GUIA PRACTICA - REGISTRO MAESTRO DETALLE EN ASP NET

Especificación Funcional

OBJETIVO:

- Implementar el registro de una orden de compra, adjuntando una proforma.
- Listar las Ordenes generadas y las proformas adjuntas

PARTE 1: REGISTRO DE LA ORDEN DE COMPRA

Paso 1: Implementar la capa de datos y de negocios

Realice lo siguiente:

1. Inicie sesión en su servidor SQL y ejecute el script "Script_ListarOrdenes" que es parte del material de la sesión. Revise el código junto a su instructor.
2. Abra su solución "DemoVentas_3CapasWEB". Agregar las clases OrdenBE y OrdenADO (que son parte de su material) a sus respectivos proyectos. Revise junto a su instructor ambas clases
3. Diríjase al proyecto "ProyVentas_ADO" e implemente en la clase OrdenADO los métodos RegistrarOrden y ListarOrden:

```
public String RegistrarOrden(OrdenBE objOrdenBE)
{
    try
    {
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_RegistrarOCompra";
         cmd.Parameters.Clear();
         cmd.Parameters.AddWithValue("@vfecoco", objOrdenBE.FecOco);
         cmd.Parameters.AddWithValue("@vcodprv", objOrdenBE.CodPrv);
         cmd.Parameters.AddWithValue("@vfecate", objOrdenBE.FecAte);
         cmd.Parameters.AddWithValue("@vestoco", objOrdenBE.EstOco);
         cmd.Parameters.AddWithValue("@vUsu_registro", objOrdenBE.Usu_Registro);
         // Parametro de salida
        cmd.Parameters.Add(new SqlParameter("@vnumoco", SqlDbType.Char, 6));
        cmd.Parameters["@vnumoco"].Direction = ParameterDirection.Output;
         // Parametro tipo tabla con los detalles
        cmd.Parameters.Add(new SqlParameter("@detalles", SqlDbType.Structured));
        cmd.Parameters["@detalles"].Value = objOrdenBE.DetallesOC;
        cnx.Open();
        cmd.ExecuteNonQuery();
         // Retorna el NUMOCO generado.
        return cmd.Parameters["@vnumoco"].Value.ToString ();
    }
    catch (Exception ex)
    {
        return String.Empty;
    }
public DataTable ListarOrden()
{
    try
    {

        DataSet dts = new DataSet();
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_ListarOrden";
        cmd.Parameters.Clear();
        SqlDataAdapter ada = new SqlDataAdapter(cmd);
        ada.Fill(dts, "Ordenes");
        return dts.Tables["Ordenes"];

    }
    catch (SqlException ex)
    {
        throw new Exception(ex.Message);
    }

}
}
```

4. Ahora en la clase ProveedorADO (que ya la tiene en su capa de datos) agregue los siguientes métodos:

```
// Para el registro de OC...
public DataTable ListarProveedoresActivos()
{
    DataSet dts = new DataSet();
    cnx.ConnectionString = MiConexion.GetCnx();
    cmd.Connection = cnx;
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.CommandText = "usp_ListarProveedoresActivos";
    try
    {
        //Codifique
        cmd.Parameters.Clear();
        SqlDataAdapter ada = new SqlDataAdapter(cmd);
        ada.Fill(dts, "ProveedoresActivos");
        return dts.Tables["ProveedoresActivos"];
    }
    catch (SqlException ex)
    {
        throw new Exception(ex.Message);
    }

}

public DataTable ListarProductosProveedor(String cod_prv)
{
    DataSet dts = new DataSet();
    cnx.ConnectionString = MiConexion.GetCnx();
    cmd.Connection = cnx;
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.CommandText = "usp_ListarProductosProveedor";
    try
    {
        cmd.Parameters.Clear();
        cmd.Parameters.AddWithValue("@cod_prv", cod_prv);
        SqlDataAdapter ada = new SqlDataAdapter(cmd);
        ada.Fill(dts, "ProductosProveedor");
        return dts.Tables["ProductosProveedor"];
    }
    catch (SqlException ex)
    {
        throw new Exception(ex.Message);
    }

}
```

ListarProveedoresActivos es el método que permitirá seleccionar proveedores activos al momento de registrar la OC y **ListarProductosProveedor** permite solo filtrar productos que abastece un determinado proveedor. Con ambos métodos llenaremos los combos de **cboProveedor** y **cboProducto** en la capa de presentación.

