Gestor BD

Creación del proyecto

1. Abrir Visual como administrador.
2. Seleccionar un proyecto nuevo **Biblioteca de clases (Class Library)**, darle nombre: (p. ej.: *GestorBD*), eliminar *clase1.cs*.
3. Agregación de las clases del componente, por cada clase: menú breve del *Proyecto*, Agregar, Componente…, seleccionar plantilla Clase de componentes (Component Class), darle nombre (p. ej.: *GestorBD*, sin alterar el .cs), <Agregar>, en la ventana que se abre dar clic en: haga clic aquí para cambiar a la vista Código.
4. Declaración de propiedades, métodos y eventos de cada clase del componente. Se declaran como en cualquier clase de Visual C#.

```
//Agregar al inicio del archivo:
using System.Data;
using System.Data.OleDb;

      /* Esta clase contiene rutinas para consultar, dar de alta, dar de baja y
cambiar la información de una BD de Oracle. Con esta clase se crea una biblioteca
.dll para que pueda ser usada en cualquier aplicación de Visual C#.
       Nota:
        Para poder conectarse a una BD de otro DBMS, basta con cambiar los parámetros
en el constructor del objeto de conexión. Hay que darlos según el DBMS en cuestión.
*/

    //=====================================================================
    //Definición de los objetos de ADO.Net para la manipulación de la BD.
    OleDbDataAdapter odaCons = new OleDbDataAdapter();
    OleDbDataAdapter odaAct = new OleDbDataAdapter();
    private const int OK = 1;
    //Puede definirse como Property.
    public OleDbConnection conex = new OleDbConnection();
```

Constructor

```
    //Constructor: establece la conexión a una BD Oracle o Access.
    public GestorBD(String prov, String user, String pass, String ds)
    {
        conex = new OleDbConnection("Provider=" + prov + ";" +
          "User ID=" + user + ";Password=" + pass + ";Data Source=" + ds + ";");
    }

    //Constructor: establece la conexión a una BD SQL Server.
    public GestorBD(String prov, String server, String user, String pass, String db)
    {
        conex = new OleDbConnection("Provider=" + prov + ";Data Source=" + server +
          ";User ID=" + user + ";Password=" + pass + ";Initial catalog=" + db +";");
    }
```

Consulta

```
/* ====================================================================
   Ejecuta una consulta sobre la BD. La consulta se da como parámetro,
   Parámetros de entrada: la consulta y el nombre de la tabla resultante.
   Parámetro de salida: el DataSet generado.
 */
public void consBD(String cadSql, System.Data.DataSet dsCons, String tabla)
{
    try
    {
        odaCons.SelectCommand =
          new OleDbCommand(cadSql, conex); //Define la consulta.

        if (!dsCons.Tables.Contains(tabla))
        {      //Establece el
            dsCons.Tables.Add(tabla);   //nombre de la tabla resultante.
            odaCons.FillSchema(dsCons, SchemaType.Source, tabla);
        }
        dsCons.Clear();                //Borra resultados anteriores.
        odaCons.Fill(dsCons, tabla);   //Ejecuta la consulta.
    }
    catch (OleDbException err)
    {
        Console.WriteLine(err.Message);
    }
}
```

Alta

```
//====================================================================
//Efectúa una inserción de datos en una tabla de la BD.
//La instrucción de inserción se da como parámetro.
//La rutina regresa OK o un código, si hubo error.
public int altaBD(String cadSql)
{
    int cant, result;

    try
    {
        odaAct.InsertCommand =
          new OleDbCommand(cadSql, conex);   //Define la inserción.

        conex.Open();
        cant = odaAct.InsertCommand.ExecuteNonQuery();   //La ejecuta.
        result = OK;
    }
    catch (OleDbException err)
    {
        Console.WriteLine(err.Message);
        result = err.ErrorCode;
    }
    conex.Close();
    return result;
}
```

Baja

```
//====================================================================
```

```
//Efectúa un borrado de datos en una tabla de la BD.
//La instrucción de borrado se da como parámetro.
//La rutina regresa OK o un código, si hubo error.
public int bajaBD(String cadSql)
{
    int cant, result;

    try
    {
        odaAct.DeleteCommand =
          new OleDbCommand(cadSql, conex);   //Define la inserción.

        conex.Open();
        cant = odaAct.DeleteCommand.ExecuteNonQuery();   //La ejecuta.
        result = OK;
    }
    catch (OleDbException err)
    {
        Console.WriteLine(err.Message);
        result = err.ErrorCode;
    }
    conex.Close();
    return result;
}
```

Modificación

```
//=====================================================================
//Efectúa un cambio de datos en una tabla de la BD.
//La instrucción de cambio se da como parámetro.
//La rutina regresa OK o un código, si hubo error.
public int cambiaBD(String cadSql)
{
    int cant, result;

    try
    {
        odaAct.UpdateCommand =
          new OleDbCommand(cadSql, conex);   //Define la inserción.

        conex.Open();
        cant = odaAct.UpdateCommand.ExecuteNonQuery();   //La ejecuta.
        result = OK;
    }
    catch (OleDbException err)
    {
        Console.WriteLine(err.Message);
        result = err.ErrorCode;
    }
    conex.Close();
    return result;
}
```

5. Compilación del componente: una vez que se han construido las clases, hay que compilarlos: menú Compilar, Generar *componente*. Guardar el proyecto (sin crear nuevo directorio). Esto genera el código correspondiente

|

en un archivo *componente.dll* en el directorio bin/ Debug (o bin/ Release) del proyecto.

Aplicación de Cliente-Servidor

Conexión a la base de datos

Nota: Antes de iniciar la conexión y si no se hizo cuando se empezó a utilizar el SQL Developer: ir al servidor y copiar desde Docs\InstalConfig a los archivos: sqlnet.ora y tnsnames.ora, después pegarlos en la carpeta C:Oracle\product\12.1.0\ client_1\network\admin.

Crear proyecto> Archivo>Nuevo>Proyecto>Visual C#>Aplicación de Windows Forms

1. Seleccionar menú Ver y Explorador de servidores, para visualizar a este último.
2. Dar clic izquierdo, menú breve de Conexiones de datos, seleccionar Agrega conexión…
3. Clic a <Cambiar…> y elegir el tipo del Origen de datos. Para Access: *Archivo de bases de datos* **Microsoft Access**; **para Oracle**: *Base de datos de Oracle y Proveedor de datos de OLE DB*; **para SQL Server**: *Microsoft SQL Server*.
4. Seleccionar el Archivo de base de datos (para Access), el Nombre del servidor (para Oracle: oracle u oraculo.itam.mx) o el Nombre del servidor, la autenticación de SQL Server y la base de datos (para SQL Server).
5. Decidir si se quiere guardar la contraseña en la cadena de conexión (en el caso de Oracle, **sí hay que** guardar la contraseña en la cadena de conexión).
6. <Probar conexión>, <Aceptar>.
7. Aquí ya se pueden ver los elementos que tiene la base de datos.

Nota: en el caso de Access, antes de hacer la conexión se tiene que decidir si la base de datos va estar en el proyecto (en \bin\Debug) o fuera de él. Después, hacer la conexión.

|

Agregar GestorBD

Se debe agregar la referencia al ***.dll***:

1. Seleccionar el proyecto de prueba
2. Menú breve
3. Agregar referencia…
4. Pestaña Examinar
5. <Examinar…>
6. Seleccionar el archivo .dll (que debe estar en el directorio bin/Debug (o bin/ Release) de la solución del componente)
7. <Aceptar>.

Clase Comunes

```
class Comunes
    {
        //Aquí se colocan los métodos (y variables) de uso común en todo el
proyecto.
        /* Agrega en un ComboBox (primer parámetro), los datos que están en un
DataSet(segundo parámetro), en la tabla y columna dadas (tercer y cuarto
parámetros). Los datos se agregan sin repetición. */
        public void cargaCombo(ComboBox cbo, DataSet ds, String tabla, String col)
        {
            DataTable tabResul;

            cbo.Items.Clear();
            //La columna 'col' es la que tiene los datos que se agregarán al combo.
            tabResul = ds.Tables[tabla];
            foreach (DataRow fila in tabResul.Rows)
                if (!cbo.Items.Contains(fila[col].ToString().Trim()))
                    cbo.Items.Add(fila[col].ToString().Trim());
        }
    }
```

Cargar un comboBox con GestorBD y Comunes

```
// Carga el combo de Productos.
GestorBD.GestorBD GestorBD = new GestorBD.GestorBD("MSDAORA", "bd08", "padbel",
"oracle");
cadSql = "select Nombre from T4Producto";
DataSet dsArtículo = new DataSet();
GestorBD.consBD(cadSql, dsArtículo, "Productos");
comunes.cargaCombo(cbProductos, dsArtículo, "Productos", "Nombre");
```

Cargar un comboBox manualmente

```
//Carga ComboBox de Pago
cbTipoPago.Items.Add("Contado");
cbTipoPago.Items.Add("Crédito");
```

Salir de la aplicación

```
private void adiosPopóToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
```

```
        }
```

Mandar llamar a otra forma

```
private void procesosToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Procesos.Procesos procesos = new Procesos.Procesos();
            procesos.Show();
        }
```

Fecha en formato Oracle de un dateTimePicker

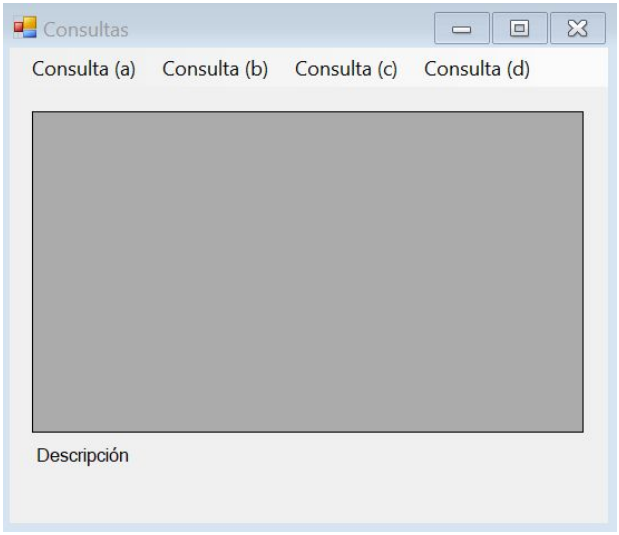
```
string fecha, sdia, smes;
int dia, mes;
//Construye la fecha para el formato de Oracle.
dia = dateTimePicker1.Value.Day;
if (dia < 10) {
       sdia = "0" + dia;
} else {
       sdia = "" + dia;
}
mes = dateTimePicker1.Value.Month;
if (mes < 10) {
       smes = "0" + mes;
} else {
       smes = "" + mes;
}
fecha = sdia + "/" + smes + "/" + (dateTimePicker1.Value.Year-2000);

//Construye la fecha para el formato de Oracle.
      fecha = "date'" + dtpFecha.Value.Year + "-" + dtpFecha.Value.Month + "-" +
        dtpFecha.Value.Day + "'";
```

Llenar DataGridView

```
DataSet dsCompras = new DataSet();
            GestorBD.GestorBD GestorBD = new GestorBD.GestorBD("MSDAORA", "bd08",
"padbel", "oracle");
            cadSql = " (query) ";
            GestorBD.consBD(cadSql, dsCompras, "Nombre");
            dtConsulta.DataSource = dsCompras.Tables["Nombre"];
```

|

Consultar Vistas



1. Crear en oracle una vista de la consulta deseada:

Consulta a: dado el nombre de una cadena, obtener el nombre de los clientes cuyo saldo global es menor al 10% del monto total de los artículos que compraron.

```
create view (Nombre de la vista) as select cad.nombre as Cadena_Comercial,cl.nombre as Cliente from T4Cliente cl,T4Registrado r,T4Cadena cad where cl.IdCliente=r.IdCliente and r.RFC=cad.RFC and cl.idCliente in (select idCliente from T4Compra c,T4Factura f,T4Sucursal s where c.Folio=f.folio and c.Folio=f.folio and f.IdSucursal=s.IdSucursal group by idCliente having sum(c.montoTotal)>(sum(f.saldoActual)*10));
```

2. En el DataGridView, seleccionar la flechita y abrir el comboBox de “Seleccionar origen de datos”.
3. Elegir “Agregar origen de datos del proyecto…”. Seguir los pasos indicados, y para la conexión seleccionar la existente.
4. Seleccionar “Vistas” y buscar la correcta. Ponerle un nombre al dataSet.

