

1) Crear GestorPresupuesto.cs en carpeta servicios (tiene una instancia dao de IPresupuestoDao.cs, se ocupa de la logica como que me diga el proximo presupuesto, creo un private IPresupuestoDAO dao siendo dao un objeto y le doy directamente la interfaz que contiene los metodos, uso la interfaz pq me obliga a tener los metodos hechos, dao es la interfaz que llama a PresupuestoDao.cs junto con sus metodos)

```
1  using Carpinteria_Refactorizado.accesoDatos;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Carpinteria_Refactorizado.servicios
9  {
10     0 referencias
11     class GestorPresupuesto
12     {
13         private IPresupuestoDAO dao;
14
15         0 referencias
16         public int ProximoPresupuesto()
17         {
18             return dao.ObtenerProximoNroPresupuesto();
19         }
20     }
```

2) Crear IPresupuestoDAO.cs en carpeta accesoDatos (interfaz que contiene los metodos que debe implementar PResupuestoDAO al acceder a datos, me obliga a implementar los metodos en PresupuestoDAO.cs)

```
1  using Carpinteria_Refactorizado.dominio;
2  using System.Data;
3
4  namespace Carpinteria_Refactorizado.accesoDatos
5  {
6     2 referencias
7     interface IPresupuestoDAO
8     {
9         // Todo lo que va a hacer contra la Base de Datos la clase de PresupuestoDAO
10
11         2 referencias
12         int ObtenerProximoNroPresupuesto(); // metodo que estaba en Frm_Alta_Presupuesto.cs
13         1 referencia
14         DataTable ListarProductos(); // metodo que estaba en Frm_Alta_Presupuesto.cs
15         1 referencia
16         bool Crear(Presupuesto oPresupuesto); // metodo que estaba en Presupuesto.cs
17     }
18 }
```

3) Crear PresupuestoDAO.cs en carpeta accesoDatos (va a contener los metodos encargados del presupuesto que utilizan la BD y contienen codigo SQL)

```
1 using Carpinteria_Refactorizado.dominio;
2 using System;
3 using System.Collections.Generic;
4 using System.Data;
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace Carpinteria_Refactorizado.accesoDatos
10 {
11     0 referencias
12     class PresupuestoDAO : IPresupuestoDAO
13     {
14         // La interfaz asociada me obliga a implementar estos metodos
15         2 referencias
16         public int ObtenerProximoNroPresupuesto()
17         {
18             throw new NotImplementedException();
19         }
20
21         1 referencia
22         public DataTable ListarProductos()
23         {
24             throw new NotImplementedException();
25         }
26
27         1 referencia
28         public bool Crear(Presupuesto oPresupuesto)
29         {
30             throw new NotImplementedException();
31         }
32     }
33 }
```

APLICACIÓN DE ESTO SACANDO EL PROXIMO_PRESUOPUESTO DESDE FORM

1)

```
1 using Carpinteria_Refactorizado.accesoDatos;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace Carpinteria_Refactorizado.servicios
9 {
10     2 referencias
11     class GestorPresupuesto
12     {
13         private IPresupuestoDAO dao;
14
15         1 referencia
16         public int ProximoPresupuesto()
17         {
18             return dao.ObtenerProximoNroPresupuesto();
19         }
20     }
```

2)

```
1 using Carpinteria_Refactorizado.dominio;
2 using System.Data;
3
4 namespace Carpinteria_Refactorizado.accesoDatos
5 {
6     2 referencias
7     interface IPresupuestoDAO
8     {
9         // Todo lo que va a hacer contra la Base de Datos la clase de PresupuestoDAO
10
11         2 referencias
12         int ObtenerProximoNroPresupuesto(); // metodo que estaba en Frm_Alta_Presupuesto.cs
13         1 referencia
14         DataTable ListarProductos(); // metodo que estaba en Frm_Alta_Presupuesto.cs
15         1 referencia
16         bool Crear(Presupuesto oPresupuesto); // metodo que estaba en Presupuesto.cs
17     }
18 }
```

3)

```
10 namespace Carpinteria_Refactorizado.accesoDatos
11 {
12     0 referencias
13     class PresupuestoDAO : IPresupuestoDAO
14     {
15         // La interfaz asociada me obliga a implementar estos metodos
16         2 referencias
17         public int ObtenerProximoNroPresupuesto()
18         {
19             SqlConnection cnn = new SqlConnection(@"Data Source=LAPTOP-8EMMHC7Q;Initial Catalog=carpinteria_db;Integrated Security=True");
20             cnn.Open();
21
22             // Command proximo ID
23             SqlCommand cmd = new SqlCommand();
24             cmd.Connection = cnn;
25
26             // Command Type para el Tipo de Comando que quiero ejecutar
27             // cmd.CommandText = CommandType.Text; ejecutamos sql como texto plano
28             cmd.CommandType = CommandType.StoredProcedure;
29             cmd.CommandText = "SP_PROXIMO_ID";
30
31             SqlParameter param = new SqlParameter();
32             param.ParameterName = "@next";
33             param.SqlDbType = SqlDbType.Int;
34             param.Direction = ParameterDirection.Output;
35
36             cmd.Parameters.Add(param);
37             cmd.ExecuteNonQuery(); // no estoy esperando que el SP me devuelva un SELECT
38             cnn.Close();
39
40             return (int)param.Value;
41         }
42     }
```

4) Patron Factory para quitar new GestorPresupuesto()

```
16 public partial class Frm_Alta_Presupuesto : Form
17 {
18     private Presupuesto oPresupuesto;
19     private GestorPresupuesto gestor;
20     private bool banderaUpdate = false;
21     2 referencias
22     public Frm_Alta_Presupuesto()
23     {
24         InitializeComponent();
25
26         consultarUltimoPresupuesto();
27         cargarCombo();
28
29         // Valores por defecto
30         txtFecha.Text = DateTime.Now.ToString("dd/MM/yyyy");
31         txtCliente.Text = "CONSUMIDOR FINAL";
32         txtDescuento.Text = "0";
33         cboProducto.DropDownStyle = ComboBoxStyle.DropDownList;
34
35         // Crear un nuevo Objeto Presupuesto
36         oPresupuesto = new Presupuesto();
37         gestor = new GestorPresupuesto();
38     }
39     1 referencia
40     private void cargarCombo()...
41
42     1 referencia
43     private void consultarUltimoPresupuesto()
44     {
45         lblNro.Text = "Presupuesto Nro: " + gestor.ProximoPresupuesto();
46     }
47 }
```

5) Creamos patron factory dentro de carpeta accesoDatos

Primero)

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Carpinteria_Refactorizado.accesoDatos
8 {
9     1 referencia
10     abstract class AbstractDAOFactory
11     {
12         1 referencia
13         public abstract IPresupuestoDAO CrearPresupuestoDAO();
14     }
15 }
```

Segundo)

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Carpinteria_Refactorizado.accesoDatos
8  {
9      0 referencias
10     class DAOFactory : AbstractDAOFactory
11     {
12         1 referencia
13         public override IPresupuestoDAO CrearPresupuestoDAO()
14         {
15             return new PresupuestoDAO();
16         }
17     }
```

Tercero) Tengo un gestor de presupuesto que tiene un constructor, ese constructor tiene un factory y ese factory lo que hace es crear el presupuesto.

```
1  using Carpinteria_Refactorizado.accesoDatos;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Carpinteria_Refactorizado.servicios
9  {
10     3 referencias
11     class GestorPresupuesto
12     {
13         private IPresupuestoDAO dao;
14
15         1 referencia
16         public GestorPresupuesto(AbstractDAOFactory factory)
17         {
18             dao = factory.CrearPresupuestoDAO();
19         }
20
21         1 referencia
22         public int ProximoPresupuesto()
23         {
24             return dao.ObtenerProximoNroPresupuesto();
25         }
26     }
```

Cuarto)

```
15 namespace Carpinteria_Refactorizado.gui
16 {
17     6 referencias
18     public partial class Frm_Alta_Presupuesto : Form
19     {
20         private Presupuesto oPresupuesto;
21         private GestorPresupuesto gestor;
22         private bool banderaUpdate = false;
23         2 referencias
24         public Frm_Alta_Presupuesto()
25         {
26             InitializeComponent();
27
28             consultarUltimoPresupuesto();
29             cargarCombo();
30
31             // Valores por defecto
32             txtFecha.Text = DateTime.Now.ToString("dd/MM/yyyy");
33             txtCliente.Text = "CONSUMIDOR FINAL";
34             txtDescuento.Text = "0";
35             cboProducto.DropDownStyle = ComboBoxStyle.DropDownList;
36
37             // Crear un nuevo Objeto Presupuesto
38             oPresupuesto = new Presupuesto();
39             gestor = new GestorPresupuesto(new DAOFactory());
40
41             1 referencia
42             private void cargarCombo()...
43
44             1 referencia
45             private void consultarUltimoPresupuesto()
46             {
47                 lblNro.Text = "Presupuesto Nro: " + gestor.ProximoPresupuesto();
48             }
49         }
50     }
51 }
```

6) Ahora lo mismo con el método CargarProducto() del Form

```
IPresupuestoDAO.cs  PresupuestoDAO.cs  GestorPresupuesto.cs  AbstractDAOFactory.cs  DAOFactory.cs  Frm_Alta_Presupuesto.cs*
Carpinteria_Refactorizado
38 }
39
40 1 referencia
41 private void cargarCombo()
42 {
43
44     // tabla.Rows[0]; // cada fila que tenga va a ser un DataRow
45
46     cboProducto.DataSource = tabla;
47     cboProducto.DisplayMember = tabla.Columns[1].ColumnName; // n_producto
48     cboProducto.ValueMember = tabla.Columns[0].ColumnName; // id_producto
49
50 }
51 }
```

```
IPresupuestoDAO.cs PresupuestoDAO.cs x GestorPresupuesto.cs AbstractDAOFactory.cs DAOFactory.cs Frm_Alta_Presupuesto.cs* Presupuesto.cs
Carpinteria_Refactorizado
38 return (int)param.value;
39 }
40
41 1 referencia
42 public DataTable ListarProductos()
43 {
44     SqlConnection cnn = new SqlConnection(@"Data Source=LAPTOP-8EMNHC7Q;Initial Catalog=carpinteria_db;Integrated Security=True");
45     cnn.Open();
46
47     // Command Productos
48     SqlCommand cmd2 = new SqlCommand("SP CONSULTAR_PRODUCTOS", cnn);
49     cmd2.CommandType = CommandType.StoredProcedure;
50
51     DataTable tabla = new DataTable();
52     tabla.Load(cmd2.ExecuteReader());
53
54     cnn.Close();
55
56     return tabla;
57 }
```

```
IPresupuestoDAO.cs PresupuestoDAO.cs x GestorPresupuesto.cs x AbstractDAOFactory.cs DAOFactory.cs Frm_Alta_Presupuesto.cs* Presupuesto.cs
Carpinteria_Refactorizado
1 using Carpinteria_Refactorizado.accesoDatos;
2 using System;
3 using System.Collections.Generic;
4 using System.Data;
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace Carpinteria_Refactorizado.servicios
10 {
11     3 referencias
12     class GestorPresupuesto
13     {
14         private IPresupuestoDAO dao;
15
16         1 referencia
17         public GestorPresupuesto(AbstractDAOFactory factory)
18         {
19             dao = factory.CrearPresupuestoDAO();
20
21         1 referencia
22         public int ProximoPresupuesto()
23         {
24             return dao.ObtenerProximoNroPresupuesto();
25         }
26
27         0 referencias
28         public DataTable ObtenerProductos()
29         {
30             return dao.ListarProductos();
31         }
32     }
33 }
```

```
PresupuestoDAO.cs GestorPresupuesto.cs AbstractDAOFactory.cs DAOFactory.cs Frm_Alta_Presupuesto.cs* x IPresupuestoDAO.cs Presupuesto.cs
Carpinteria_Refactorizado
39
40 1 referencia
41 private void cargarCombo()
42 {
43     DataTable tabla = gestor.ObtenerProductos();
44
45     // tabla.Rows[0]; // cada fila que tenga va a ser un DataRow
46
47     cboProducto.DataSource = tabla;
48     cboProducto.DisplayMember = tabla.Columns[1].ColumnName; // n_producto
49     cboProducto.ValueMember = tabla.Columns[0].ColumnName; // id_producto
50 }
```

7) Ahora con el botón de Aceptar que el confirmar se encuentra en la clase Presupuesto

```
GestorPresupuesto.cs  AbstractDAOFactory.cs  DAOFactory.cs  Frm_Alta_Presupue
Carpinteria_Refactorizado
1  using Carpinteria_Refactorizado.accesoDatos;
2  using Carpinteria_Refactorizado.dominio;
3  using System;
4  using System.Collections.Generic;
5  using System.Data;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Carpinteria_Refactorizado.servicios
11 {
12     3 referencias
13     class GestorPresupuesto
14     {
15         private IPresupuestoDAO dao;
16
17         1 referencia
18         public GestorPresupuesto(AbstractDAOFactory factory)
19         {
20             dao = factory.CrearPresupuestoDAO();
21         }
22
23         1 referencia
24         public int ProximoPresupuesto()
25         {
26             return dao.ObtenerProximoNroPresupuesto();
27         }
28
29         1 referencia
30         public DataTable ObtenerProductos()
31         {
32             return dao.ListarProductos();
33         }
34
35         0 referencias
36         public bool ConfirmarPresupuesto(Presupuesto oPresupuesto)
37         {
38             return dao.Crear(oPresupuesto);
39         }
40     }
41 }
```



```

GestorPresupuesto.cs  AbstractDAOFactory.cs  DAOFactory.cs  Frm_Alta_Presupuesto.cs  PresupuestoDAO.cs  IPresupuestoDAO.cs  Presupuesto.cs*
Carpinteria_Refactorizado  Carpinteria_Refactorizado.accesoDatos.PresupuestoDAO  Crear(Presupuesto oPresupuesto)

58 public bool Crear(Presupuesto oPresupuesto)
59 {
60     bool resultado = true;
61     SqlConnection cnn = new SqlConnection();
62     SqlTransaction trans = null;
63
64     try
65     {
66         cnn.ConnectionString = @"Data Source=LAPTOP-8EMNHC70;Initial Catalog=carpinteria_db;Integrated Security=True";
67         cnn.Open();
68         trans = cnn.BeginTransaction();
69
70         SqlCommand cmd = new SqlCommand("SP_INSERTAR_MAESTRO", cnn, trans);
71         cmd.CommandType = CommandType.StoredProcedure;
72         cmd.Parameters.AddWithValue("@cliente", oPresupuesto.Cliente);
73         cmd.Parameters.AddWithValue("@dto", oPresupuesto.Descuento);
74         cmd.Parameters.AddWithValue("@total", oPresupuesto.Total);
75
76         SqlParameter param = new SqlParameter("@presupuesto_nro", SqlDbType.Int);
77         param.Direction = ParameterDirection.Output;
78         cmd.Parameters.Add(param);
79         cmd.ExecuteNonQuery();
80         int presupuestoNro = Convert.ToInt32(param.Value);
81         int cDetalles = 1; // es el ID que forma de la PK doble entre IDPresupuesto E ID_DETALLE
82
83         foreach (DetallePresupuesto det in oPresupuesto.Detalles)
84         {
85             SqlCommand cmdDet = new SqlCommand("SP_INSERTAR_DETALLE", cnn);
86             cmdDet.CommandType = CommandType.StoredProcedure;
87             cmdDet.Transaction = trans;
88             cmdDet.Parameters.AddWithValue("@presupuesto_nro", presupuestoNro);
89             cmdDet.Parameters.AddWithValue("@detalle", cDetalles);
90             cmdDet.Parameters.AddWithValue("@id_producto", det.Producto.IdProducto);
91             cmdDet.Parameters.AddWithValue("@cantidad", det.Cantidad);
92             cmdDet.ExecuteNonQuery();
93             cDetalles++;
94         }
95
96         trans.Commit();
97     }
98     catch (Exception ex)

```

```

PresupuestoDAO.cs  GestorPresupuesto.cs  AbstractDAOFactory.cs  DAOFactory.cs  Frm_Alta_Presupuesto.cs  IPresupuestoDAO.cs  Presupuesto.cs
Carpinteria_Refactorizado  Carpinteria_Refactorizado.gui.Frm_Alta_Presupuesto  GuardarPresupuesto()

124 private void dgvDetalles_CellContentClick(object sender, DataGridViewCellEventArgs e)
125 {
126 }
127
128 private void btnAceptar_Click(object sender, EventArgs e)
129 {
130 }
131
132 private void GuardarPresupuesto()
133 {
134     oPresupuesto.Cliente = txtCliente.Text;
135     oPresupuesto.Descuento = Convert.ToDouble(txtDescuento.Text);
136     oPresupuesto.Fecha = Convert.ToDateTime(txtFecha.Text);
137     oPresupuesto.Total = Convert.ToDouble(txtTotal.Text);
138
139     if (banderaUpdate)
140     {
141         if (oPresupuesto.Actualizar())
142         {
143             MessageBox.Show("Presupuesto Actualizado", "Informe", MessageBoxButtons.OK, MessageBoxIcon.Information);
144             this.Dispose();
145         }
146         else
147         {
148             MessageBox.Show("ERROR. No se pudo registrar el presupuesto", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
149         }
150     }
151     else
152     {
153         if (gestor.ConfirmarPresupuesto(oPresupuesto))
154         {
155             MessageBox.Show("Presupuesto registrado", "Informe", MessageBoxButtons.OK, MessageBoxIcon.Information);
156             this.Dispose();
157         }
158         else
159         {
160             MessageBox.Show("ERROR. No se pudo registrar el presupuesto", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
161         }
162     }
163 }