5. En el proyecto "ProyVentas_BL" copie la clase **OrdenBL** (incluida en el material). Verá que ya se tiene el llamado al método **RegistrarOrden** implementado en el punto 3:

```
//Agregar...
using ProyVentas_ADO;
using ProyVentas_BE;

namespace ProyVentas_BL
{
    public class OrdenBL
    {
        OrdenADO objOrdenADO = new OrdenADO();

        public string RegistrarOrden(OrdenBE objOrdenBE)
        {
            return objOrdenADO.RegistrarOrden(objOrdenBE);
        }

        public DataTable ListarOrden()
        {
            return objOrdenADO.ListarOrden();
        }
```

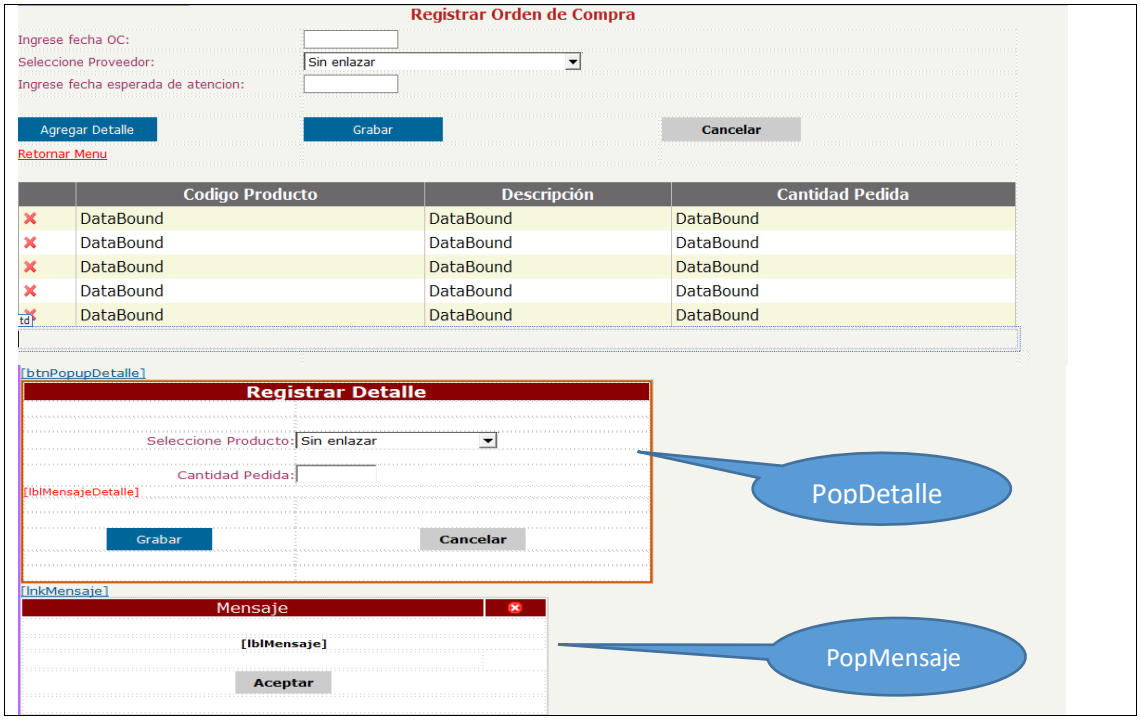
6. Para terminar con la capa de Lógica de Negocios (BL) diríjase a la clase **ProveedorBL** y agregue los métodos que invoquen a los que implementó en la capa de datos en el punto 4 y luego compile la solución:

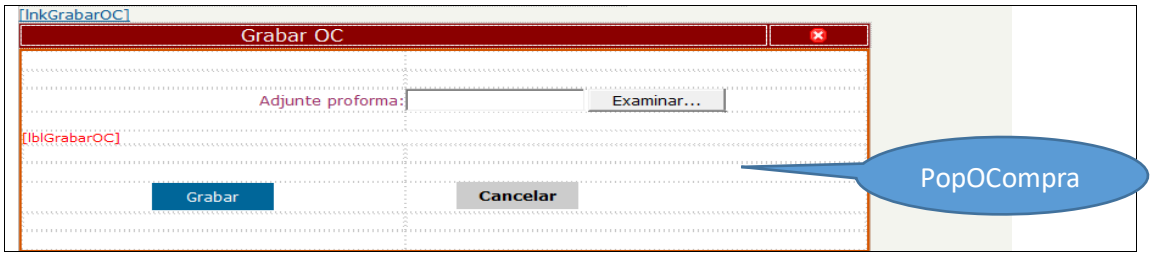
```
public DataTable ListarProveedoresActivos()
{
    return objProveedorADO.ListarProveedoresActivos();
}

public DataTable ListarProductosProveedor(String cod_prv)
{
    return objProveedorADO.ListarProductosProveedor(cod_prv);
}
```

Paso 2: Implementar la capa de presentación para el registro de la orden de compra.

Realice lo siguiente

1. En la capa de presentación, proyecto "SitioWEB_VentasGUI", copie el WebForm "RegistrarOrden.aspx" (que está en sus materiales del presente ejercicio) en la carpeta Transacciones del sitio. Este formulario tiene el siguiente diseño:





Como se aprecia hay 3 ModalPopup

- **PopDetalle:** Para el registro de cada detalle de la OC.
- **PopMensaje:** Para el envío de mensajes Popup
- **PopOCompra:** Para el paso final del registro de la OC adjuntando la proforma

2. Este WebForm está parcialmente codificado, es parte de la especificación concluir con su codificación. Empezaremos codificando el evento Load.

```
protected void Page_Load(object sender, EventArgs e)
{
    try
    {
        if (Page.IsPostBack == false)
        {
            // Para manejar la fecha en formato dd-mm-yyyy
            Thread.CurrentThread.CurrentUICulture = new CultureInfo("es-ES");
            // Pintamos la fecha de hoy por defecto
            txtFecOco.Text = DateTime.Now.Date.ToShortDateString();
            txtFecAte.Text = DateTime.Now.Date.ToShortDateString();
            // Enlazamos el combo de proveedores (solo los proveedores  activos)
            ProveedorBL objProveedorBL = new ProveedorBL();
            DataTable dt = objProveedorBL.ListarProveedoresActivos();
            DataRow dr;
            dr = dt.NewRow();
            dr["cod_prv"] = "X";
            dr["raz_soc_prv"] = "--Seleccione--";
            dt.Rows.InsertAt(dr, 0);
            cboProveedor.DataSource = dt;
            cboProveedor.DataValueField = "cod_prv";
            cboProveedor.DataTextField = "raz_soc_prv";
            cboProveedor.DataBind();

            //Creamos el datatable en memoria para almacenar los detalles
            CrearTabla();

        }
    }
    catch (Exception ex)
    {
        lblMensajePopup.Text = ex.Message;
        PopMensaje.Show();
    }

}
```

