Sistema punto de venta

C# - SQL Server -Visual Studio

Módulos desarrollados

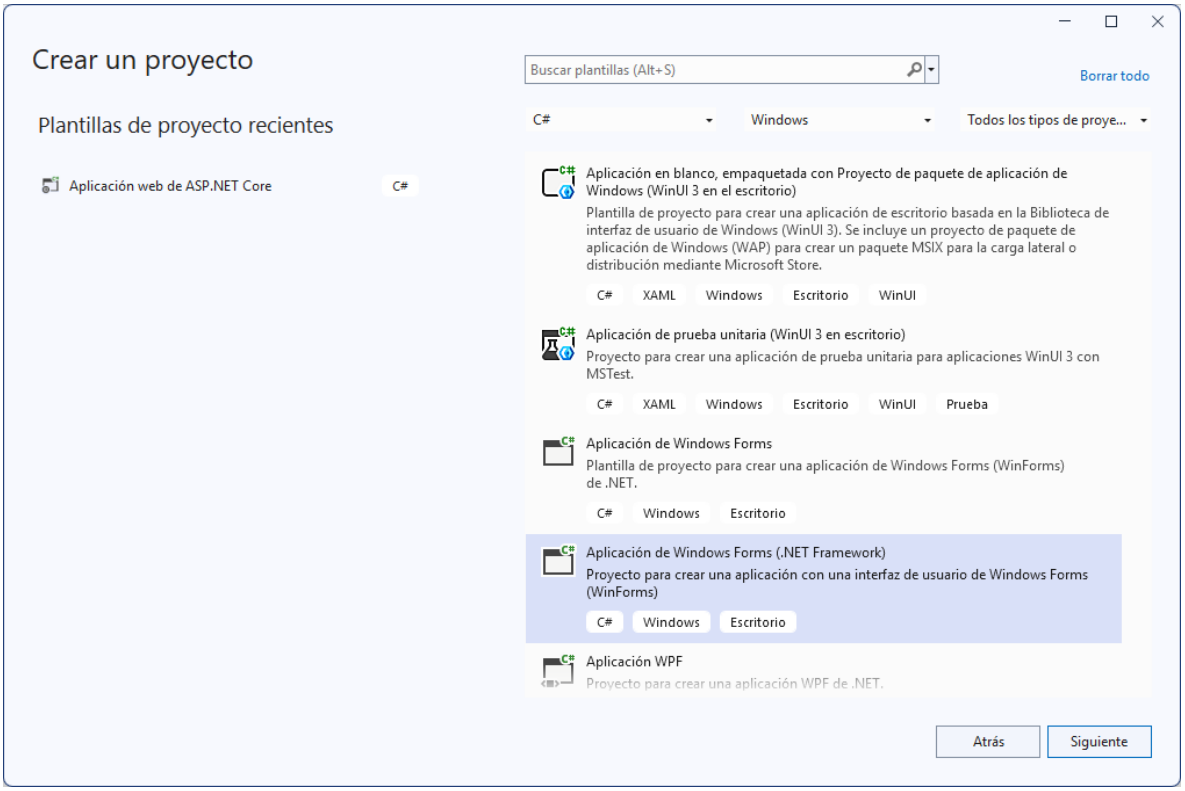
- Registro de clientes
- Administrador
- Restricciones
- Artículos
- Identificación por código (clientes y artículos)
- Ventas
- Cancelación de venta
- Facturación
- Inventario

