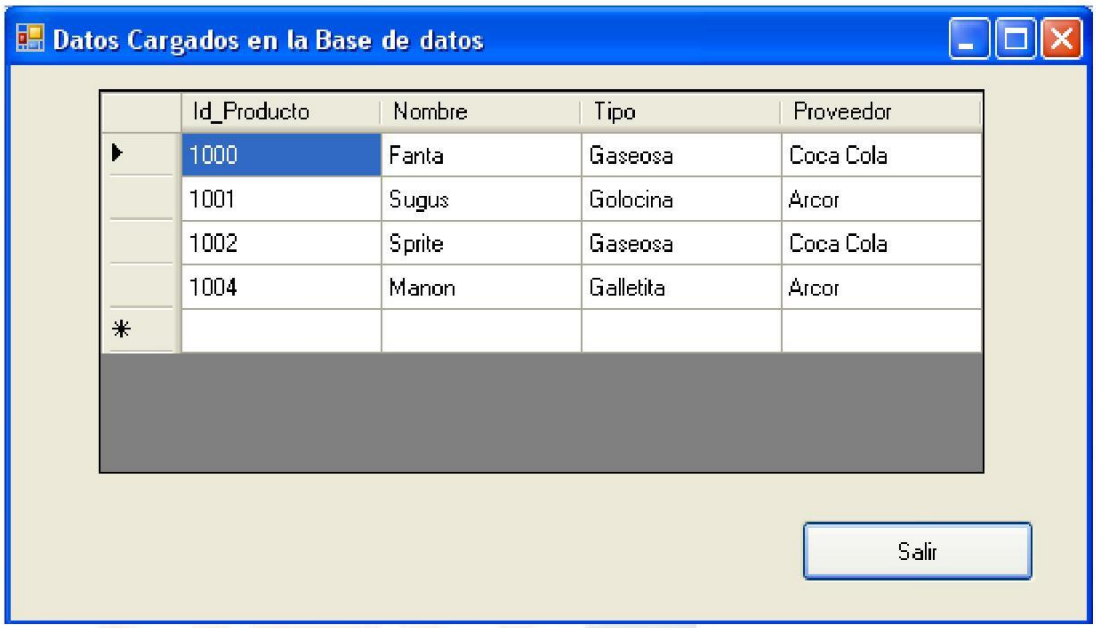
Introducción

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 _dataSet;

    private SqlDataAdapter _dataAdapter;

    private SqlCommand _Select;

    private SqlCommand _Insert;

    private SqlCommand _Update;

    private SqlCommand _Delete;