3. Como se aprecia se preparan las cajas de texto de las fechas, se carga el combo de proveedores y se invoca el método **CrearTabla** (que ya está implementado como parte de la plantilla). Este método lo que hace es crear un DataTable en memoria, el cual almacenará todos los detalles que el usuario vaya registrando en la orden. Para que esta tabla en memoria no pierda sus valores la guardamos en una variable de sesión y la enlazamos al gridView "grDetalles". A continuación, el código del método CrearTabla, solo con la intención de entender mejor la guía, dado que el método ya está en la plantilla por lo que no hace falta su codificación.

```
private void CrearTabla()
{
    mitb = new DataTable("TBCompras");
    //Creando las columnas para la tabla temporal de  detalle de compra
    //Columna Codigo
    DataColumn Ccodigo = new DataColumn("Cod_Pro");
    Ccodigo.DataType = Type.GetType("System.String");
    mitb.Columns.Add(Ccodigo);
    //Columna Descripcion
    DataColumn Cdescripcion = new DataColumn("Des_Pro");
    Cdescripcion.DataType = Type.GetType("System.String");
    mitb.Columns.Add(Cdescripcion);
    //Columna Cantidad
    Ccantidad = new DataColumn("Can_Ped");
    Ccantidad.DataType = Type.GetType("System.Int16");
    mitb.Columns.Add(Ccantidad);
    //definimos la PK
    mitb.PrimaryKey = new DataColumn[] { mitb.Columns["Cod_Pro"] };
    grDetalles.DataSource = mitb;
    grDetalles.DataBind();
    Session["Detalles"] = mitb;

}
```

4. Implementemos ahora lo necesario para registrar los detalles (por ahora en el DataTable creado en el punto anterior). Codificaremos código para el evento **SelectedIndexChange** del combo **cboProveedor** para que el combo **cboProductos** (del ModalPopup PopDetalle) muestre sólo los productos abastecidos por el proveedor seleccionado:

```
protected void cboProveedor_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        // Si el gridView ya tiene filas es porque ya registro detalles de OC y no puede cambiar de proveedor
        if (grDetalles.Rows.Count > 0)
        {
            throw new Exception("Ya selecciono productos, no puede cambiar de proveedor.");

        }

        // Mostramos solo los productos del proveedor seleccionado
        DataTable dt = objProveedorBL.ListarProductosProveedor(cboProveedor.SelectedValue.ToString());
        DataRow dr;
        dr = dt.NewRow();
        dr["cod_pro"] = "X";
        dr["des_pro"] = "--Seleccione--";
        dt.Rows.InsertAt(dr, 0);
        cboProducto.DataSource = dt;
        cboProducto.DataValueField = "cod_pro";
        cboProducto.DataTextField = "des_pro";
        cboProducto.DataBind();
        cboProducto.SelectedValue = "";
    }
    catch (Exception ex)
    {
        lblMensajePopup.Text = ex.Message;
        PopMensaje.Show();
    }

}
```

Se observa cómo se valida que si el gridview **grDetalles** ya tiene detalles incluidos no es posible cambiar el proveedor. Si aún no se ha seleccionado ningún detalle podemos cambiar de proveedor y con ello se llenará el combo de productos con los productos abastecidos por el proveedor seleccionado.

5. Codificaremos ahora el evento clic del botón **btnAgregar**, el cual mostrara el modalPopup **PopDetalles** para seleccionar el producto y su cantidad. Los productos serán solo los abastecidos por proveedor seleccionado

```
protected void btnAgregar_Click(object sender, EventArgs e)
{
    try
    {
        if (cboProveedor.SelectedValue == "X")
        {
            throw new Exception("Debe seleccionar un proveedor.");
        }
        // Mostramos el popup  de detalle
        cboProducto.SelectedIndex = 0;
        txtCanPed.Text = String.Empty;
        lblMensajeDetalle.Text = String.Empty;
        PopDetalle.Show();

    }
    catch (Exception ex)
    {
        lblMensajePopup.Text = ex.Message;
        PopMensaje.Show();
    }

}
```