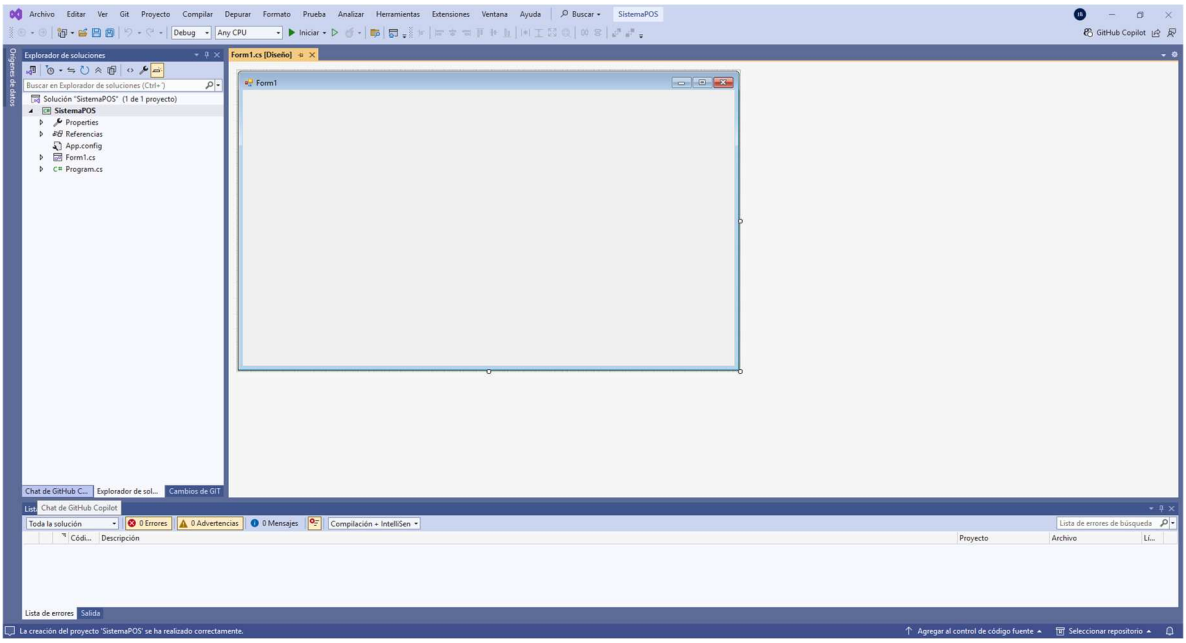
Crear Base de datos

Pegar script

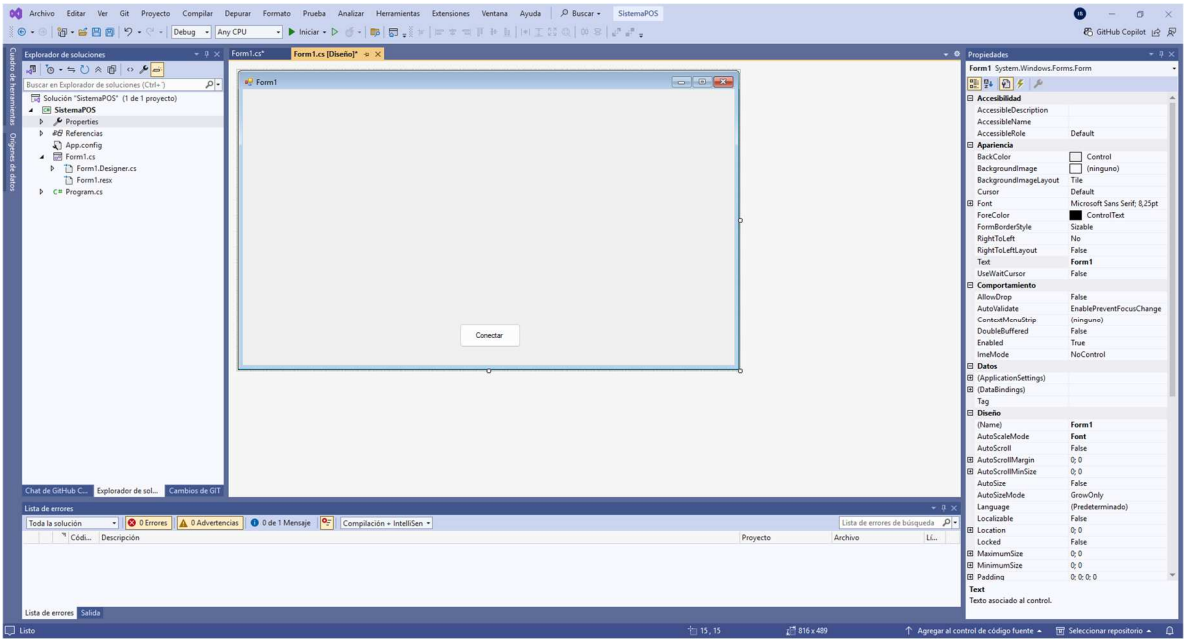