   Nota: Si se elegirán varias vistas para mostrar en un mismo DGV, seleccionar la opción de “Ninguno” en “Seleccionar origen de datos” al terminar de agregar todas las vistas con su respectivo DataSet. En el pageLoad automáticamente se creará el siguiente código:

```
private void Consultas_Load(object sender, EventArgs e)
        {
            // TODO: esta línea de código carga datos en la tabla 'dsSucursalMontoTotal.SUCURSALMONTOTOTAL' Puede moverla o quitarla según sea necesario.
```

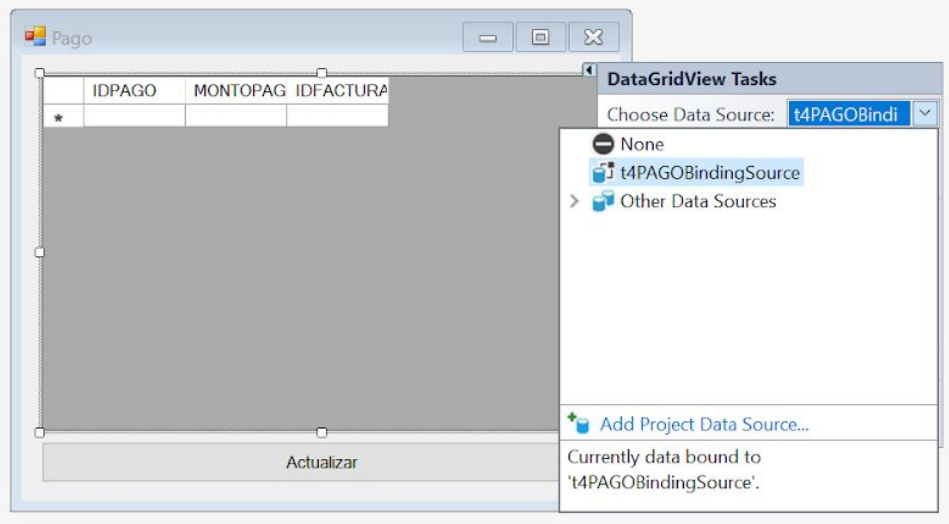
```
this.sUCURSALMONTOTOTALTableAdapter.Fill(this.dsSucursalMontoTotal.SUCURSALMONTOTOTA
L);
            // TODO: esta línea de código carga datos en la tabla
'dsSucursalesCadena.SUCURSALESCADENA' Puede moverla o quitarla según sea necesario.

this.sUCURSALESCADENATableAdapter.Fill(this.dsSucursalesCadena.SUCURSALESCADENA);
            // TODO: esta línea de código carga datos en la tabla
'dsClientesSinCompras.CLIENTESSINCOMPRAS' Puede moverla o quitarla según sea
necesario.

this.cLIENTESSINCOMPRASTableAdapter.Fill(this.dsClientesSinCompras.CLIENTESSINCOMPRA
S);
            // TODO: esta línea de código carga datos en la tabla
'dsClientesSaldo10.CLIENTESSALDO10' Puede moverla o quitarla según sea necesario.

this.cLIENTESSALDO10TableAdapter.Fill(this.dsClientesSaldo10.CLIENTESSALDO10);
        }
```

5. Añadir el código correspondiente para mostrar la consulta:

```
        private void consultaaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            descripción.Text = "Nombre de los clientes cuyo saldo global es menor al
10% del monto \ntotal de los artículos que compraron.";
            descripción.Visible = true;
            dataGridView1.DataSource = this.cLIENTESSALDO10BindingSource;
        }

        private void consultabToolStripMenuItem_Click(object sender, EventArgs e)
        {
            descripción.Text = "Nombre de los clientes que no han hecho compras este
mes, así \ncomo el nombre de las cadenas donde tienen crédito.";
            descripción.Visible = true;
            dataGridView1.DataSource = this.cLIENTESSINCOMPRASBindingSource;
        }

        private void consultacToolStripMenuItem_Click(object sender, EventArgs e)
        {
            descripción.Text = "Nombre de las sucursales, por cadena, y el monto
total de ventas \n(menor a $50,000) que realizaron el mes pasado.";
            descripción.Visible = true;
            dataGridView1.DataSource = this.sUCURSALMONTOTOTALBindingSource;
        }

        private void consultadToolStripMenuItem_Click(object sender, EventArgs e)
        {
            descripción.Text = "Nombre de la cadena que tiene más sucursales, así
como el nombre \ny domicilio de éstas.";
            descripción.Visible = true;
            dataGridView1.DataSource = this.sUCURSALESCADENABindingSource;
        }
```

|

Modificar tablas



1. Seguir los mismos pasos que al consultar una vista, pero en lugar de seleccionar “Vistas”, seleccionar “Tablas” y elegir la deseada. Además, en “Seleccionar origen de datos”, elegir el origen correspondiente a la tabla. Se creará su código correspondiente en el pageLoad:

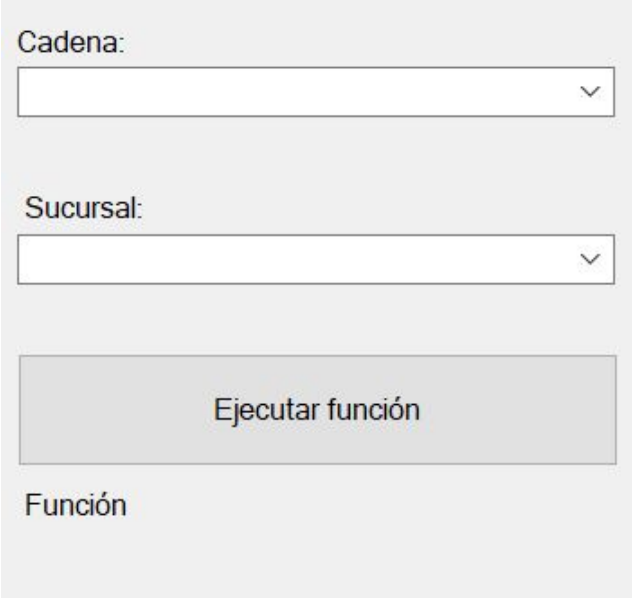
```
private void Pago_Load(object sender, EventArgs e)
        {
            // TODO: esta línea de código carga datos en la tabla 'dsPago.T4PAGO'
Puede moverla o quitarla según sea necesario.
            this.t4PAGOTableAdapter.Fill(this.dsPago.T4PAGO);
        }
```

2. Para poder habilitar la habilitación de la tabla, se debe agregar la siguiente línea de código en el botón correspondiente:

```
this.t4PAGOTableAdapter.Update(this.dsPago.T4PAGO);
```

Ejecutar una función almacenada

```
create or replace function tiposDeArticulos(cadena
char,sucursal char) return integer is
cantArt integer;
begin
    select count(v.IdProd) into cantArt
    from T4Vende v,T4Sucursal s,T4Cadena cad
    where cad.Nombre=cadena and s.Nombre=sucursal and
v.IdSucursal=s.IdSucursal;
    return cantArt;
end;
```

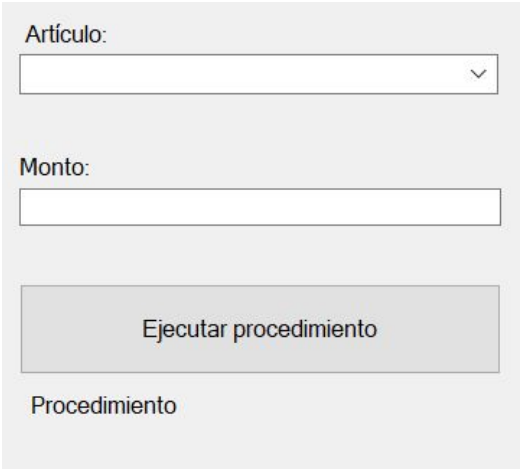



```
private void btnFunción_Click(object sender, EventArgs e) {
            try {
                //1- Abrir la conexión a la BD.
                OleDbConnection cnOracle = new OleDbConnection("Provider=MSDAORA;
Data Source=oracle;User ID=bd08;Password=padbel");
                cnOracle.Open();
                OleDbCommand función = new OleDbCommand();
                función.Connection = cnOracle;

                //2- Especificar el llamado a la función.
                función.CommandText = "tiposDeArticulos";
                función.CommandType = CommandType.StoredProcedure;

                //3- Especificar los parámetros:
                    //a) Valor que regresará la función.
                OleDbParameter parámetro = new OleDbParameter("RETURN_VALUE",
OleDbType.Integer,4, ParameterDirection.ReturnValue, false, 4, 0, "Salida",
DataRowVersion.Current, 0);
                función.Parameters.Add(parámetro);

                    //b) Parámetros de entrada.
                String cadena = cbCadenas.Text;
                parámetro = new OleDbParameter("cadena", cadena);
                función.Parameters.Add(parámetro);
                String sucursal = cbSucursales.Text;
                parámetro = new OleDbParameter("sucursal", sucursal);
                función.Parameters.Add(parámetro);

                //4- Ejecutar el procedimiento (en general: el subprograma).
                try {
                    función.ExecuteNonQuery();
                    //5- Recuperar el valor regresado por la función.
                    int cantArt =
Convert.ToInt16(función.Parameters["RETURN_VALUE"].Value);
                    MessageBox.Show(cadena+" "+sucursal+" vende "+cantArt+"
artículos distintos.");
                }
```

```
            catch (OleDbException err) {
                MessageBox.Show(err.Message);
            }

            //6- Cerrar la conexión a la BD.
            cnOracle.Close();
        } catch {
            MessageBox.Show("Debe llenar todos los campos.");
        }
    }
```

Ejecutar un procedimiento almacenado

```
create or replace procedure unaGangaT5(producto char,monto
float,n out real, cc out char) is
begin
    select count(distinct o.RFC) into n
    from T4Ofrece o,T4Producto p
    where o.IdProd=p.IdProd
    and o.PrecioU<monto and p.Nombre=producto;
    select c.Nombre into cc
    from T4Cadena c,T4Ofrece o,T4Producto p
     where c.RFC=o.RFC and o.IdProd=p.IdProd and o.PrecioU<monto
     and p.Nombre=producto and rownum=1;
end;
```




```
private void btnProcedimiento_Click(object sender, EventArgs e) {
        try {
            //1- Abrir la conexión a la BD.
            OleDbConnection cnOracle = new OleDbConnection("Provider=MSDAORA;
Data Source=oracle;" +
              "User ID=bd08;Password=padbel");
            cnOracle.Open();
            OleDbCommand procedimiento = new OleDbCommand();
            procedimiento.Connection = cnOracle;

            //2- Especificar el llamado al procedimiento  (en general: al
subprograma).
            procedimiento.CommandText = "unaGangaT5";
            procedimiento.CommandType = CommandType.StoredProcedure;
```

```
                //3- Especificar los parámetros:
                //a) Los de entrada:
                String artículo = cbProductos.Text;
                OleDbParameter parámetro = new OleDbParameter("producto", artículo);
                procedimiento.Parameters.Add(parámetro);
                Double monto = Double.Parse(tbMonto.Text);
                parámetro = new OleDbParameter("monto", monto);
                procedimiento.Parameters.Add(parámetro);

                //b) Los de salida:
                parámetro = new OleDbParameter("n", OleDbType.Double, 8,
                  ParameterDirection.Output, false, 12, 0, "Salida1",
DataRowVersion.Current, 0);
                procedimiento.Parameters.Add(parámetro);
                parámetro = new OleDbParameter("cc", OleDbType.Char, 8,
                  ParameterDirection.Output, false, 12, 0, "Salida2",
DataRowVersion.Current, 0);
                procedimiento.Parameters.Add(parámetro);

                //4- Ejecutar el procedimiento (en general: el subprograma).
                try {
                    procedimiento.ExecuteNonQuery();
                    //5- Recuperar el (los) valor(es) regresado(s) por medio del (de
los) parámetro(s) de salida.
                    Double cantCadenas =
Convert.ToDouble(procedimiento.Parameters["n"].Value);
                    String cadena =
Convert.ToString(procedimiento.Parameters["cc"].Value);
                    if (cantCadenas > 0) {
                        MessageBox.Show(cantCadenas + " cadenas comerciales ofrecen
" + artículo + " a un precio menor a $" + monto + ".\n"
                            + "Una de ellas es " + cadena + ".");
                    } else {
                        MessageBox.Show(cantCadenas + " cadenas comerciales ofrecen
" + artículo + " a un precio menor a $" + monto + ".");
                    }
                } catch (OleDbException err) {
                    MessageBox.Show(err.Message);
                }

                //6- Cerrar la conexión a la BD.
                cnOracle.Close();
            } catch {
                MessageBox.Show("Debe llenar todos los campos.");
            }
                                        }
```