6. Para ir agregando los detalles en el DataTable, en el evento clic del botón **btnGrabarDetalle** (que está dentro del modalpopup PopDetalle) se debe codificar lo siguiente:

```
protected void btnGrabarDetalle_Click(object sender, EventArgs e)
{
    try
    {
        // Validamos el detalle antes de registrarlo en memoria
        if (cboProducto.SelectedValue == "X")
        {
            throw new Exception("Debe seleccionar un producto");
        }
        if (txtCanPed.Text == "")
        {
            throw new Exception("Debe ingresar una cantidad");
        }
        // Si todo esta OK casteamos la variable de sesion "Detalles" a DataTable
        mitb = (DataTable)Session["Detalles"];
        //Creamos una fila y la instanciamos como fila de mitb
        DataRow dr;
        dr = mitb.NewRow();
        //Llenamos los campos con lo ingresado en el Popup de detalle
        dr["Cod_Pro"] = cboProducto.SelectedValue.ToString();
        dr["Des_Pro"] = cboProducto.SelectedItem.ToString();
        dr["Can_Ped"] = Convert.ToInt16(txtCanPed.Text);
        //Agregamos la fila a la coleccion de filas de detalle
        mitb.Rows.Add(dr);
        // Lo mostramos en la grilla
        grDetalles.DataSource = mitb;
        grDetalles.DataBind();
    }
    catch (Exception ex)
    {
        lblMensajeDetalle.Text = "Error: " + ex.Message;
        PopDetalle.Show();
    }

}
```

Se aprecia cómo se agrega al DataTable creado en memoria el producto seleccionado y su cantidad pedida para luego mostrarlo en el gridview de detalle **grDetalles**. Para hacerlo validamos que se haya seleccionado un producto e ingresado una cantidad lógicamente.

7. Es parte del proceso corregir un detalle que por error se registró. Como todo está en memoria podemos rectificar un detalle eliminándolo del DataTable en memoria. Para ello se codificará el evento **RowCommand** del **grDetalles**:

```
protected void grDetalles_RowCommand(object sender, GridViewCommandEventArgs e)
{
    try
    {
        // Obtenemos el indice de la fila seleccionada
        int indicefila = Convert.ToInt16(e.CommandArgument);

        // Si se pulso en el boton eliminar , eliminamos el detalle de memoria
        if (e.CommandName == "Eliminar")
        {
            mitb = (DataTable)Session["Detalles"];
            mitb.Rows.RemoveAt(indicefila);
            grDetalles.DataSource = mitb;
            grDetalles.DataBind();
            Session["Detalles"] = mitb;
        }
    }
    catch (Exception ex)
    {
        lblMensajePopup.Text = ex.Message;
        PopMensaje.Show();
    }
}
```

Cabe señalar que el gridview **grDetalles** tiene una columna de tipo button llamada "Eliminar" definida en la propiedad CommandName. Al hacer clic sobre esa columna se remueve de memoria la fila asociada al detalle en memoria.

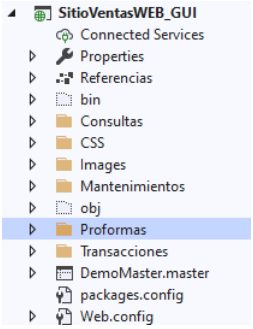
8. Ahora, para registrar la orden de compra, adjuntando su proforma, explicaremos el evento clic del botón **btnVerGrabar** que ya está como parte de la plantilla

```
protected void btnVerGrabar_Click(object sender, EventArgs e)
{
    try
    {
        mitb = (DataTable)Session["Detalles"];

        // Validamos las fechas...
        if (txtFecOco.Text == "")
        {
            throw new Exception("Debe ingresar fecha de OC.");
        }

        if (txtFecAte.Text == "")
        {
            throw new Exception("Debe ingresar fecha de Atencion.");
        }

        if (Convert.ToDateTime(txtFecOco.Text) > Convert.ToDateTime(txtFecAte.Text))
        {
            throw new Exception("La fecha de orden no puede ser mayor que la de atencion.");

        }
        //Si no hay detalles, no se puede registrar la  orden
        if (mitb.Rows.Count == 0)
        {
            throw new Exception("No puede registrar una orden sin detalles.");
        }

        // Si todo esta OK mostramos el ModalPopup para el paso final de la generacion de la OC
        lblGrabarOC.Text = "";
        PopOCompra.Show();
    }
    catch (Exception ex)
    {
        lblMensajePopup.Text = ex.Message;
        PopMensaje.Show();
    }

}
```

Como se observa, para poder mostrar el modalPopup **mpGrabarOC** se hacen validaciones de fechas y de que se tenga al menos un detalle en el gridview **grDetalles**. Si todas las validaciones son correctas se muestra el modalPopup **mpGrabarOC** que permitirá adjuntar la proforma y registrar en la base de datos la orden de compra.