Conectando Visual Studio con Sql Server

Nuevo proyecto

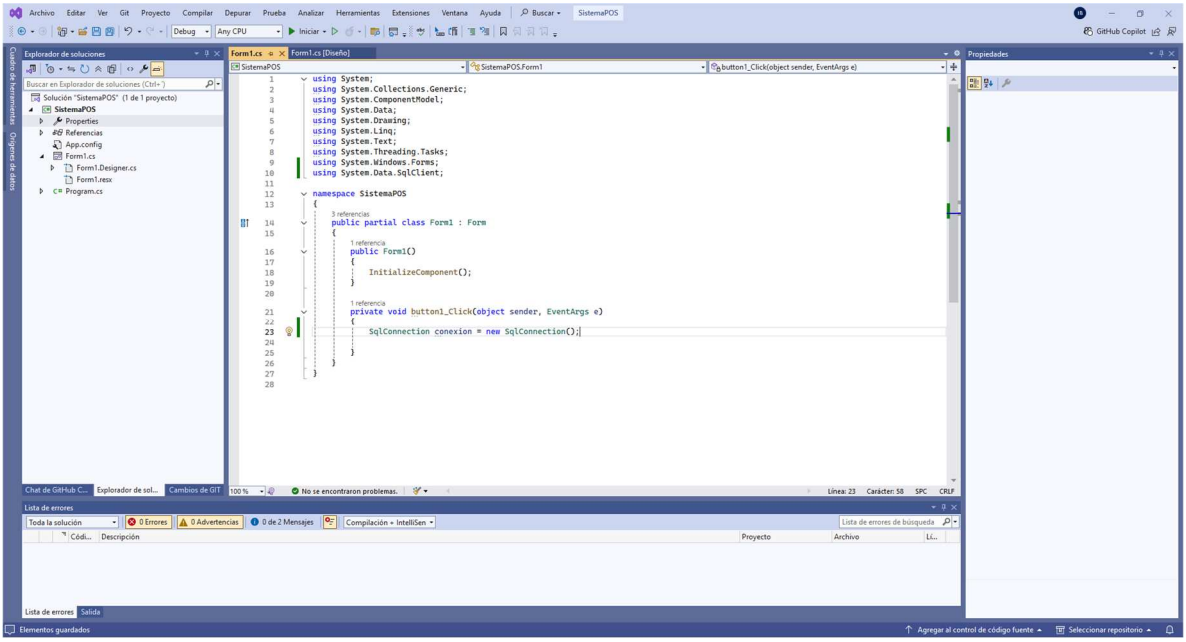




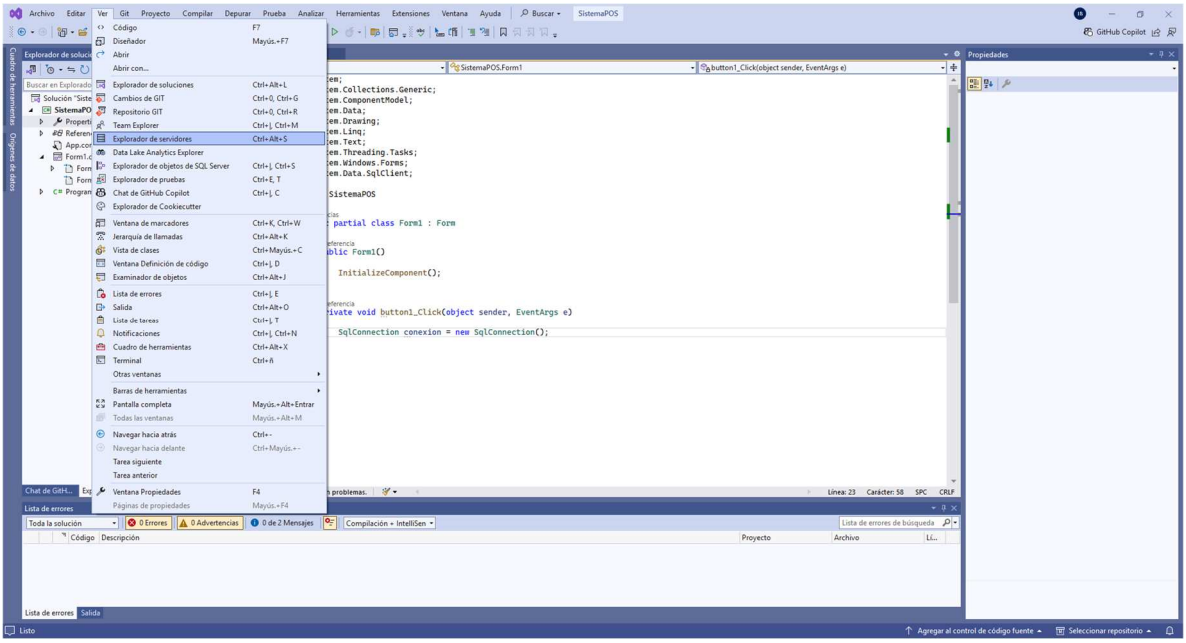
Añadir Boton - Conectar



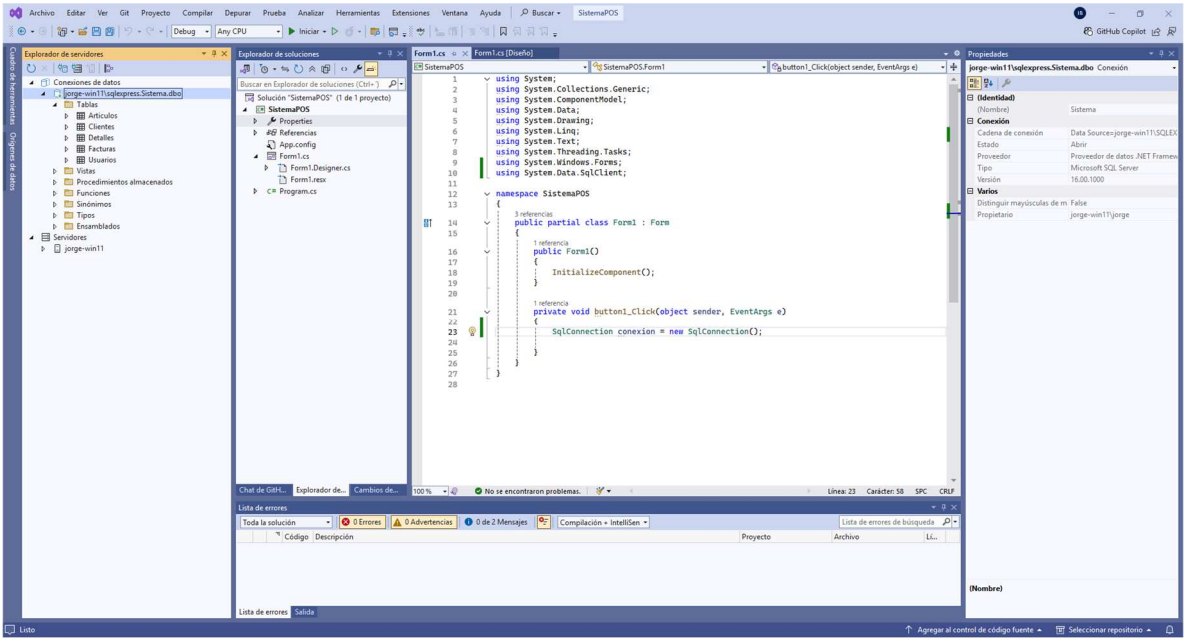
Agregamos conexión a sql server



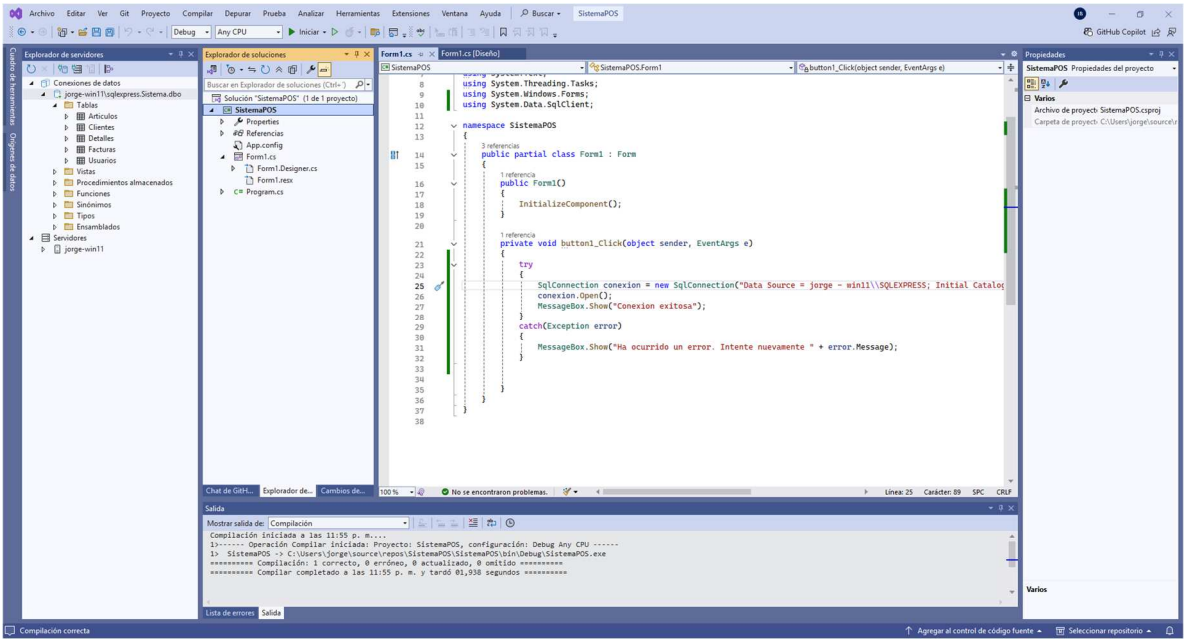
Explorador de servidores



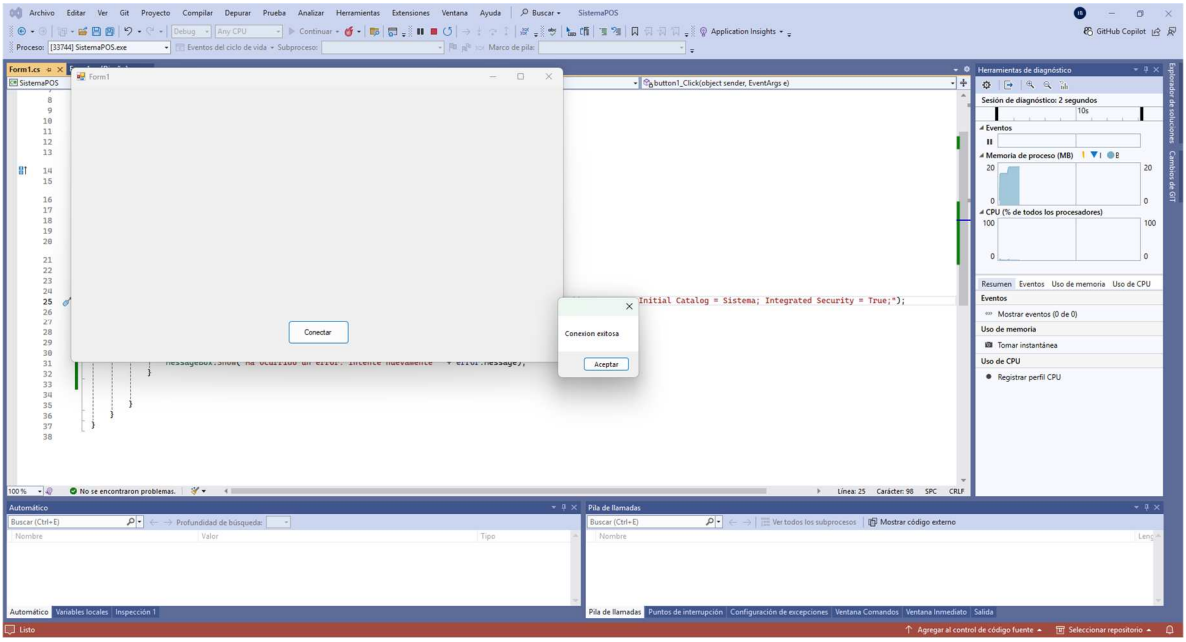
Conectarse a la base de datos



