# 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

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 |

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

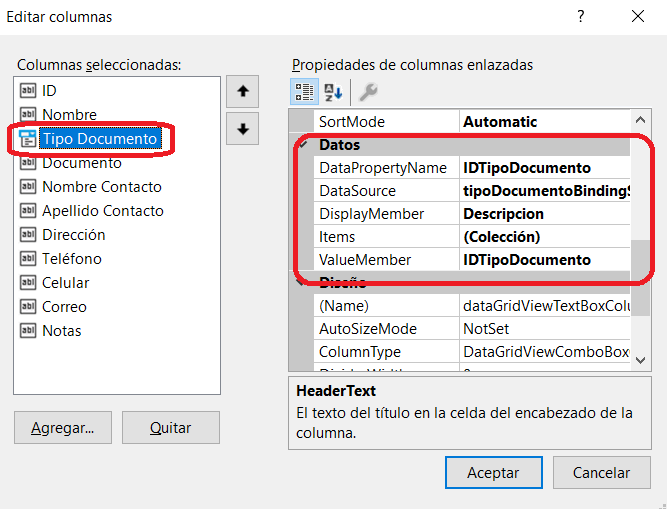
Descripción generada automáticamente

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:



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;

proveedorBindingNavigatorSaveItem.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 proveedorBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

if (!Validarcampos()) return;

this.Validate();

this.proveedorBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.dSAppComercial);

Habilitar(false);

}

# 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 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |