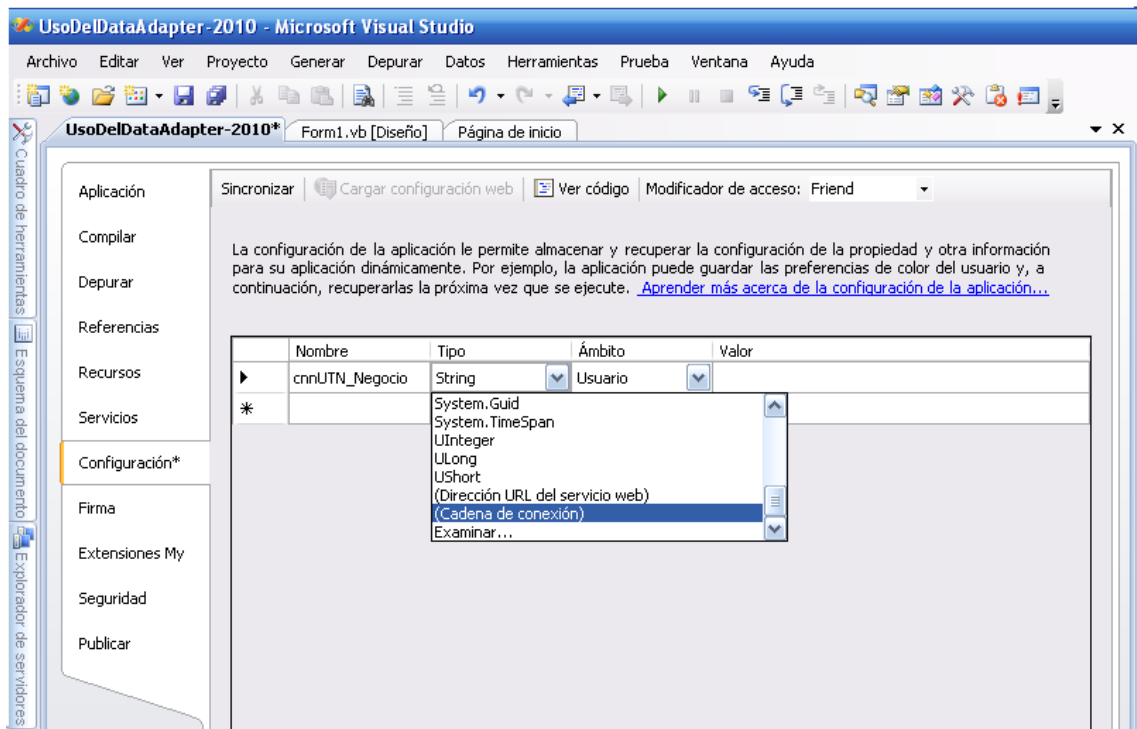
```
Dim MiStrConecion As String = "Data Source=VALEN-FEDE-06\SQLEXPRESS;Initial
Catalog=UTN_NEGOCIO;Integrated Security=True"
```

Nota: Cambiar el nombre del servidor de la cadena de conexión.

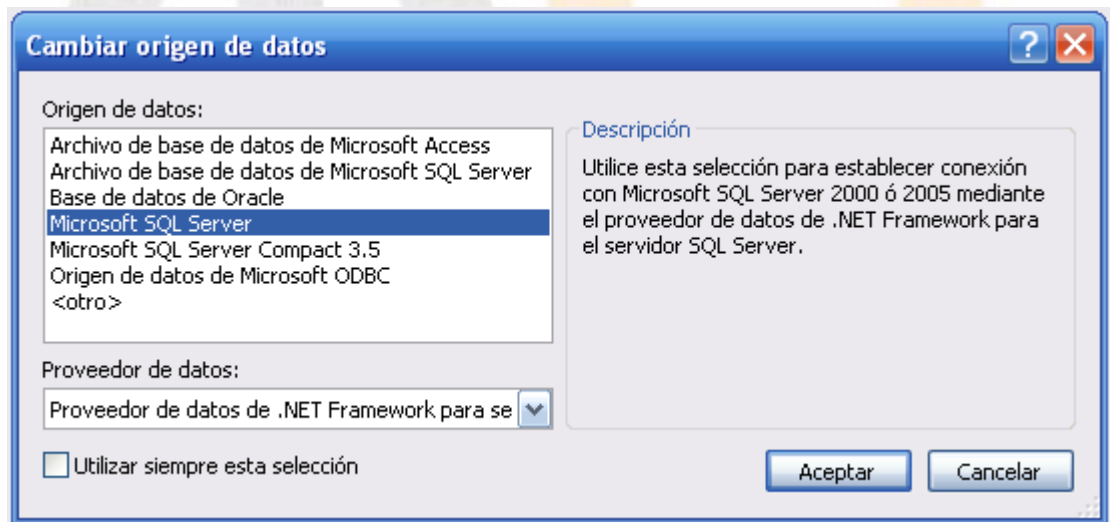
2. Podemos configurarlo en la solapa 'Configuración' de nuestro proyecto.



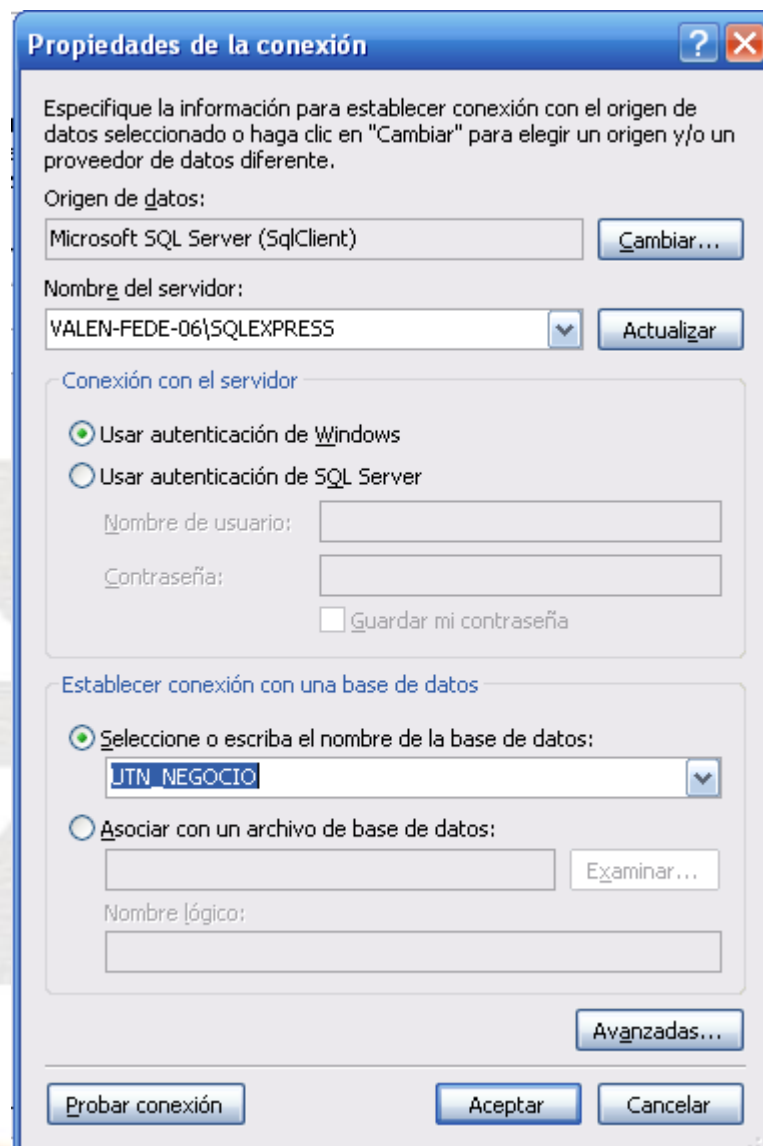
- a) Seleccionamos la opción cadena de conexión (Connection String) en la columna "Tipo"



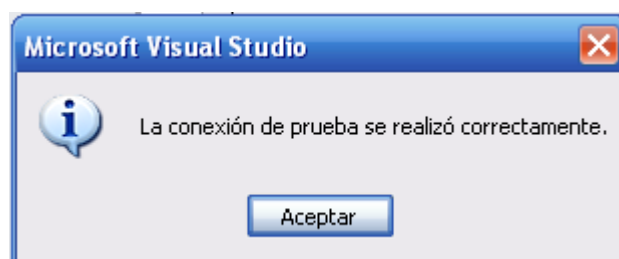
- b) En la columna 'Valor' seleccionamos el proveedor de datos.



c) Seleccionamos el servidor y la base de datos



d) Por último, probamos la conexión a la base de datos.



Ahora podemos utilizar la cadena de conexión de la siguiente manera:

CÓDIGO:

```
_Connection = New SqlConnection(My.Settings.cnnUTN_Negocio)
```

Una vez que la conexión está configurada, vamos a crear los distintos comandos con sus correspondientes sentencias SQL (**SELECT**, **INSERT**, **UPDATE** y **DELETE**).

Dentro de las sentencias utilizaremos parámetros por los cuales pasaremos los valores para cada instrucción. Estos parámetros los identificaremos anteponiendo el carácter '@'.

- `_Select = New SqlCommand("SELECT * FROM Productos", _Connection)`
- `_Insert = New SqlCommand("INSERT INTO Productos (Nombre,Tipo,Proveedor) VALUES (@Nombre, @Tipo, @Proveedor)", _Connection)`
- `_Update = New SqlCommand("UPDATE Productos SET Nombre = @Nombre, Tipo = @Tipo ,Proveedor = @Proveedor WHERE Id_Producto = @Id_Producto", _Connection)`
- `_Delete = New SqlCommand("DELETE FROM Productos WHERE Id_Producto = @Id_Producto", _Connection)`

Instanciamos el objeto SqlDataAdapter y le pasamos los comandos que va a utilizar para cada transacción SQL. Luego configuramos todos los parámetros a estos comandos.

CÓDIGO :

```
_dataAdapter = New SqlDataAdapter()  
  
With _dataAdapter  
    .SelectCommand = _Select  
    .InsertCommand = _Insert  
    .UpdateCommand = _Update  
    .DeleteCommand = _Delete  
  
    .InsertCommand.Parameters.Add("@Nombre", SqlDbType.VarChar, 50, "Nombre")  
    .InsertCommand.Parameters.Add("@Tipo", SqlDbType.VarChar, 50, "Tipo")  
    .InsertCommand.Parameters.Add("@Proveedor", SqlDbType.VarChar, 50, "Proveedor")  
  
    .UpdateCommand.Parameters.Add("@Id_Producto", SqlDbType.Int, 18, "Id_Producto")  
    .UpdateCommand.Parameters.Add("@Nombre", SqlDbType.VarChar, 50, "Nombre")  
    .UpdateCommand.Parameters.Add("@Tipo", SqlDbType.VarChar, 50, "Tipo")  
    .UpdateCommand.Parameters.Add("@Proveedor", SqlDbType.VarChar, 50, "Proveedor")  
  
    .DeleteCommand.Parameters.Add("@Id_Producto", SqlDbType.Int, 18, "Id_Producto")  
End With
```

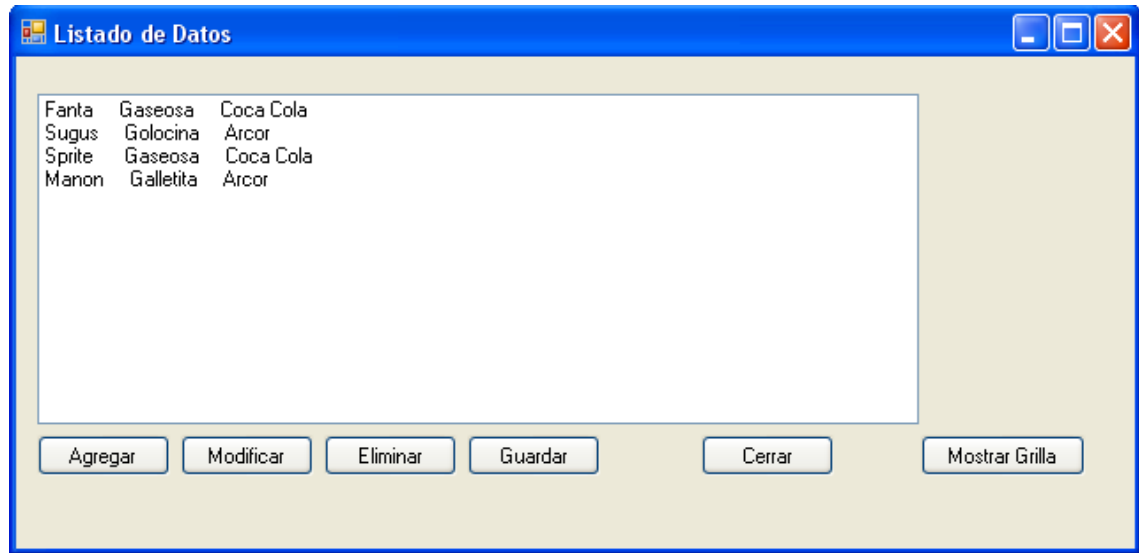
Cargar el DataSet

Para poder cargar el objeto DataSet, debemos utilizar el método 'Fill' del objeto SqlDataAdapter (que recibe al DataSet como parámetro) que carga con los registros del origen de datos (para este ejemplo la base de datos UTN_Negocio) al objeto DataSet.