    private SqlConnection _Connection;
```

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.

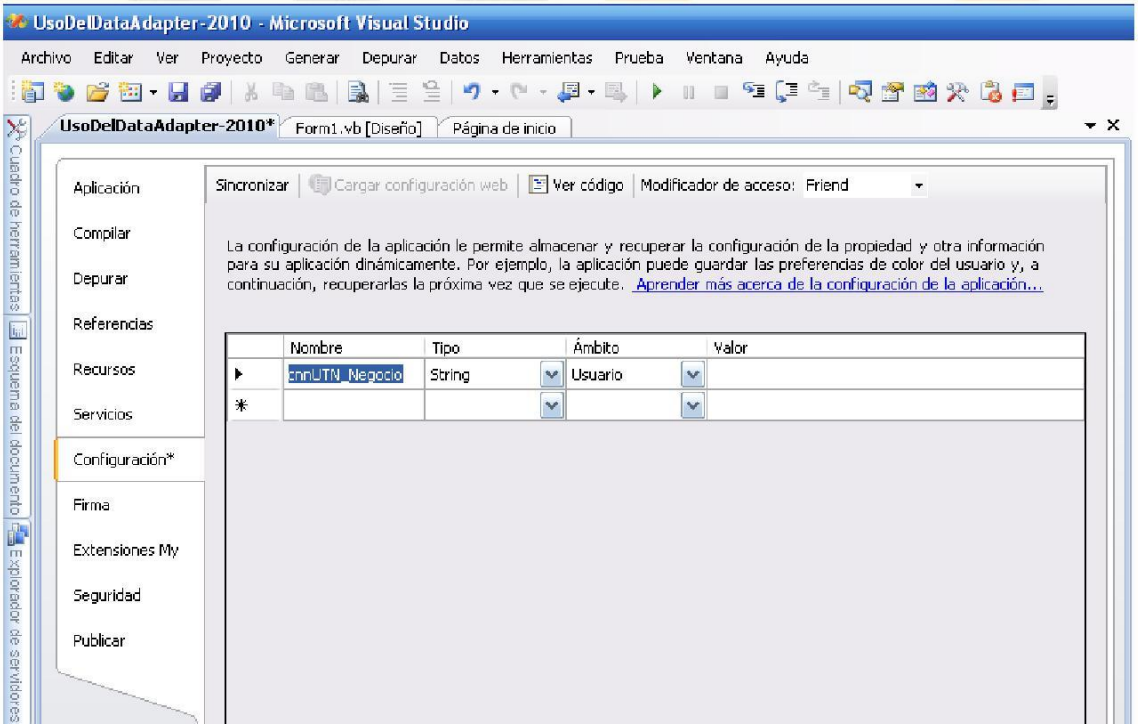
1. Podemos escribir la cadena de conexión en el parámetro del constructor.

CÓDIGO:

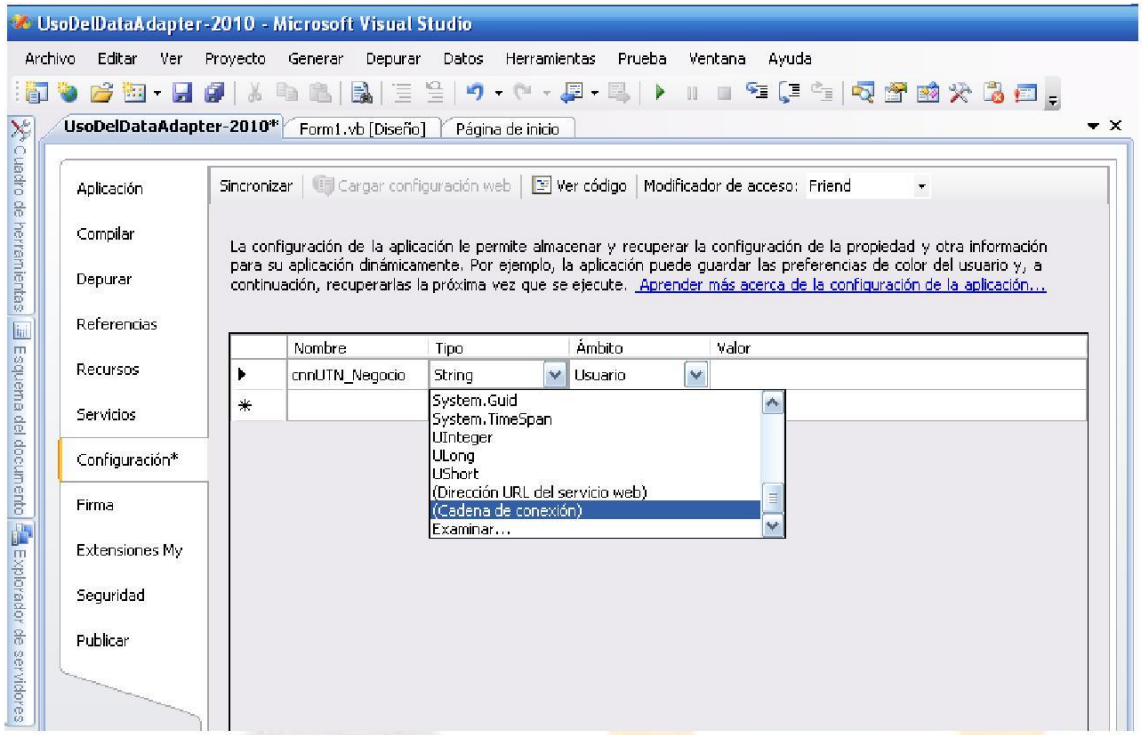
```
String MiStrConexion = "Data Source=Homero\SQLEXPRESS;Initial