Seleccionar un valor directamente del Grid

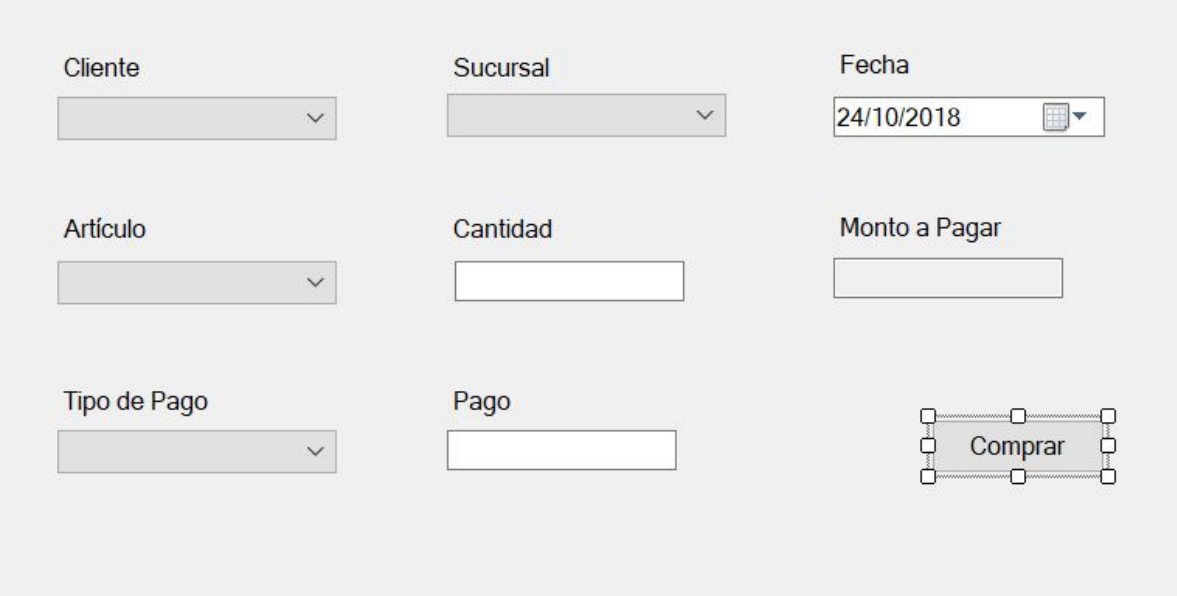
```
//Para obtener el valor de la celda seleccionada al darle clic en el DataGrid
    //existen tres maneras (necesariamente hay que dar clic en la celda elegida):
       //1) nombreDataGrid.CurrentCell.Value;
       //2) nombreDataGrid[ÍndiceColumna, ÍndiceFila].Value;
       //3) nombreDataGrid[NombreColumna, ÍndiceFila].Value
       private void dtgGeneral_CellContentClick(object sender,
DataGridViewCellEventArgs e) {
       int fila, col; Object valor;
```

```
        //Obtiene fila, columna y valor de la celda elegida.
        fila = dtgGeneral.CurrentCell.RowIndex; col =
dtgGeneral.CurrentCell.ColumnIndex;
        //valor = dtgGeneral.CurrentCell.Value;
        //MessageBox.Show("Fila: " + fila + " Columna: " + col + " Valor: " + valor);

        //Obtiene sólo el valor.
        MessageBox.Show("Valor: " + dtgGeneral[col, fila].Value.ToString());
        }
```

Confirmar alta/baja/modificación

```
DialogResult botón = MessageBox.Show("¿Se elimina el registro con folio= " +
folio.ToString(),
        "Eliminación", MessageBoxButtons.YesNo);
//Si se selecciona el botón Yes, del MessageBox, elimina el registro asociado.
        if (botón == DialogResult.Yes)
        {
        //Construye la cadena de eliminación/alta/modificación y la envía para su
ejecución.
        }
```

Alta y Modificación

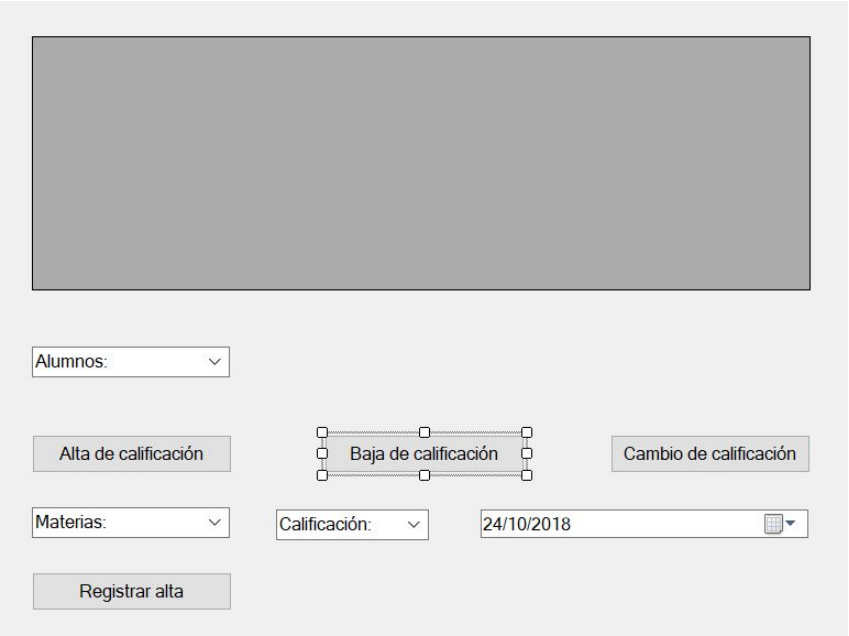


Ejemplo de la tarea 5

```
//Variables de la clase
DataRow fila;
int idCliente = 0;
int idProd = 0;
int folio = 0;
int factura = 0;
int sucursal = 0;
int pago = 0;
```

```
|


double saldoactual = Double.Parse(txtMonto.Text) - Double.Parse(txtPago.Text);

// Cadenas de consulta para llenar los DropDownList
cadSql = "select c.idCliente, idprod, max(com.folio) as Folio, max(f.idfactura) as
Factura, s.idSucursal, max(pa.idpago) as Pago from T4Cliente c, t4producto p,
t4compra com, t4factura f, t4pago pa, t4sucursal s where c.nombre =
'"+cbCliente.Text+"' and p.nombre = '"+cbArticulos.Text+"' and
s.nombre='"+cbSucursal.Text+"' group by(c.idCliente,p.idprod,s.idsucursal)";

GestorBD.consBD(cadSql, dsInfo, "Info");
// Recupera la información de la consulta con un DataRow y almacena los valores en
las variables declaradas previamente
fila = dsInfo.Tables["Info"].Rows[0];
idCliente = Convert.ToInt16(fila["IdCliente"]);
idProd = Convert.ToInt16(fila["IdProd"]);
folio = Convert.ToInt16(fila["Folio"]) +10;
factura = Convert.ToInt16(fila["Factura"]) + 10;
sucursal = Convert.ToInt16(fila["IdSucursal"]);
pago = Convert.ToInt16(fila["Pago"]) + 10;

// Cadenas de inserción de valores en las tablas
String insertCompra = "insert into T4Compra values ("+folio+",'" + dtpFecha.Text +
"'," + txtMonto.Text + ","+idCliente+") ";
String insertFactura = "insert into T4Factura values
("+factura+","+saldoactual+","+indicadorPago+", "+folio+","+sucursal+")";
String insertPago = "insert into T4Pago values ("+pago+"," + txtPago.Text +
","+factura+")";
String insertDe = "insert into T4De values ("+folio+","+idProd+"," +
txtCanttidad.Text +")";

//Cadena de Actualización
String update = "update t4vende set disponibles =disponibles-"+txtCanttidad.Text+"
where idsucursal="+sucursal+"";
// Inserta los valores en la base de datos
GestorBD.altaBD(insertCompra);
GestorBD.altaBD(insertFactura);
GestorBD.altaBD(insertPago);
GestorBD.altaBD(insertDe);
GestorBD.cambiaBD(update);
```

