# 1 – Introducción

## Repositorio

Creamos el repositorio en GitHub y lo llamamos AppComercial.

Debe tener el gitignore de VisualStudio.

## Solución

Copiamos la ruta del repositorio y vamos a Visual Studio y clonamos el Repositorio en la Carpeta C:\AppComercial

Dentro de la Solución AppComercial creamos los Proyectos:

* Aplicación de Windows Forms (.NET Framework), y lo llamamos AppComercial
* Biblioteca de Clases (.NET Framework), y lo llamamos CADAppCOmercial

## Formulario Login

En el Proyecto AppComercial cambiamos el nombre del formulario Form1 a frmLogin

Le ponemos estas Propiedades y Controles:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmLogin |
|  | AcceptButton | btnAceptar |
|  | BackColor | SkyBlue |
|  | CancelButton | btnCancelar |
|  | FormBorderStyle | FixedDialog |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | Text | Ingreso al Sistema |
|  | Size | 413,280 |
|  |  |  |
| Label | (Name) | label1 |
|  | Text | &Usuario; |
|  |  |  |
| TextBox | (Name) | txtUsuario |
|  |  |  |
| Label | (Name) | Label2 |
|  | Text | &Clave; |
|  | PasswordChar | \* |
|  |  |  |
| PictureBox | Image | Cargamos la imagen como recurso local |
|  |  |  |
| Button | (Name) | btnAceptar |
|  | Text | &Aceptar |
|  | Image | Cargamos la imagen como recurso local |
|  | ImageAlign | TopCenter |
|  | TextAlign | BottomCenter |
|  |  |  |
| Button | (Name) | btnCancelar |
|  | Text | &Cancelar |
|  | Image | Cargamos la imagen como recurso local |
|  | ImageAlign | TopCenter |
|  | TextAlign | BottomCenter |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |

# 2 - Iniciando

## Código del frmLogin

En el código del frmLogin ponemos:

|  |
| --- |
| using System;  using System.Windows.Forms;  namespace AppComercial  {  public partial class frmLogin : Form  {  public frmLogin()  {  InitializeComponent();  }  private void frmLogin\_Load(object sender, EventArgs e)  {  }  private void btnCancelar\_Click(object sender, EventArgs e)  {  this.Close();  }  private void btnAceptar\_Click(object sender, EventArgs e)  {  errorProvider1.Clear();  if (txtUsuario.Text==string.Empty)  {  errorProvider1.SetError(txtUsuario, "Debe ingresar un Usuario");  txtUsuario.Focus();  return;  }  if (txtClave.Text == string.Empty)  {  errorProvider1.SetError(txtClave, "Debe ingresar una Clave");  txtClave.Focus();  return;  }  }  }  } |

## Usuario TableAdapter

En el Proyecto **CADAppComercial** hacemos clic derecho – Agregar - Nuevo Elemento – Datos – Conjunto de Datos y lo llamamos **DSAppComercial**.

Agregamos en el explorador de servidores la Base de Datos.

Arrastramos las Tabla **Usuario** y **Rol**.

En **UsuarioTableAdapter** hacemos clic derecho, agregar Consulta, Crear un nuevo procedimiento almacenado, SELECT que devuelve un solo valor

|  |
| --- |
| SELECT 1 FROM Usuario WHERE IDUsuario=@IDUsuario AND Clave=@Clave |

Como nombre le ponemos **ValidaUsuario.**

Hay que crear una Clase que permita utilizar esa consulta.

## Clase CADUsuario

En el Proyecto **CADAppComercial** creamos la Clase **CADUsuario**.

# 3 - Login

|  |
| --- |
| using CADAppComercial.DSAppComercialTableAdapters;  namespace CADAppComercial  {  public class CADUsuario  {  private static UsuarioTableAdapter adaptador = new UsuarioTableAdapter();  public static bool ValidaUsuario(string IDUsuario, string Clave)  {  if (adaptador.ValidaUsuario(IDUsuario, Clave) == null)  {  return false;  }  else  {  return true;  }  }  }  } |

Compilamos la solución.

En el Proyecto **AppComercial** agregamos la Referencia del proyecto **CADAppComercial**

## Verificando Usuario y Clave contra la Base de Datos

En **frmLogin** hacemos:

|  |
| --- |
| using CADAppComercial;  using System;  using System.Windows.Forms;  namespace AppComercial  {  public partial class frmLogin : Form  {  public frmLogin()  {  InitializeComponent();  }  private void frmLogin\_Load(object sender, EventArgs e)  {  }  private void btnCancelar\_Click(object sender, EventArgs e)  {  this.Close();  }  private void btnAceptar\_Click(object sender, EventArgs e)  {  errorProvider1.Clear();    if (txtUsuario.Text==string.Empty)  {  errorProvider1.SetError(txtUsuario, "Debe ingresar un Usuario");  txtUsuario.Focus();  return;  }    if (txtClave.Text == string.Empty)  {  errorProvider1.SetError(txtClave, "Debe ingresar una Clave");  txtClave.Focus();  return;  }  if(!CADUsuario.ValidaUsuario(txtUsuario.Text,txtClave.Text))  {  MessageBox.Show("Usuario o Clave no válidos", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);  txtUsuario.Text = string.Empty;  txtClave.Text = string.Empty;  txtUsuario.Focus();  return;  }  frmPrincipal miForm = new frmPrincipal();  miForm.Show();  this.Hide();  }  }  } |

## Formulario Principal

Creamos el **frmPrincipal**:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmPrincipal |
|  | BackColor |  |
|  | IsMdiContainer | True |
|  | WindowsState | Maximized |
|  | Text | Sistema Comercial |
|  |  |  |

# 4-Menú

En el **frmPrincipal** agregamos desde el Cuadro de Herramientas un **MenuStrip**.

 

 



# 5 – CRUD Clientes, Barra de herramientas

## Para que al cerrar el frmPrincipal se cierre la Aplicación

En el código del **frmPrincipal** ponemos:

|  |
| --- |
| using System.Windows.Forms;  namespace AppComercial  {  public partial class frmPrincipal : Form  {  public frmPrincipal()  {  InitializeComponent();  }  private void frmPrincipal\_FormClosing(object sender, FormClosingEventArgs e)  {  Application.Exit();  }  private void salirToolStripMenuItem\_Click(object sender, System.EventArgs e)  {  this.Close();  }  }  } |

## Formulario Clientes (primera versión)

Creamos el formulario **frmClientesTemp.** Le ponemos estas Propiedades y Controles:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmClientesTemp |
|  | Text | Clientes |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  |  |  |
| ToolStrip | (Name) | toolStrip1 |
|  | Agregamos 4 botones (para navegar entre registros), un separador, y 6 botones más (Nuevo, Editar, Guardar, Cancelar, Borrar y Buscar).  Le ponemos íconos en la propiedad Image.    Les ponemos nombres:   * tsbPrimero * tsbAnterior * tsbSiguiente * tsbUltimo * tsbEditar * tsbNuevo * tsbBorrar * tsbGuardar * tsbCancelar * tsbBuscar   Agregamos los ToolStripText | |

# 6 – CRUD Clientes, diseño

## Abrir el Formulario de Cliente

Para poder llamar el formulario de Clientes desde el Menú Principal, ponemos en el código del **frmPrincipal**:

|  |
| --- |
| private void clientesToolStripMenuItem\_Click(object sender, System.EventArgs e)  {  frmClientesTemp miForm = new frmClientesTemp();  miForm.MdiParent = this;  miForm.Show();  } |

## DataSet DSAppComercial

En el Proyecto **AppComercial** creamos el DataSet **DSAppComercial** y le agregamos la Tabla **TiposDocumento**



## Formulario Clientes

Agregamos al formulario **frmClientesTemp:**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Label | (Name) | label1 |
|  | Text | ID Cliente: |
|  |  |  |
| TextBox | (Name) | txtIDCliente |
|  | ReadOnly | True |
|  |  |  |
| Label | (Name) | label2 |
|  | Text | Tipo Documento: |
|  |  |  |
| ComboBox | (Name) | cbTipoDocumento |
|  | Enabled | False |
|  | BackColor | Aquamarine |
|  | Datos de origen |  |
|  |  |  |
| Label | (Name) | Label3 |
|  | Text | Documento: |
|  |  |  |
| TextBox | (Name) | txtDocumento: |
|  | ReadOnly | True |
|  | BackColor | Aquamarine |
|  |  |  |
| Label | (Name) | Label4 |
|  | Text | Nombre Comercial: |
|  |  |  |
| TextBox | (Name) | txtNombreComercial |
|  | ReadOnly | True |
|  | BackColor | Aquamarine |
|  |  |  |
| Label | (Name) | Label5 |
|  | Text | Nombre Contacto: |
|  |  |  |
| TextBox | (Name) | txtNombreContacto |
|  | ReadOnly | True |
|  | BackColor | Aquamarine |
|  |  |  |
| Label | (Name) | Label6 |
|  | Text | Apellido Contacto: |
|  |  |  |
| TextBox | (Name) | txtApellidoContacto |
|  | ReadOnly | True |
|  | BackColor | Aquamarine |
|  |  |  |
| Label | (Name) | Label7 |
|  | Text | Dirección: |
|  |  |  |
| TextBox | (Name) | txtDireccion |
|  | ReadOnly | True |
|  | BackColor | White |
|  |  |  |
| Label | (Name) | Label8 |
|  | Text | Teléfono: |
|  |  |  |
| TextBox | (Name) | txtTeléfono |
|  | ReadOnly | True |
|  | BackColor | White |
|  |  |  |
| Label | (Name) | Label9 |
|  | Text | Celular: |
|  |  |  |
| TextBox | (Name) | txtCelular |
|  | ReadOnly | True |
|  | BackColor | White |

# 7 – CRUD Clientes, cargar datos en Grid

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Label | (Name) | Label10 |
|  | Text | Correo: |
|  |  |  |
| TextBox | (Name) | txtCorreo |
|  | ReadOnly | True |
|  | BackColor | White |
|  |  |  |
| Label | (Name) | Label11 |
|  | Text | Aniversario: |
|  |  |  |
| DateTimePicker | (Name) | dtpAniversario |
|  | Enabled | False |
|  | BackColor | White |
|  |  |  |
| Label | (Name) | Label12 |
|  | Text | Notas: |
|  |  |  |
| TextBox | (Name) | txtNotas |
|  | ReadOnly | True |
|  | BackColor | White |
|  | Multiline | True |
|  | ScrollBars | Vertical |
|  |  |  |
| DataGridView | (Name) | dgvDatos |
|  | Anchor | Los 4 costados marcados |
|  | DefaultCellStyle | Font Arial Narrow 7 |
|  | AlternatingRowsDefaultCells | Font Arial Narrow 7 y BackColor=192; 255; 255 |

Hacemos clic en el triangulito negro que sale arriba a la derecha de la grilla y ahí hacemos:



En el Proyecto **CADAppComercial**,en el DataSet **DSAppComercial**, ponemos las Tablas **Clientes** y **TiposDocumento**:



En el Proyecto **CADAppComercial**, creamos la Clase **CADCliente**:

|  |
| --- |
| using CADAppComercial.DSAppComercialTableAdapters;  namespace CADAppComercial  {  public class CADCliente  {  private static ClienteTableAdapter adaptador = new ClienteTableAdapter();  public static DSAppComercial.ClienteDataTable GetData()  {  return adaptador.GetData();  }  }  } |

En el código del **frmClientesTemp** ponemos:

|  |
| --- |
| using CADAppComercial;  using System;  using System.Windows.Forms;  namespace AppComercial  {  public partial class frmClientesTemp : Form  {  public frmClientesTemp()  {  InitializeComponent();  }  private void frmClientesTemp\_Load(object sender, EventArgs e)  {  this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);  cbTipoDocumento.SelectedIndex = -1;  dgvDatos.DataSource = CADCliente.GetData();  dgvDatos.AutoResizeColumns();  }  private void fillActivosToolStripButton\_Click(object sender, EventArgs e)  {  try  {  this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);  }  catch (System.Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  }  private void fillActivosToolStripButton\_Click\_1(object sender, EventArgs e)  {  try  {  this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);  }  catch (System.Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  }  private void fillActivosToolStripButton\_Click\_2(object sender, EventArgs e)  {  try  {  this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);  }  catch (System.Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  }  }  } |

# 8 – CRUD Clientes, botones de navegación

En el código del **frmClientesTemp** ponemos:

|  |
| --- |
| using CADAppComercial;  using System;  using System.Windows.Forms;  namespace AppComercial  {  public partial class frmClientesTemp : Form  {  private int i = 0;  public frmClientesTemp()  {  InitializeComponent();  }  private void frmClientesTemp\_Load(object sender, EventArgs e)  {  this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);  cbTipoDocumento.SelectedIndex = -1;  dgvDatos.DataSource = CADCliente.GetData();  MostrarRegistro();  dgvDatos.AutoResizeColumns();  }  private void MostrarRegistro()  {  if (dgvDatos.Rows.Count == 0) return;  txtIDCliente.Text = dgvDatos.Rows[i].Cells["IDCliente"].Value.ToString();  txtNombreComercial.Text = dgvDatos.Rows[i].Cells["NombreComercial"].Value.ToString();  txtNombreContacto.Text = dgvDatos.Rows[i].Cells["NombresContacto"].Value.ToString();  txtApellidoContacto.Text = dgvDatos.Rows[i].Cells["ApellidosContacto"].Value.ToString();  txtDocumento.Text = dgvDatos.Rows[i].Cells["Documento"].Value.ToString();  txtDireccion.Text = dgvDatos.Rows[i].Cells["Direccion"].Value.ToString();  txtTelefono1.Text = dgvDatos.Rows[i].Cells["Telefono1"].Value.ToString();  txtTelefono2.Text = dgvDatos.Rows[i].Cells["Telefono2"].Value.ToString();  txtDireccion.Text = dgvDatos.Rows[i].Cells["Direccion"].Value.ToString();  txtNotas.Text = dgvDatos.Rows[i].Cells["Notas"].Value.ToString();  txtCorreo.Text = dgvDatos.Rows[i].Cells["Correo"].Value.ToString();  cbTipoDocumento.SelectedValue= dgvDatos.Rows[i].Cells["IDTipoDocumento"].Value;  try  {  dtpAniversario.Value = Convert.ToDateTime(dgvDatos.Rows[i].Cells["Aniversario"].Value);  }  catch (Exception)  {  dtpAniversario.Value = DateTime.Now;  }  }  private void fillActivosToolStripButton\_Click(object sender, EventArgs e)  {  try  {  this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);  }  catch (System.Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  }  private void fillActivosToolStripButton\_Click\_1(object sender, EventArgs e)  {  try  {  this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);  }  catch (System.Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  }  private void fillActivosToolStripButton\_Click\_2(object sender, EventArgs e)  {  try  {  this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);  }  catch (System.Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  }  private void tsbSiguiente\_Click(object sender, EventArgs e)  {  if(i>=dgvDatos.Rows.Count-1)return;  i++;  MostrarRegistro();  }  private void tsbAnterior\_Click(object sender, EventArgs e)  {  if (i == 0) return;  i--;  MostrarRegistro();  }  private void tsbPrimero\_Click(object sender, EventArgs e)  {  i=0;  MostrarRegistro();  }  private void tsbUltimo\_Click(object sender, EventArgs e)  {  i = dgvDatos.Rows.Count-1;  MostrarRegistro();  }  }  } |

# 9 – CRUD Clientes, botones editar y nuevo

A los botones de Guardar y Cancelar les ponemos Enabled en False

# 10 - CRUD Clientes, botón guardar

En el DataSet **DSAppComercial** del Proyecto **CADAppComercial**,en la Tabla **Clientes** creamos los métodos:

ClienteInsert

INSERT INTO [dbo].[Cliente] ([IDTipoDocumento], [Documento], [NombreComercial], [NombresContacto], [ApellidosContacto], [Direccion], [Telefono1], [Telefono2], [Correo], [Notas], [Aniversario]) VALUES (@IDTipoDocumento, @Documento, @NombreComercial, @NombresContacto, @ApellidosContacto, @Direccion, @Telefono1, @Telefono2, @Correo, @Notas, @Aniversario);

ClienteUpdate

UPDATE [dbo].[Cliente] SET

[IDTipoDocumento] = @IDTipoDocumento,

[Documento] = @Documento,

[NombreComercial] = @NombreComercial,

[NombresContacto] = @NombresContacto,

[ApellidosContacto] = @ApellidosContacto,

[Direccion] = @Direccion,

[Telefono1] = @Telefono1,

[Telefono2] = @Telefono2,

[Correo] = @Correo,

[Notas] = @Notas,

[Aniversario] = @Aniversario

WHERE ([IDCliente] = @IDCliente)

ClienteDelete

DELETE FROM [dbo].[Cliente] WHERE ([IDCliente] = @IDCliente)

# 11 – CRUD Clientes, botón borrar

En la Clase **CADCliente** hacemos:

using CADAppComercial.DSAppComercialTableAdapters;

using System;

namespace CADAppComercial

{

public class CADCliente

{

private static ClienteTableAdapter adaptador = new ClienteTableAdapter();

public static DSAppComercial.ClienteDataTable GetData()

{

return adaptador.GetData();

}

public static void ClienteInsert(

int IDTipoDocumento,

string Documento,

string NombreComercial,

string NombresContacto,

string ApellidosContacto,

string Direccion,

string Telefono1,

string Telefono2,

string Correo,

string Notas,

DateTime Aniversario)

{

adaptador.ClienteInsert(IDTipoDocumento, Documento, NombreComercial, NombresContacto, ApellidosContacto, Direccion, Telefono1, Telefono2, Correo, Notas, Aniversario);

}

public static void ClienteUpdate(

int IDTipoDocumento,

string Documento,

string NombreComercial,

string NombresContacto,

string ApellidosContacto,

string Direccion,

string Telefono1,

string Telefono2,

string Correo,

string Notas,

DateTime Aniversario,

int IDCliente)

{

adaptador.ClienteUpdate(IDTipoDocumento, Documento, NombreComercial, NombresContacto, ApellidosContacto, Direccion, Telefono1, Telefono2, Correo, Notas, Aniversario, IDCliente);

}

public static void ClienteDelete(int IDCliente)

{

adaptador.ClienteDelete(IDCliente);

}

}

}

En el código del **frmClientesTemp** ponemos:

using CADAppComercial;

using System;

using System.Windows.Forms;

namespace AppComercial

{

public partial class frmClientesTemp : Form

{

private int i = 0;

private bool nuevo;

public frmClientesTemp()

{

InitializeComponent();

}

private void frmClientesTemp\_Load(object sender, EventArgs e)

{

this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);

dgvDatos.DataSource = CADCliente.GetData();

MostrarRegistro();

dgvDatos.AutoResizeColumns();

}

private void MostrarRegistro()