```

8) Aplicar patron Singleton

En el Form solo quedan los métodos referentes a la Interfaz

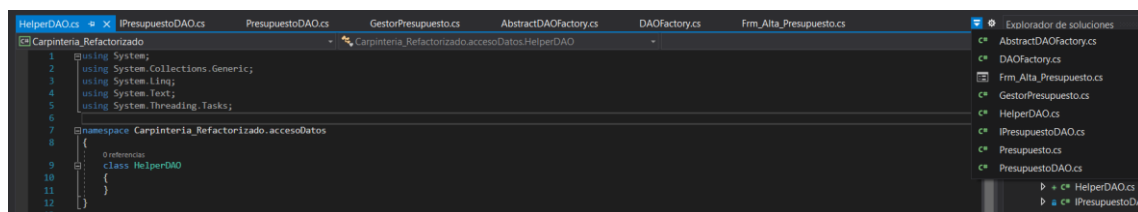
El Gestor se va a encargar de mandar a llamar a las funciones

El DAO repite lo de creación de conexión por lo tanto deberíamos crear una clase manipuladora que me permite tener siempre una instancia

Primero creamos una clase en accesoDatos a la cual la pueda llamar desde el DAO y no tener que siempre repetir lo de la conexión.

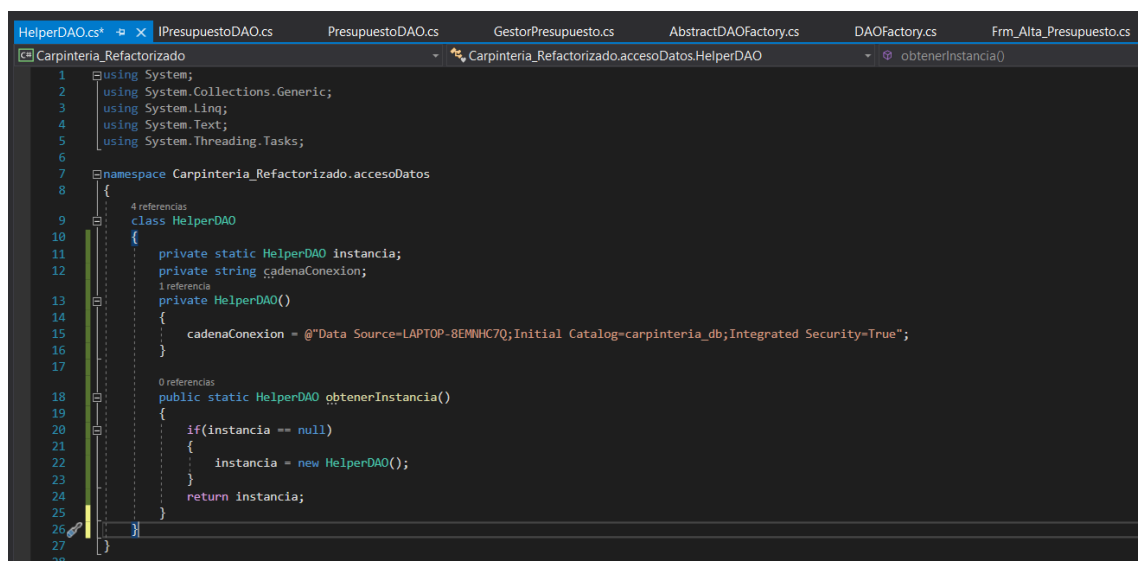
Primero) Crear helperDAO que seria un gestor de DAO o ayudante de DAO en carpeta accesoDatos

En HelperDAO.cs vamos a hacer toda la funcionalidad genérica que tenga que ver con los datos, quitar todo lo que se repite en PresupuestoDAO referido a la conexión a la BD.



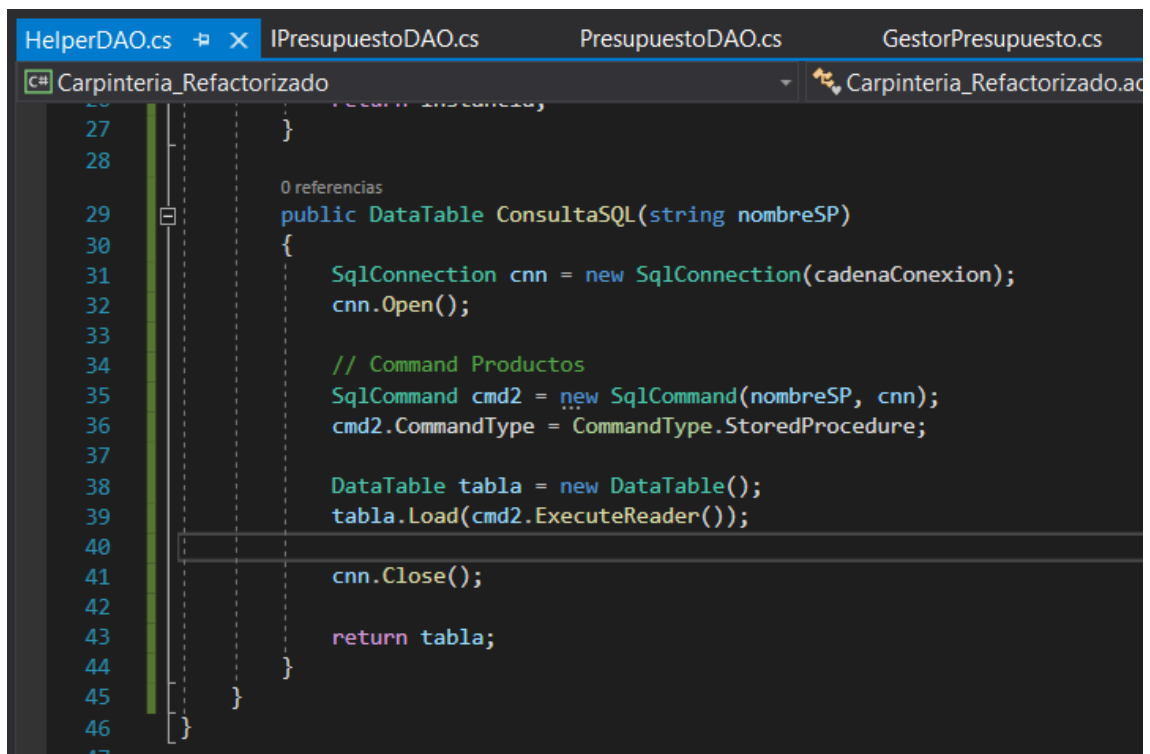
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Carpinteria_Refactorizado.accesoDatos
8 {
9     0 referencias
10     class HelperDAO
11     {
12     }
13 }
```

Segundo) Creamos el Patron Singleton para la instancia pero ahora tengo que generar a partir de ahí todos los métodos necesarios que me den la funcionalidad.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Carpinteria_Refactorizado.accesoDatos
8 {
9     4 referencias
10     class HelperDAO
11     {
12         private static HelperDAO instancia;
13         private string cadenaConexion;
14         1 referencia
15         private HelperDAO()
16         {
17             cadenaConexion = @"Data Source=LAPTOP-8EMWKC7Q;Initial Catalog=carpinteria_db;Integrated Security=True";
18         }
19         0 referencias
20         public static HelperDAO obtenerInstancia()
21         {
22             if(instancia == null)
23             {
24                 instancia = new HelperDAO();
25             }
26             return instancia;
27         }
28     }
```

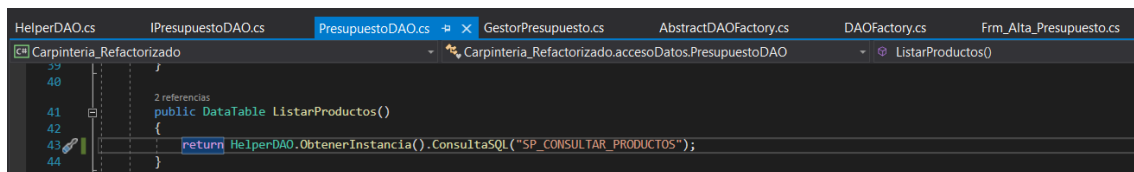
Tercero) Crear los métodos de las consultas repetitivas



The screenshot shows the Visual Studio IDE with the file `IPresupuestoDAO.cs` open. The code defines a method `ConsultarSQL` that takes a string `nombreSP` as a parameter and returns a `DataTable`. The method implementation is as follows:

```
27     }
28
29     0 referencias
30     public DataTable ConsultarSQL(string nombreSP)
31     {
32         SqlConnection cnn = new SqlConnection(cadenaConexion);
33         cnn.Open();
34
35         // Command Productos
36         SqlCommand cmd2 = new SqlCommand(nombreSP, cnn);
37         cmd2.CommandType = CommandType.StoredProcedure;
38
39         DataTable tabla = new DataTable();
40         tabla.Load(cmd2.ExecuteReader());
41
42         cnn.Close();
43
44         return tabla;
45     }
46 }
```

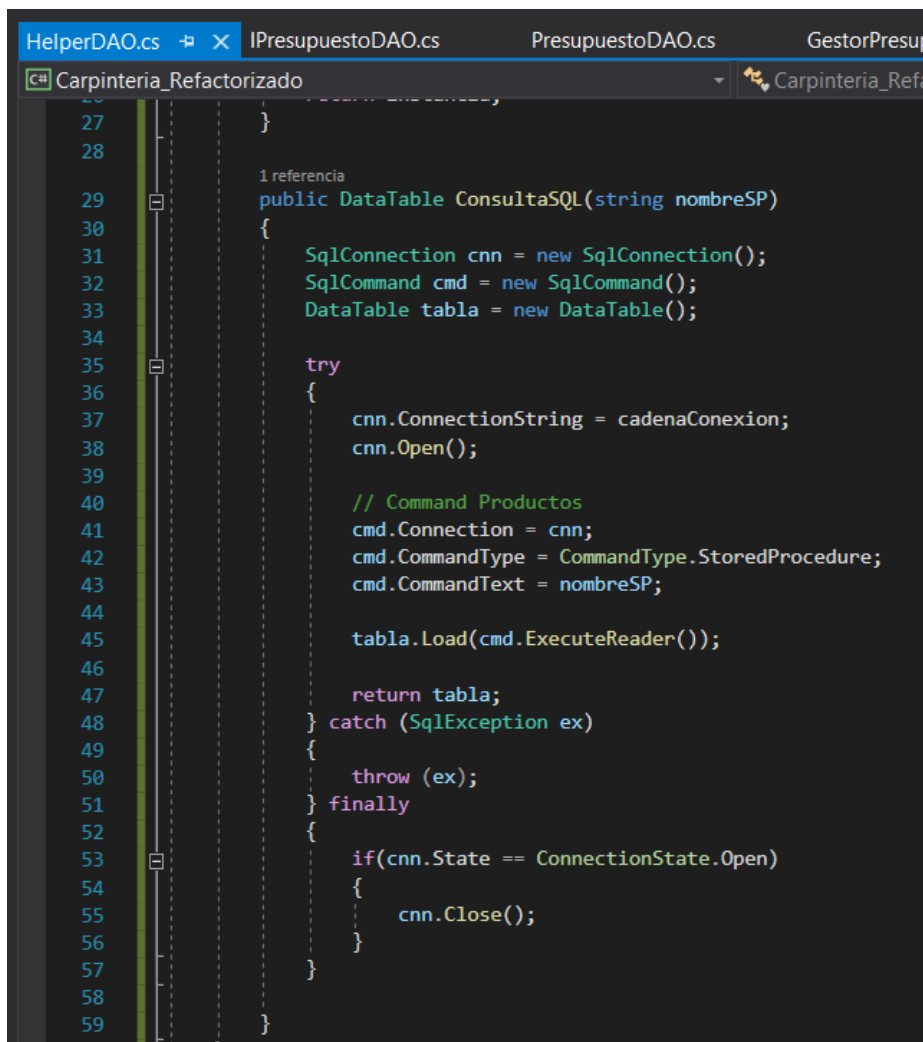
Cuarto) Ejecutar este método desde PresupuestoDAO



The screenshot shows the Visual Studio IDE with the file `PresupuestoDAO.cs` open. The code defines a method `ListarProductos` that returns a `DataTable`. The method implementation is as follows:

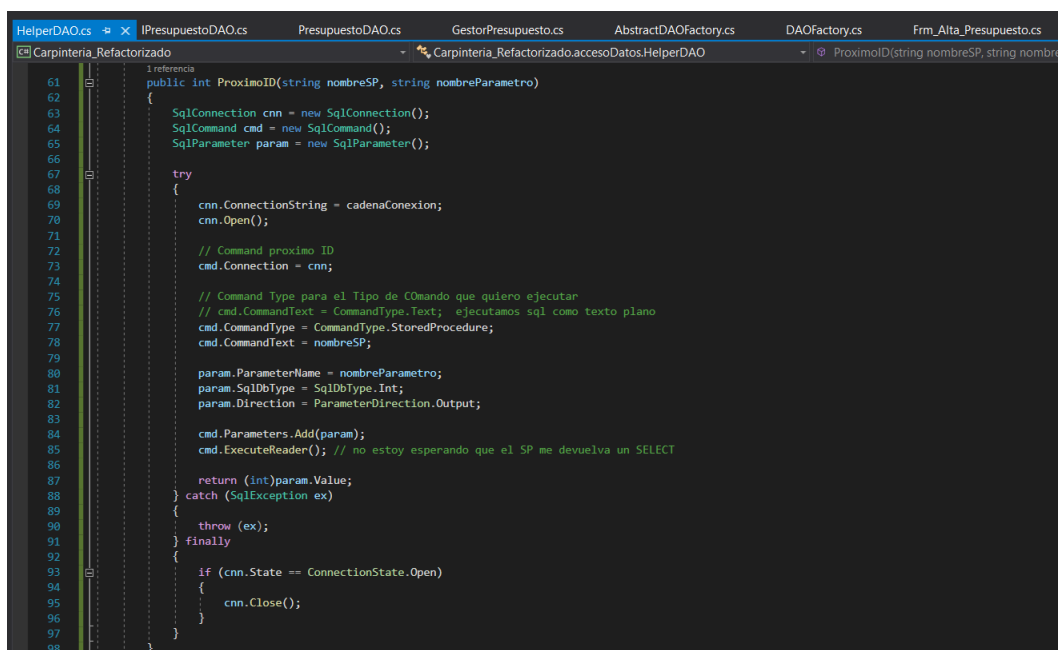
```
39     }
40
41     2 referencias
42     public DataTable ListarProductos()
43     {
44         return HelperDAO.ObtenerInstancia().ConsultarSQL("SP_CONSULTAR_PRODUCTOS");
45     }
46 }
```

Quinto) Optimizar método de ConsultaSQL en HelperDAO



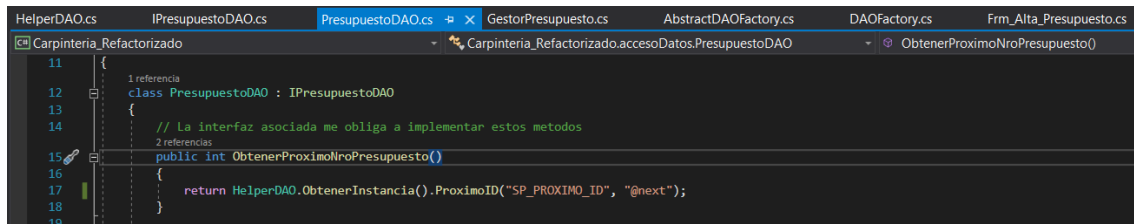
```
27 }
28
29 1 referencia
30 public DataTable ConsultaSQL(string nombreSP)
31 {
32     SqlConnection cnn = new SqlConnection();
33     SqlCommand cmd = new SqlCommand();
34     DataTable tabla = new DataTable();
35
36     try
37     {
38         cnn.ConnectionString = cadenaConexion;
39         cnn.Open();
40
41         // Command Productos
42         cmd.Connection = cnn;
43         cmd.CommandType = CommandType.StoredProcedure;
44         cmd.CommandText = nombreSP;
45
46         tabla.Load(cmd.ExecuteReader());
47
48         return tabla;
49     } catch (SqlException ex)
50     {
51         throw (ex);
52     } finally
53     {
54         if(cnn.State == ConnectionState.Open)
55         {
56             cnn.Close();
57         }
58     }
59 }
```

Sexto) Crear método en HelperDAO para obtener el próximo ID



```
61 1 referencia
62 public int ProximoID(string nombreSP, string nombreParametro)
63 {
64     SqlConnection cnn = new SqlConnection();
65     SqlCommand cmd = new SqlCommand();
66     SqlParameter param = new SqlParameter();
67
68     try
69     {
70         cnn.ConnectionString = cadenaConexion;
71         cnn.Open();
72
73         // Command proximo ID
74         cmd.Connection = cnn;
75
76         // Command Type para el Tipo de Comando que quiero ejecutar
77         // cmd.CommandText = CommandType.Text; ejecutamos sql como texto plano
78         cmd.CommandType = CommandType.StoredProcedure;
79         cmd.CommandText = nombreSP;
80
81         param.ParameterName = nombreParametro;
82         param.SqlDbType = SqlDbType.Int;
83         param.Direction = ParameterDirection.Output;
84
85         cmd.Parameters.Add(param);
86         cmd.ExecuteReader(); // no estoy esperando que el SP me devuelva un SELECT
87
88         return (int)param.Value;
89     } catch (SqlException ex)
90     {
91         throw (ex);
92     } finally
93     {
94         if (cnn.State == ConnectionState.Open)
95         {
96             cnn.Close();
97         }
98     }
99 }
```