9. En el evento clic del botón **btnGrabarOC** se debe registrar la orden de compra en la base de datos, obtener el nuevo número de orden de compra generada y con ese número enviar la proforma adjunta en el fileUpLoad **fulOC** a una carpeta en el servidor llamada **Proformas**. Cada proforma tendrá como nombre el número de la orden de compra asociada.

NOTA: No olvide crear la carpeta Proformas dentro de su sitio WEB, para que en ella se envíen las proformas adjuntas:



El código siguiente valida la existencia del archivo adjunto, que este sea PDF y no tenga un tamaño mayor a 4MB. De estar todo correcto, se registra la orden de compra (cabecera y detalle en la BD) y se envía la proforma, nombrándola con el número de orden generado, para por ultimo enviar un mensaje en el modalpopup **mpMensaje** indicando que se registró exitosamente la orden de compra:

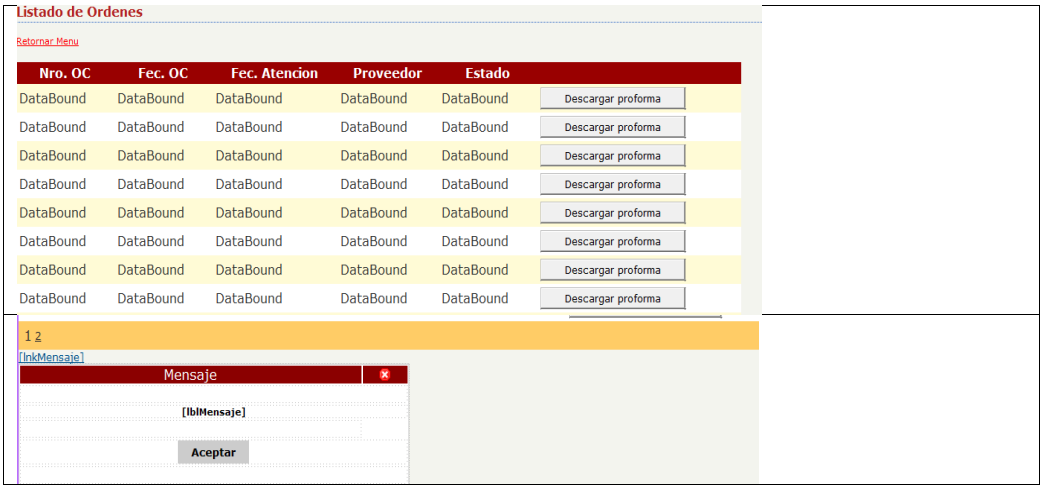
```
protected void btnGrabarOC_Click(object sender, EventArgs e)
{
    try
    {
        if (fulOC.HasFile == false)
        {
            throw new Exception("Debe adjuntar proforma");
        }
        // Si  adjunto la proforma se valida que sea PDF y del 4MG
        String strNombreArchivo = fulOC.FileName;
        // Obtenemos la extension del archivo seleccionado...
        String strExtension = Path.GetExtension(strNombreArchivo).ToLower();
        // Si la extension es PDF
        if (strExtension != ".pdf" && fulOC.PostedFile.ContentLength != 4200000)
        {
            throw new Exception("El archivo no es PDF o excede los 4Mb.");
        }

        OrdenBE objOrdenBE = new OrdenBE();
        OrdenBL objOrdenBL = new OrdenBL();

        //Asignamos valores de cabecera ( El nro lo genera el SP)
        objOrdenBE.FecOco = Convert.ToDateTime(txtFecOco.Text);
        objOrdenBE.FecAte = Convert.ToDateTime(txtFecAte.Text);
        objOrdenBE.EstOco = "1";  // Se registra como pendiente
        objOrdenBE.CodPrv = cboProveedor.SelectedValue;
        objOrdenBE.Usu_Registro = "jleon";

        // Asignamos los detalles a la propiedad respectiva
        objOrdenBE.DetallesOC = mitb;

        // Se evalua el exito del metodo
        String strNumOC = objOrdenBL.RegistrarOrden(objOrdenBE);
        if (strNumOC == String.Empty)
        {
            throw new Exception("Error , no se registró la orden. Revise");
        }
        else
        {
            // Definimos la ruta destino de los documentos...
            String strRuta = Server.MapPath("/") + @"Proformas\";
            strRuta += strNumOC + ".pdf";
            // Enviamos el archivo al servidor...
            fulOC.SaveAs(strRuta);
            // Reinicio los controles y  la tabla por si se desea registrar una nueva orden de compra
            txtFecAte.Text = DateTime.Now.ToShortDateString();
            txtFecOco.Text = DateTime.Now.ToShortDateString();
            cboProveedor.SelectedIndex = 0;
            CrearTabla();
            // Se envia mensaje
            lblMensajePopup.Text = "Se registró la orden Nro: " + strNumOC + " exitosamente.";
            PopMensaje.Show();

        }
    }
    catch (Exception ex)
    {
        lblGrabarOC.Text = ex.Message;
        PopOCompra.Show();
    }
}
```