{

if (dgvDatos.Rows.Count == 0) return;

txtIDCliente.Text = dgvDatos.Rows[i].Cells["IDCliente"].Value.ToString();

txtNombreComercial.Text = dgvDatos.Rows[i].Cells["NombreComercial"].Value.ToString();

txtNombreContacto.Text = dgvDatos.Rows[i].Cells["NombresContacto"].Value.ToString();

txtApellidoContacto.Text = dgvDatos.Rows[i].Cells["ApellidosContacto"].Value.ToString();

txtDocumento.Text = dgvDatos.Rows[i].Cells["Documento"].Value.ToString();

txtDireccion.Text = dgvDatos.Rows[i].Cells["Direccion"].Value.ToString();

txtTelefono1.Text = dgvDatos.Rows[i].Cells["Telefono1"].Value.ToString();

txtTelefono2.Text = dgvDatos.Rows[i].Cells["Telefono2"].Value.ToString();

txtDireccion.Text = dgvDatos.Rows[i].Cells["Direccion"].Value.ToString();

txtNotas.Text = dgvDatos.Rows[i].Cells["Notas"].Value.ToString();

txtCorreo.Text = dgvDatos.Rows[i].Cells["Correo"].Value.ToString();

cbTipoDocumento.SelectedValue= dgvDatos.Rows[i].Cells["IDTipoDocumento"].Value;

try

{

dtpAniversario.Value = Convert.ToDateTime(dgvDatos.Rows[i].Cells["Aniversario"].Value);

}

catch (Exception)

{

dtpAniversario.Value = DateTime.Now;

}

}

private void fillActivosToolStripButton\_Click(object sender, EventArgs e)

{

try

{

this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

}

private void fillActivosToolStripButton\_Click\_1(object sender, EventArgs e)

{

try

{

this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

}

private void fillActivosToolStripButton\_Click\_2(object sender, EventArgs e)

{

try

{

this.tipoDocumentoTableAdapter.FillActivos(this.dSAppComercial.TipoDocumento);

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

}

private void tsbSiguiente\_Click(object sender, EventArgs e)

{

if(i>=dgvDatos.Rows.Count-1)return;

i++;

MostrarRegistro();

}

private void tsbAnterior\_Click(object sender, EventArgs e)

{

if (i == 0) return;

i--;

MostrarRegistro();

}

private void tsbPrimero\_Click(object sender, EventArgs e)

{

i=0;

MostrarRegistro();

}

private void tsbUltimo\_Click(object sender, EventArgs e)

{

i = dgvDatos.Rows.Count-1;

MostrarRegistro();

}

private void tsbModificar\_Click(object sender, EventArgs e)

{

Habilitar(false);

nuevo = false;

}

private void Habilitar(bool campo)

{

tsbPrimero.Enabled = campo;

tsbUltimo.Enabled = campo;

tsbSiguiente.Enabled = campo;

tsbAnterior.Enabled = campo;

tsbModificar.Enabled = campo;

tsbNuevo.Enabled = campo;

tsbBorrar.Enabled = campo;

tsbGuardar.Enabled = !campo;

tsbCancelar.Enabled = !campo;

tsbBuscar.Enabled = campo;

txtApellidoContacto.ReadOnly = campo;

txtCorreo.ReadOnly = campo;

txtDireccion.ReadOnly = campo; ;

txtDocumento.ReadOnly = campo;

txtNombreComercial.ReadOnly = campo;

txtNombreContacto.ReadOnly = campo;

txtNotas.ReadOnly = campo;

txtTelefono1.ReadOnly = campo;

txtTelefono2.ReadOnly = campo;

cbTipoDocumento.Enabled = !campo;

dtpAniversario.Enabled = !campo;

cbTipoDocumento.Focus();

}

private void tsbBuscar\_Click(object sender, EventArgs e)

{

}

private void tsbCancelar\_Click(object sender, EventArgs e)

{

Habilitar(true);

MostrarRegistro();

}

private void tsbNuevo\_Click(object sender, EventArgs e)

{

Habilitar(false);

nuevo = true;

LimpiarCampos();

}

private void LimpiarCampos()

{

txtIDCliente.Text = string.Empty;

txtApellidoContacto.Text = string.Empty;

txtCorreo.Text = string.Empty;

txtDireccion.Text = string.Empty;

txtDocumento.Text = string.Empty;

txtNombreComercial.Text = string.Empty;

txtNombreContacto.Text = string.Empty;

txtNotas.Text = string.Empty;

txtTelefono1.Text = string.Empty;

txtTelefono2.Text = string.Empty;

cbTipoDocumento.SelectedIndex = -1;

dtpAniversario.Value = DateTime.Now;

cbTipoDocumento.Focus();

}

private void tsbGuardar\_Click(object sender, EventArgs e)

{

if (!Validarcampos()) return;

if (nuevo)

{

CADCliente.ClienteInsert(

(int)cbTipoDocumento.SelectedValue,

txtDocumento.Text,

txtNombreComercial.Text,

txtNombreContacto.Text,

txtApellidoContacto.Text,

txtDireccion.Text,

txtTelefono1.Text,

txtTelefono2.Text,

txtCorreo.Text,

txtNotas.Text,

dtpAniversario.Value);

}

else

{

CADCliente.ClienteUpdate(

(int)cbTipoDocumento.SelectedValue,

txtDocumento.Text,

txtNombreComercial.Text,

txtNombreContacto.Text,

txtApellidoContacto.Text,

txtDireccion.Text,

txtTelefono1.Text,

txtTelefono2.Text,

txtCorreo.Text,

txtNotas.Text,

dtpAniversario.Value,

Convert.ToInt32(txtIDCliente.Text));

}

Habilitar(true);

dgvDatos.DataSource = null;

dgvDatos.DataSource = CADCliente.GetData();

if (nuevo) tsbUltimo\_Click(sender, e);

MostrarRegistro();

}

private bool Validarcampos()

{

errorProvider1.Clear();

if (cbTipoDocumento.SelectedIndex==-1)

{

errorProvider1.SetError(cbTipoDocumento, "Debe seleccionar un Tipo de Documento");

cbTipoDocumento.Focus();

return false;

}

if (txtDocumento.Text==string.Empty)

{

errorProvider1.SetError(txtDocumento, "Debe ingresar un N° de Documento");

txtDocumento.Focus();

return false;

}

if (txtNombreComercial.Text == string.Empty)

{

errorProvider1.SetError(txtNombreComercial, "Debe ingresar un Nombre Comercial");

txtNombreComercial.Focus();

return false;

}

if (txtNombreContacto.Text == string.Empty)

{

errorProvider1.SetError(txtNombreContacto, "Debe ingresar un Nombre de Contacto");

txtNombreContacto.Focus();

return false;

}

if (txtApellidoContacto.Text == string.Empty)

{

errorProvider1.SetError(txtApellidoContacto, "Debe ingresar un Apellido de Contacto");

txtApellidoContacto.Focus();

return false;

}

return true;

}

private void tsbBorrar\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Registro actual?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

CADCliente.ClienteDelete(Convert.ToInt32(txtIDCliente.Text));

dgvDatos.DataSource = null;

dgvDatos.DataSource = CADCliente.GetData();

if (i != 0) i--;

MostrarRegistro();

}

}

}

# 12 al 17 – CRUD Proveedores Parte I a VI

En **DSAppComercial** del Proyecto **AppComercial**, desde el Explorador de Servidores arrastramos la Tabla **Proveedores**:



Creamos el Formulario **frmProveedores**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmProveedores |
|  | Text | Proveedores |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | Size | 1296; 676 |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |

En el evento Load agregamos

dgvDatos.AutoResizeColumns();

Arrastramos desde Orígenes de Datos la Tabla Proveedores y de esta manera creamos la grilla.

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| DataGridView | (Name) | dgvDatos |
|  | Anchor | Los 4 costados marcados |
|  | DefaultCellStyle | Font Arial Narrow 7 |
|  | AlternatingRowsDefaultCells | Font Arial Narrow 7 y BackColor=192; 255; 255 |

Hacemos clic en el triangulito negro que sale arriba a la derecha de la grilla y ahí hacemos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ahí elegimos Editar Columnas y cambiamos/corregimos los Nombres de las Columnas:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

TipoDocumento para que no salga el ID sino la Descripción, hacemos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para mostrar este formulario hacemos en frmPrincipal:

private void proveedoresToolStripMenuItem\_Click(object sender, System.EventArgs e)

{

frmProveedores miForm = new frmProveedores();

miForm.MdiParent = this;

miForm.Show();

}

Desde Orígenes de Datos arrastramos campo por campo de la Tabla Proveedores:



Para el IDTipoDocumento, antes de arrástralo, hacemos clic derecho y elegimos ComboBox



Hacemos Ver – Orden de Tabulación y le damos el orden que queremos.

En el ComBox de Tipo de Documento hacemos:



Y en las Propiedades hacemos:



A todos los TextBox les ponemos ReadOnly = True Al ComboBox le ponemos Enabled = False

En la Barra de Herramientas:

* Agregamos los botones de Editar, Cancelar y Buscar
* Les ponemos nombres según el estándar de los 3 botones ya existentes (nuevo, grabar y borrar)
* Cambiamos las imágenes de todos los íconos
* Cambiamos el orden los íconos a Editar-Nuevo-Borrar-Grabar-Cancelar-Buscar
* Les ponemos Enabled = True a todos excepto a Grabar y Cancelar que les ponemos Enabled = False
* Les ponemos los ToolTipText

Creamos el método Habilitar:

private void Habilitar(bool campo)

{

iDTipoDocumentoComboBox.Enabled = campo;

documentoTextBox.ReadOnly = !campo;

nombreTextBox.ReadOnly = !campo;

nombresContactoTextBox.ReadOnly = !campo;

apellidosContactoTextBox.ReadOnly = !campo;

direccionTextBox.ReadOnly = !campo;

telefono1TextBox.ReadOnly = !campo;

telefono2TextBox.ReadOnly = !campo;

correoTextBox.ReadOnly = !campo;

notasTextBox.ReadOnly = !campo;

bindingNavigatorEditItem.Enabled = !campo;

bindingNavigatorAddNewItem.Enabled = !campo;

bindingNavigatorDeleteItem.Enabled = !campo;

bindingNavigatorSaveItem.Enabled = campo;

bindingNavigatorCancelItem.Enabled = campo;

bindingNavigatorSearchItem.Enabled = !campo;

bindingNavigatorMoveFirstItem.Enabled = !campo;

bindingNavigatorMovePreviousItem.Enabled = !campo;

bindingNavigatorMoveNextItem.Enabled = !campo;

bindingNavigatorMoveLastItem.Enabled = !campo;

bindingNavigatorPositionItem.Enabled = !campo;

bindingNavigatorCountItem.Enabled = !campo;

}

Al botón Editar le ponemos:

private void bindingNavigatorEditItem\_Click(object sender, EventArgs e)

{

Habilitar(true);

}

Al botón Cancelar le ponemos:

private void bindingNavigatorCancelItem\_Click(object sender, EventArgs e)

{

this.proveedorBindingSource.CancelEdit();

errorProvider1.Clear();

Habilitar(false);

}

Creamos la Clase RegexUtilities:

using System;

using System.Globalization;

using System.Text.RegularExpressions;

public class RegexUtilities

{

bool invalid = false;

public bool IsValidEmail(string strIn)

{

invalid = false;

if (String.IsNullOrEmpty(strIn))

return false;

// Use IdnMapping class to convert Unicode domain names.

try

{

strIn = Regex.Replace(strIn, @"(@)(.+)$", this.DomainMapper,

RegexOptions.None, TimeSpan.FromMilliseconds(200));

}

catch (RegexMatchTimeoutException)

{

return false;

}

if (invalid)

return false;

// Return true if strIn is in valid e-mail format.

try

{

return Regex.IsMatch(strIn,

@"^(?("")(""[^""]+?""@)|(([0-9a-z]((\.(?!\.))|[-!#\$%&'\\*\+/=\?\^`\{\}\|~\w])\*)(?<=[0-9a-z])@))" +

@"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]\*[0-9a-z]\*\.)+[a-z0-9]{2,24}))$",

RegexOptions.IgnoreCase, TimeSpan.FromMilliseconds(250));

}

catch (RegexMatchTimeoutException)

{

return false;

}

}

private string DomainMapper(Match match)

{

// IdnMapping class with default property values.

IdnMapping idn = new IdnMapping();

string domainName = match.Groups[2].Value;

try

{

domainName = idn.GetAscii(domainName);

}

catch (ArgumentException)

{

invalid = true;

}

return match.Groups[1].Value + domainName;

}

}

Creamos el método ValidarCampos

private bool Validarcampos()

{

errorProvider1.Clear();

if (iDTipoDocumentoComboBox.SelectedIndex == -1)

{

errorProvider1.SetError(iDTipoDocumentoComboBox, "Debe seleccionar un Tipo de Documento");

iDTipoDocumentoComboBox.Focus();

return false;

}

if (documentoTextBox.Text == string.Empty)

{

errorProvider1.SetError(documentoTextBox, "Debe ingresar un N° de Documento");

documentoTextBox.Focus();

return false;

}

if (nombreTextBox.Text == string.Empty)

{

errorProvider1.SetError(nombreTextBox, "Debe ingresar un Nombre de Proveedor");

nombreTextBox.Focus();

return false;

}

if (nombresContactoTextBox.Text == string.Empty)

{

errorProvider1.SetError(nombresContactoTextBox, "Debe ingresar un Nombre de Contacto");

nombresContactoTextBox.Focus();

return false;

}

if (apellidosContactoTextBox.Text == string.Empty)

{

errorProvider1.SetError(apellidosContactoTextBox, "Debe ingresar un Apellido de Contacto");

apellidosContactoTextBox.Focus();

return false;

}

if (correoTextBox.Text != string.Empty)

{

RegexUtilities regexUtilities = new RegexUtilities();

if (!regexUtilities.IsValidEmail(correoTextBox.Text))

{

errorProvider1.SetError(correoTextBox, "Si ingresa un correo, éste debe ser válido");

correoTextBox.Focus();

return false;

}

}

return true;

}

Al botón Guardar le ponemos:

private void bindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

if (!Validarcampos()) return;

this.Validate();

this.proveedorBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

Habilitar(false);

}

En las propiedades de proveedorBindingNavigator hacemos:



Al botón Nuevo le ponemos:

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

Habilitar(true);

proveedorBindingSource.AddNew();

iDTipoDocumentoComboBox.Focus();

}

Creamos el método ArmaNombre que lo usaremos para cuando el Proveedor sea una persona natural y no una empresa:

private void nombresContactoTextBox\_TextChanged(object sender, EventArgs e)

{

ArmaNombre();

}

private void apellidosContactoTextBox\_TextChanged(object sender, EventArgs e)

{

ArmaNombre();

}

private void ArmaNombre()

{

if(iDTipoDocumentoComboBox.SelectedIndex==0)

{

nombreTextBox.Text = nombresContactoTextBox.Text + " " + apellidosContactoTextBox.Text;

}

}

Al botón Borrar le ponemos:

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Registro actual?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

this.Validate();

this.proveedorBindingSource.RemoveAt(proveedorBindingSource.Position);

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

}

* **Falta validar al momento de borrar que el Proveedor no tenga Compras asociadas**

Así queda todo el código del **frmProveedores**:

using System;

using System.Windows.Forms;

namespace AppComercial

{

public partial class frmProveedores : Form

{

public frmProveedores()

{

InitializeComponent();

}

private void frmProveedores\_Load(object sender, EventArgs e)

{

this.tipoDocumentoTableAdapter.Fill(this.dSAppComercial.TipoDocumento);

this.proveedorTableAdapter.Fill(this.dSAppComercial.Proveedor);

dgvDatos.AutoResizeColumns();

}

private void bindingNavigatorEditItem\_Click(object sender, EventArgs e)

{

Habilitar(true);

}

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

Habilitar(true);

proveedorBindingSource.AddNew();

iDTipoDocumentoComboBox.Focus();

}

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Registro actual?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

this.Validate();

this.proveedorBindingSource.RemoveAt(proveedorBindingSource.Position);

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

}

private void bindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

if (!Validarcampos()) return;

this.Validate();

this.proveedorBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

Habilitar(false);

}

private void bindingNavigatorCancelItem\_Click(object sender, EventArgs e)

{

this.proveedorBindingSource.CancelEdit();

errorProvider1.Clear();

Habilitar(false);

}

private void Habilitar(bool campo)

{

iDTipoDocumentoComboBox.Enabled = campo;

documentoTextBox.ReadOnly = !campo;

nombreTextBox.ReadOnly = !campo;

nombresContactoTextBox.ReadOnly = !campo;

apellidosContactoTextBox.ReadOnly = !campo;

direccionTextBox.ReadOnly = !campo;

telefono1TextBox.ReadOnly = !campo;

telefono2TextBox.ReadOnly = !campo;

correoTextBox.ReadOnly = !campo;

notasTextBox.ReadOnly = !campo;

bindingNavigatorEditItem.Enabled = !campo;

bindingNavigatorAddNewItem.Enabled = !campo;

bindingNavigatorDeleteItem.Enabled = !campo;

bindingNavigatorSaveItem.Enabled = campo;

bindingNavigatorCancelItem.Enabled = campo;

bindingNavigatorSearchItem.Enabled = !campo;

bindingNavigatorMoveFirstItem.Enabled = !campo;

bindingNavigatorMovePreviousItem.Enabled = !campo;

bindingNavigatorMoveNextItem.Enabled = !campo;

bindingNavigatorMoveLastItem.Enabled = !campo;

bindingNavigatorPositionItem.Enabled = !campo;

bindingNavigatorCountItem.Enabled = !campo;

}

private void nombresContactoTextBox\_TextChanged(object sender, EventArgs e)

{

ArmaNombre();

}

private void apellidosContactoTextBox\_TextChanged(object sender, EventArgs e)

{

ArmaNombre();

}

private void ArmaNombre()

{

if(iDTipoDocumentoComboBox.SelectedIndex==0)

{

nombreTextBox.Text = nombresContactoTextBox.Text + " " + apellidosContactoTextBox.Text;

}

}

private bool Validarcampos()

{

errorProvider1.Clear();

if (iDTipoDocumentoComboBox.SelectedIndex == -1)

{

errorProvider1.SetError(iDTipoDocumentoComboBox, "Debe seleccionar un Tipo de Documento");

iDTipoDocumentoComboBox.Focus();

return false;

}

if (documentoTextBox.Text == string.Empty)

{

errorProvider1.SetError(documentoTextBox, "Debe ingresar un N° de Documento");

documentoTextBox.Focus();

return false;

}

if (nombreTextBox.Text == string.Empty)

{

errorProvider1.SetError(nombreTextBox, "Debe ingresar un Nombre de Proveedor");

nombreTextBox.Focus();

return false;

}

if (nombresContactoTextBox.Text == string.Empty)

{

errorProvider1.SetError(nombresContactoTextBox, "Debe ingresar un Nombre de Contacto");

nombresContactoTextBox.Focus();

return false;

}

if (apellidosContactoTextBox.Text == string.Empty)

{

errorProvider1.SetError(apellidosContactoTextBox, "Debe ingresar un Apellido de Contacto");

apellidosContactoTextBox.Focus();

return false;

}

if (correoTextBox.Text != string.Empty)

{

RegexUtilities regexUtilities = new RegexUtilities();

if (!regexUtilities.IsValidEmail(correoTextBox.Text))

{

errorProvider1.SetError(correoTextBox, "Si ingresa un correo, éste debe ser válido");

correoTextBox.Focus();

return false;

}

}

return true;

}

}

}

Hay que hacer de manera similar al **frmProveedores** los siguientes:

* Clientes **Ok**
* Bodegas **Ok**
* Conceptos **Ok**
* Departamentos
* IVAs
* Unidades de Medida
* Tipos de Documentos

# 18 al 20 – Búsqueda de Proveedores

Creamos el Formulario **frmBusquedaProveedor**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmBusquedaProveedor |
|  | Text | Búsqueda de Proveedores |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | Size | 1296; 676 |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |
|  |  |  |
| DataGridView | (Name) | dgvDatos |
|  | Anchor | Los 4 costados marcados |
|  | DefaultCellStyle | Font Arial Narrow 7 |
|  | AlternatingRowsDefaultCells | Font Arial Narrow 7 y BackColor=192; 255; 255 |

Hacemos clic en el triangulito negro que sale arriba a la derecha de la grilla y ahí hacemos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Y elegimos el Origen de datos:



Ahí elegimos Editar Columnas y cambiamos/corregimos los Nombres de las Columnas:

Interfaz de usuario gráfica

Descripción generada automáticamente

TipoDocumento para que no salga el ID sino la Descripción, hacemos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En el **frmProveedores** le ponemos al botón de búsqueda (para abrir de forma **modal** el frmBusquedaProveedor):

private void bindingNavigatorSearchItem\_Click(object sender, EventArgs e)

{

frmBusquedaProveedor miBusqueda = new frmBusquedaProveedor();

miBusqueda.ShowDialog();

}

En el evento Load del frmBusquedaProveedor agregamos:

private void frmBusquedaProveedor\_Load(object sender, EventArgs e)

{

this.proveedorTableAdapter.Fill(this.dSAppComercial.Proveedor);

dgvDatos.AutoResizeColumns();

}

Hacemos clic en el triangulito negro que sale arriba a la derecha de la grilla y ahí elegimos Agregar Consulta:



Y ponemos:

SELECT IDProveedor, Nombre, IDTipoDocumento, Documento,

NombresContacto, ApellidosContacto, Direccion,

Telefono1, Telefono2, Correo, Notas

FROM dbo.Proveedor

WHERE Documento LIKE @Documento

AND Nombre LIKE @Nombre

AND NombresContacto LIKE @NombresContacto

AND ApellidosContacto LIKE @ApellidosContacto

Al botón **FillBy1** le ponemos la propiedad DisplayStyle =Image y le cargamos en Image el ícono de la lupa.

Le ponemos como nombre **tsbBuscar**

Y como **ToolTipText** le ponemos Aplica los criterios de búsqueda seleccionados

Agregamos al formulario:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| GroupBox | Text | Tipo criterio |
|  |  |  |
| RadioButton | (Name) | rbtContenga |
|  | Text | Contenga… |
|  | Checked | True |
|  |  |  |
| RadioButton | (Name) | rbtEmpiece |
|  | Text | Empiece por… |
|  |  |  |
| RadioButton | (Name) | rbtTermina |
|  | Text | Termina en… |
|  |  |  |
| RadioButton | (Name) | rbtIgual |
|  | Text | Igual a… |
|  |  |  |
| Copiamos desde frmLogin los botones Aceptar y Cancelar y los pegamos en este formulario. | | |
|  |  |  |
| Button | (Name) | btnQuitarFiltros |
|  | Text | Quitar Filtros |
|  | TextAlign | MiddleCenter |
|  |  |  |

Escribimos el código para el botón buscar:

private void fillByToolStripButton\_Click(object sender, EventArgs e)

{

string documento, nombre, nombreContacto, apellidoContacto;

if(rbtContenga.Checked)

{

documento = "%" + documentoToolStripTextBox.Text + "%";

nombre = "%" + nombreToolStripTextBox.Text + "%";

nombreContacto = "%" + nombresContactoToolStripTextBox.Text + "%";

apellidoContacto = "%" + apellidosContactoToolStripTextBox.Text + "%";

}

else if (rbtEmpiece.Checked)

{

documento = documentoToolStripTextBox.Text + "%";

nombre = nombreToolStripTextBox.Text + "%";

nombreContacto = nombresContactoToolStripTextBox.Text + "%";

apellidoContacto = apellidosContactoToolStripTextBox.Text + "%";

}

else if (rbtTermina.Checked)

{

documento = "%" + documentoToolStripTextBox.Text;

nombre = "%" + nombreToolStripTextBox.Text;

nombreContacto = "%" + nombresContactoToolStripTextBox.Text;

apellidoContacto = "%" + apellidosContactoToolStripTextBox.Text;

}

else

{

documento =documentoToolStripTextBox.Text;

nombre = nombreToolStripTextBox.Text;

nombreContacto = nombresContactoToolStripTextBox.Text;

apellidoContacto = apellidosContactoToolStripTextBox.Text;

}

try

{

this.proveedorTableAdapter.FillBy(

this.dSAppComercial.Proveedor,

documento,

nombre,

nombreContacto,

apellidoContacto);

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

}

Escribimos el código para el botón Quitar Filtros:

private void btnQuitarFiltros\_Click(object sender, EventArgs e)

{

documentoToolStripTextBox.Text=string.Empty;

nombreToolStripTextBox.Text = string.Empty;

nombresContactoToolStripTextBox.Text = string.Empty;

apellidosContactoToolStripTextBox.Text = string.Empty;

fillByToolStripButton\_Click(sender, e);

}

Para devolver al frmProveedores el Proveedor elegido tenemos que hacer lo siguiente.

En **frmBusquedaProveedores** definimos una atributo privado

private int idElegido;

public int IDElegido { get => idElegido; }

A los botones Aceptar y Cancelar le ponemos este código:

private void btnCancelar\_Click(object sender, EventArgs e)

{

idElegido = 0;

this.Close();

}

private void btnAceptar\_Click(object sender, EventArgs e)

{

if(dgvDatos.Rows.Count==0)

{

idElegido = 0;

}

else if (dgvDatos.SelectedRows.Count!=0)

{

idElegido = Convert.ToInt32(dgvDatos.SelectedRows[0].Cells[0].Value);

}

else

{

idElegido = Convert.ToInt32(dgvDatos.Rows[0].Cells[0].Value);

}

this.Close();

}

En el **frmProveedores** ponemos:

private void bindingNavigatorSearchItem\_Click(object sender, EventArgs e)

{

frmBusquedaProveedor miBusqueda = new frmBusquedaProveedor();

miBusqueda.ShowDialog();

if (miBusqueda.IDElegido == 0) return;

int position = proveedorBindingSource.Find("IDProveedor", miBusqueda.IDElegido);

proveedorBindingSource.Position = position;

}

# 21 – Validación de documento único

En el método Grabar corregimos:

private void bindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

if (!Validarcampos()) return;

this.Validate();

this.proveedorBindingSource.EndEdit();

try

{

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

errorProvider1.Clear();

}

catch (Exception)

{

errorProvider1.SetError(documentoTextBox, "Este N° de Documento ya está asignado");

documentoTextBox.Focus();

return;

}

Habilitar(false);

}

# 22 al 33 – CRUD Productos Parte I a XIII

En **DSAppComercial** del Proyecto **AppComercial**, desde el Explorador de Servidores arrastramos las Tablas **Producto, Barra y BodegaProducto**:



Creamos el Formulario **frmProductos**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmProveedores |
|  | Text | Proveedores |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | Size | 1296; 676 |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |

En el evento Load agregamos

dgvDatos.AutoResizeColumns();

barrasDataGridView.AutoResizeColumns();

bodegasDataGridView.AutoResizeColumns();

Arrastramos desde Orígenes de Datos la Tabla Productos y de esta manera creamos la grilla.

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| DataGridView | (Name) | dgvDatos |
|  | Anchor | Los 4 costados marcados |
|  | DefaultCellStyle | Font Arial Narrow 7 |
|  | AlternatingRowsDefaultCells | Font Arial Narrow 7 y BackColor=192; 255; 255 |

Hacemos clic en el triangulito negro que sale arriba a la derecha de la grilla y ahí hacemos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ahí elegimos Editar Columnas y cambiamos/corregimos los Nombres de las Columnas:

Interfaz de usuario gráfica

Descripción generada automáticamente

IVA, Departamento y Unidad: para que no salga el ID sino la Descripción, hacemos:



Para cambiar el formato de la columna precio hacemos:

Interfaz de usuario gráfica

Descripción generada automáticamente 

Para mostrar este formulario hacemos en frmPrincipal:

private void productosToolStripMenuItem1\_Click(object sender, System.EventArgs e)

{

frmProductos miForm = new frmProductos();

miForm.MdiParent = this;

miForm.Show();

}

Desde Orígenes de Datos arrastramos campo por campo de la Tabla Productos

Hacemos Ver – Orden de Tabulación y le damos el orden que queremos.

En los ComBox Departamento, IVA y Unidad hacemos:



Y en las Propiedades hacemos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

A todos los TextBox les ponemos ReadOnly = True Al ComboBox le ponemos Enabled = False

En la Barra de Herramientas:

* Agregamos los botones de Editar, Cancelar y Buscar
* Les ponemos nombres según el estándar de los 3 botones ya existentes (nuevo, grabar y borrar)
* Cambiamos las imágenes de todos los íconos
* Cambiamos el orden los íconos a Editar-Nuevo-Borrar-Grabar-Cancelar-Buscar
* Les ponemos Enabled = True a todos excepto a Grabar y Cancelar que les ponemos Enabled = False
* Les ponemos los ToolTipText

Agregamos:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Button | (Name) | btnBuscarImagen |
|  | Text | … |
|  |  |  |
| PictureBox | (Name) | pbxImagen |
|  | SizeMode | Zoom |
|  |  |  |
| GroupBox | (Name) | gbxBarras |
|  | Text | Códigos de Barras |
|  |  |  |
| DataGridView | (Name) | barrasDataGridView |
|  |  |  |
| Button | (Name) | btnAgregarBarra |
|  | Text | Agregar |
|  |  |  |
| Button | (Name) | btnElminarBarra |
|  | Text | Elminar |
|  |  |  |
| GroupBox | (Name) | gbxBodegas |
|  | Text | Parámetros de Bodega |
|  |  |  |
| DataGridView | (Name) | bodegasDataGridView |
|  |  |  |
| Button | (Name) | btnAgregarBodega |
|  | Text | Agregar / Modificar |

A los DataGridView barrasDataGridView y bodegasDataGridView le agregamos el origen de datos

Les ponemos el formato.

Les corregimos los títulos de la columna.

Les damos formatos a las columnas numéricas.

A los DataGridView barrasDataGridView y bodegasDataGridView les agregamos la consulta con el WHERE IdProducto =IdProducto

Hacemos Ver – Orden de Tabulación y le damos el orden que queremos.

En las propiedades de productoBindingNavigator hacemos:



Creamos el método Habilitar:

private void Habilitar(bool campo)

{

descripcionTextBox.ReadOnly = !campo;

iDDepartamentoComboBox.Enabled = campo;

precioTextBox.ReadOnly = !campo;

iDIVAComboBox.Enabled = campo;

btnBuscarImagen.Enabled = campo;

notasTextBox.ReadOnly = !campo;

iDMedidaComboBox.Enabled = campo;

medidaTextBox.ReadOnly = !campo;

btnAgregarBarra.Enabled = campo;

btnElminarBarra.Enabled = campo;

btnAgregarBodega.Enabled = campo;

bindingNavigatorEditItem.Enabled = !campo;

bindingNavigatorAddNewItem.Enabled = !campo;

bindingNavigatorDeleteItem.Enabled = !campo;

bindingNavigatorSaveItem.Enabled = campo;

bindingNavigatorCancelItem.Enabled = campo;

bindingNavigatorSearchItem.Enabled = !campo;

bindingNavigatorMoveFirstItem.Enabled = !campo;

bindingNavigatorMovePreviousItem.Enabled = !campo;

bindingNavigatorMoveNextItem.Enabled = !campo;

bindingNavigatorMoveLastItem.Enabled = !campo;

bindingNavigatorPositionItem.Enabled = !campo;

bindingNavigatorCountItem.Enabled = !campo;

}

Al botón Editar le ponemos:

private void bindingNavigatorEditItem\_Click(object sender, EventArgs e)

{

Habilitar(true);

}

Al botón Cancelar le ponemos:

private void bindingNavigatorCancelItem\_Click(object sender, EventArgs e)

{

this.productoBindingSource.CancelEdit();

errorProvider1.Clear();

Habilitar(false);

}

Creamos el método ValidarCampos

private bool Validarcampos()

{

errorProvider1.Clear();

if (descripcionTextBox.Text == string.Empty)

{

errorProvider1.SetError(descripcionTextBox, "Debe ingresar una Descripción");

descripcionTextBox.Focus();

return false;

}

if (iDDepartamentoComboBox.SelectedIndex == -1)

{

errorProvider1.SetError(iDDepartamentoComboBox, "Debe seleccionar un Departamento");

iDDepartamentoComboBox.Focus();

return false;

}

if (precioTextBox.Text == string.Empty)

{

errorProvider1.SetError(precioTextBox, "Debe ingresar un Precio");

precioTextBox.Focus();

return false;

}

decimal precio;

if (!decimal.TryParse(precioTextBox.Text,out precio))

{

errorProvider1.SetError(precioTextBox, "Debe ingresar un valor numérico");

precioTextBox.Focus();

return false;

}

if (precio <=0)

{

errorProvider1.SetError(precioTextBox, "Debe ingresar un valor mayor a 0");

precioTextBox.Focus();

return false;

}

errorProvider1.Clear();

if (iDIVAComboBox.SelectedIndex == -1)

{

errorProvider1.SetError(iDIVAComboBox, "Debe seleccionar un IVA");

iDIVAComboBox.Focus();

return false;

}

errorProvider1.Clear();

if (iDMedidaComboBox.SelectedIndex == -1)

{

errorProvider1.SetError(iDMedidaComboBox, "Debe seleccionar una Unidad");

iDMedidaComboBox.Focus();

return false;

}

if (medidaTextBox.Text == string.Empty)

{

errorProvider1.SetError(medidaTextBox, "Debe ingresar una Medida");

medidaTextBox.Focus();

return false;

}

float medida;

if (!float.TryParse(medidaTextBox.Text, out medida))

{

errorProvider1.SetError(medidaTextBox, "Debe ingresar un valor numérico");

medidaTextBox.Focus();

return false;

}

if (medida <= 0)

{

errorProvider1.SetError(medidaTextBox, "Debe ingresar un valor mayor a 0");

medidaTextBox.Focus();

return false;

}

return true;

}

Al botón Guardar le ponemos:

private void bindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

if (!Validarcampos()) return;

this.Validate();

this.productoBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

Habilitar(false);

}

Al botón Nuevo le ponemos:

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

Habilitar(true);

productoBindingSource.AddNew();

iDDepartamentoComboBox.Focus();

}

Al botón Borrar le ponemos:

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Registro actual?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

this.Validate();

this.productoBindingSource.RemoveAt(productoBindingSource.Position);

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

}

Agregamos:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| OpenFileDIalog | (Name) | openFileDIalog |

Bajamos unas imágenes de internet y las guardamos en C:\AppComercial\AppComercial\Imágenes

Ponemos en el código:

private void btnBuscarImagen\_Click(object sender, EventArgs e)

{

openFileDialog1.ShowDialog();

imagenTextBox.Text = openFileDialog1.FileName;

CargarImagen();

}

private void CargarImagen()

{

if(imagenTextBox.Text==string.Empty)

{

pbxImagen.Image = null;

}

else

{

if(File.Exists(imagenTextBox.Text))

{

pbxImagen.Load(imagenTextBox.Text);

}

}

}

Y agregamos:

private void frmProductos\_Load(object sender, EventArgs e)

{

// TODO: esta línea de código carga datos en la tabla 'dSAppComercial.Bodega' Puede moverla o quitarla según sea necesario.

this.bodegaTableAdapter.Fill(this.dSAppComercial.Bodega);

this.medidaTableAdapter.Fill(this.dSAppComercial.Medida);

this.iVATableAdapter.Fill(this.dSAppComercial.IVA);

this.departamentoTableAdapter.Fill(this.dSAppComercial.Departamento);

this.productoTableAdapter.Fill(this.dSAppComercial.Producto);

LlenarGrillas();

CargarImagen();

dgvDatos.AutoResizeColumns();

barrasDataGridView.AutoResizeColumns();

bodegasDataGridView.AutoResizeColumns();

}

…

private void bindingNavigatorMovePreviousItem\_Click(object sender, EventArgs e)

{

LlenarGrillas();

CargarImagen();

}

private void LlenarGrillas()

{

this.barraTableAdapter.FillBy(this.dSAppComercial.Barra,Convert.ToInt32(iDProductoTextBox.Text));

this.bodegaProductoTableAdapter.FillBy(this.dSAppComercial.BodegaProducto, Convert.ToInt32(iDProductoTextBox.Text));

}

private void bindingNavigatorMoveFirstItem\_Click(object sender, EventArgs e)

{

LlenarGrillas();

CargarImagen();

}

private void bindingNavigatorMoveNextItem\_Click(object sender, EventArgs e)

{

LlenarGrillas();

CargarImagen();

}

private void bindingNavigatorMoveLastItem\_Click(object sender, EventArgs e)

{

LlenarGrillas();

CargarImagen();

}

private void dgvDatos\_CellClick(object sender, DataGridViewCellEventArgs e)

{

LlenarGrillas();

CargarImagen();

}

Creamos el formulario **frmBarras**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmBarras |
|  | Text | Códigos de Barras |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | AcceptButton | btnAceptar |
|  | CancelButton | btnCancelar |
|  | Size | 302; 190 |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |
|  |  |  |
| Label | (Name) | label1 |
|  | Text | Código: |
|  |  |  |
| TextBox | (Name) | barraTextBox |
|  |  |  |
| Copiamos y pegamos los botones Aceptar y Cancelar desde el frmLogin | | |

Hacemos Ver – Orden de Tabulación y le damos el orden que queremos.

Para abrir este formulario, en el frmProductos agregamos este código al botón Agregar de Barras:

private void btnAgregarBarra\_Click(object sender, EventArgs e)

{

frmBarras miForm = new frmBarras();

miForm.ShowDialog();

}

En el DataSet DSAppComercial del Proyecto CADAppComercial, agregamos la Tabla Barra

Le agregamos una Consulta que devuelva un solo valor:

SELECT 1 FROM Barra

WHERE Barra=@Barra

y le ponemos como nombre **BarraExisteBarra**

Creamos otra Consulta tipo INSERT

INSERT INTO [dbo].[Barra] ([IDProducto], [Barra]) VALUES (@IDProducto, @Barra);

y le ponemos como nombre **BarraInsertBarra**

Creamos otra Consulta tipo DELETE

DELETE FROM [dbo].[Barra] WHERE ([Barra] = @Barra)

y le ponemos como nombre **BarraDeleteBarra**

Creamos una Clase llamada **CADBarra**

using CADAppComercial.DSAppComercialTableAdapters;

namespace CADAppComercial

{

public class CADBarra

{

private static BarraTableAdapter adaptador = new BarraTableAdapter();

public static bool ExisteBarra(long Barra)

{

if (adaptador.BarraExisteBarra(Barra) == null)

{

return false;

}

else

{

return true;

}

}

public static void BarraInsert(

int IDProducto,

long Barra)

{

adaptador.BarraInsertBarra(IDProducto,Barra);

}

public static void BarraDelete(long Barra)

{

adaptador.BarraDeleteBarra(Barra);

}

}

}

Ponemos en el código del frmBarras:

using CADAppComercial;

using System;

using System.Windows.Forms;

namespace AppComercial

{

public partial class frmBarras : Form

{

private long barra=0;

public long Barra { get => barra; }

public frmBarras()

{

InitializeComponent();

}

private void btnAceptar\_Click(object sender, EventArgs e)

{

errorProvider1.Clear();

if(barraTextBox.Text==string.Empty)

{

errorProvider1.SetError(barraTextBox, "Debe ingresar un Código de Barras");

barraTextBox.Focus();

return;

}

if (!long.TryParse(barraTextBox.Text,out barra))

{

errorProvider1.SetError(barraTextBox, "El Código de Barras debe ser un N° entero");

barraTextBox.Focus();

return;

}

if (barra <1000000)

{

errorProvider1.SetError(barraTextBox, "El Código de Barras debe ser un N° de por lo menos 7 cifras");

barraTextBox.Focus();

return;

}

if (CADBarra.ExisteBarra(barra))

{

errorProvider1.SetError(barraTextBox, "Este Código de Barras ya existe para otro Producto");

barraTextBox.Focus();

return;

}

this.Close();

}

private void btnCancelar\_Click(object sender, EventArgs e)

{

barra = 0;

this.Close();

}

}

}

Y en el **frmProducto** hacemos:

private void btnAgregarBarra\_Click(object sender, EventArgs e)

{

frmBarras miForm = new frmBarras();

miForm.ShowDialog();

if (miForm.Barra == 0) return;

CADBarra.BarraInsert(Convert.ToInt32(iDProductoTextBox.Text), miForm.Barra);

this.barraTableAdapter.FillBy(this.dSAppComercial.Barra, Convert.ToInt32(iDProductoTextBox.Text));

}

private void btnEliminarBarra\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Código de Barras?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

long barra = (long) barrasDataGridView.Rows[barraBindingSource.Position].Cells[0].Value;

CADBarra.BarraDelete(barra);

this.barraTableAdapter.FillBy(this.dSAppComercial.Barra, Convert.ToInt32(iDProductoTextBox.Text));

}

Cremoas el formulario **frmParametrosBodega**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmParametrosBodega |
|  | Text | ParámetrosBodega |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | AcceptButton | btnAceptar |
|  | CancelButton | btnCancelar |
|  | Size | 592; 316 |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |
|  |  |  |
| Label | (Name) | label1 |
|  | Text | Bodega |
|  |  |  |
| ComboBox | (Name) | bodegaComboBox |
|  | Llenamos el origen de datos |  |
|  |  |  |
| Label | (Name) | label2 |
|  | Text | Stock |
|  |  |  |
| TextBox | (Name) | stockTextBox |
|  | ReadOnly | True |
|  |  |  |
| Label | (Name) | label3 |
|  | Text | Cantidad Mínima: |
|  |  |  |
| NumericUpDown | (Name) | minimoNumericUpDown |
|  | Minimum | 1 |
|  | Maximum | 999999 |
|  |  |  |
| Label | (Name) | Label4 |
|  | Text | Cantidad Máxima: |
|  |  |  |
| NumericUpDown | (Name) | maximoNumericUpDown |
|  | Minimum | 1 |
|  | Maximum | 999999 |
|  |  |  |
| Label | (Name) | label5 |
|  | Text | Días Repos.: |
|  |  |  |
| NumericUpDown | (Name) | diasReposicionNumericUpDown |
|  | Minimum | 1 |
|  | Maximum | 999999 |
|  |  |  |
| Label | (Name) | label6 |
|  | Text | Cant. Mín.: |
|  |  |  |
| NumericUpDown | (Name) | Cantidad MinimaNumericUpDown |
|  | Minimum | 1 |
|  | Maximum | 999999 |
|  |  |  |
| Copiamos y pegamos los botones Aceptar y Cancelar desde el frmLogin | | |

En el DataSet DSAppComercial del Proyecto CADAppComercial, agregamos la Tabla BodegaProducto

Le agregamos una Consulta que devuelva filas:

SELECT IDBodega, IDProducto, Stock, Minimo, Maximo, DiasReposicion, CantidadMinima

FROM dbo.BodegaProducto

WHERE IDBodega=@IDBodega AND IDProducto=@IDProducto

y le ponemos como nombre **BodegaProductoGetBodegaProductoByIDBodegaAndIDProducto**

Creamos otra Consulta tipo INSERT

INSERT INTO [dbo].[BodegaProducto] ([IDBodega], [IDProducto], [Stock], [Minimo], [Maximo], [DiasReposicion], [CantidadMinima]) VALUES (@IDBodega, @IDProducto, 0, @Minimo, @Maximo, @DiasReposicion, @CantidadMinima);

y le ponemos como nombre **BodegaProductoInsert**

Creamos otra Consulta tipo UPDATE

UPDATE [dbo].[BodegaProducto] SET

[Minimo] = @Minimo,

[Maximo] = @Maximo,

[DiasReposicion] = @DiasReposicion,

[CantidadMinima] = @CantidadMinima

WHERE [IDBodega] = @IDBodega AND [IDProducto] = @IDProducto

y le ponemos como nombre **BodegaProductoUpdate**

Creamos una Clase llamada **CADBodegaProducto**

|  |  |
| --- | --- |
| **CADBodegaProducto** | **Comentarios** |
| using CADAppComercial.DSAppComercialTableAdapters;  namespace CADAppComercial  {  public class CADBodegaProducto  {  public int IDBodega { get; set; }  public int IDProducto { get; set; }  public float Stock { get; set; }  public float Minimo { get; set; }  public float Maximo { get; set; }  public int DiasReposicion { get; set; }  public float CantidadMinima { get; set; }  private static BodegaProductoTableAdapter adaptador = new BodegaProductoTableAdapter();  public static CADBodegaProducto BodegaProductoGetBodegaProductoByIDBodegaAndIDProducto(int IDBodega, int IDProducto)  {  CADBodegaProducto miBodegaProducto = null;  DSAppComercial.BodegaProductoDataTable miTabla = adaptador.BodegaProductoGetBodegaProductoByIDBodegaAndIDProducto(IDBodega, IDProducto);  if (miTabla.Rows.Count==0)  {  return miBodegaProducto;  }  else  {  DSAppComercial.BodegaProductoRow miRegistro = (DSAppComercial.BodegaProductoRow)miTabla.Rows[0];  miBodegaProducto = new CADBodegaProducto();  miBodegaProducto.CantidadMinima = (float)miRegistro.CantidadMinima;  miBodegaProducto.DiasReposicion = miRegistro.DiasReposicion;  miBodegaProducto.IDBodega = miRegistro.IDBodega;  miBodegaProducto.IDProducto = miRegistro.IDProducto;  miBodegaProducto.Minimo = (float)miRegistro.Minimo;  miBodegaProducto.Maximo = (float)miRegistro.Maximo;  miBodegaProducto.Stock = (float)miRegistro.Stock;  return miBodegaProducto;  }  }  public static void BodegaProductoUpdate(  int IDBodega,  int IDProducto,  float Minimo,  float Maximo,  int DiasReposicion,  float CantidadMinima)  {  try  {  adaptador.BodegaProductoInsert(IDBodega, IDProducto, Minimo, Maximo, DiasReposicion, CantidadMinima);  }  catch (System.Exception)  {  adaptador.BodegaProductoUpdate(Minimo, Maximo, DiasReposicion, CantidadMinima,IDBodega,IDProducto);  }  }  }  } | Creamos las propiedades que son los campos de la Tabla BodegaProducto  Creamos el adaptador  Creamos el objeto CADBodegaProducto = null  Obtenemos la fila de la Tabla BodegaProducto y la guardamos en miTabla  Si la Tabla no tiene filas devolvemos el objeto miBodegaProducto nulo  Y sino, convertimos el primer registro de la tabla (miRegistro) en el objeto miBodegaProducto que es lo que se retorna  En un solo método ponemos el INSERT y el UPDATE (usando try-catch) |

En el frmParametrosBodega hacemos:

|  |  |
| --- | --- |
| **frmParametrosBodega** | **Comentarios** |
| using CADAppComercial;  using System;  using System.Windows.Forms;  namespace AppComercial  {  public partial class frmParametrosBodega : Form  {  private int idProducto;  public int IDProducto  {  set { idProducto = value; }  }  public frmParametrosBodega()  {  InitializeComponent();  }  private void frmParametrosBodega\_Load(object sender, EventArgs e)  {  this.bodegaTableAdapter.Fill(this.dSAppComercial.Bodega);  bodegaComboBox.SelectedIndex = -1;  }  private void bodegaComboBox\_SelectedIndexChanged(object sender, EventArgs e)  {  if (bodegaComboBox.SelectedIndex == -1) return;    CADBodegaProducto miBodegaProducto = CADBodegaProducto.BodegaProductoGetBodegaProductoByIDBodegaAndIDProducto((int)bodegaComboBox.SelectedValue,idProducto) ;  if (miBodegaProducto == null)  {  stockTextBox.Text = string.Empty;  cantidadMinimaNumericUpDown.Value = 1;  diasReposicionNumericUpDown.Value = 1;  minimoNumericUpDown.Value = 1;  maximoNumericUpDown.Value = 1;  diasReposicionNumericUpDown.Value = 1;  return;  }  stockTextBox.Text = miBodegaProducto.Stock.ToString();  cantidadMinimaNumericUpDown.Value = (decimal)miBodegaProducto.Minimo;  diasReposicionNumericUpDown.Value = miBodegaProducto.DiasReposicion;  minimoNumericUpDown.Value = (decimal) miBodegaProducto.Minimo;  maximoNumericUpDown.Value = (decimal) miBodegaProducto.Maximo;  }  private void btnAceptar\_Click(object sender, EventArgs e)  {  errorProvider1.Clear();  if (bodegaComboBox.SelectedIndex == -1)  {  errorProvider1.SetError(bodegaComboBox, "Debe seleccionar una Bodega");  bodegaComboBox.Focus();  return;  }  CADBodegaProducto.BodegaProductoUpdate((int)bodegaComboBox.SelectedValue, idProducto,(float)minimoNumericUpDown.Value, (float)maximoNumericUpDown.Value, (int)diasReposicionNumericUpDown.Value, (float)cantidadMinimaNumericUpDown.Value);  this.Close();  }  }  } | Propiedad para recibir el IDProducto desde frmProducto  Al elegir una Bodega llena los casilleros con los valores encontrados, o los pone en valores iniciales si es una bodega que no tenía parámetros.  Guarda (Insert o Update) los parámetros de la bodega seleccionada |

En **frmProductos** hacemos, para abrir el **frmParametrosBodega**:

private void btnAgregarBodega\_Click(object sender, EventArgs e)

{

frmParametrosBodega miForm = new frmParametrosBodega();

miForm.IDProducto = Convert.ToInt32(iDProductoTextBox.Text);

miForm.ShowDialog();

this.bodegaProductoTableAdapter.FillBy(this.dSAppComercial.BodegaProducto, Convert.ToInt32(iDProductoTextBox.Text));

}

## Crear Barra de Herramientas en el Menú Principal

Agregamos un ToolStrip al Menú Principal.

Le agregamos 3 botones

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| ToolStrip |  |  |
| Button | (Name) | tsbClientes |
|  | DisplayStyle | ImageAndText |
|  | Text | Clientes |
|  | Image | Cargamos el ícono |
|  |  |  |
| Button | (Name) | tsbProveedores |
|  | DisplayStyle | ImageAndText |
|  | Text | Proveedores |
|  | Image | Cargamos el ícono |
|  |  |  |
| Button | (Name) | tsbProductos |
|  | DisplayStyle | ImageAndText |
|  | Text | Proveedores |
|  | Image | Cargamos el ícono |

# 34 al 36 CRUD Usuarios

En **DSAppComercial** del Proyecto **AppComercial**, desde el Explorador de Servidores arrastramos la Tabla **Usuarios** y **Rol**

Creamos el Formulario **frmUsuarios**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmUsuarios |
|  | Text | Usuarios |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | Size | 1316; 696 |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |

En el evento Load agregamos

dgvDatos.AutoResizeColumns();

Arrastramos desde Orígenes de Datos la Tabla Usuarios y de esta manera creamos la grilla.

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| DataGridView | (Name) | dgvDatos |
|  | Anchor | Los 4 costados marcados |
|  | DefaultCellStyle | Font Arial Narrow 7 |
|  | AlternatingRowsDefaultCells | Font Arial Narrow 7 y BackColor=192; 255; 255 |

Hacemos clic en el triangulito negro que sale arriba a la derecha de la grilla y ahí hacemos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ahí elegimos Editar Columnas y cambiamos/corregimos los Nombres de las Columnas:

Rol debe ser tipo ComboBox:

Para mostrar este formulario hacemos en frmPrincipal:

Agregamos un Label y un TextBox para la confirmación de Clave

A los TextBox Clave y Confirmación le ponemos PasswordChar \*

private void productosToolStripMenuItem1\_Click(object sender, System.EventArgs e)

{

frmUsuarios miForm = new frmUsuarios ();

miForm.MdiParent = this;

miForm.Show();

}

Desde Orígenes de Datos arrastramos campo por campo de la Tabla Usuarios

A fechaModificacionClaveDateTimePicker ke ponemos Format short

Hacemos Ver – Orden de Tabulación y le damos el orden que queremos.

Al ComBox Rol le ponemos el origen de datos

Y en las Propiedades hacemos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

A todos los TextBox les ponemos ReadOnly = True Al ComboBox le ponemos Enabled = False

En la Barra de Herramientas:

* Agregamos los botones de Editar, Cancelar y Buscar
* Les ponemos nombres según el estándar de los 3 botones ya existentes (nuevo, grabar y borrar)
* Cambiamos las imágenes de todos los íconos
* Cambiamos el orden los íconos a Editar-Nuevo-Borrar-Grabar-Cancelar-Buscar
* Les ponemos Enabled = True a todos excepto a Grabar y Cancelar que les ponemos Enabled = False
* Les ponemos los ToolTipText

A todos los eventos de cambios de registros le ponemos

confirmacionTextBox.Text = claveTextBox.Text;

Creamos el método Habilitar:

private void Habilitar(bool campo)

{

iDUsuarioTextBox.ReadOnly = false;

iDRolComboBox.Enabled = campo;

nombresTextBox.ReadOnly = !campo;

apellidosTextBox.ReadOnly = !campo;

claveTextBox.ReadOnly = !campo;

confirmacionTextBox.ReadOnly = !campo;

correoTextBox.ReadOnly = !campo;

bindingNavigatorEditItem.Enabled = !campo;

bindingNavigatorAddNewItem.Enabled = !campo;

bindingNavigatorDeleteItem.Enabled = !campo;

bindingNavigatorSaveItem.Enabled = campo;

bindingNavigatorCancelItem.Enabled = campo;

bindingNavigatorMoveFirstItem.Enabled = !campo;

bindingNavigatorMovePreviousItem.Enabled = !campo;

bindingNavigatorMoveNextItem.Enabled = !campo;

bindingNavigatorMoveLastItem.Enabled = !campo;

bindingNavigatorPositionItem.Enabled = !campo;

bindingNavigatorCountItem.Enabled = !campo;

}

Al botón Editar le ponemos:

private void bindingNavigatorEditItem\_Click(object sender, EventArgs e)

{

Habilitar(true);

iDUsuarioTextBox.ReadOnly = true;

nuevo =false;

iDRolComboBox.Focus();

}

Al botón Cancelar le ponemos:

private void bindingNavigatorCancelItem\_Click(object sender, EventArgs e)

{

this.usuarioBindingSource.CancelEdit();

errorProvider1.Clear();

Habilitar(false);

}

En el DataSet DSAppComercial del Proyecto CADAppComercial, agregamos la Tabla Usuarios

Le agregamos una Consulta que devuelva un solo valor:

SELECT 1 FROM Usuario

WHERE IDUsuario=@IDUsuario

y le ponemos como nombre **UsuarioExisteUsuario**

En la Clase **CADUsuario** agregamos el método **UsuarioExisteusuario**

using CADAppComercial.DSAppComercialTableAdapters;

namespace CADAppComercial

{

public class CADUsuario

{

private static UsuarioTableAdapter adaptador = new UsuarioTableAdapter();

public static bool ValidaUsuario(string IDUsuario, string Clave)

{

if (adaptador.ValidaUsuario(IDUsuario, Clave) == null)

{

return false;

}

else

{

return true;

}

}

public static bool ExisteUsuario(string IDUsuario)

{

if (adaptador.UsuarioExisteUsuario(IDUsuario) == null)

{

return false;

}

else

{

return true;

}

}

}

}

Creamos el método ValidarCampos

private bool Validarcampos()

{

errorProvider1.Clear();

if (nuevo)

{

if (iDUsuarioTextBox.Text == string.Empty)

{

errorProvider1.SetError(iDUsuarioTextBox, "Debe ingresar un IDUsuario");

iDUsuarioTextBox.Focus();

return false;

}

if (CADUsuario.ExisteUsuario(iDUsuarioTextBox.Text))

{

errorProvider1.SetError(iDUsuarioTextBox, "Este IDUsuario ya existe");

iDUsuarioTextBox.Focus();

return false;

}

}

if (iDRolComboBox.SelectedIndex == -1)

{

errorProvider1.SetError(iDRolComboBox, "Debe seleccionar un Rol");

iDRolComboBox.Focus();

return false;

}

if (nombresTextBox.Text == string.Empty)

{

errorProvider1.SetError(nombresTextBox, "Debe ingresar un Nombre");

nombresTextBox.Focus();

return false;

}

if (apellidosTextBox.Text == string.Empty)

{

errorProvider1.SetError(apellidosTextBox, "Debe ingresar un Apellido");

apellidosTextBox.Focus();

return false;

}

if (claveTextBox.Text == string.Empty)

{

errorProvider1.SetError(claveTextBox, "Debe ingresar una Clave");

claveTextBox.Focus();

return false;

}

if (confirmacionTextBox.Text == string.Empty)

{

errorProvider1.SetError(confirmacionTextBox, "Debe ingresar una Confirmación de Clave");

confirmacionTextBox.Focus();

return false;

}

if (claveTextBox.Text != confirmacionTextBox.Text)

{

errorProvider1.SetError(claveTextBox, "La Clave y su Confirmación no son iguales");

errorProvider1.SetError(confirmacionTextBox, "La Clave y su Confirmación no son iguales");

claveTextBox.Focus();

return false;

}

if (correoTextBox.Text == string.Empty)

{

errorProvider1.SetError(correoTextBox, "Debe ingresar un Correo");

correoTextBox.Focus();

return false;

}

RegexUtilities regexUtilities = new RegexUtilities();

if (!regexUtilities.IsValidEmail(correoTextBox.Text))

{

errorProvider1.SetError(correoTextBox, "El correo ingresado no es válido");

correoTextBox.Focus();

return false;

}

return true;