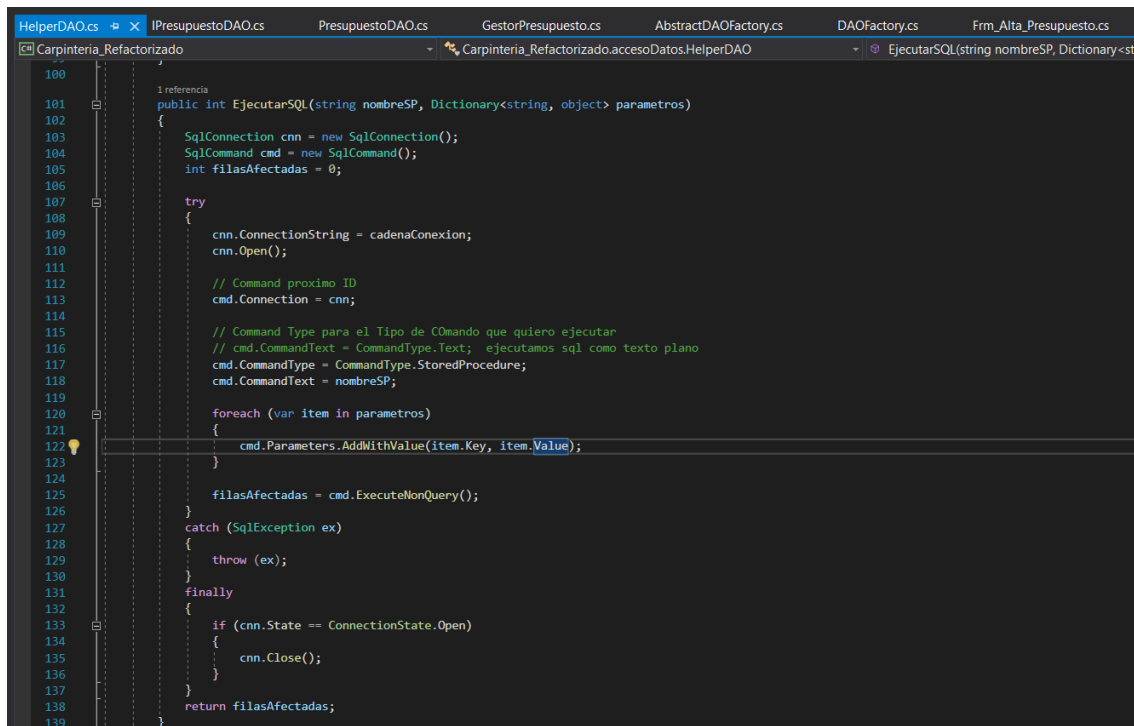
Séptimo) Refactorizar método de obtención de próximo ID en PresupuestoDAO



```
HelperDAO.cs | IPresupuestoDAO.cs | PresupuestoDAO.cs | GestorPresupuesto.cs | AbstractDAOFactory.cs | DAOFactory.cs | Frm_Alta_Presupuesto.cs
Carpinteria_Refactorizado | Carpinteria_Refactorizado | Carpinteria_Refactorizado.accesoDatos.PresupuestoDAO | ObtenerProximoNroPresupuesto()

11
12 1 referencia
13 class PresupuestoDAO : IPresupuestoDAO
14 {
15     // La interfaz asociada me obliga a implementar estos metodos
16     // 2 referencias
17     public int ObtenerProximoNroPresupuesto()
18     {
19         return HelperDAO.ObtenerInstancia().ProximoID("SP_PROXIMO_ID", "@next");
20     }
21 }
```

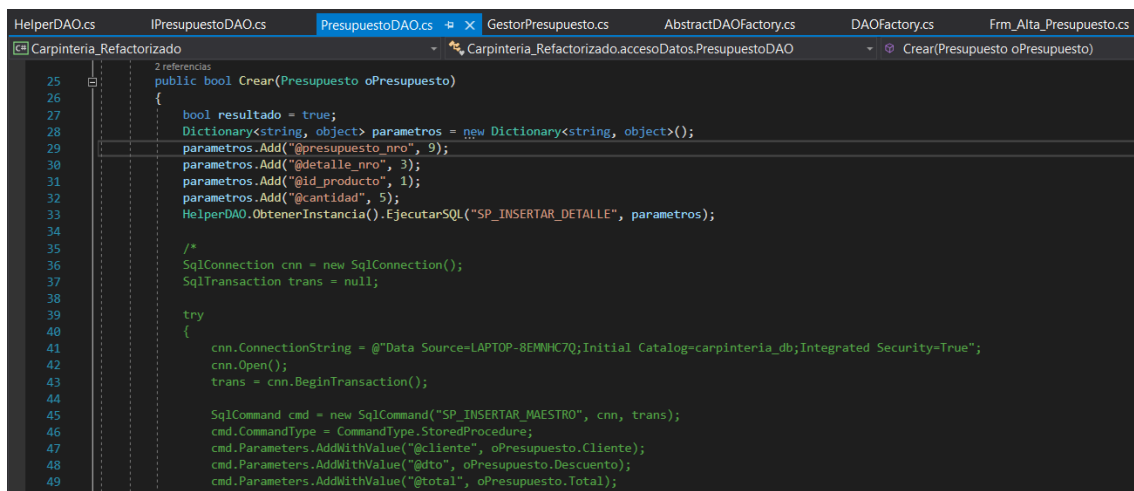
Octavo) Crear método en HelperDAO para Crear nuevo Presupuesto (INCOMPLETO)



```
HelperDAO.cs | IPresupuestoDAO.cs | PresupuestoDAO.cs | GestorPresupuesto.cs | AbstractDAOFactory.cs | DAOFactory.cs | Frm_Alta_Presupuesto.cs
Carpinteria_Refactorizado | Carpinteria_Refactorizado | Carpinteria_Refactorizado.accesoDatos.HelperDAO | EjecutarSQL(string nombreSP, Dictionary<string, object> parametros)

100
101 1 referencia
102 public int EjecutarSQL(string nombreSP, Dictionary<string, object> parametros)
103 {
104     SqlConnection cnn = new SqlConnection();
105     SqlCommand cmd = new SqlCommand();
106     int filasAfectadas = 0;
107
108     try
109     {
110         cnn.ConnectionString = cadenaConexion;
111         cnn.Open();
112
113         // Command proximo ID
114         cmd.Connection = cnn;
115
116         // Command Type para el Tipo de Comando que quiero ejecutar
117         // cmd.CommandText = CommandType.Text; ejecutamos sql como texto plano
118         cmd.CommandType = CommandType.StoredProcedure;
119         cmd.CommandText = nombreSP;
120
121         foreach (var item in parametros)
122         {
123             cmd.Parameters.AddWithValue(item.Key, item.Value);
124         }
125
126         filasAfectadas = cmd.ExecuteNonQuery();
127     }
128     catch (SqlException ex)
129     {
130         throw (ex);
131     }
132     finally
133     {
134         if (cnn.State == ConnectionState.Open)
135         {
136             cnn.Close();
137         }
138     }
139     return filasAfectadas;
140 }
```

Noveno) Refactorizar método de obtención de próximo ID en PresupuestoDAO (INCOMPLETO)



```
HelperDAO.cs | IPresupuestoDAO.cs | PresupuestoDAO.cs | GestorPresupuesto.cs | AbstractDAOFactory.cs | DAOFactory.cs | Frm_Alta_Presupuesto.cs
Carpinteria_Refactorizado | Carpinteria_Refactorizado | Carpinteria_Refactorizado.accesoDatos.PresupuestoDAO | Crear(Presupuesto oPresupuesto)

25 2 referencias
26 public bool Crear(Presupuesto oPresupuesto)
27 {
28     bool resultado = true;
29     Dictionary<string, object> parametros = new Dictionary<string, object>();
30     parametros.Add("@presupuesto_nro", 9);
31     parametros.Add("@detalle_nro", 3);
32     parametros.Add("@id_producto", 1);
33     parametros.Add("@cantidad", 5);
34     HelperDAO.ObtenerInstancia().EjecutarSQL("SP_INSERTAR_DETALLE", parametros);
35
36     /*
37     SqlConnection cnn = new SqlConnection();
38     SqlTransaction trans = null;
39
40     try
41     {
42         cnn.ConnectionString = @"Data Source=LAPTOP-8EMWHC7Q;Initial Catalog=carpinteria_db;Integrated Security=True";
43         cnn.Open();
44         trans = cnn.BeginTransaction();
45
46         SqlCommand cmd = new SqlCommand("SP_INSERTAR_MAESTRO", cnn, trans);
47         cmd.CommandType = CommandType.StoredProcedure;
48         cmd.Parameters.AddWithValue("@cliente", oPresupuesto.Cliente);
49         cmd.Parameters.AddWithValue("@dto", oPresupuesto.Descuento);
50         cmd.Parameters.AddWithValue("@total", oPresupuesto.Total);
51     }
52     catch (Exception ex)
53     {
54         resultado = false;
55     }
56     finally
57     {
58         if (trans != null) trans.Dispose();
59         if (cnn.State == ConnectionState.Open) cnn.Close();
60     }
61     return resultado;
62 }
```