Catalog=UTN_NEGOCIO;Integrated Security=True";
```

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

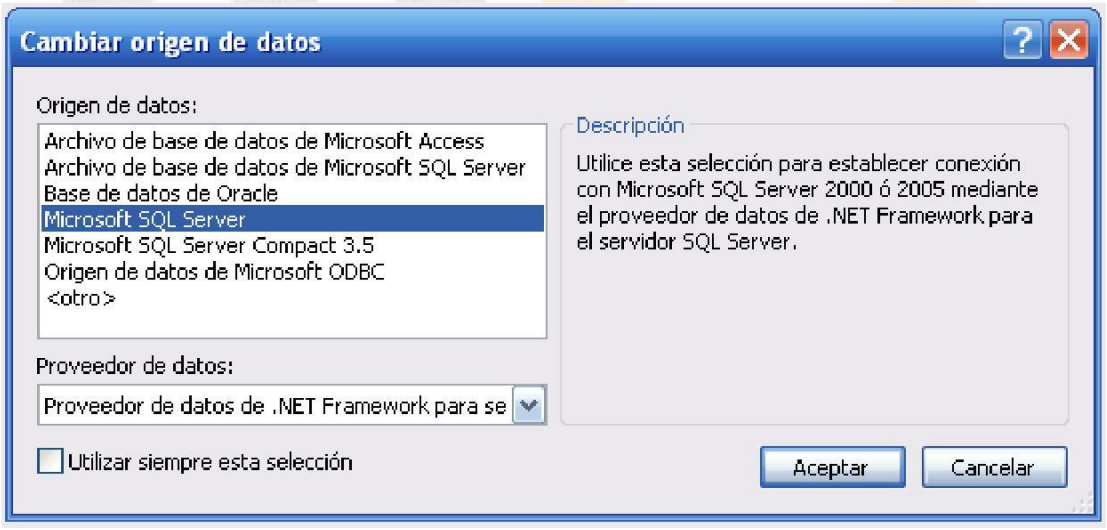
2. Podemos configurarlo en la solapa „Configuración" de nuestro proyecto.



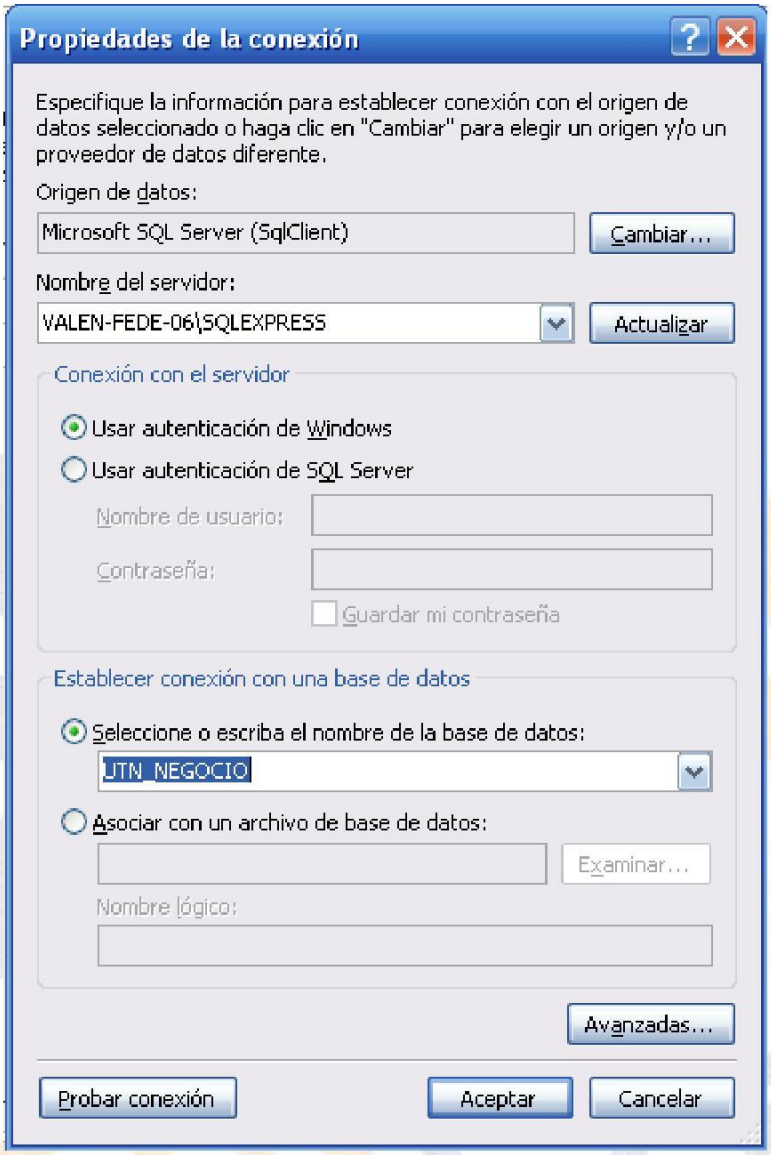
a) Seleccionamos la opción cadena de conexión (Connection String) en la columna "Tipo"



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:

```
This._Connection = new SqlConnection(Properties.Settings.Default.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 „@".

- ```
  this_Select = new SqlCommand("SELECT * FROM Productos", this._Connection);
  ```
- ```
  this._Insert = new SqlCommand("INSERT INTO Productos
  (Nombre,Tipo,Proveedor) VALUES (@Nombre, @Tipo, @Proveedor)",
  this._Connection);
  ```
- ```
  this._Update = new SqlCommand("UPDATE Productos SET Nombre = @Nombre,
  Tipo = @Tipo ,Proveedor = @Proveedor WHERE Id_Producto =
  @Id_Producto", this._Connection);
  ```
- ```
  this._Delete = new SqlCommand("DELETE FROM Productos WHERE
  Id_Producto = @Id_Producto", this._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 :

```
 this._dataAdapter = new SqlDataAdapter();



 this._dataAdapter.SelectCommand = this._Select;
 this._dataAdapter.InsertCommand = this._Insert;
 this._dataAdapter.UpdateCommand = this._Update;
 this._dataAdapter.DeleteCommand = this._Delete;

this._dataAdapter.InsertCommand.Parameters.Add("@Nombre", SqlDbType.VarChar, 50,
"Nombre");
this._dataAdapter.InsertCommand.Parameters.Add("@Tipo", SqlDbType.VarChar, 50,
"Tipo");
this._dataAdapter.InsertCommand.Parameters.Add("@Proveedor", SqlDbType.VarChar,
50, "Proveedor");


this._dataAdapter.UpdateCommand.Parameters.Add("@Id_Producto", SqlDbType.Int, 18,
"Id_Producto");
this._dataAdapter.UpdateCommand.Parameters.Add("@Nombre", SqlDbType.VarChar, 50,
"Nombre");
this._dataAdapter.UpdateCommand.Parameters.Add("@Tipo", SqlDbType.VarChar, 50,
"Tipo");
this._dataAdapter.UpdateCommand.Parameters.Add("@Proveedor", SqlDbType.VarChar,
50, "Proveedor");



this._dataAdapter.DeleteCommand.Parameters.Add("@Id_Producto", SqlDbType.Int, 18,
"Id_Producto");
```

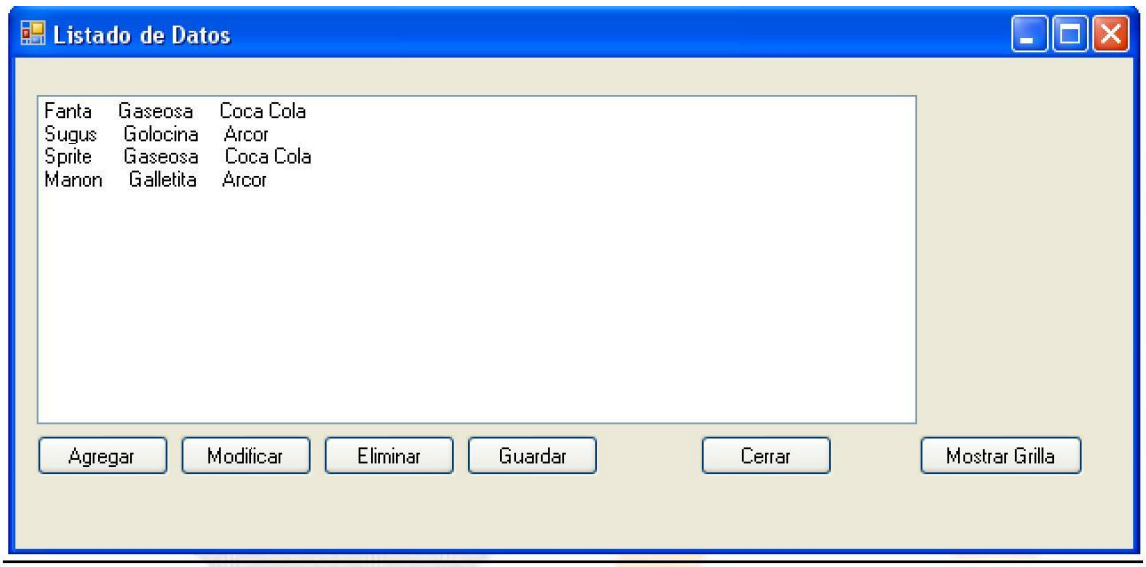
Cargar el DataSet

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 void TraerDatos()
        {
            try
            {
                this._dataSet = new DataSet();

                this._dataAdapter.Fill(this._dataSet);

            }
            catch (Exception ex)
            {
                MessageBox.Show("Se ha producido un error", ex.Message);
            }
        }
```