PARTE 2: LISTAR LAS ORDENES DE COMPRA Y DESCARGAR LA PROFORMA ADJUNTA

Paso 1: Codificar la capa de datos

Paso 2: Codificar la capa de presentación

1. Copie el WebForm WebListarOrdenes (proporcionado en sus materiales) a la carpeta Transacciones de su proyecto WEB SitioVentasWEB_GUI. Se tiene el siguiente diseño:



Se tiene un gridView llamado **grvOC** donde se mostrarán los campos asociados a todas las ordenes registradas, y una columna para descargar la proforma asociada. La grilla tiene una paginación local para facilitar la navegación y también se incluye un modalPopup **PopMensaje** para envíos de mensajes Popup.

2. El formulario está parcialmente codificado, faltando solo la parte referente a la descarga de la proforma. En el evento **RowCommand** del gridView **grvOC** codifique lo siguiente:

```
protected void grvOC_RowCommand(object sender, GridViewCommandEventArgs e)
{
    try
    {
        // Obtenemos la fila donde se hizo click en el boton Descargar
        Int16 fila = Convert.ToInt16(e.CommandArgument);

        // Si se  invoco a la columna Descargar
        if (e.CommandName == "Descargar")
        {
            // Se obtiene el nombre de archivo a descargar
            String strNomArchivo = grvOC.Rows[fila].Cells[0].Text + ".pdf";
            // Obtenemosla ruta del  archivo de la carpeta Proformas
            String strRuta = Server.MapPath("/") + @"Proformas\" + strNomArchivo;
            if (File.Exists(strRuta) == true)
            {
                // Se codifica los eventos en el cliente para la descarga del archivo
                Response.Clear();

                Response.AddHeader("content-disposition", string.Format("attachment;filename={0}", strNomArchivo));
                Response.ContentType = "application/octet-stream";

                Response.WriteFile(Server.MapPath("/") +@"~/Proformas" + strNomArchivo);
                Response.End();

            }
            else
            {
                throw new Exception("Archivo no existe");

            }

        }
    }
    catch (Exception ex)
    {
        lblMensajePopup.Text = "Error: " + ex.Message;
        PopMensaje.Show();
    }
}
}
```

Al hacer clic en el botón con CommandName Descargar se procede a la descarga de la proforma asociada. Si la orden no tiene proforma asociada se envía el respectivo mensaje.

NOTA:

Para probar la parte 1 y 2 de esta guía adicione al formulario Transacciones un control treeview con nodos que direccionen a los formularios RegistrarOrden y WebListarOrdenes tal como se explicó en clase.