}

Al botón Guardar le ponemos:

private void bindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

if (!Validarcampos()) return;

this.Validate();

fechaModificacionClaveDateTimePicker.Value = DateTime.Now;

this.usuarioBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

Habilitar(false);

}

En las propiedades de usuarioBindingNavigator hacemos AddNewItem = (ninguno) y DeleteItem = (ninguno)

Al botón Nuevo le ponemos:

Habilitar(true);

nuevo = true;

iDUsuarioTextBox.ReadOnly = false;

confirmacionTextBox.Clear();

fechaModificacionClaveDateTimePicker.Value = DateTime.Now;

usuarioBindingSource.AddNew();

iDUsuarioTextBox.Focus();

Al botón Borrar le ponemos:

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Registro actual?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

this.Validate();

this.usuarioBindingSource.RemoveAt(usuarioBindingSource.Position);

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

}

# 37 al 43 – Seguridad de la Aplicación

En el DataSet DSAppComercial del Proyecto CADAppComercial, en la Tabla Usuarios agregamos una Consulta que devuelva filas:

SELECT IDUsuario, Nombres, Apellidos, Clave, FechaModificacionClave, IDRol, Correo

FROM dbo.Usuario

WHERE IDUsuario=@IDUsuario

y le ponemos como nombre **UsuarioGetUsuarioByIDUsuario**

En **CADUsuario** agregamos:

using CADAppComercial.DSAppComercialTableAdapters;

using System;

namespace CADAppComercial

{

public class CADUsuario

{

public string IDUsuario { get; set; }

public string Nombres { get; set; }

public string Apellidos { get; set; }

public string Clave { get; set; }

public DateTime FechaModificacionClave { get; set; }

public int IDRol { get; set; }

public string Correo { get; set; }

private static UsuarioTableAdapter adaptador = new UsuarioTableAdapter();

public static bool ValidaUsuario(string IDUsuario, string Clave)

{

if (adaptador.ValidaUsuario(IDUsuario, Clave) == null)

{

return false;

}

else

{

return true;

}

}

public static bool ExisteUsuario(string IDUsuario)

{

if (adaptador.UsuarioExisteUsuario(IDUsuario) == null)

{

return false;

}

else

{

return true;

}

}

public static CADUsuario UsuarioGetUsuarioByIDUsuario(string IDUsuario)

{

CADUsuario miUsuario = null;

DSAppComercial.UsuarioDataTable miTabla = adaptador.UsuarioGetUsuarioByIDUsuario(IDUsuario);

if (miTabla.Rows.Count == 0)

{

return miUsuario;

}

else

{

DSAppComercial.UsuarioRow miRegistro = (DSAppComercial.UsuarioRow)miTabla.Rows[0];

miUsuario = new CADUsuario();

miUsuario.Apellidos= miRegistro.Apellidos;

miUsuario.Clave = miRegistro.Clave;

miUsuario.Correo = miRegistro.Correo;

miUsuario.FechaModificacionClave = miRegistro.FechaModificacionClave;

miUsuario.IDRol = miRegistro.IDRol;

miUsuario.IDUsuario = miRegistro.IDUsuario;

miUsuario.Nombres = miRegistro.Nombres;

return miUsuario;

}

}

}

}

En el código del **frmPrincipal** agregamos:

using CADAppComercial;

using System.Windows.Forms;

namespace AppComercial

{

public partial class frmPrincipal : Form

{

private CADUsuario usuarioLogueado;

public CADUsuario UsuarioLogueado

{

get => usuarioLogueado;

set => usuarioLogueado = value;

}

………………..

En el código del **frmLogin** agregamos:

……

frmPrincipal miForm = new frmPrincipal();

miForm.UsuarioLogueado = CADUsuario.UsuarioGetUsuarioByIDUsuario(txtUsuario.Text);

miForm.Show();

this.Hide();

…..

Al **frmPrincipal** le agregamos:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| StatusStrip | (Name) | statusStrip1 |
| toolStripStatusLabel | (Name) | nombresUsuarioToolStripStatusLabel |

En el evento Load del **frmPrincipal** ponemos:

private void frmPrincipal\_Load(object sender, System.EventArgs e)

{

nombresUsuarioToolStripStatusLabel.Text = "Usuario: " + usuarioLogueado.Nombres + " " + usuarioLogueado.Apellidos;

}

## Cambio de Clave

Creamos el Formulario **frmCambioClave**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmProveedores |
|  | Text | Proveedores |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | Size | 327; 272 |
|  | AcceptButton | btnAceptar |
|  | CancelButton | btnCancelar |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |
|  |  |  |
| Label | (Name) | label1 |
|  | Text | Clave anterior: |
|  |  |  |
| TextBox | (Name) | claveAnteriorTextBox |
|  | PasswordChar | \* |
|  |  |  |
| Label | (Name) | label2 |
|  | Text | Nueva Clave: |
|  | PasswordChar | \* |
|  |  |  |
| TextBox | (Name) | nuevaClaveTextBox |
|  |  |  |
| Label | (Name) | label1 |
|  | Text | Confirmación: |
| PasswordChar | \* |  |
|  |  |  |
| TextBox | (Name) | confirmacionTextBox |
|  |  |  |
| Copiamos y pegamos los botones Aceptar y Cancelar desde el frmLogin | | |

Para abrirlo **de forma modal** desde el Formulario Principal, le ponemos al código del **frmPrincipal**:

private void usuarioToolStripMenuItem\_Click(object sender, System.EventArgs e)

{

frmUsuarios miForm = new frmUsuarios();

miForm.UsuarioLogueado = usuarioLogueado;

miForm.ShowDialog();

}

Hacemos Ver – Orden de Tabulación y le damos el orden que queremos.

En el DataSet DSAppComercial del Proyecto CADAppComercial, en la Tabla Usuarios agregamos una Consulta UPDATE:

UPDATE [dbo].[Usuario] SET

[Clave] = @Clave,

[FechaModificacionClave] = @FechaModificacionClave

WHERE [IDUsuario] = @IDUsuario

y le ponemos como nombre **UsuarioUpdateClave**

En **CADUsuario** agregamos:

public static void UsuarioUpdateClave(

string Clave,

string IDUsuario

)

{

adaptador.UsuarioUpdateClave(Clave, DateTime.Now, IDUsuario);

}

Al botón Aceptar del **frmCambioClave** le ponemos:

private void btnAceptar\_Click(object sender, EventArgs e)

{

errorProvider1.Clear();

if (claveAnteriorTextBox.Text == string.Empty)

{

errorProvider1.SetError(claveAnteriorTextBox, "Debe ingresar la Clave actual");

claveAnteriorTextBox.Focus();

return;

}

if (claveAnteriorTextBox.Text != usuarioLogueado.Clave)

{

errorProvider1.SetError(claveAnteriorTextBox, "Clave actual incorrecta");

claveAnteriorTextBox.Focus();

return ;

}

if (nuevaClaveTextBox.Text == string.Empty)

{

errorProvider1.SetError(nuevaClaveTextBox, "Debe ingresar la Nueva Clave");

nuevaClaveTextBox.Focus();

return;

}

if (confirmacionTextBox.Text == string.Empty)

{

errorProvider1.SetError(confirmacionTextBox, "Debe ingresar la Confirmacion");

confirmacionTextBox.Focus();

return;

}

if (nuevaClaveTextBox.Text != confirmacionTextBox.Text)

{

errorProvider1.SetError(nuevaClaveTextBox, "La nueva Clave y su confirmación no coinciden");

errorProvider1.SetError(confirmacionTextBox, "La nueva Clave y su confirmación no coinciden");

confirmacionTextBox.Focus();

return;

}

CADUsuario.UsuarioUpdateClave(nuevaClaveTextBox.Text, usuarioLogueado.IDUsuario);

MessageBox.Show("Cambio de Clave realizado con éxito", "Confirmación", MessageBoxButtons.OK, MessageBoxIcon.Information);

usuarioLogueado.Clave = nuevaClaveTextBox.Text;

usuarioLogueado.FechaModificacionClave = DateTime.Now;

this.Close();

}

## Obligar a cambiar la Clave después de un cierto tiempo

En el **frmPrincipal** hacemos:

private void frmPrincipal\_Load(object sender, System.EventArgs e)

{

nombresUsuarioToolStripStatusLabel.Text = "Usuario: " + usuarioLogueado.Nombres + " " + usuarioLogueado.Apellidos;

VerificaCambioClave(sender, e);

}

private void VerificaCambioClave(object sender, System.EventArgs e)

{

if(usuarioLogueado.FechaModificacionClave.AddDays(30)<DateTime.Now)

{

cambioDeClaveToolStripMenuItem\_Click(sender, e);

}

}

En el **frmCambioClave**, sacamos el código del evento btnCancelar\_Click, y ponemos este otro método al evento **frmCambioClave\_FormClosing:**

private void frmCambioClave\_FormClosing(object sender, FormClosingEventArgs e)

{

if (usuarioLogueado.FechaModificacionClave.AddDays(30) < DateTime.Now)

{

errorProvider1.SetError(nuevaClaveTextBox, "Debe cambiar la Clave");

nuevaClaveTextBox.Focus();

e.Cancel = true;

return;

}

}

## Cambio de Usuario

Creamos el Formulario **frmCambioUsuario**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmProveedores |
|  | Text | Proveedores |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | Size | 413;280 |
|  | AcceptButton | btnAceptar |
|  | CancelButton | btnCancelar |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |

Para abrirlo **de forma modal** desde el Formulario Principal, le ponemos al código del **frmPrincipal**:

private void cambioDeUsuarioToolStripMenuItem\_Click(object sender, EventArgs e)

{

frmCambioUsuario miForm = new frmCambioUsuario();

miForm.UsuarioLogueado = usuarioLogueado;

miForm.ShowDialog();

if(miForm.UsuarioLogueado!=null)

{

usuarioLogueado = miForm.UsuarioLogueado;

}

nombresUsuarioToolStripStatusLabel.Text = "Usuario: " + usuarioLogueado.Nombres + " " + usuarioLogueado.Apellidos;

}

Hacemos Ver – Orden de Tabulación y le damos el orden que queremos.

Al botón **Aceptar** del **frmCambioUsuario** le ponemos:

private void btnAceptar\_Click(object sender, EventArgs e)

{

errorProvider1.Clear();

if (txtUsuario.Text == string.Empty)

{

errorProvider1.SetError(txtUsuario, "Debe ingresar un Usuario");

txtUsuario.Focus();

return;

}

if (txtClave.Text == string.Empty)

{

errorProvider1.SetError(txtClave, "Debe ingresar una Clave");

txtClave.Focus();

return;

}

if (!CADUsuario.ValidaUsuario(txtUsuario.Text, txtClave.Text))

{

MessageBox.Show("Usuario o Clave no válidos", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtUsuario.Text = string.Empty;

txtClave.Text = string.Empty;

txtUsuario.Focus();

return;

}

usuarioLogueado = CADUsuario.UsuarioGetUsuarioByIDUsuario(txtUsuario.Text);

this.Close();

}

Al botón **Cancelar** del **frmCambioUsuario** le ponemos:

private void btnCancelar\_Click(object sender, EventArgs e)

{

usuarioLogueado = null;

this.Close();

}

## Permisos según Rol

Hay 3 Roles:

Tabla

Descripción generada automáticamente

En la Tabla **PermisoRol** se define lo que cada Rol puede hacer en cada Formulario:

Tabla

Descripción generada automáticamente

En el DataSet DSAppComercial del Proyecto **CADAppComercial**, agregamos la Tabla **PermisoRol**.

Le agregamos las Consultas que devuelve un solo valor:

SELECT 1 FROM PermisoRol

WHERE IDRol=@IDRol AND PuedeVer=1 AND Formulario=@Formulario

y le ponemos como nombre **PermisoRolPuedeVer**

SELECT 1 FROM PermisoRol

WHERE IDRol=@IDRol AND PuedeModificar=1 AND Formulario=@Formulario

y le ponemos como nombre **PermisoRolPuedeModificar**

SELECT 1 FROM PermisoRol

WHERE IDRol=@IDRol AND PuedeBorrar=1 AND Formulario=@Formulario

y le ponemos como nombre **PermisoRolPuedeBorrar**

Creamos una Clase llamada **CADPermisoRol**

using CADAppComercial.DSAppComercialTableAdapters;

namespace CADAppComercial

{

public class CADPermisoRol

{

private static PermisoRolTableAdapter adaptador = new PermisoRolTableAdapter();

public static bool PermisoRolPuedeVer(int IDRol, string Formulario)

{

if (adaptador.PermisoRolPuedeVer(IDRol,Formulario)==null)

{

return false;

}

else

{

return true;

}

}

public static bool PermisoRolPuedeModificar(int IDRol, string Formulario)

{

if (adaptador.PermisoRolPuedeModificar(IDRol, Formulario) == null)

{

return false;

}

else

{

return true;

}

}

public static bool PermisoRolPuedeBorrar(int IDRol, string Formulario)

{

if (adaptador.PermisoRolPuedeBorrar(IDRol, Formulario) == null)

{

return false;

}

else

{

return true;

}

}

}

}

En **frmPrincipal** ponemos:

private void frmPrincipal\_Load(object sender, System.EventArgs e)

{

nombresUsuarioToolStripStatusLabel.Text = "Usuario: " + usuarioLogueado.Nombres + " " + usuarioLogueado.Apellidos;

VerificaCambioClave(sender, e);

VerificarPermisos();

}

private void VerificarPermisos()

{

proveedoresToolStripMenuItem.Visible = CADPermisoRol.PermisoRolPuedeVer(usuarioLogueado.IDRol, "frmProveedores");

tsbProveedores.Visible = CADPermisoRol.PermisoRolPuedeVer(usuarioLogueado.IDRol, "frmProveedores");

}

……………..

private void cambioDeUsuarioToolStripMenuItem\_Click(object sender, EventArgs e)

{

frmCambioUsuario miForm = new frmCambioUsuario();

miForm.UsuarioLogueado = usuarioLogueado;

miForm.ShowDialog();

if(miForm.UsuarioLogueado!=null)

{

usuarioLogueado = miForm.UsuarioLogueado;

nombresUsuarioToolStripStatusLabel.Text = "Usuario: " + usuarioLogueado.Nombres + " " + usuarioLogueado.Apellidos;

VerificarPermisos();

}

}

Al **frmProveedores** le ponemos:

private CADUsuario usuarioLogueado;

public CADUsuario UsuarioLogueado

{

get => usuarioLogueado;

set => usuarioLogueado = value;

}

Y en el **frmPrincipal** cuando llamamos al **frmProveedores** agregamos:

private void proveedoresToolStripMenuItem\_Click(object sender, System.EventArgs e)

{

frmProveedores miForm = new frmProveedores();

miForm.MdiParent = this;

miForm.UsuarioLogueado = usuarioLogueado;

miForm.Show();

}

En el **frmProveedores** hacemos:

private void frmProveedores\_Load(object sender, EventArgs e)

{

this.tipoDocumentoTableAdapter.Fill(this.dSAppComercial.TipoDocumento);

this.proveedorTableAdapter.Fill(this.dSAppComercial.Proveedor);

dgvDatos.AutoResizeColumns();

VerificarPermisos();

}

private void VerificarPermisos()

{

bindingNavigatorAddNewItem.Enabled = CADPermisoRol.PermisoRolPuedeModificar(usuarioLogueado.IDRol, this.Name);

bindingNavigatorEditItem.Enabled = CADPermisoRol.PermisoRolPuedeModificar(usuarioLogueado.IDRol, this.Name);

bindingNavigatorDeleteItem.Enabled = CADPermisoRol.PermisoRolPuedeBorrar(usuarioLogueado.IDRol, this.Name);

}

……………

private void bindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

if (!Validarcampos()) return;

this.Validate();

this.proveedorBindingSource.EndEdit();

try

{

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

errorProvider1.Clear();

}

catch (Exception)

{

errorProvider1.SetError(documentoTextBox, "Este N° de Documento ya está asignado");

documentoTextBox.Focus();

return;

}

Habilitar(false);

VerificarPermisos();

}

private void bindingNavigatorCancelItem\_Click(object sender, EventArgs e)

{

this.proveedorBindingSource.CancelEdit();

errorProvider1.Clear();

Habilitar(false);

VerificarPermisos();

}

# 44 al 59 - Compras

## Poner íconos a la aplicación

Descargar íconos como PNG de 32x32 y descargar en formato ICO

En los menúes de frmPrincipal se carga el PNG en la propiedad Image

En cada formulario se carga el ICO en la propiedad Icon

## Formulario Compras

Creamos el Formulario **frmCompras**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmCompras |
|  | Text | Compras |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | Size | 1316; 696 |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |
|  |  |  |
| Label | (Name) | label1 |
|  | Text | Fecha: |
|  |  |  |
| DateTimePicker | (Name) | fechaDateTimePicker |
|  |  |  |
| Label | (Name) | label2 |
|  | Text | Proveedor: |
|  |  |  |
| ComboBox | (Name) | proveedorComboBox |
|  |  |  |
| Button | (Name) | btnBuscarProveedor |
|  | Text | … |
|  |  |  |
| Label | (Name) | label3 |
|  | Text | Bodega: |
|  |  |  |
| ComboBox | (Name) | bodegaComboBox |

Cuando hacemos una Compra afectamos las Tablas

* Compra
* CompraDetalle
* BodegaProducto
* Kardex

En el **frmPrincipal** agregamos:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| ToolStripButton | (Name) | tsbCompras |
|  | DisplayStyle | ImageAndText |
|  | Text | Compras |
|  | Image | Cargamos el ícono |

Y en el código del **frmPrincipal** agregamos:

private void comprasToolStripMenuItem\_Click(object sender, EventArgs e)

{

frmCompras miForm = new frmCompras();

miForm.MdiParent = this;

miForm.UsuarioLogueado = usuarioLogueado;

miForm.Show();

}

private void tsbCompras\_Click(object sender, EventArgs e)

{

comprasToolStripMenuItem\_Click(sender, e);

}

En el **frmCompras** ponemos:

private CADUsuario usuarioLogueado;

public CADUsuario UsuarioLogueado

{

get => usuarioLogueado;

set => usuarioLogueado = value;

}

Agregamos los orígenes de datos a los ComboBox de Proveedores y de Bodegas

Al **btnBuscarProveedor** le ponemos:

private void btnBuscarProveedor\_Click(object sender, EventArgs e)

{

frmBusquedaProveedor miBusqueda = new frmBusquedaProveedor();

miBusqueda.ShowDialog();

if (miBusqueda.IDElegido == 0) return;

proveedorComboBox.SelectedValue = miBusqueda.IDElegido;

}

En los ComboBox ComboBox de Proveedores y de Bodegas agregamos Consulta y le ponemos ORDER BY (Nombre en un caso y Descripción en el otro).

Esto genera una barra arriba, copiamos el código del FillBy y reemplazamos en el evento Load del Formulario las instrucciones de llenado de los ComboBox.

Y Borramos la barra y el código asociado.

En el evento Load del **frmCompras** hacemos:

private void frmCompras\_Load(object sender, EventArgs e)

{

this.bodegaTableAdapter.FillBy(this.dSAppComercial.Bodega);

this.proveedorTableAdapter.FillBy1(this.dSAppComercial.Proveedor);

proveedorComboBox.SelectedIndex = -1;

bodegaComboBox.SelectedIndex = -1;

fechaDateTimePicker.Value = DateTime.Now;

}

Agregamos al **frmCompras**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Label | (Name) | label4 |
|  | Text | Producto: |
|  |  |  |
| TextBox | (Name) | productoTextBox |
|  |  |  |
| Button | (Name) | btnBuscarProducto |
|  | Text | … |
|  |  |  |
| Label | (Name) | productoLabel |
|  | Text | Descripción del Producto |
|  | ForeColor | Blue |
|  | Font | Negrita |
|  |  |  |
| PictureBox | (Name) | pbxImagen |

Al evento **Validating** del **productoTextBox** le ponemos:

private void productoTextBox\_Validating(object sender, System.ComponentModel.CancelEventArgs e)

{

errorProvider1.Clear();

pbxImagen.Image = null;

if (productoTextBox.Text==string.Empty)

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un Código de Producto o un Código de Barras");

return;

}

long producto;

if (!long.TryParse(productoTextBox.Text, out producto))

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un valor numérico");

productoTextBox.Focus();

return;

}

if (producto <= 0)

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un valor mayor a 0");

productoTextBox.Focus();

return;

}

}

En **DSAppComercial** del Proyecto **CADAppComercial** agregamos la Tabla **Producto**

Agregamos una Consulta que devuelva filas

SELECT IDProducto, Descripcion, IDDepartamento, IDIVA, Precio, Notas, Imagen, IDMedida, Medida

FROM dbo.Producto

WHERE IDProducto=@IDProducto

y la llamamos **ProductoGetProductoByIDProducto**

Agregamos otra Consulta que devuelva filas

SELECT Producto.IDProducto, Descripcion, IDDepartamento, IDIVA, Precio, Notas, Imagen, IDMedida, Medida

FROM Barra INNER JOIN Producto ON Barra.IDProducto=Producto.IDProducto

WHERE Barra=@Barra

y la llamamos **ProductoGetProductoByIDBarra**

Creamos la Clase **CADProducto**

using CADAppComercial.DSAppComercialTableAdapters;

namespace CADAppComercial

{

public class CADProducto

{

public int IDProducto { get; set; }

public string Descripcion { get; set; }

public int IDDepartamento { get; set; }

public int IDIVA { get; set; }

public decimal Precio { get; set; }

public string Notas { get; set; }

public string Imagen { get; set; }

public string IDMedida { get; set; }

public float Medida { get; set; }

private static ProductoTableAdapter adaptador = new ProductoTableAdapter();

public static CADProducto ProductoGetProductoByIDProducto(int IDProducto)

{

CADProducto miProducto = null;

DSAppComercial.ProductoDataTable miTabla = adaptador.ProductoGetProductoByIDProducto(IDProducto);

if (miTabla.Rows.Count==0)

{

return miProducto;

}

else

{

DSAppComercial.ProductoRow miRegistro = (DSAppComercial.ProductoRow)miTabla.Rows[0];

miProducto = new CADProducto();

miProducto.Descripcion = miRegistro.Descripcion;

miProducto.IDDepartamento = miRegistro.IDDepartamento;

miProducto.IDIVA = miRegistro.IDIVA;

miProducto.IDMedida = miRegistro.IDMedida;

miProducto.IDProducto = miRegistro.IDProducto;

miProducto.Imagen = miRegistro.Imagen;

miProducto.Medida = (float) miRegistro.Medida;

miProducto.Notas = miRegistro.Notas;

miProducto.Precio = miRegistro.Precio;

return miProducto;

}

}

public static CADProducto ProductoGetProductoByIDBarra(long Barra)

{

CADProducto miProducto = null;

DSAppComercial.ProductoDataTable miTabla = adaptador.ProductoGetProductoByIDBarra(IDBarra);

if (miTabla.Rows.Count == 0)

{

return miProducto;

}

else

{

DSAppComercial.ProductoRow miRegistro = (DSAppComercial.ProductoRow)miTabla.Rows[0];

miProducto = new CADProducto();

miProducto.Descripcion = miRegistro.Descripcion;

miProducto.IDDepartamento = miRegistro.IDDepartamento;

miProducto.IDIVA = miRegistro.IDIVA;

miProducto.IDMedida = miRegistro.IDMedida;

miProducto.IDProducto = miRegistro.IDProducto;

miProducto.Imagen = miRegistro.Imagen;

miProducto.Medida = (float)miRegistro.Medida;

miProducto.Notas = miRegistro.Notas;

miProducto.Precio = miRegistro.Precio;

return miProducto;

}

}

}

}

Agregamos al **frmCompras**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Label | (Name) | label5 |
|  | Text | Cantidad: |
|  |  |  |
| TextBox | (Name) | cantidadTextBox |
|  |  |  |
| Button | (Name) | agregarButton |
|  | Text | Agregar |
|  | TextAlign | MiddleRight |
|  | Image | Cargamos el ícono |
|  | ImageAlign | MiddleLeft |
|  |  |  |
| Button | (Name) | eliminarButton |
|  | Text | Eliminar |
|  | TextAlign | MiddleRight |
|  | Image | Cargamos el ícono |
|  | ImageAlign | MiddleLeft |
|  |  |  |
| DataGridVIew | (Name) | dgvDatos |
|  |  |  |
| Label | (Name) | label6 |
|  | Text | Costo: |
|  |  |  |
| TextBox | (Name) | costoTextBox |
|  |  |  |
| Label | (Name) | Label7 |
|  | Text | % Descuento: |
|  |  |  |
| TextBox | (Name) | porcentajeDescuentoTextBox |

En **frmCompras** creamos:

List<DetalleCompra> misDetalles = new List<DetalleCompra>();

CADProducto ultimoProducto = null;

private void frmCompras\_Load(object sender, EventArgs e)

{

this.bodegaTableAdapter.FillBy(this.dSAppComercial.Bodega);

this.proveedorTableAdapter.FillBy1(this.dSAppComercial.Proveedor);

proveedorComboBox.SelectedIndex = -1;

bodegaComboBox.SelectedIndex = -1;

fechaDateTimePicker.Value = DateTime.Now;

productoLabel.Text = string.Empty;

dgvDatos.DataSource = misDetalles;

}

En el evento **Validating** del **productoTextBox** hacemos

private void productoTextBox\_Validating(object sender, System.ComponentModel.CancelEventArgs e)

{

errorProvider1.Clear();

if (productoTextBox.Text==string.Empty)

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un Código de Producto o un Código de Barras");

return;

}

long producto;

if (!long.TryParse(productoTextBox.Text, out producto))

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un valor numérico");

productoTextBox.Focus();

return;

}

if (producto <= 0)

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un valor mayor a 0");

productoTextBox.Focus();

return;

}

CADProducto miProducto = CADProducto.ProductoGetProductoByIDProducto((int)producto);

if (miProducto==null)

{

miProducto = CADProducto.ProductoGetProductoByIDBarra(producto);

}

if (miProducto == null)

{

errorProvider1.SetError(productoTextBox, "Producto no existe");

productoLabel.Text = string.Empty;

pbxImagen.Image = null;

ultimoProducto = null;

}

else

{

productoLabel.Text = miProducto.Descripcion;

ultimoProducto = miProducto;

if (miProducto.Imagen == string.Empty)

{

pbxImagen.Image = null;

}

else

{

if (File.Exists(miProducto.Imagen))

{

pbxImagen.Load(miProducto.Imagen);

}

else

{

pbxImagen.Image = null;

}

}

}

}

Al **btnBuscarProducto** le ponemos:

private void btnBuscarProducto\_Click(object sender, EventArgs e)

{

frmBusquedaProducto miBusqueda = new frmBusquedaProducto();

miBusqueda.ShowDialog();

if (miBusqueda.IDElegido == 0) return;

productoTextBox.Text = miBusqueda.IDElegido.ToString();

productoTextBox\_Validating(sender, new System.ComponentModel.CancelEventArgs());

}

Creamos la Clase **DetalleCompra**

namespace AppComercial

{

public class DetalleCompra

{

public int IDProducto { get; set; }

public string Descripcion { get; set; }

public decimal Costo { get; set; }

public float Cantidad { get; set; }

public float PorcentajeIVA { get; set; }

public float PorcentajeDescuento { get; set; }

public decimal valorBruto { get { return Costo \* (decimal) Cantidad/(1+(decimal)PorcentajeIVA); } }

public decimal valorIVA { get { return Costo \* (decimal)Cantidad - valorBruto; } }

public decimal valorDescuento { get { return valorBruto \* (decimal)PorcentajeDescuento; } }

public decimal valorNeto { get { return Costo \* (decimal)Cantidad-valorDescuento; } }

}

}

En **DSAppComercial** del Proyecto **CADAppComercial** agregamos la Tabla **IVA**

Agregamos una Consulta que devuelva filas

SE SELECT IDIVA, Descripcion, Tarifa

FROM dbo.IVA

WHERE IDIVA=@IDIVA

y la llamamos **IVAGetIVAByIDIVA**

Creamos la Clase **CADIVA**

using CADAppComercial.DSAppComercialTableAdapters;

namespace CADAppComercial

{

public class CADIVA

{

public int IDIVA { get; set; }

public string Descripcion { get; set; }

public float Tarifa { get; set; }

private static IVATableAdapter adaptador = new IVATableAdapter();

public static CADIVA IVAGetIVAByIDIVA(int IDIVA)

{

CADIVA miIVA = null;

DSAppComercial.IVADataTable miTabla = adaptador.IVAGetIVAByIDIVA(IDIVA);

if (miTabla.Rows.Count == 0)

{

return miIVA;

}

else

{

DSAppComercial.IVARow miRegistro = (DSAppComercial.IVARow)miTabla.Rows[0];

miIVA = new CADIVA();

miIVA.Descripcion = miRegistro.Descripcion;

miIVA.IDIVA = miRegistro.IDIVA;

miIVA.Tarifa = (float) miRegistro.Tarifa;

return miIVA;

}

}

}

}

En **DSAppComercial** del Proyecto **CADAppComercial** vamos a la Tabla Producto yen los campos Notas e Imagen le ponemos la propiedad NullValue en (Empty):



Al botón Agregar (agregarButton) le ponemos:

private void agregarButton\_Click(object sender, EventArgs e)

{

errorProvider1.Clear();

if (ultimoProducto==null)

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un Producto");

productoTextBox.Focus();

return;

}

if (cantidadTextBox.Text == string.Empty)

{

errorProvider1.SetError(cantidadTextBox, "Debe ingresar una Cantidad");

return;

}

float cantidad;

if (!float.TryParse(cantidadTextBox.Text, out cantidad))

{

errorProvider1.SetError(cantidadTextBox, "Debe ingresar un valor numérico");

cantidadTextBox.Focus();

return;

}

if (cantidad <= 0)

{

errorProvider1.SetError(cantidadTextBox, "Debe ingresar un valor mayor a 0");

cantidadTextBox.Focus();

return;

}

if (costoTextBox.Text == string.Empty)

{

errorProvider1.SetError(costoTextBox, "Debe ingresar un Costo");

return;

}

decimal costo;

if (!decimal.TryParse(costoTextBox.Text, out costo))

{

errorProvider1.SetError(costoTextBox, "Debe ingresar un valor numérico");

costoTextBox.Focus();

return;

}

if (costo <= 0)

{

errorProvider1.SetError(costoTextBox, "Debe ingresar un valor mayor a 0");

costoTextBox.Focus();

return;

}

float porcentajeDescuento = 0;

if (porcentajeDescuentoTextBox.Text !=string.Empty)

{

if (!float.TryParse(porcentajeDescuentoTextBox.Text, out porcentajeDescuento))

{

errorProvider1.SetError(porcentajeDescuentoTextBox, "Debe ingresar un valor numérico");

porcentajeDescuentoTextBox.Focus();

return;

}

if (porcentajeDescuento < 0 || porcentajeDescuento>100)

{

errorProvider1.SetError(porcentajeDescuentoTextBox, "Debe ingresar un valor igual o mayor a 0 y menor o igual a 100");

porcentajeDescuentoTextBox.Focus();

return;

}

porcentajeDescuento /= 100;

}

DetalleCompra miDetalle = new DetalleCompra();

miDetalle.Cantidad = cantidad;

miDetalle.Costo = costo;

miDetalle.Descripcion = ultimoProducto.Descripcion;

miDetalle.IDProducto = ultimoProducto.IDProducto;

miDetalle.PorcentajeDescuento = porcentajeDescuento;

miDetalle.PorcentajeIVA = CADIVA.IVAGetIVAByIDIVA(ultimoProducto.IDIVA).Tarifa;

misDetalles.Add(miDetalle);

dgvDatos.DataSource = null;

dgvDatos.DataSource = misDetalles;

}

Agregamos a la Clase **DetalleCompra**

namespace AppComercial

{

public class DetalleCompra

{

public int IDProducto { get; set; }

public string Descripcion { get; set; }

public decimal Costo { get; set; }

public float Cantidad { get; set; }

public float PorcentajeIVA { get; set; }

public float PorcentajeDescuento { get; set; }

public decimal valorBruto { get { return Costo \* (decimal) Cantidad; } }

public decimal valorIVA { get { return valorBruto \* (decimal) PorcentajeIVA; } }

public decimal valorDescuento { get { return valorBruto \* (decimal)PorcentajeDescuento; } }

public decimal valorNeto { get { return valorBruto + valorIVA - valorDescuento; } }

}

}

Personalizamos la cuadrícula por código:

private void PersonalizaGrid()

{

dgvDatos.Columns["IDProducto"].HeaderText = "ID Producto";

dgvDatos.Columns["IDProducto"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

dgvDatos.Columns["Descripcion"].HeaderText = "Descripción";

dgvDatos.Columns["Costo"].HeaderText = "Costo";

dgvDatos.Columns["Costo"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["Costo"].DefaultCellStyle.Format = "C2";

dgvDatos.Columns["Cantidad"].HeaderText = "Cantidad";

dgvDatos.Columns["Cantidad"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["Cantidad"].DefaultCellStyle.Format = "N2";

dgvDatos.Columns["PorcentajeIVA"].HeaderText = "% IVA";

dgvDatos.Columns["PorcentajeIVA"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["PorcentajeIVA"].DefaultCellStyle.Format = "P2";

dgvDatos.Columns["PorcentajeDescuento"].HeaderText = "% Descuento";

dgvDatos.Columns["PorcentajeDescuento"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["PorcentajeDescuento"].DefaultCellStyle.Format = "P2";

dgvDatos.Columns["ValorBruto"].HeaderText = "Valor Bruto";

dgvDatos.Columns["ValorBruto"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["ValorBruto"].DefaultCellStyle.Format = "C2";

dgvDatos.Columns["ValorIVA"].HeaderText = "Valor IVA";

dgvDatos.Columns["ValorIVA"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["ValorIVA"].DefaultCellStyle.Format = "C2";

dgvDatos.Columns["ValorDescuento"].HeaderText = "Valor Descuento";

dgvDatos.Columns["ValorDescuento"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["ValorDescuento"].DefaultCellStyle.Format = "C2";

dgvDatos.Columns["ValorNeto"].HeaderText = "Valor Neto";

dgvDatos.Columns["ValorNeto"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["ValorNeto"].DefaultCellStyle.Format = "C2";

dgvDatos.Columns["IDProducto"].Width = 80;

dgvDatos.Columns["Descripcion"].Width = 200;

dgvDatos.Columns["Cantidad"].Width = 80;

dgvDatos.Columns["Costo"].Width = 80;

dgvDatos.Columns["PorcentajeIVA"].Width = 80;

dgvDatos.Columns["PorcentajeDescuento"].Width = 80;

dgvDatos.Columns["ValorBruto"].Width = 80;

dgvDatos.Columns["ValorIVA"].Width = 80;

dgvDatos.Columns["ValorDescuento"].Width = 80;

dgvDatos.Columns["ValorNeto"].Width = 80;

}

Modificamos/Agregamos en el botón Agregar:

private void agregarButton\_Click(object sender, EventArgs e)

……

DetalleCompra miDetalle = new DetalleCompra();

miDetalle.Cantidad = cantidad;

miDetalle.Costo = costo;

miDetalle.Descripcion = ultimoProducto.Descripcion;

miDetalle.IDProducto = ultimoProducto.IDProducto;

miDetalle.PorcentajeDescuento = porcentajeDescuento;

miDetalle.PorcentajeIVA = CADIVA.IVAGetIVAByIDIVA(ultimoProducto.IDIVA).Tarifa;

misDetalles.Add(miDetalle);

RefrescaGrid();

LimpiarControles();

}

private void LimpiarControles()

{

ultimoProducto = null;

productoTextBox.Text = string.Empty;

productoLabel.Text = string.Empty;

cantidadTextBox.Text = string.Empty;

costoTextBox.Text = string.Empty;

porcentajeDescuentoTextBox.Text = string.Empty;

productoTextBox.Focus();

}

private void RefrescaGrid()

{

dgvDatos.DataSource = null;

dgvDatos.DataSource = misDetalles;

PersonalizaGrid();

}

Agregamos al **frmCompras**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Label | (Name) | label8 |
|  | Text | Totales: |
|  |  |  |
| Label | (Name) | label9 |
|  | Text | Items |
|  |  |  |
| Label | (Name) | label10 |
|  | Text | Valor Bruto |
|  |  |  |
| Label | (Name) | label11 |
|  | Text | Valor IVA |
|  |  |  |
| Label | (Name) | label12 |
|  | Text | Valor Desc |
|  |  |  |
| Label | (Name) | label13 |
|  | Text | Valor Neto |
|  |  |  |
| TextBox | (Name) | totalItemTextBox |
|  | ReadOnly | True |
|  | TextAlign | Right |
|  |  |  |
| TextBox | (Name) | totalBrutoTextBox |
|  | ReadOnly | True |
|  | TextAlign | Right |
|  |  |  |
| TextBox | (Name) | totalIVATextBox |
|  | ReadOnly | True |
|  | TextAlign | Right |
|  |  |  |
| TextBox | (Name) | totalDescuentoTextBox |
|  | ReadOnly | True |
|  | TextAlign | Right |
|  |  |  |
| TextBox | (Name) | totalNetoTextBox |
|  | ReadOnly | True |
|  | TextAlign | Right |

En el **frmCompras** ponemos:

…

private int totalItems = 0;

private decimal totalBruto = 0;

private decimal totalIVA = 0;

private decimal totalDescuento = 0;

private decimal totalNeto = 0;

…

Y en el código del botón Agregar agregamos:

….

DetalleCompra miDetalle = new DetalleCompra();

miDetalle.Cantidad = cantidad;

miDetalle.Costo = costo;

miDetalle.Descripcion = ultimoProducto.Descripcion;

miDetalle.IDProducto = ultimoProducto.IDProducto;

miDetalle.PorcentajeDescuento = porcentajeDescuento;

miDetalle.PorcentajeIVA = CADIVA.IVAGetIVAByIDIVA(ultimoProducto.IDIVA).Tarifa;

misDetalles.Add(miDetalle);

totalItems += 1;

totalBruto += miDetalle.valorBruto;

totalIVA += miDetalle.valorIVA;

totalDescuento += miDetalle.valorDescuento;

totalNeto += miDetalle.valorNeto;

RefrescaGrid();

LimpiarControles();

}

Agregamos al **frmCompras**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Button | (Name) | btnGrabar |
|  | Image | Cargamos el ícono |
|  |  |  |
| Button | (Name) | btnCancelar |
|  | Image | Cargamos el ícono |

Al botón Cancelar le ponemos:

private void btnCancelar\_Click(object sender, EventArgs e)

{

this.Close();

}

Para grabar hay que guardar en estas Tablas:

* Compra
* Kardex
* CompraDetalle
* BodegaProducto

En **DSAppComercial** del Proyecto **CADAppComercial** agregamos las Tablas **Compra, CompraDetalle** y **Kardex** (BodegaProducto ya estraba agregada)

En **Compra** agregamos una Consulta INSERT

INSERT INTO [dbo].[Compra] ([Fecha], [IDProveedor], [IDBodega]) VALUES (@Fecha, @IDProveedor, @IDBodega);

SELECT IDCompra FROM Compra WHERE (IDCompra = SCOPE\_IDENTITY())

y la llamamos **CompraInsertCompra**

Una vez guardada le hacemos clic derecho-configurar-siguiente y elegimos “Un solo valor”

Creamos la Clase **CADCompra**

using CADAppComercial.DSAppComercialTableAdapters;

using System;

namespace CADAppComercial

{

public class CADCompra

{

private static CompraTableAdapter adaptador = new CompraTableAdapter();

public static int CompraInsertCompra(

DateTime Fecha,

int IDProveedor,

int IDBodega)

{

return (int) adaptador.CompraInsertCompra(Fecha, IDProveedor, IDBodega);

}

}

}

En **BodegaProducto** agregamos una Consulta UPDATE

UPDATE [dbo].[BodegaProducto] SET

[Stock] = [Stock]+@Cantidad

WHERE (IDBodega = @IDBodega) AND (IDProducto = @IDProducto)

y la llamamos **BodegaProductoActualizaStock**

En **CADBodegaProducto** creamos el método **BodegaProductoActualizaStock**

public static void BodegaProductoActualizaStock(float Cantidad, int IDBodega, int IDProducto)

{

adaptador.BodegaProductoActualizaStock(Cantidad, IDBodega, IDProducto);

}

En la Tabla **Kardex** agregamos una Consulta que **devuelve filas**

SELECT IDKardex, IDBodega, IDProducto, Fecha, Documento, Entrada, Salida, Saldo, UltimoCosto, CostoPromedio FROM dbo.Kardex

WHERE IDBodega=@IDBodega AND IDProducto=@IDProducto AND IDKardex=(SELECT MAX(IDKardex) FROM Kardex WHERE IDBodega=@IDBodega AND IDProducto=@IDProducto)

Y la llamamos **KardexUltimoKardex**

Creamos la Clase **CADKardex**

using CADAppComercial.DSAppComercialTableAdapters;

using System;

namespace CADAppComercial

{

public class CADKardex

{

public int IDKardex { get; set; }

public int IDBodega { get; set; }

public int IDProducto { get; set; }

public DateTime Fecha { get; set; }

public string Documento { get; set; }

public float Entrada { get; set; }

public float Salida { get; set; }

public float Saldo { get; set; }

public decimal UltimoCosto { get; set; }

public decimal CostoPromedio { get; set; }

private static KardexTableAdapter adaptador = new KardexTableAdapter();

public static CADKardex KardexUltimoKardex(int IDBodega,int IDProducto)

{

CADKardex miKardex = null;

DSAppComercial.KardexDataTable miTabla = adaptador.KardexUltimoKardex(IDBodega, IDProducto);

if (miTabla.Rows.Count == 0)

{

return miKardex;

}

else

{

DSAppComercial.KardexRow miRegistro = (DSAppComercial.KardexRow)miTabla.Rows[0];

miKardex = new CADKardex();

miKardex.IDKardex = miRegistro.IDKardex;

miKardex.IDBodega = miRegistro.IDBodega;

miKardex.IDProducto = miRegistro.IDProducto;

miKardex.Fecha = miRegistro.Fecha;

miKardex.Documento = miRegistro.Documento;

miKardex.Entrada = (float) miRegistro.Entrada;

miKardex.Salida = (float) miRegistro.Salida;

miKardex.Saldo = (float) miRegistro.Saldo;

miKardex.UltimoCosto = miRegistro.UltimoCosto;

miKardex.CostoPromedio = miRegistro.CostoPromedio;

return miKardex;

}

}

}

}

En la Tabla **Kardex** agregamos una Consulta que **INSERT**

INSERT INTO [dbo].[Kardex] ([IDBodega], [IDProducto], [Fecha], [Documento], [Entrada], [Salida], [Saldo], [UltimoCosto], [CostoPromedio]) VALUES (@IDBodega, @IDProducto, @Fecha, @Documento, @Entrada, @Salida, @Saldo, @UltimoCosto, @CostoPromedio);

SELECT IDKardex

FROM Kardex

WHERE (IDKardex = SCOPE\_IDENTITY())

y la llamamos **KardexInsertKardex**

Una vez guardada le hacemos clic derecho-configurar-siguiente y elegimos “Un solo valor”

Agregamos en **CADKardex**

public static int KardexInsertKardex(

int IDBodega,

int IDProducto,

DateTime Fecha,

string Documento,

float Entrada,

float Salida,

float Saldo,

decimal UltimoCosto,

decimal CostoPromedio)

{

return (int) adaptador.KardexInsertKardex(IDBodega, IDProducto, Fecha, Documento, Entrada, Salida, Saldo, UltimoCosto, CostoPromedio);

}

En la Tabla **CompraDetalle** agregamos una Consulta que **INSERT**

INSERT INTO [dbo].[CompraDetalle] ([IDCompra], [IDProducto], [Descripcion], [Costo], [Cantidad], [IDKardex], [PorcentajeIVA], [PorcentajeDescuento]) VALUES (@IDCompra, @IDProducto, @Descripcion, @Costo, @Cantidad, @IDKardex, @PorcentajeIVA, @PorcentajeDescuento);

y la llamamos **CompraDetalleInsertCompraDetalle**

Creamos la Clase **CADCompraDetalle**

using CADAppComercial.DSAppComercialTableAdapters;

namespace CADAppComercial

{

public class CADCompraDetalle

{

private static CompraDetalleTableAdapter adaptador = new CompraDetalleTableAdapter();

public static void CompraDetalleInsertCompraDetalle(

int IDCompra,

int IDProducto,

string Descripcion,

decimal Costo,

float Cantidad,

int IDKardex,

float PorcentajeIVA,

float PorcentajeDescuento)

{

adaptador.CompraDetalleInsertCompraDetalle(IDCompra, IDProducto, Descripcion, Costo, Cantidad, IDKardex, PorcentajeIVA, PorcentajeDescuento);

}

}

}

Al botón **Grabar** le ponemos:

private void btnGrabar\_Click(object sender, EventArgs e)

{

errorProvider1.Clear();

if (proveedorComboBox.SelectedIndex == -1)

{

errorProvider1.SetError(proveedorComboBox, "Debe seleccionar un Proveedor");

proveedorComboBox.Focus();

return;

}

if (bodegaComboBox.SelectedIndex == -1)

{

errorProvider1.SetError(bodegaComboBox, "Debe seleccionar una Bodega");

bodegaComboBox.Focus();

return;

}

if (misDetalles.Count==0)

{

errorProvider1.SetError(productoTextBox, "Debe ingresar al menos un Producto");

productoTextBox.Focus();

return;

}

DialogResult rta = MessageBox.Show(

"¿Está seguro de guardar la Compra?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

int IDProveedor = (int)proveedorComboBox.SelectedValue;

int IDBodega=(int)bodegaComboBox.SelectedValue;

DateTime fecha = fechaDateTimePicker.Value;

//Grabamos la Cabecera de la Compra

int IDCompra = CADCompra.CompraInsertCompra(

fechaDateTimePicker.Value,

IDProveedor,

IDBodega);

//Grabamos el Detalle de la Compra

foreach (DetalleCompra midetalle in misDetalles)

{

//Actualizamos la Tabla BodegaProducto

CADBodegaProducto miBodegaProducto = CADBodegaProducto.BodegaProductoGetBodegaProductoByIDBodegaAndIDProducto(IDBodega,midetalle.IDProducto);

if (miBodegaProducto==null)

{

CADBodegaProducto.BodegaProductoUpdate(IDBodega, midetalle.IDProducto, 1, 1, 1, 1);

}

CADBodegaProducto.BodegaProductoActualizaStock(midetalle.Cantidad, IDBodega, midetalle.IDProducto);

//Actualizamos el Kardex

CADKardex miKardex = CADKardex.KardexUltimoKardex(IDBodega, midetalle.IDProducto);

int IDKardex;

float nuevoSaldo;

decimal nuevoCostoPromedio;

decimal nuevoUltimoCosto;

if (miKardex == null)

{

nuevoSaldo = midetalle.Cantidad;

nuevoCostoPromedio = midetalle.valorNeto/(decimal)midetalle.Cantidad;

nuevoUltimoCosto = nuevoCostoPromedio;

}

else

{

nuevoSaldo = miKardex.Saldo + midetalle.Cantidad;

nuevoCostoPromedio = (miKardex.CostoPromedio \* (decimal)miKardex.Saldo

+ midetalle.valorNeto) / (decimal)nuevoSaldo;

nuevoUltimoCosto = midetalle.valorNeto / (decimal)midetalle.Cantidad;

}

IDKardex = CADKardex.KardexInsertKardex(

IDBodega,

midetalle.IDProducto,

fecha,

string.Format("CO-{0}", IDCompra),

midetalle.Cantidad,

0,

nuevoSaldo,

nuevoUltimoCosto

nuevoCostoPromedio);

//Actualizamos CompraDetalle

CADCompraDetalle.CompraDetalleInsertCompraDetalle(

IDCompra,

midetalle.IDProducto,

midetalle.Descripcion,

midetalle.Costo,

midetalle.Cantidad,

IDKardex, midetalle.PorcentajeIVA,

midetalle.PorcentajeDescuento);

}

MessageBox.Show(

string.Format("La Compra {0}, fue grabada de forma exitosa",IDCompra),

"Confirmación",

MessageBoxButtons.OK,

MessageBoxIcon.Information);

totalItems = 0;

totalBruto = 0;

totalIVA = 0;

totalDescuento = 0;

totalNeto = 0;

pbxImagen.Image = null;

proveedorComboBox.SelectedIndex = -1;

bodegaComboBox.SelectedIndex = -1;

misDetalles.Clear();

pbxImagen.Image = null;

RefrescaGrid();

proveedorComboBox.Focus();

}

Inicializamos las Tablas:

DELETE FROM CompraDetalle

DELETE FROM Compra

DELETE FROM Kardex

DELETE FROM BodegaProducto

--Inicializar los indices de las tablas

DBCC CHECKIDENT (CompraDetalle, RESEED,0)

DBCC CHECKIDENT (Compra, RESEED,0)

DBCC CHECKIDENT (Kardex, RESEED,0)

Agregamos al **frmCompras**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Button | (Name) | eliminarTodoButton |
|  | Image | Cargamos el ícono |
|  |  |  |
| Button | (Name) | eliminarLineaButton |
|  | Image | Cargamos el ícono |

Al botón eliminar todo le ponemos:

private void eliminarTodoButton\_Click(object sender, EventArgs e)

{

errorProvider1.Clear();

if (misDetalles.Count == 0) return;

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el todas las líneas de la Compra?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

misDetalles.Clear();

totalItems = 0;

totalBruto = 0;

totalIVA = 0;

totalDescuento = 0;

totalNeto = 0;

RefrescaGrid();

pbxImagen.Image = null;

}

Al botón eliminar línea le ponemos:

private void eliminarLineaButton\_Click(object sender, EventArgs e)

{

errorProvider1.Clear();

if (misDetalles.Count == 0) return;

if(dgvDatos.SelectedRows.Count==0)

{

misDetalles.RemoveAt(misDetalles.Count - 1);

RefrescaGrid();

}

}

Al **RefrescaGrid** lo actualizamos:

private void RefrescaGrid()

{

dgvDatos.DataSource = null;

dgvDatos.DataSource = misDetalles;

totalItems = 0;

totalBruto = 0;

totalIVA = 0;

totalDescuento = 0;

totalNeto = 0;

foreach (DetalleCompra miDetalle in misDetalles)

{

totalItems += 1;

totalBruto += miDetalle.valorBruto;

totalIVA += miDetalle.valorIVA;

totalDescuento += miDetalle.valorDescuento;

totalNeto += miDetalle.valorNeto;

}

totalItemTextBox.Text = string.Format("{0:N0}", totalItems);

totalBrutoTextBox.Text = string.Format("{0:C2}", totalBruto);

totalIVATextBox.Text = string.Format("{0:C2}", totalIVA);

totalDescuentoTextBox.Text = string.Format("{0:C2}", totalDescuento);

totalNetoTextBox.Text = string.Format("{0:C2}", totalNeto);

PersonalizaGrid();

}

Al **frmCompras** le cargamos el evento **FormClosing**:

private void frmCompras\_FormClosing(object sender, FormClosingEventArgs e)

{

if(misDetalles.Count>0)

{

DialogResult rta = MessageBox.Show(

"La Compra tiene productos cargados, ¿está seguro de salir sin terminar la Compra?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No)

{

e.Cancel = true;

}

}

}

# 60 al 62 Correcciones varias

En el frmProducto, buscamos el productoBindingSource y vamos a las propiedades y buscamos el método PositionChanged:

private void productoBindingSource\_PositionChanged(object sender, EventArgs e)

{

LlenarGrillas();

CargarImagen();

}

Y podemos borrar:

private void bindingNavigatorMoveFirstItem\_Click(object sender, EventArgs e)

{

LlenarGrillas();

CargarImagen();

}

private void bindingNavigatorMoveNextItem\_Click(object sender, EventArgs e)

{

LlenarGrillas();

CargarImagen();

}

private void bindingNavigatorMoveLastItem\_Click(object sender, EventArgs e)

{

LlenarGrillas();

CargarImagen();

}

private void dgvDatos\_CellClick(object sender, DataGridViewCellEventArgs e)

{

LlenarGrillas();

CargarImagen();

}

Y agregamos

private void LlenarGrillas()

{

if (iDProductoTextBox.Text == string.Empty) return;

this.barraTableAdapter.FillBy(this.dSAppComercial.Barra,Convert.ToInt32(iDProductoTextBox.Text));

this.bodegaProductoTableAdapter.FillBy(this.dSAppComercial.BodegaProducto, Convert.ToInt32(iDProductoTextBox.Text));

}

## Como no guardar la ruta completa de las imágenes

Cambiamos al buscar la imagen:

private void btnBuscarImagen\_Click(object sender, EventArgs e)

{

openFileDialog1.ShowDialog();

imagenTextBox.Text = openFileDialog1.SafeFileName;

CargarImagen();

}

Y cambiamos

private void CargarImagen()

{

if(imagenTextBox.Text==string.Empty)

{

pbxImagen.Image = null;

}

else

{

if(File.Exists("Images\\"+imagenTextBox.Text))

{

pbxImagen.Load("Images\\" + imagenTextBox.Text);

}

}

}

Creamos la Carpeta **Images** dentro de **AppComercial\bin\Debug\**:

C:\AppComercial\AppComercial\AppComercial\bin\Debug\Images

En **frmCompras**, dentro de **productoTextBox**\_Validating modificamos

…………

productoLabel.Text = miProducto.Descripcion;

ultimoProducto = miProducto;

if (miProducto.Imagen == string.Empty)

{

pbxImagen.Image = null;

}

else

{

if (File.Exists("Images\\" + miProducto.Imagen))

{

if (miProducto.Imagen == string.Empty)

{

pbxImagen.Image = null;

}

else

{

if (File.Exists("Images\\" + miProducto.Imagen))

{

pbxImagen.Load("Images\\" + miProducto.Imagen);

}

}

}

else

{

pbxImagen.Image = null;

}

}

}

## Controlar que no se pueda borrar un Proveedor que tenga Compras

Vamos al DataSet **DSAppComercial** del Proyecto **CADAppComercial** y en la tabla **Compra** agregamos una Consulta que devuelve un valor:

SELECT TOP 1 1 FROM Compra

WHERE IDProveedor =@IDProveedor

y la llamamos **ProveedorTieneCompras**

En la Tabla Kardex agregamos una Consulta que devuelve un valor:

SELECT TOP 1 1 FROM Kardex

WHERE IDProducto=@IDProducto

y la llamamos **KardexProductoTieneMovimientos**

En la Tabla Kardex agregamos una Consulta que devuelve un valor:

SELECT TOP 1 1 FROM Kardex

WHERE IDBodega=@IDBodega

y la llamamos **KardexBodegaTieneMovimientos**

En **CADCompras** creamos el método

public static bool ProveedorTieneCompras(int IDProveedor)

{

if(adaptador.ProveedorTieneCompras(IDProveedor)==null)

{

return false;

}

return true;

}

En **CADKardex** creamos los métodos:

public static bool KardexProductoTieneMovimientos(int IDProducto)

{

if (adaptador.KardexProductoTieneMovimientos(IDProducto) == null)

{

return false;

}

return true;

}

public static bool KardexBodegaTieneMovimientos(int IDBodega)

{

if (adaptador.KardexBodegaTieneMovimientos(IDBodega) == null)

{

return false;

}

return true;

}

En **frmProveedores** hacemos en el método de borrar:

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Registro actual?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

if (CADCompra.ProveedorTieneCompras(Convert.ToInt32(iDProveedorTextBox.Text)))

{

MessageBox.Show(

"No se puede borrar Proveedor porque tiene movimientos",

"Error",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

return;

}

this.Validate();

this.proveedorBindingSource.RemoveAt(proveedorBindingSource.Position);

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

}

En **frmBodegas** hacemos en el método de borrar:

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Registro actual?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

if (CADKardex.KardexBodegaTieneMovimientos(Convert.ToInt32(iDBodegaTextBox.Text)))

{

MessageBox.Show(

"No se puede borrar Bodega porque tiene movimientos",

"Error",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

return;

}

this.Validate();

this.bodegaBindingSource.RemoveAt(bodegaBindingSource.Position);

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

}

En **frmProductos** hacemos en el método de borrar:

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Registro actual?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

if (CADKardex.KardexProductoTieneMovimientos(Convert.ToInt32(iDProductoTextBox.Text)))

{

MessageBox.Show(

"No se puede borrar Producto porque tiene movimientos",

"Error",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

return;

}

this.Validate();

this.productoBindingSource.RemoveAt(productoBindingSource.Position);

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

}

## Borrar un Producto sin Movimientos pero que tiene Barras y Parámetros de BodegaProducto

Vamos al DataSet **DSAppComercial** del Proyecto **CADAppComercial** y en la tabla **Barra** agregamos una Consulta **DELETE**:

DELETE FROM [dbo].[Barra]

WHERE IDProducto = @IDProducto

y la llamamos **DeleteBarraByIDProducto**

En la tabla **BodegaProducto** agregamos una Consulta **DELETE**:

DELETE FROM [dbo].[BodegaProducto]

WHERE IDProducto = @IDProducto

y la llamamos **DeleteBodegaProductoByIDProducto**

En **CADBarra** hacemos:

public static void DeleteBarraByIDProducto(int IDProducto)