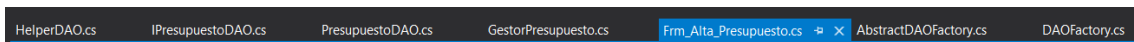
COSAS PARA HACER:

1. Terminar lo que no COMPLETO BOTTA (NO FUNCIONA)
2. Refactorizar presupuesto actualizar
3. Agregar los ENUM para la creación de formularios como POLLIOTTO
4. Mostrar detalle desde la misma pantalla
5. Armar Editar como Botta (2:02:11)
6. Btn_editar
7. Update en frm_alta_presupuesto

Teniendo todo esto:

1. Luego de tener el Update y la Parte de Botta pasar todo a HelperDAO.
2. Dividir y organizar lo que falta

ORDEN:



PASOS

1. Frm_Alta_Presupuesto
2. GestorPresupuesto
3. IPresupuestoDAO
4. PresupuestoDAO
5. AbstractDAOFactory
6. DAOFactory
7. En GestorPresupuesto aplico el factory
8. Llamo el factory desde Frm_Alta_Presupuesto en gestor
9. HelperDao para el Patron Singleton

Para armar carga datos UPDATE:

1. Frm_alta_presupuesto tener Presupuesto oPresupuesto
2. En Acción.UPDATE llamar función cargarPresupuesto con idPresupuesto
3. En cargarPresupuesto dentro de formPrincipal llamar a this.oPresupuesto para que dentro se almacene el resultado de Presupuesto obtenerPresupuestoPorId(int nro) que estará en IPresupuestoDAO
4. En PresupuestoDAO creo una función public Presupuesto ObtenerPresupuestoPorId(int nro) que me retorne dao.GetById(nro) desde el dao que es el GestorPresupuesto.
5. Creo una función GetById(int id) en HelperDAO que me devuelva el Presupuesto Completo Junto a sus Detalles con el Presupuesto Cargado.

Pasos en mi Proyecto

1. En frm_alta_presupuesto private Presupuesto oPresupuesto;
2. En Acción.UPDATE llamar función CargarPresupuesto(int nro_presupuesto)
3. CargarPresupuesto(int nro_presupuesto) llama al GestorPresupuesto que con la función ObtenerPresupuestoPorID(nro) llena el oPresupuesto mediante this.oPresupuesto = servicio.ObtenerPresupuestoPorID(nro); donde servicio es la instancia de GestorPresupuesto.
4. Creo la función ObtenerPresupuestoPorID(nro) en GestorPresupuesto retornando el Dao
5. En IPresupuestoDAO creo: public Presupuesto ObtenerPresupuestoPorID(int nro) para luego implementar el método en PresupuestoDAO
6. En PresupuestoDAO creo la función GetByID(int id) que me rellena tanto el oPresupuesto como el oDetalles y me los devuelve.
7. Defino el SQL en HelperDAO y luego me devuelve el objeto Presupuesto y lo voy pasando.

PASOS OBTENER PRESUPUESTOS LISTA

1. En Frm_Alta_Presupuesto

- a.

```
private GestorPresupuesto gestor;
```
- b.

```
if (modo.Equals(Accion.UPDATE))
{
    banderaUpdate = true;
    this.Text = "EDITAR PRESUPUESTO";
    CargarPresupuesto(nro_presupuesto);
}
```
- c.

```
private void CargarPresupuesto(int idPresupuesto)
{
    this.oPresupuesto = gestor.ObtenerPresupuestoPorID(idPresupuesto);
    txtCliente.Text = oPresupuesto.Cliente;
    txtFecha.Text = oPresupuesto.Fecha.ToString("dd/MM/yyyy");
    txtDescuento.Text = oPresupuesto.Descuento.ToString();
    lblNro.Text = "Presupuesto Nro: " + oPresupuesto.PresupuestoNro.ToString();

    dgvDetalles.Rows.Clear();
    foreach (DetallePresupuesto oDetalle in oPresupuesto.Detalles)
    {
        dgvDetalles.Rows.Add(new object[] { "", oDetalle.Producto.Nombre, oDetalle.Producto.Precio, oDetalle.Cantidad, oDetalle.CalcularSubtotal() });
    }
    calcularTotales();
}
```
- d.

```
gestor.ObtenerPresupuestoPorID(idPresupuesto);
```

2. En GestorPresupuesto

- a.

```
public Presupuesto ObtenerPresupuestoPorID(int nro)
{
    return dao.ObtenerPresupuestoPorID(nro);
}
```

3. En IPresupuestoDAO

- a.

```
Presupuesto ObtenerPresupuestoPorID(int nro);
```

4. En PresupuestoDAO

- a.

```
public Presupuesto ObtenerPresupuestoPorID(int id)
{
    return HelperDAO.ObtenerInstancia().GetById(id);
}
```

5. En HelperDAO


```

public Presupuesto GetById(int id)
{
    Presupuesto oPresupuesto = new Presupuesto();
    SqlConnection cnn = new SqlConnection();
    cnn.ConnectionString = cadenaConexion;
    cnn.Open();
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = cnn;
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.CommandText = "SP_CONSULTAR_PRESUPUESTO_POR_ID";
    cmd.Parameters.AddWithValue("@nro", id);
    SqlDataReader reader = cmd.ExecuteReader();
    bool esPrimerRegistro = true;

    while (reader.Read())
    {
        if (esPrimerRegistro)
        {
            //solo para el primer resultado recuperamos los datos del MAESTRO:
            oPresupuesto.PresupuestoNro = Convert.ToInt32(reader["presupuesto_nro"].ToString());
            oPresupuesto.Cliente = reader["cliente"].ToString();
            oPresupuesto.Fecha = Convert.ToDateTime(reader["fecha"].ToString());
            oPresupuesto.Descuento = Convert.ToDouble(reader["descuento"].ToString());
            oPresupuesto.PresupuestoNro = Convert.ToInt32(reader["presupuesto_nro"].ToString());
            oPresupuesto.Total = Convert.ToDouble(reader["total"].ToString());
            esPrimerRegistro = false;
        }

        DetallePresupuesto oDetalle = new DetallePresupuesto();
        Producto oProducto = new Producto();
        oProducto.IdProducto = Convert.ToInt32(reader["id_producto"].ToString());
        oProducto.Nombre = reader["n_producto"].ToString();
        oProducto.Precio = Convert.ToDouble(reader["precio"].ToString());
        oProducto.Activo = reader["activo"].ToString().Equals("S");
        oDetalle.Producto = oProducto;
        oDetalle.Cantidad = Convert.ToInt32(reader["cantidad"].ToString());
        esPrimerRegistro = false;
        oPresupuesto.AgregarDetalle(oDetalle);
    }
    return oPresupuesto;
}

```

a.

PASOS CONSULTAR PRESUPUESTO CON SU DETALLE

1. En Frm_Consultar_Presupuesto

```

public partial class Frm_Consultar_Presupuestos : Form
{
    private Presupuesto oPresupuesto;
    private GestorPresupuesto gestor;

    1 referencia
    public Frm_Consultar_Presupuestos()
    {
        InitializeComponent();

        oPresupuesto = new Presupuesto();
        gestor = new GestorPresupuesto(new DAOFactory());
    }
}

```

a.

```
private void btnConsultar_Click(object sender, EventArgs e)
{
    dgvResultados.Rows.Clear();
    Validar();
}

2 referencias
private void Validar()
{
    if (dtpFechaDesde.Value.Date >= dtpFechaHasta.Value.Date)
    {
        MessageBox.Show("Ingrese Fechas Validas");
        return;
    }

    DateTime fechaDesde = dtpFechaDesde.Value.Date;
    DateTime fechaHasta = dtpFechaHasta.Value.Date;
    // MessageBox.Show(fechaDesde.ToString());
    string cliente = txtCliente.Text.Trim();
    string baja;
    baja = chkBaja.Checked == true ? "S" : "N";

    List<Parametro> filtros = new List<Parametro>();

    Parametro fecha_desde = new Parametro();
    fecha_desde.Nombre = "@fecha_desde";
    fecha_desde.Value = fechaDesde;
    filtros.Add(fecha_desde);
    filtros.Add(new Parametro("@fecha_hasta", fechaHasta));

    filtros.Add(new Parametro("@cliente", cliente));

    filtros.Add(new Parametro("@datos_baja", baja));

    ConsultarPresupuesto(filtros);
}
```

b.

```
private void ConsultarPresupuesto(List<Parametro> filtros)
{
    List<Presupuesto> lst = gestor.ConsultarPresupuestos(filtros);

    dgvResultados.Rows.Clear();
    foreach (Presupuesto oPresupuesto in lst)
    {
        dgvResultados.Rows.Add(new object[] {
            oPresupuesto.PresupuestoNro,
            oPresupuesto.Fecha.ToString("dd/MM/yyyy"),
            oPresupuesto.Cliente,
            oPresupuesto.Descuento,
            oPresupuesto.Total,
            oPresupuesto.GetFechaBajaFormato()
        });
    }
}
```

c.

2. En Parametro

```

4 referencias
class Parametro
{
    0 referencias
    public Parametro()
    {
    }

    0 referencias
    public Parametro(string nombre, object valor)
    {
        Nombre = nombre;
        Valor = valor;
    }

    1 referencia
    public string Nombre { get; set; }
    1 referencia
    public object Valor { get; set; }
}

```

a.

3. En GestorPresupuesto

```

1 referencia
public List<Presupuesto> ConsultarPresupuestos(List<Parametro> filtros)
{
    return dao.ConsultarPresupuestos(filtros);
}

```

a.

4. En IPresupuestoDAO

```

2 referencias
List<Presupuesto> ConsultarPresupuestos(List<Parametro> criterios);

```

a.

5. En PresupuestoDAO

```

2 referencias
public List<Presupuesto> ConsultarPresupuestos(List<Parametro> criterios)
{
    return HelperDAO.ObtenerInstancia().GetByFilters(criterios);
}

```

a.

6. En HelperDAO

```

public List<Presupuesto> GetByFilters(List<Parametro> criterios)
{
    List<Presupuesto> lst = new List<Presupuesto>();
    DataTable table = new DataTable();
    SqlConnection cnn = new SqlConnection();
    cnn.ConnectionString = cadenaConexion;

    try
    {
        cnn.Open();

        SqlCommand cmd = new SqlCommand("SP_CONSULTAR_PRESUPUESTOS", cnn);
        cmd.CommandType = CommandType.StoredProcedure;
        foreach (Parametro p in criterios)
        {
            cmd.Parameters.AddWithValue(p.Nombre, p.Valor);
        }

        table.Load(cmd.ExecuteReader());
        //mapear los registros como objetos del dominio:

        foreach (DataRow row in table.Rows)
        {
            //Por cada registro creamos un objeto del dominio
            Presupuesto oPresupuesto = new Presupuesto();
            oPresupuesto.Cliente = row["cliente"].ToString();
            oPresupuesto.Fecha = Convert.ToDateTime(row["fecha"].ToString());
            oPresupuesto.Descuento = Convert.ToDouble(row["descuento"].ToString());
            oPresupuesto.PresupuestoNro = Convert.ToInt32(row["presupuesto_nro"].ToString());
            oPresupuesto.Total = Convert.ToDouble(row["total"].ToString());
            //validar que fecha_baja no es null:
            if (!row["fecha_baja"].Equals(DBNull.Value))
                oPresupuesto.FechaBaja = Convert.ToDateTime(row["fecha_baja"].ToString());

            lst.Add(oPresupuesto);
        }

        cnn.Close();
    }
    catch (SqlException)
    {
        lst = null;
    }
}

```

a.

```

catch (SqlException)
{
    lst = null;
}

```

b.

```

return lst;

```

PASOS ELIMINAR PRESUPUESTO

1. En Frm_Consular_Presupuestos

- a.
- ```

4 referencias
public partial class Frm_Consultar_Presupuestos : Form
{
 private Presupuesto oPresupuesto;
 private GestorPresupuesto gestor;

 1 referencia
 public Frm_Consultar_Presupuestos()
 {
 InitializeComponent();

 oPresupuesto = new Presupuesto();
 gestor = new GestorPresupuesto(new DAOFactory());
 }

```
- b.
- ```

private void btnEliminar_Click(object sender, EventArgs e)
{
    int idPresupuesto = Int32.Parse(dgvResultados.CurrentRow.Cells["colNro"].Value.ToString());

    DialogResult result = MessageBox.Show("Esta seguro que desea dar de baja?", "Baja de Presupuesto", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
    if (result == DialogResult.Yes)
    {
        if (eliminarPresupuesto(idPresupuesto))
        {
            // MessageBox.Show(idPresupuesto.ToString());
            // dgvResultados.Rows.Remove(dgvResultados.CurrentRow);
            MessageBox.Show("Presupuesto eliminado!", "Confirmación", MessageBoxButtons.OK, MessageBoxIcon.Information);
            this.btnConsultar_Click(null, null);
        }
        else
        {
            MessageBox.Show("Error al intentar borrar el presupuesto!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```
- c.
- ```

private bool eliminarPresupuesto(int idPresupuesto)
{
 bool resultado = gestor.RegistrarBajaPresupuesto(idPresupuesto);

 SqlConnection cnn = new SqlConnection();
 SqlTransaction trans = null;

 // Realizar una baja logica mediante la fecha_baja SQL SP y agregar transaccion
 try
 {
 cnn.ConnectionString = @"Data Source=LAPTOP-8EMNHC7Q;Initial Catalog=carpinteria_db;Integrated Security=True";
 cnn.Open();
 trans = cnn.BeginTransaction();

 SqlCommand cmdPre = new SqlCommand("SP_ELIMINAR_PRESUPUESTO", cnn, trans);
 cmdPre.CommandType = CommandType.StoredProcedure;
 cmdPre.Transaction = trans;
 cmdPre.Parameters.AddWithValue("@presupuesto_nro", idPresupuesto);
 cmdPre.ExecuteNonQuery();

 trans.Commit();
 }
 catch (Exception ex)
 {
 trans.Rollback();
 MessageBox.Show(ex.ToString());
 resultado = false;
 }
 finally
 {
 if (cnn != null && cnn.State == ConnectionState.Open)
 {
 cnn.Close();
 }
 }

 return resultado;
}

```

## 2. En GestorPresupuesto

```
public bool RegistrarBajaPresupuesto(int idPresupuesto)
{
 return dao.RegistrarBajaPresupuesto(idPresupuesto);
}
```

a.

3. En IPresupuestoDAO

```
bool RegistrarBajaPresupuesto(int idPresupuesto);
```

a.

4. En PresupuestoDAO

```
public bool RegistrarBajaPresupuesto(int idPresupuesto)
{
 return HelperDAO.ObtenerInstancia().Delete(idPresupuesto);
}
```

a.

5. En HelperDAO

```
public bool Delete(int idPresupuesto)
{
 SqlConnection cnn = new SqlConnection();
 cnn.ConnectionString = cadenaConexion;
 SqlTransaction t = null;
 int affected = 0;
 try
 {
 cnn.Open();
 t = cnn.BeginTransaction();
 SqlCommand cmd = new SqlCommand("SP_REGISTRAR_BAJA_PRESUPUESTOS", cnn, t);
 cmd.CommandType = CommandType.StoredProcedure;
 cmd.Parameters.AddWithValue("@presupuesto_nro", idPresupuesto);
 affected = cmd.ExecuteNonQuery();
 t.Commit();
 }
 catch (SqlException)
 {
 t.Rollback();
 }
 finally
 {
 if (cnn != null && cnn.State == ConnectionState.Open)
 cnn.Close();
 }

 return affected == 1;
}
```

a.

## PARA INSERTAR NUEVO PRESUPUESTO

### 1. En Frm\_Alta\_Presupuesto

- a.
- ```
public partial class Frm_Alta_Presupuesto : Form
{
    private Presupuesto oPresupuesto;
    private GestorPresupuesto gestor;
    private bool banderaUpdate = false;
    private Accion modo;

    3 referencias
    public Frm_Alta_Presupuesto(Accion modo, int nro_presupuesto)
    {
        InitializeComponent();

        // Crear un nuevo Objeto Presupuesto
        this.modo = modo;
        oPresupuesto = new Presupuesto();
        gestor = new GestorPresupuesto(new DAOFactory());
    }

    private void GuardarPresupuesto()
    {
        oPresupuesto.Cliente = txtCliente.Text;
        oPresupuesto.Descuento = Convert.ToDouble(txtDescuento.Text);
        oPresupuesto.Fecha = Convert.ToDateTime(txtFecha.Text);
        oPresupuesto.Total = Convert.ToDouble(txtTotal.Text);

        if (banderaUpdate)
        {
            if (oPresupuesto.Actualizar())
            {
                MessageBox.Show("Presupuesto Actualizado", "Informe", MessageBoxButtons.OK, MessageBoxIcon.Information);
                this.Dispose();
            }
            else
            {
                MessageBox.Show("ERROR. No se pudo registrar el presupuesto", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        else
        {
            if (gestor.ConfirmarPresupuesto(oPresupuesto))
            {
                MessageBox.Show("Presupuesto registrado", "Informe", MessageBoxButtons.OK, MessageBoxIcon.Information);
                this.Dispose();
            }
            else
            {
                MessageBox.Show("ERROR. No se pudo registrar el presupuesto", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```
- b.