Baja



Ejemplo de Cliente-Servidor

```
//Realiza el proceso para eliminar una calificación de Historial.
       //Se usa el folio, de la fila elegida en el grid, para hacer la baja.
       private void btnBaja_Click(object sender, EventArgs e) {
       int folio; String cadSql;
      DialogResult botón;

       //Toma el folio de la calificación seleccionada en el data grid.
       folio = Convert.ToInt16(dtgGeneral["Folio",
dtgGeneral.CurrentRow.Index].Value);
       botón = MessageBox.Show("¿Se elimina el registro con folio= " +
folio.ToString(),
       "Eliminación", MessageBoxButtons.YesNo);

       //Si se selecciona el botón Yes, del MessageBox, elimina el registro
asociado.
       if (botón == DialogResult.Yes)
       {
       //Construye la cadena de eliminación y la envía para su ejecución.
       cadSql = "delete from Historial where Folio = " + folio;
       if (GestorBD.bajaBD(cadSql) == OK)
       MessageBox.Show("Se eliminó la calificación del folio " + folio + "
exitosamente");
       else
       MessageBox.Show("No se pudo eliminar la calificación");
                                                  }
```

|

Aplicación de Internet

Variables comunes de clase

```
GestorBD.GestorBD GestorBD;
string cadSql;
DataSet DsGeneral = new DataSet();
DataRow fila;
Comunes objComún = new Comunes();  //Para manejar las rutinas de uso común.
private const int OK= 1;
```

Conexión a la BD al cargar la página

```
protected void Page_Load(object sender, EventArgs e) {

       //Extraer la fecha actual para ponerla en un label
       lblFechaHora.Text = DateTime.Now.ToString();

       //NOTA: IsPostBack da false la 1a. vez que se entra a la página.
       //Da true las veces subsecuentes.
       if (!IsPostBack) {
       //Hace la conexión a la BD. Recuerda cambiar al nombre y contraseña propio
              GestorBD = new GestorBD.GestorBD("MSDAORA", "bd01", "fradan",
"oracle");
              Session["GestorBD"] = GestorBD;
       }
  }
```

LogIn

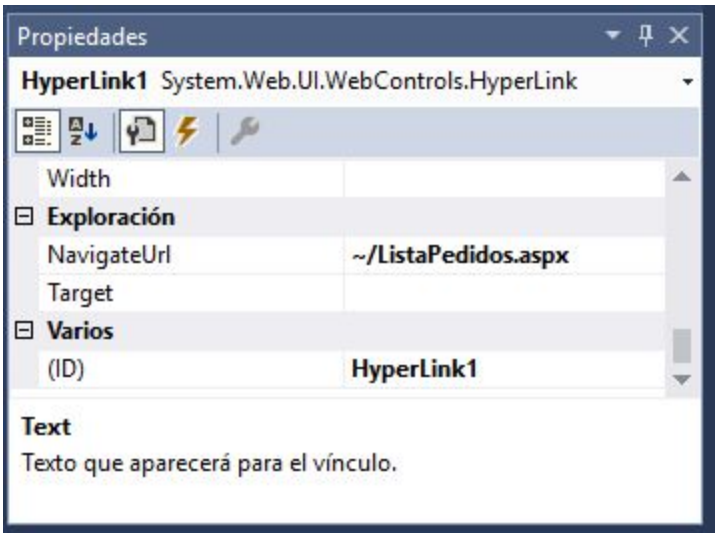
```
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e) {

       if (valida()) {
              Session["rfc"] = Login1.UserName;  //RFC del usuario registrado.
              Server.Transfer("Menu.aspx");      //Transferencia a pagina deseada
       }
  }

//Método generado manualmente para validar
bool valida() {
       //Recupera a GestorBD. Siempre hacer esto
       GestorBD = (GestorBD.GestorBD)Session["GestorBD"];
       cadSql = "select * from pcusuarios where rfc='" + Login1.UserName +
       "' and passw='" + Login1.Password + "'";
       GestorBD.consBD(cadSql, DsGeneral, "Temp");
       //Verifica que la consulta regrese algún resultado
       if (DsGeneral.Tables["Temp"].Rows.Count != 0)
       return true;
       else
       return false;
}
```

Hyperlink

Cuando se usen *Hyperlinks*, la página a la que llegaremos se pone en propiedades cuando estamos en el diseñador

|

Ejemplo: Se dirigía a ListaPedidos



TIP

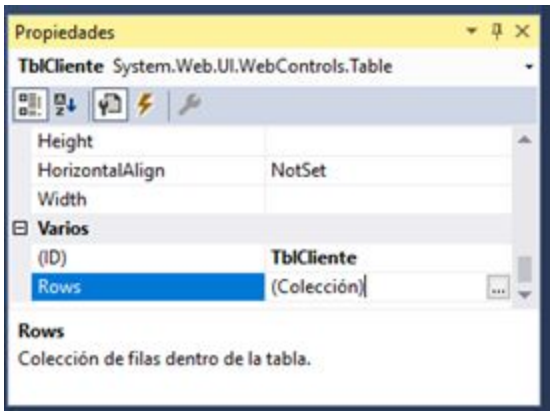
Si necesitas prohibir el acceso de ciertos elementos a ciertos usuarios, usa la función del visible (poniendo todos en false y sólo true cuando se requiera)

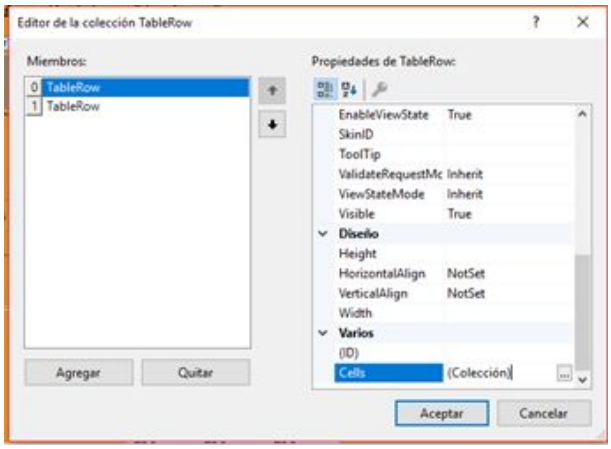
```
if (tipo== "Cli")
{
        HyperLink1.Visible = true;
        HyperLink4.Visible = true;
}
```

Tables

Configuración

(En el diseñador se deben agregar las filas y celdas necesarias)