{

adaptador.DeleteBarraByIDProducto(IDProducto);

}

En **CADBodegaProducto** hacemos:

public static void DeleteBodegaProductoByIDProducto(int IDProducto)

{

adaptador.DeleteBodegaProductoByIDProducto(IDProducto);

}

En el método borrar de **frmProducto** hacemos:

private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)

{

DialogResult rta = MessageBox.Show(

"¿Está seguro de borrar el Registro actual?",

"Confirmación",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2);

if (rta == DialogResult.No) return;

if (CADKardex.KardexProductoTieneMovimientos(Convert.ToInt32(iDProductoTextBox.Text)))

{

MessageBox.Show(

"No se puede borrar Producto porque tiene movimientos",

"Error",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

return;

}

this.Validate();

CADBarra.DeleteBarraByIDProducto(Convert.ToInt32(iDProductoTextBox.Text));

CADBodegaProducto.DeleteBodegaProductoByIDProducto(Convert.ToInt32(iDProductoTextBox.Text));

this.productoBindingSource.RemoveAt(productoBindingSource.Position);

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

}

# 63 al 70 - Ventas

Creamos un Procedimiento almacenado para limpiar las Tablas antes de las pruebas y lo llamamos **InicializaInventario**:

CREATE PROCEDURE InicializaInventario AS

DELETE FROM BodegaProducto

DELETE FROM DevolucionClienteDetalle

DELETE FROM DevolucionCliente

DELETE FROM DevolucionProveedorDetalle

DELETE FROM DevolucionProveedor

DELETE FROM CompraDetalle

DELETE FROM Compra

DELETE FROM InventarioDetalle

DELETE FROM Inventario

DELETE FROM SalidaDetalle

DELETE FROM Salida

DELETE FROM TrasladoDetalle

DELETE FROM Traslado

DELETE FROM VentaDetalle

DELETE FROM Venta

DELETE FROM Kardex

--Inicializar los indices de las tablas

DBCC CHECKIDENT (CompraDetalle, RESEED,0)

DBCC CHECKIDENT (Compra, RESEED,0)

DBCC CHECKIDENT (DevolucionClienteDetalle, RESEED,0)

DBCC CHECKIDENT (DevolucionCliente, RESEED,0)

DBCC CHECKIDENT (DevolucionProveedorDetalle, RESEED,0)

DBCC CHECKIDENT (DevolucionProveedor, RESEED,0)

DBCC CHECKIDENT (InventarioDetalle, RESEED,0)

DBCC CHECKIDENT (Inventario, RESEED,0)

DBCC CHECKIDENT (SalidaDetalle, RESEED,0)

DBCC CHECKIDENT (Salida, RESEED,0)

DBCC CHECKIDENT (TrasladoDetalle, RESEED,0)

DBCC CHECKIDENT (Traslado, RESEED,0)

DBCC CHECKIDENT (VentaDetalle, RESEED,0)

DBCC CHECKIDENT (Venta, RESEED,0)

DBCC CHECKIDENT (Kardex, RESEED,0)

Y lo ejecutamos así:

EXEC InicializaInventario

## Formulario Ventas

Creamos el Formulario **frmVentas**

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| Formulario | (Name) | frmVentas |
|  | Text | Ventas |
|  | BackColor | SkyBlue |
|  | MaximizeBox | False |
|  | MinimizeBox | False |
|  | StartPosition | CenterScreen |
|  | FormBorderStyle | FixedDialog |
|  | Size | 1316; 696 |
|  | Icon | Cargamos el ìcono |
|  |  |  |
| ErrorProvider | (Name) | errorProvider1 |
|  |  |  |
| Label | (Name) | label1 |
|  | Text | Fecha: |
|  |  |  |
| DateTimePicker | (Name) | fechaDateTimePicker |
|  |  |  |
| Label | (Name) | label2 |
|  | Text | Cliente: |
|  |  |  |
| ComboBox | (Name) | clienteComboBox |
|  |  |  |
| Button | (Name) | btnBuscarCliente |
|  | Text | … |
|  |  |  |
| Label | (Name) | label3 |
|  | Text | Bodega: |
|  |  |  |
| ComboBox | (Name) | bodegaComboBox |
|  |  |  |
| Label | (Name) | label4 |
|  | Text | Producto: |
|  |  |  |
| TextBox | (Name) | productoTextBox |
|  |  |  |
| Button | (Name) | btnBuscarProducto |
|  | Text | … |
|  |  |  |
| Label | (Name) | productoLabel |
|  | Text | Descripción del Producto |
|  | ForeColor | Blue |
|  | Font | Negrita |
|  |  |  |
| PictureBox | (Name) | pbxImagen |
|  |  |  |
| Label | (Name) | label5 |
|  | Text | Cantidad: |
|  |  |  |
| TextBox | (Name) | cantidadTextBox |
|  |  |  |
| Button | (Name) | agregarButton |
|  | Text | Agregar |
|  | TextAlign | MiddleRight |
|  | Image | Cargamos el ícono |
|  | ImageAlign | MiddleLeft |
|  |  |  |
| Button | (Name) | eliminarButton |
|  | Text | Eliminar |
|  | TextAlign | MiddleRight |
|  | Image | Cargamos el ícono |
|  | ImageAlign | MiddleLeft |
|  |  |  |
| DataGridVIew | (Name) | dgvDatos |
|  |  |  |
| Label | (Name) | label6 |
|  | Text | Precio: |
|  |  |  |
| TextBox | (Name) | precioTextBox |
|  | ReadOnly | true |
|  | BackGroundColor | LightBlue |
|  | Font | Negrita |
|  | ForeColor | Blue |
|  |  |  |
| Label | (Name) | Label7 |
|  | Text | % Descuento: |
|  |  |  |
| TextBox | (Name) | porcentajeDescuentoTextBox |

Cuando hacemos una Compra afectamos las Tablas

* Venta
* VentaDetalle
* BodegaProducto
* Kardex

En el **frmPrincipal** agregamos:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Propiedad** | **Valor** |
| ToolStripButton | (Name) | tsbVentas |
|  | DisplayStyle | ImageAndText |
|  | Text | Ventas |
|  | Image | Cargamos el ícono |
|  | ImageScaling | None |
|  | **Los ìconos deben ser archivos ICO de 32x32** | |

Y en el código del **frmPrincipal** agregamos:

private void ventasToolStripMenuItem\_Click(object sender, EventArgs e)

{

frmVentas miForm = new frmVentas();

miForm.MdiParent = this;

miForm.UsuarioLogueado = usuarioLogueado;

miForm.Show();

}

private void tsbVentas\_Click(object sender, EventArgs e)

{

ventasToolStripMenuItem\_Click(sender, e);

}

Le ponemos el origen de datos a los ComboBox de clientes y de bodegas.

En **frmVentas** ponemos:

private CADUsuario usuarioLogueado;

public CADUsuario UsuarioLogueado

{

get => usuarioLogueado;

set => usuarioLogueado = value;

}

private int totalItems = 0;

private decimal totalBruto = 0;

private decimal totalIVA = 0;

private decimal totalDescuento = 0;

private decimal totalNeto = 0;

List<DetalleVenta> misDetalles = new List<DetalleVenta>();

CADProducto ultimoProducto = null;

public frmVentas()

{

InitializeComponent();

}

private void frmVentas\_Load(object sender, EventArgs e)

{

this.clienteTableAdapter.FillBy1(this.dSAppComercial.Cliente);

this.bodegaTableAdapter.FillBy(this.dSAppComercial.Bodega);

clienteComboBox.SelectedIndex = -1;

bodegaComboBox.SelectedIndex = -1;

productoLabel.Text = string.Empty;

}

Al botón Buscar Clientes le ponemos:

private void btnBuscarCliente\_Click(object sender, EventArgs e)

{

frmBusquedaCliente miBusqueda = new frmBusquedaCliente();

miBusqueda.ShowDialog();

if (miBusqueda.IDElegido == 0) return;

clienteComboBox.SelectedValue = miBusqueda.IDElegido;

}

Al botón Buscar Productos le ponemos:

private void btnBuscarProducto\_Click(object sender, EventArgs e)

{

frmBusquedaProducto miBusqueda = new frmBusquedaProducto();

miBusqueda.ShowDialog();

if (miBusqueda.IDElegido == 0) return;

productoTextBox.Text = miBusqueda.IDElegido.ToString();

productoTextBox\_Validating(sender, new System.ComponentModel.CancelEventArgs());

}

Al **productoTextBox** le cargamos el evento **Validating**:

private void productoTextBox\_Validating(object sender, System.ComponentModel.CancelEventArgs e)

{

errorProvider1.Clear();

if (productoTextBox.Text == string.Empty)

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un Código de Producto o un Código de Barras");

return;

}

long producto;

if (!long.TryParse(productoTextBox.Text, out producto))

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un valor numérico");

productoTextBox.Focus();

return;

}

if (producto <= 0)

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un valor mayor a 0");

productoTextBox.Focus();

return;

}

CADProducto miProducto = CADProducto.ProductoGetProductoByIDProducto((int)producto);

if (miProducto == null)

{

miProducto = CADProducto.ProductoGetProductoByIDBarra(producto);

}

if (miProducto == null)

{

errorProvider1.SetError(productoTextBox, "Producto no existe");

productoLabel.Text = string.Empty;

pbxImagen.Image = null;

ultimoProducto = null;

}

else

{

productoLabel.Text = miProducto.Descripcion;

ultimoProducto = miProducto;

if (miProducto.Imagen == string.Empty)

{

pbxImagen.Image = null;

}

else

{

if (File.Exists("Images\\" + miProducto.Imagen))

{

if (miProducto.Imagen == string.Empty)

{

pbxImagen.Image = null;

}

else

{

if (File.Exists("Images\\" + miProducto.Imagen))

{

pbxImagen.Load("Images\\" + miProducto.Imagen);

}

}

}

else

{

pbxImagen.Image = null;

}

}

}

}

Creamos la Clase **DetalleVenta**

namespace AppComercial

{

class DetalleVenta

{

public int IDProducto { get; set; }

public string Descripcion { get; set; }

public decimal Precio { get; set; }

public float Cantidad { get; set; }

public float PorcentajeIVA { get; set; }

public float PorcentajeDescuento { get; set; }

public decimal valorBruto { get { return Precio \* (decimal)Cantidad / (1 + (decimal)PorcentajeIVA); } }

public decimal valorIVA { get { return Precio \* (decimal)Cantidad - valorBruto; } }

public decimal valorDescuento { get { return valorBruto \* (decimal)PorcentajeDescuento; } }

public decimal valorNeto { get { return Precio \* (decimal)Cantidad - valorDescuento; } }

}

}

Hacemos Ver – Orden de Tabulación y le damos el orden que queremos.

Agregamos al **frmVentas**:

private void agregarButton\_Click(object sender, EventArgs e)

{

errorProvider1.Clear();

if (ultimoProducto == null)

{

errorProvider1.SetError(productoTextBox, "Debe ingresar un Producto");

productoTextBox.Focus();

return;

}

if (cantidadTextBox.Text == string.Empty)

{

errorProvider1.SetError(cantidadTextBox, "Debe ingresar una Cantidad");

return;

}

float cantidad;

if (!float.TryParse(cantidadTextBox.Text, out cantidad))

{

errorProvider1.SetError(cantidadTextBox, "Debe ingresar un valor numérico");

cantidadTextBox.Focus();

return;

}

if (cantidad <= 0)

{

errorProvider1.SetError(cantidadTextBox, "Debe ingresar un valor mayor a 0");

cantidadTextBox.Focus();

return;

}

float porcentajeDescuento = 0;

if (porcentajeDescuentoTextBox.Text != string.Empty)

{

if (!float.TryParse(porcentajeDescuentoTextBox.Text, out porcentajeDescuento))

{

errorProvider1.SetError(porcentajeDescuentoTextBox, "Debe ingresar un valor numérico");

porcentajeDescuentoTextBox.Focus();

return;

}

if (porcentajeDescuento < 0 || porcentajeDescuento > 100)

{

errorProvider1.SetError(porcentajeDescuentoTextBox, "Debe ingresar un valor igual o mayor a 0 y menor o igual a 100");

porcentajeDescuentoTextBox.Focus();

return;

}

porcentajeDescuento /= 100;

}

decimal precio= ultimoProducto.Precio;

DetalleVenta miDetalle = new DetalleVenta();

miDetalle.Cantidad = cantidad;

miDetalle.Precio = precio;

miDetalle.Descripcion = ultimoProducto.Descripcion;

miDetalle.IDProducto = ultimoProducto.IDProducto;

miDetalle.PorcentajeDescuento = porcentajeDescuento;

miDetalle.PorcentajeIVA = CADIVA.IVAGetIVAByIDIVA(ultimoProducto.IDIVA).Tarifa;

misDetalles.Add(miDetalle);

pbxImagen.Image = null;

RefrescaGrid();

LimpiarControles();

}

private void LimpiarControles()

{

ultimoProducto = null;

productoTextBox.Text = string.Empty;

productoLabel.Text = string.Empty;

cantidadTextBox.Text = string.Empty;

precioTextBox.Text = string.Empty;

porcentajeDescuentoTextBox.Text = string.Empty;

productoTextBox.Focus();

}

private void RefrescaGrid()

{

dgvDatos.DataSource = null;

dgvDatos.DataSource = misDetalles;

totalItems = 0;

totalBruto = 0;

totalIVA = 0;

totalDescuento = 0;

totalNeto = 0;

foreach (DetalleVenta miDetalle in misDetalles)

{

totalItems += 1;

totalBruto += miDetalle.valorBruto;

totalIVA += miDetalle.valorIVA;

totalDescuento += miDetalle.valorDescuento;

totalNeto += miDetalle.valorNeto;

}

totalItemTextBox.Text = string.Format("{0:N0}", totalItems);

totalBrutoTextBox.Text = string.Format("{0:C2}", totalBruto);

totalIVATextBox.Text = string.Format("{0:C2}", totalIVA);

totalDescuentoTextBox.Text = string.Format("{0:C2}", totalDescuento);

totalNetoTextBox.Text = string.Format("{0:C2}", totalNeto);

PersonalizaGrid();

}

private void PersonalizaGrid()

{

dgvDatos.Columns["IDProducto"].HeaderText = "ID Producto";

dgvDatos.Columns["IDProducto"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

dgvDatos.Columns["Descripcion"].HeaderText = "Descripción";

dgvDatos.Columns["Precio"].HeaderText = "Precio";

dgvDatos.Columns["Precio"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["Precio"].DefaultCellStyle.Format = "C2";

dgvDatos.Columns["Cantidad"].HeaderText = "Cantidad";

dgvDatos.Columns["Cantidad"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["Cantidad"].DefaultCellStyle.Format = "N2";

dgvDatos.Columns["PorcentajeIVA"].HeaderText = "% IVA";

dgvDatos.Columns["PorcentajeIVA"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["PorcentajeIVA"].DefaultCellStyle.Format = "P2";

dgvDatos.Columns["PorcentajeDescuento"].HeaderText = "% Descuento";

dgvDatos.Columns["PorcentajeDescuento"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["PorcentajeDescuento"].DefaultCellStyle.Format = "P2";

dgvDatos.Columns["ValorBruto"].HeaderText = "Valor Bruto";

dgvDatos.Columns["ValorBruto"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["ValorBruto"].DefaultCellStyle.Format = "C2";

dgvDatos.Columns["ValorIVA"].HeaderText = "Valor IVA";

dgvDatos.Columns["ValorIVA"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["ValorIVA"].DefaultCellStyle.Format = "C2";

dgvDatos.Columns["ValorDescuento"].HeaderText = "Valor Descuento";

dgvDatos.Columns["ValorDescuento"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["ValorDescuento"].DefaultCellStyle.Format = "C2";

dgvDatos.Columns["ValorNeto"].HeaderText = "Valor Neto";

dgvDatos.Columns["ValorNeto"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

dgvDatos.Columns["ValorNeto"].DefaultCellStyle.Format = "C2";

//dgvDatos.AutoResizeColumns();

dgvDatos.Columns["IDProducto"].Width = 80;

dgvDatos.Columns["Descripcion"].Width = 200;

dgvDatos.Columns["Cantidad"].Width = 80;

dgvDatos.Columns["Precio"].Width = 80;

dgvDatos.Columns["PorcentajeIVA"].Width = 80;

dgvDatos.Columns["PorcentajeDescuento"].Width = 80;

dgvDatos.Columns["ValorBruto"].Width = 80;

dgvDatos.Columns["ValorIVA"].Width = 80;

dgvDatos.Columns["ValorDescuento"].Width = 80;

dgvDatos.Columns["ValorNeto"].Width = 80;

}

# PROCEDIMIENTOS ALMACENADOS

|  |  |  |
| --- | --- | --- |
| **Video** | **Nombre** | **Sirve para** |
| 2 | ValidaUsuario | Validar Usuario y Clave contra la Base de Datos |
| 10 | ClienteInsert | Para insertar un Cliente |
| 10 | ClienteUpdate | Para modificar un Cliente |
| 10 | ClienteDelete | Para borrar un Cliente |
| 30 | BarraExisteBarra | Para ver si un Código de Barra ya existe |
| 30 | BarraInsertBarra | Para grabar un Código de Barra |
| 31 | BodegaProductoGetBodegaProductoByIDBodegaAndIDProducto | Obtiene una fila de BodegaProducto por IDBodega y IDProducto |
| 33 | BodegaProductoInsert | Para grabar un nuevo Parámetro de Bodega |
| 33 | BodegaProductoUpdate | Para actualizar un Parámetro de Bodega |
| 35 | UsuarioExisteUsuario | Para verificar si un Usuario existe |
| 37 | UsuarioGetUsuarioByIDUsuario | Obtiene una fila de Usuarios por IDUsuario |
| 39 | UsuarioUpdateClave | Cambiar Clave de Usuario |
| 41 | PermisoRolPuedeVer | Para ver si un IDRol en un Formulario puede ver |
| 41 | PermisoRolPuedeModificar | Para ver si un IDRol en un Formulario puede modificar |
| 41 | PermisoRolPuedeBorrar | Para ver si un IDRol en un Formulario puede borrar |
| 46 | ProductoGetProductoByIDProducto | Obtiene una fila de Productos por IDProducto |
| 49 | IVAGetIVAByIDIVA | Obtiene una fila de IVA por IDIVA |
| 52 | CompraInsertCompra | Inserta la Cabecera de una Compra |
| 54 | BodegaProductoActualizaStock | Actualiza Stock en BodegaProducto |
| 56 | KardexInsertKardex | Inserta un Kardex |
| 57 | CompraDetalleInsertCompraDetalle | Inserta una CompraDetalle |
| 61 | ProveedorTieneCompras | Dice si un Proveedor tiene Compras |
| 61 | KardexProductoTieneMovimientos | Dice si un Producto tiene Movimientos |
| 61 | KardexBodegaTieneMovimientos | Dice si una Bodega tiene Movimientos |
| 62 | DeleteBarraByIDProducto | Borra las Barras de un Producto |
| 62 | DeleteBodegaProductoByIDProducto | Borra las BodegaProducto de un Producto |
|  |  |  |