CÓDIGO:

```
Private Sub TraerDatos()  
    Try  
        _dataSet = New DataSet()  
        _dataAdapter.Fill(_dataSet)  
    Catch ex As Exception  
        MessageBox.Show ("Se ha producido un error: " & ex.Message)  
    End Try  
End Sub
```


Mostrar los Datos



Los datos los mostraremos en un ListBox que llamaremos lstDatos.

Antes de mostrar los datos debemos preguntar por el estado de cada fila, ya que cuando borramos una fila, el DataSet no la borra físicamente, sino que la marca como 'Deleted' y realiza el borrado sobre la base de datos solo cuando llamamos al método 'Update' del objeto SqlDataAdapter.

El código para llegar al valor de una fila en un DataSet será:

CÓDIGO:

```
Private Sub MostrarDatos()
    Me.lstDatos.Items.Clear()

    For Each fila As DataRow In _dataSet.Tables(0).Rows
        'Pregunto si el estado de la fila es distinto de Deleted
        If fila.RowState <> DataRowState.Deleted Then
            'agrego los datos a mi lista
            Me.lstDatos.Items.Add(fila(1).ToString() _
                & "      " & fila(2).ToString() _
                & "      " & fila(3).ToString())
        End If
    Next

End Sub
```

Eliminar Datos

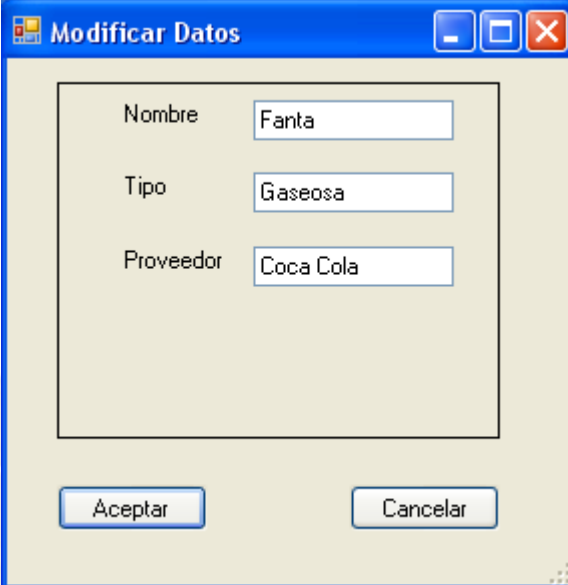
Para eliminar una fila de un DataSet, utilizamos el método Delete() del objeto DataSet, indicándole el número del índice de la fila que vamos a borrar, como en este ejemplo el dato que vamos a borrar fue seleccionado de la lista, utilizaremos dicho índice.

CÓDIGO:

```
Private Sub EliminarRegistro()  
    Try  
        _dataSet.Tables(0).Rows(Me.lstDatos.SelectedIndex).Delete()  
        'Actualizamos la lista  
        Call MostrarDatos()  
    Catch ex As Exception  
        MessageBox.Show("Seleccione un Producto de la lista")  
    End Try  
End Sub
```

Modificar Datos

Para modificar los datos utilizaremos un formulario de este tipo:



Para poder recibir y enviar datos a este formulario, lo haremos a través de propiedades que devuelven o asignan los datos de los TextBox del formulario.

Por ejemplo, para la propiedad 'Nombre' devolvemos el contenido de 'txtNombre.Text' y asignamos el valor a la propiedad 'Text' de este cuadro de texto.

Nota: Cambiar el valor de la propiedad Modifiers de Friend a Protected.

CÓDIGO:

```
Public Property Nombre() As String
    Get
        Return (txtNombre.Text)
    End Get
    Set(ByVal value As String)
        txtNombre.Text = value
    End Set
End Property
```

Una vez hecho esto para cada TextBox, debemos mostrar los datos a modificar en el formulario.