2. En GestorPresupuesto

- a.
- ```
public bool ConfirmarPresupuesto(Presupuesto oPresupuesto)
{
 return dao.Crear(oPresupuesto);
}
```

### 3. En IPresupuestoDAO

- a.
- ```
2 referencias
bool RegistrarBajaPresupuesto(int idPresupuesto);
```

4. En PresupuestoDAO

```
public bool Crear(Presupuesto oPresupuesto)
{
    /*
    Dictionary<string, object> parametros = new Dictionary<string, object>();
    parametros.Add("@presupuesto_nro", 18);
    parametros.Add("@detalle_nro", 5);
    parametros.Add("@id_producto", 1);
    parametros.Add("@cantidad", 5);
    HelperDAO.ObtenerInstancia().EjecutarSQL("SP_INSERTAR_DETALLE", parametros);
    */

    return HelperDAO.ObtenerInstancia().Save(oPresupuesto);
}
```

a.

5. En HelperDAO

```
public bool Save(Presupuesto oPresupuesto)
{
    SqlConnection cnn = new SqlConnection();
    SqlTransaction trans = null;
    bool resultado = true;

    try
    {
        cnn.ConnectionString = cadenaConexion;
        cnn.Open();
        trans = cnn.BeginTransaction();

        SqlCommand cmd = new SqlCommand("SP_INSERTAR_MAESTRO", cnn, trans);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@cliente", oPresupuesto.Cliente);
        cmd.Parameters.AddWithValue("@dto", oPresupuesto.Descuento);
        cmd.Parameters.AddWithValue("@total", oPresupuesto.Total);

        SqlParameter param = new SqlParameter("@presupuesto_nro", SqlDbType.Int);
        param.Direction = ParameterDirection.Output;
        cmd.Parameters.Add(param);
        cmd.ExecuteNonQuery();
        int presupuestoNro = Convert.ToInt32(param.Value);
        int cDetalles = 1; // es el ID que forma de la PK doble entre ID_PRESUPUESTO E ID_DETALLE
        int filasAfectadas = 0;

        foreach (DetallePresupuesto det in oPresupuesto.Detalles)
        {
            SqlCommand cmdDet = new SqlCommand("SP_INSERTAR_DETALLE", cnn);
            cmdDet.CommandType = CommandType.StoredProcedure;
            cmdDet.Transaction = trans;
            cmdDet.Parameters.AddWithValue("@presupuesto_nro", presupuestoNro);
            cmdDet.Parameters.AddWithValue("@detalle", cDetalles);
            cmdDet.Parameters.AddWithValue("@id_producto", det.Producto.IdProducto);
            cmdDet.Parameters.AddWithValue("@cantidad", det.Cantidad);
            cmdDet.ExecuteNonQuery();

            cDetalles++;
        }
    }
}
```

a.


```
    }

    trans.Commit();
}
catch (Exception ex)
{
    trans.Rollback();
    resultado = false;
}
finally
{
    if (cnn != null && cnn.State == ConnectionState.Open)
    {
        cnn.Close();
    }
}

return resultado;
}
```

b.

PASOS PARA ACTUALIZAR PRESUPUESTO

1. En Frm_Alta_Presupuesto

a.

```
public enum Accion
{
    CREATE,
    READ,
    UPDATE,
    DELETE
}

8 referencias
public partial class Frm_Alta_Presupuesto : Form
{
    private Presupuesto oPresupuesto;
    private GestorPresupuesto gestor;
    private bool banderaUpdate = false;
    private Accion modo;

    3 referencias
    public Frm_Alta_Presupuesto(Accion modo, int nro_presupuesto)
    {
        InitializeComponent();

        // Crear un nuevo Objeto Presupuesto
        this.modo = modo;
        oPresupuesto = new Presupuesto();
        gestor = new GestorPresupuesto(new DAOFactory());
    }
}
```

b.

```
private void btnAceptar_Click(object sender, EventArgs e)
{
    if (txtCliente.Text == "")
    {
        MessageBox.Show("Debe ingresar un cliente!", "Control",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }

    if (dgvDetalles.Rows.Count == 0)
    {
        MessageBox.Show("Debe ingresar al menos detalle!",
            "Control", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }

    if (txtCliente.Text.Trim() == "")
    {
        MessageBox.Show("Debe ingresar un tipo de cliente", "Validaciones", MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtCliente.Focus();
        return;
    }

    if (txtDescuento.Text.Trim() == "")
    {
        MessageBox.Show("Debe ingresar el porcentaje de descuento", "Validaciones", MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtCliente.Focus();
        return;
    }

    GuardarPresupuesto();
}
```

```

private void GuardarPresupuesto()
{
    oPresupuesto.Cliente = txtCliente.Text;
    oPresupuesto.Descuento = Convert.ToDouble(txtDescuento.Text);
    oPresupuesto.Fecha = Convert.ToDateTime(txtFecha.Text);
    oPresupuesto.Total = Convert.ToDouble(txtTotal.Text);

    if(banderaUpdate)
    {
        if (gestor.ActualizarPresupuesto(oPresupuesto))
        {
            MessageBox.Show("Presupuesto Actualizado", "Informe", MessageBoxButtons.OK, MessageBoxIcon.Information);
            this.Dispose();
        }
        else
        {
            MessageBox.Show("ERROR. No se pudo registrar el presupuesto", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    } else
    {
        if (gestor.ConfirmarPresupuesto(oPresupuesto))
        {
            MessageBox.Show("Presupuesto registrado", "Informe", MessageBoxButtons.OK, MessageBoxIcon.Information);
            this.Dispose();
        }
        else
        {
            MessageBox.Show("ERROR. No se pudo registrar el presupuesto", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

c.

2. En GestorPresupuesto

```

public bool ActualizarPresupuesto(Presupuesto oPresupuesto)
{
    return dao.ActualizarPresupuesto(oPresupuesto);
}

```

a.

3. En IPresupuestoDAO

```

2 referencias
bool ActualizarPresupuesto(Presupuesto oPresupuesto);

```

a.

4. En PresupuestoDAO

```

2 referencias
public bool ActualizarPresupuesto(Presupuesto oPresupuesto)
{
    return HelperDAO.ObtenerInstancia().Update(oPresupuesto);
}

```

a.

5. En HelperDAO

```
public bool Update(Presupuesto oPresupuesto)
{
    bool resultado = true;
    SqlConnection cnn = new SqlConnection();
    SqlTransaction trans = null;

    try
    {
        oPresupuesto.Total = oPresupuesto.calcularTotalDesc(oPresupuesto.CalcularTotal(), oPresupuesto.Descuento);

        cnn.ConnectionString = cadenaConexion;
        cnn.Open();
        trans = cnn.BeginTransaction();

        SqlCommand cmd = new SqlCommand("SP_UPDATE_PRESUPUESTOS", cnn, trans);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@nro_presupuesto", oPresupuesto.PresupuestoNro);
        cmd.Parameters.AddWithValue("@cliente", oPresupuesto.Cliente);
        cmd.Parameters.AddWithValue("@dto", oPresupuesto.Descuento);
        cmd.Parameters.AddWithValue("@total", oPresupuesto.Total);
        cmd.ExecuteNonQuery();

        SqlCommand cmdEliminar = new SqlCommand("SP_ELIMINAR_DETALLE_PRESUPUESTO", cnn, trans);
        cmdEliminar.CommandType = CommandType.StoredProcedure;
        cmdEliminar.Parameters.AddWithValue("@presupuesto_nro", oPresupuesto.PresupuestoNro);
        cmdEliminar.ExecuteNonQuery();

        int cDetalles = 1; // es el ID que forma de la PK doble entre ID_PRESUPUESTO E ID_DETALLE
        foreach (DetallePresupuesto det in oPresupuesto.Detalles)
        {
            // SqlCommand cmdDet = new SqlCommand("SP_UPDATE_DETALLES_PRESUPUESTO", cnn);
            SqlCommand cmdDet = new SqlCommand("[SP_INSERTAR_DETALLE]", cnn);
            cmdDet.CommandType = CommandType.StoredProcedure;
            cmdDet.Transaction = trans;
            cmdDet.Parameters.AddWithValue("@presupuesto_nro", oPresupuesto.PresupuestoNro);
            cmdDet.Parameters.AddWithValue("@detalle", cDetalles);
            cmdDet.Parameters.AddWithValue("@id_producto", det.Producto.IdProducto);
            cmdDet.Parameters.AddWithValue("@cantidad", det.Cantidad);
            cmdDet.ExecuteNonQuery();
            cDetalles++;
        }
    }
}
```

a.

```
        cmdDet.ExecuteNonQuery();
        cDetalles++;
    }

    trans.Commit();
}
catch (Exception ex)
{
    trans.Rollback();
    resultado = false;
}
finally
{
    if (cnn != null && cnn.State == ConnectionState.Open)
    {
        cnn.Close();
    }
}

return resultado;
}
```

b.