Para cargar el table

Recuerda: Las rows y cells son como arreglos (Empiezan en 0). La row[0] normalmente se usa para los nombres de las celdas

```
cadSql = "select * from pcusuarios u, pcclientes c where u.rfc='" + rfc +
```

|

```
       "' and u.rfc=c.rfc";
GestorBD.consBD(cadSql, DSGeneral, "Cliente");
fila = DSGeneral.Tables["Cliente"].Rows[0];

TblCliente.Rows[1].Cells[0].Text = fila["RFC"].ToString();
TblCliente.Rows[1].Cells[1].Text = fila["Nombre"].ToString();
TblCliente.Rows[1].Cells[2].Text = fila["Domicilio"].ToString();
```

DropDownList

Recuerda:

✓ Si con el click de un elemento harás una acción… habilita el AutoPostBack

✓ Si solo obtendrás el texto de lo seleccionado entonces no es necesario AutoPostBack

Para cargar el Ddl

Recuerda: Debiste haber declarado objComún y por lo tanto importado la clase Comunes

```
//Lee sus pedidos y carga los folios en el ddl de pedidos.
       cadSql = "select * from pcpedidos where rfcc='" + rfc + "'";
       GestorBD.consBD(cadSql, DSPedidos, "Pedidos");
       objComún.cargaDDL(DdlPedidos, DSPedidos, "Pedidos", "FolioP");
```

Usar el texto seleccionado de un DDL

(Para hacer consultas)

```
if (tipo == "Cli")
{
       cadSql = "select * from PCPedidos where RFCC='" + rfc + "' and FolioP="+
       DdlPedidos.Text;
       GestorBD.consBD(cadSql, DSPedidos, "Pedidos");
       fila = DSPedidos.Tables["Pedidos"].Rows[0];
}

//Segunda alternativa: usando la información que ya está en el DataSet (más
eficiente,
//pero puede tener datos desactualizados).
//filas = (DataRow[])DSPedidos.Tables["Pedidos"].Select("FolioP=" +
DDLPedidos.Text);
//fila = filas[0];
```

Consulta

Recuerda confirmar los tipos de las variables y extraerlas acordemente

```
//Recupera objetos de Session.

GestorBD = (GestorBD.GestorBD)Session["GestorBD"];
rfc = Session["rfc"].ToString();

//Lee los datos del cliente.
```

```
cadSql = "select * from pcusuarios u where u.rfc='" + rfc + "'";
GestorBD.consBD(cadSql, DSGeneral, "Usuario");
fila = DSGeneral.Tables["Usuario"].Rows[0];
tipo = fila["Tipo"].ToString();
```

Cargar GridView

```
//Muestra los artículos del pedido.
       cadSql = "select Nombre, CantPed, CantEnt from PCDetalle d, PCArtículos a " +
      "where FolioP=" + DdlPedidos.Text + " and d.IdArt=a.IdArt";
       GestorBD.consBD(cadSql, DSArtículos, "Artículos");
       GrdArtículos.DataSource = DSArtículos.Tables["Artículos"];
       GrdArtículos.DataBind();
```

Ejecutar un procedimiento almacenado

```
//Ejemplo de ejecución de un procedimiento almacenado (en Oracle).
  //Procedimiento cantPedidosArt:
  // recibe como parámetro de entrada el nombre de un artículo y regresa como
  // parámetro de salida la cantidad de pedidos en los que aparece.
  protected void BtnEjecutaProc_Click(object sender, EventArgs e) {
       String nomArt;
       int cant;

       //1- Abrir la conexión a la BD.
       cnOracle = GestorBD.conex;
       cnOracle.Open();
       procAlm = new OleDbCommand();
       procAlm.Connection = cnOracle;

       //2- Especificar el llamado al procedimiento  (en general: al subprograma).
       procAlm.CommandText = "cantPedidosArt";
       procAlm.CommandType = CommandType.StoredProcedure;

       //3- Especificar los parámetros:
       //a) primero todos los de entrada:
       nomArt = "Portafolios";
       par = new OleDbParameter("nom", nomArt);
       procAlm.Parameters.Add(par);

       //b) luego todos los de salida (uno en este caso):
       par = new OleDbParameter("cant", OleDbType.Integer,
       4, ParameterDirection.Output, false, 4, 0, "NomArt", DataRowVersion.Current,
0);
       procAlm.Parameters.Add(par);

    //4- Ejecutar el procedimiento (en general: el subprograma).
       try {
       procAlm.ExecuteNonQuery();

       //5- Recuperar el (los) valor(es) regresado(s) por medio del (de los)
       //   parámetro(s) de salida.
       cant = Convert.ToInt16(procAlm.Parameters["cant"].Value);
```

```
        Label1.Text= "Cantidad de pedidos: " + cant;
        }
        catch (OleDbException err) {
        Label1.Text = err.Message;
        }

        //6- Cerrar la conexión a la BD.
        cnOracle.Close();
  }
```

Ejecutar una función almacenada

```
//Ejemplo de ejecución de una función almacenada (en Oracle).
  //Función cantArtsPedido:
  // recibe como parámetro de entrada el folio de un pedido,
  // y regresa la cantidad de artículos pedidos.
  protected void BtnEjecutaFunc_Click(object sender, EventArgs e) {
        int folioPed, cantArts;

        //1- Abrir la conexión a la BD.
        cnOracle = GestorBD.conex;
        cnOracle.Open();
        procAlm = new OleDbCommand();
        procAlm.Connection = cnOracle;

        //2- Especificar el llamado a la función.
        procAlm.CommandText = "cantArtsPedido";
        procAlm.CommandType = CommandType.StoredProcedure;

        //3- Especificar los parámetros:
        // a) Nota: primero hay que definir el tipo de valor que regresará la
función.
        par = new OleDbParameter("RETURN_VALUE", OleDbType.Integer,
        4, ParameterDirection.ReturnValue, false, 4, 0, "cantArts",
DataRowVersion.Current, 0);
        procAlm.Parameters.Add(par);

        // b) luego hay que definir los parámetros de entrada (uno en este caso).
        folioPed = 10;
        par = new OleDbParameter("folio", folioPed);
        procAlm.Parameters.Add(par);

        //4- Ejecutar el procedimiento (en general: el subprograma).
        try {
              procAlm.ExecuteNonQuery();

              //5- Recuperar el valor regresado por la función.
              cantArts = Convert.ToInt16(procAlm.Parameters["RETURN_VALUE"].Value);
              Label1.Text = "Cantidad de artículos: " + cantArts;
        } catch (OleDbException err) {
              Label1.Text = err.Message;
        }
```

```
        //6- Cerrar la conexión a la BD.
        cnOracle.Close();
        }
```