CÓDIGO:

```
Using frm As New FormDatos

    frm.Text = "Modificar Datos"

    frm.Nombre = _dataSet.Tables(0)
        .Rows(Me.lstDatos.SelectedIndex)("Nombre").ToString()

    frm.Tipo = _dataSet.Tables(0)
        .Rows(Me.lstDatos.SelectedIndex)("Tipo").ToString()

    frm.Proveedor = _dataSet.Tables(0)
        .Rows(Me.lstDatos.SelectedIndex)("Proveedor").ToString()

    'Mostramos el formulario
    frm.ShowDialog()
```

Dentro del formulario hay dos botones, que estarán configurados de la siguiente manera en el manejador de evento 'Click'.

Botón **btnAceptar**:

- `Me.DialogResult = Windows.Forms.DialogResult.OK`

Botón **btnCancelar**:

- `Me.DialogResult = Windows.Forms.DialogResult.Cancel`

Si el formulario devuelve un 'DialogResult.OK', entonces guardamos los datos que fueron modificados dentro del DataSet.

CÓDIGO:

```
If frm.DialogResult = Windows.Forms.DialogResult.OK Then

    Dim indice As Integer = Me.lstDatos.SelectedIndex

    _dataSet.Tables(0).Rows(indice)("Nombre") = frm.Nombre
    _dataSet.Tables(0).Rows(indice)("Tipo") = frm.Tipo
    _dataSet.Tables(0).Rows(indice)("Proveedor") = frm.Proveedor

    Call MostrarDatos()

End If
```

Agregar Datos

Para agregar datos debemos crear una nueva instancia de frmDatos, pero esta vez para agregar una nueva fila.

CÓDIGO:

```
Private Sub Agregar_Click(ByVal sender As System.Object, _  
                          ByVal e As System.EventArgs) Handles Agregar.Click  
  
    Using frm As New FormDatos  
        frm.Text = "Agregar"  
  
        If frm.ShowDialog() = Windows.Forms.DialogResult.OK Then  
            Dim fila As DataRow = Me._dataSet.Tables(0).NewRow()  
            fila(1) = frm.Nombre  
            fila(2) = frm.Tipo  
            fila(3) = frm.Proveedor  
  
            _dataSet.Tables(0).Rows.Add(fila)  
  
            Call MostrarDatos()  
        End If  
    End Using  
End Sub
```

Guardar los Datos en la Base

En el botón **btnGuardar** realizaremos la sincronización con el origen de datos (base de datos UTN_Negocio), para esto utilizaremos el método 'Update' del objeto SqlDataAdapter como lo muestran las siguientes instrucciones.

CÓDIGO:

```
Try
    _dataAdapter.Update(_dataSet)
    MessageBox.Show("Sincronización exitosa!!!")
Catch ex As Exception
    MessageBox.Show("Se ha producido un error al sincronizar.")
End Try
```

Utilización del DataReader

En el siguiente procedimiento utilizamos el objeto SqlDataReader para contar cuantos campos tiene nuestra tabla, también mostraremos sus valores.

CÓDIGO:

```
Private Sub ContarRegistros()  
  
    Dim contador As Integer  
  
    Using cn As New SqlConnection(My.Settings.cnnUTN_Negocio)  
        Dim cmd As New SqlCommand("SELECT * FROM Productos", cn)  
        cn.Open()  
  
        Using dr As SqlDataReader = cmd.ExecuteReader()  
            While dr.Read()  
                contador = contador + 1  
            End While  
        End Using  
    End Using  
  
    MessageBox.Show(String.Format("Registros: {0}", contador))  
  
End Sub
```

Nota: Realizar un procedimiento similar al anterior que utilice solamente un comando (SqlCommand) con una instrucción T-SQL.

Nota 2: Realizar un procedimiento que muestre la información de cada registro, utilizando un objeto de tipo SqlDataReader.

El estado de los campos del DataSet

En nuestro DataSet los estados de las filas van cambiando a medida que realizamos operaciones sobre ellos. Los distintos estados se pueden ver en la propiedad 'RowState'.

Aquí hay un procedimiento que nos muestra el estado de todas las filas de nuestro DataSet en un ListBox.

CÓDIGO:

```
Public Sub MostrarEstadoFilas(ByVal dataset As DataSet)

    Using frmEst As New FrmEstados
        Dim informacion As String
        frmEst.Text = "Estado de las Filas"

        For i As Integer = 0 To dataset.Tables(0).Rows.Count - 1
            With dataset
                informacion = "Fila #" & i & " "

                informacion = informacion + _
                    .Tables(0).Rows(i).RowState.ToString

                frmEst.ListaEstados.Items.Add(informacion)
            End With
        Next

        'Muestro el Formulario
        frmEst.ShowDialog()
    End Using
End Sub
```