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 void MostrarDatos()
        {
            this.lstDatos.Items.Clear();

            foreach (DataRow fila in this._dataSet.Tables[0].Rows)
            {
                if (fila.RowState != DataRowState.Deleted)
                {
                    this.lstDatos.Items.Add("Nombre: " + fila[1].ToString() + " Tipo: " +
fila[2].ToString() + " Proveedor: " + fila[3].ToString());
                }
            }
        }
```

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 void EliminarDatos()
        {
            try
            {
                this._dataSet.Tables[0].Rows[this.lstDatos.SelectedIndex].Delete();

                this.MostrarDatos();

            }
            catch (Exception ex)
            {

               MessageBox.Show("Se ha producido un error", ex.Message);
            }
        }
```

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.

CÓDIGO:

```
  public String Nombre
        {
            get { return this.txtNombre.Text; }
            set { this.txtNombre.Text = value; }

}
```

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

CÓDIGO:

```
  FormDatos frm = new FormDatos();
            frm.Text = "Modificar Datos";

            frm.Nombre =
this._dataSet.Tables[0].Rows[this.lstDatos.SelectedIndex]["Nombre"].ToString();
            frm.Tipo =
this._dataSet.Tables[0].Rows[this.lstDatos.SelectedIndex]["Tipo"].ToString();
            frm.Proveedor =
this._dataSet.Tables[0].Rows[this.lstDatos.SelectedIndex]["Proveedor"].ToString();
```

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

Botón ***btnAceptar***:

- `this.DialogResult = Windows.Forms.DialogResult.OK;`

Botón ***btnCancelar***:

- `this.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.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                int indice = this.lstDatos.SelectedIndex;

                this._dataSet.Tables[0].Rows[indice]["Nombre"] = frm.Nombre;
                this._dataSet.Tables[0].Rows[indice]["Tipo"] = frm.Tipo;
                this._dataSet.Tables[0].Rows[indice]["Proveedor"] = frm.Proveedor;

                this.MostrarDatos();
            }
```

Agregar Datos

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

CÓDIGO:

```
private void btnAgregar_Click(object sender, EventArgs e)
        {
            FormDatos frm = new FormDatos();

            frm.Text = "Agregar";

            if (frm.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                DataRow fila = this._dataSet.Tables[0].NewRow();

                fila[1] = frm.Nombre;
                fila[2] = frm.Tipo;
                fila[3] = frm.Proveedor;

                this._dataSet.Tables[0].Rows.Add(fila);

                this.MostrarDatos();
            }
        }
```

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
            {
                this._dataAdapter.Update(this._dataSet);

                MessageBox.Show("Sincronización exitosa!!!");

            }
            catch (Exception ex)
            {
                MessageBox.Show("Se ha producirdo un error al sincronizar");
            }
```

Utilización del DataReader

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

CÓDIGO:

```
private void btnContar_Click(object sender, EventArgs e)
        {
            int contador = 0;

            SqlCommand comando = new SqlCommand("Select * from Productos", this._conexion);

            try
            {
                this._conexion.Open();

                SqlDataReader lector = comando.ExecuteReader();

                while (lector.Read())
                {
                    contador = contador + 1;
                }

            }
            catch (Exception)
            {

                throw;
            }
            finally
            {
                this._conexion.Close();
            }
}
```

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 void MostrarEstadoFilas(DataSet dataset)
        {
            frmEstados frmEst = new frmEstados();
            frmEst.Text = "Estados de las Filas";

            int contador = 0;

            foreach (DataRow fila in dataset.Tables[0].Rows)
            {
                frmEst.lstEstados.Items.Add("Fila: " + contador + " Estado: " +
fila.RowState.ToString());
                contador = contador + 1;
            }

            frmEst.ShowDialog();



    }
```