Elegir qué método ejecutar con un botón

```
//=====================================================================
        //Parte 4: acciones relacionadas con la ejecución de la operación elegida.
        protected void BtnEjecuta_Click(object sender, EventArgs e) {
        String oper;

        if (Page.IsValid) {
                oper = Session["Operación"].ToString();
                switch (oper) {
                case "Alta":
                        alta();
                        break;
                case "Baja":
                        baja();
                        break;
                case "Cambio":
                        //cambio();
                        break;
                }

                TxtRFC.Visible = false; TxtNombre.Visible = false;
                TxtPassw.Visible = false;
                //TxtPassw.TextMode = TextBoxMode.Password;   //Para que aparezcan
asteriscos al dar la contra.
                TxtDomicilio.Visible = false; TxtCat.Visible = false;
                DDLTipo.Text = "Tipo de usuario"; DDLTipo.Visible = false;
DDLUsuarios.Visible = false;
                BtnAlta.Enabled = true; BtnBaja.Enabled = true; BtnCambio.Enabled =
true;
                BtnEjecuta.Enabled = false; LblMensaje.Text = "En espera";
        }
        }
```

Alta

(En este caso llamado por el botón Ejecuta)

```
//=====================================================================
        //Alta de un nuevo usuario:
        //primeramente lo da de alta en la tabla de Usuarios, verificando antes que
no exista el RFC. Después da de alta en las tablas de Clientes o Empleados, según el
tipo de usuario de que se trate.
        public void alta() {
        //Parte 1b: da de alta al nuevo usuario.
        GestorBD = (GestorBD.GestorBD)Session["GestorBD"];

        //Verifica que el RFC no exista en la BD.
        cadSql = "select * from PCUsuarios where RFC = '" + TxtRFC.Text + "'";
        GestorBD.consBD(cadSql, DsGeneral, "Usuario");
        if (DsGeneral.Tables["Usuario"].Rows.Count == 0) {
```

```
|


            //Si el RFC no existe, entonces inserta en la tabla de Usuarios.
            cadSql = "insert into PCUsuarios values('" + TxtRFC.Text + "','" +
TxtNombre.Text +
                    "','" + TxtPassw.Text + "','" +
DDLTipo.SelectedValue.ToString() + "')";

            if (GestorBD.altaBD(cadSql) == OK) {
            if (DDLTipo.SelectedValue.ToString() == "Cli") {
                    //=========================================
                    //Parte 1c: alta de un cliente.
                    cadSql = "insert into PCClientes values('" + TxtRFC.Text +
                                "','" + TxtDomicilio.Text + "')";
                    if (GestorBD.altaBD(cadSql) == OK)
                    LblMensaje.Text = "Inserción exitosa en Usuarios y Clientes";
                    else
                    LblMensaje.Text = "Error de inserción en la tabla Clientes";
            } else {
                    //=========================================
                    //Parte 1d: alta de un empleado.
                    cadSql = "insert into PCEmpleados values('" + TxtRFC.Text +
"','" + TxtCat.Text + "')";
                    if (GestorBD.altaBD(cadSql) == OK)
                    LblMensaje.Text = "Inserción exitosa en Usuarios y Empleados";
                    else
                    LblMensaje.Text = "Error de inserción en la tabla Empleados";
            }

        } else
            LblMensaje.Text = "Error: Este RFC ya existe en la BD";
    }
}
```

Baja

(En este caso llamado por el botón Ejecuta)

```
//Lee los datos de los usuarios y muestra sus RFC en el DDL de usuarios.
       //Esta rutina es usada tanto en baja, como en cambio de datos de usuarios.
       protected void leeUsuarios() {
       GestorBD = (GestorBD.GestorBD)Session["GestorBD"];
       cadSql = "select * from PCUsuarios";
       GestorBD.consBD(cadSql, DsGeneral, "DatosUsuarios");
       comunes.cargaDDL(DDLUsuarios, DsGeneral, "DatosUsuarios", "RFC");
       Session["DsGeneral"] = DsGeneral;
       }

       //Parte 2b: da de baja al usuario elegido en DDLUsuarios.
       public void baja() {
       String rfc;

       //Da de baja en Usuarios. Por el trigger se da de baja automáticamente en
       Clientes o Empleados.
       rfc = DDLUsuarios.SelectedValue;
```

```
        GestorBD = (GestorBD.GestorBD)Session["GestorBD"];
        cadSql = "delete from PCUsuarios where RFC='" + rfc + "'";
        if (GestorBD.bajaBD(cadSql) == OK)
                Response.Write("Eliminación exitosa en Usuarios");
        else
                LblMensaje.Text = "Error de eliminación en la tabla Usuarios";
        }

        //En la baja, este método habilita el botón de ejecutar operación.
        protected void DDLUsuarios_SelectedIndexChanged(object sender, EventArgs e) {

        if (DDLUsuarios.Text != " ")
                BtnEjecuta.Enabled = true;
        else
                BtnEjecuta.Enabled = false;
        }
```

Nota: Tanto el alta como la baja pueden venir de un botón específico alta/baja o de un botón de ejecución. En caso de esta última opción, recuerda deshabilitar el otro botón al elegir la opción deseada. Para lograr esto, se hace lo siguiente:

```
//=====================================================================
        //Parte 1a: acciones relacionadas con el alta de usuarios.
        //Muestra/deshabilita los controles asociados al alta.
        protected void BtnAlta_Click(object sender, EventArgs e) {
        TxtRFC.Visible = true; TxtNombre.Visible = true;
        TxtPassw.Visible = true; DDLTipo.Visible = true;
        BtnBaja.Enabled = false; BtnCambio.Enabled = false;
        LblMensaje.Text = "Operación: Alta";
        Session["Operación"] = "Alta";
        }

//=====================================================================
        //Parte 2a: acciones relacionadas con la baja de un usuario.
        protected void BtnBaja_Click(object sender, EventArgs e) {
        //Lee los datos de los usuarios y muestra sus RFC en DDLUsuarios.
        leeUsuarios();

        //Oculta/deshabilita controles.
        DDLUsuarios.Visible = true;
        BtnAlta.Enabled = false;
        BtnBaja.Enabled = false;
        BtnCambio.Enabled = false;
        LblMensaje.Text = "Operación: Baja";
        Session["Operación"] = "Baja";
        }
```

Access

Cadena para conectar con el la base de datos

```
GestorBD = new GestorBD.GestorBD("Microsoft.ACE.OLEDB.12.0","Admin", "",
"C:\\BD\\Ex2_prueba\\BD2oPedLibros.accdb");

              cadSql = "SELECT Distribuidor.* FROM Distribuidor INNER JOIN Pedido ON
Distribuidor.RfcDis = Pedido.RfcDis WHERE(((Pedido.IdPed) = 1))";

              GestorBD.consBD(cadSql, DsGrid, "Info");

              dgvDistribuidor.DataSource = DsGrid.Tables["Info"];
```