Copiar el string de conexión y compilamos

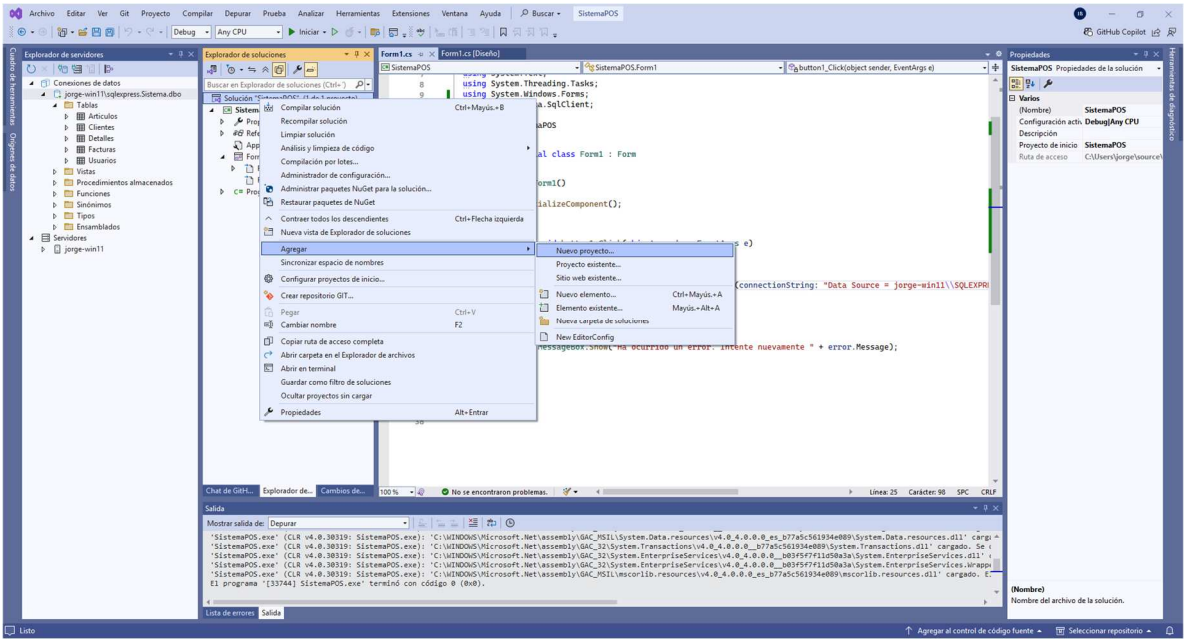


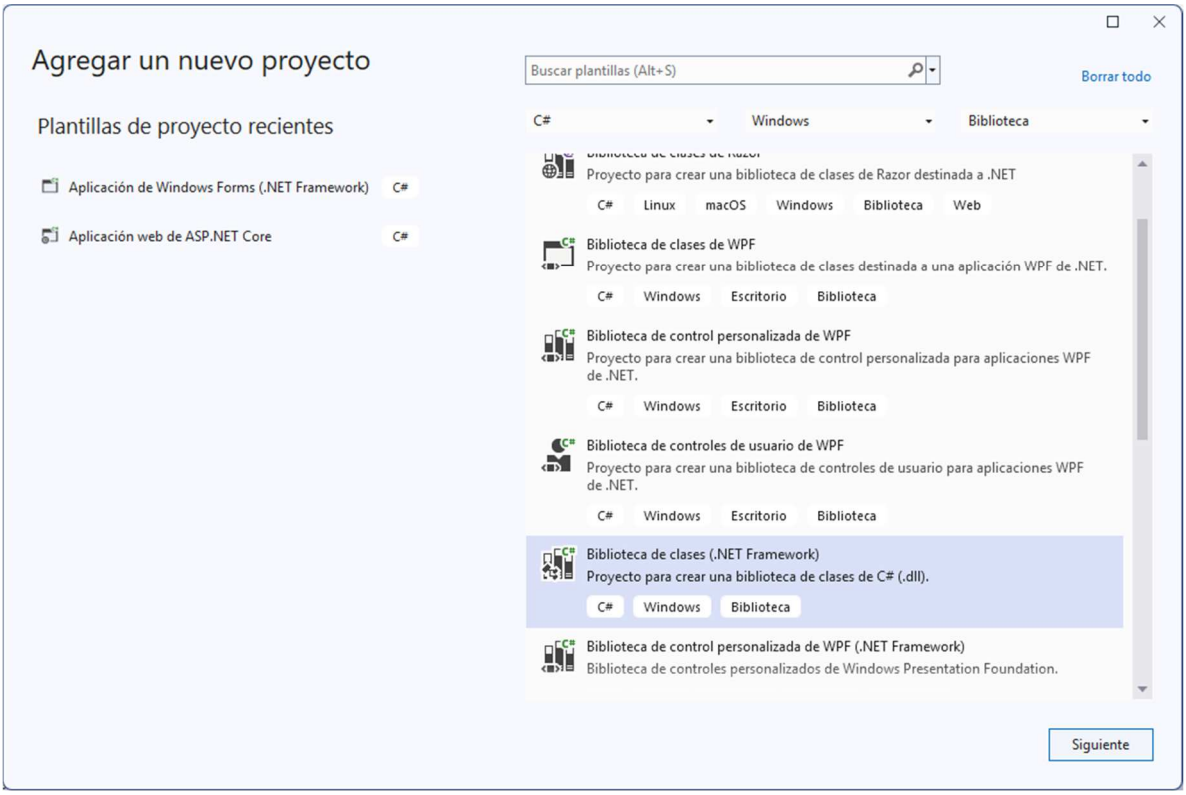
Probamos conexión

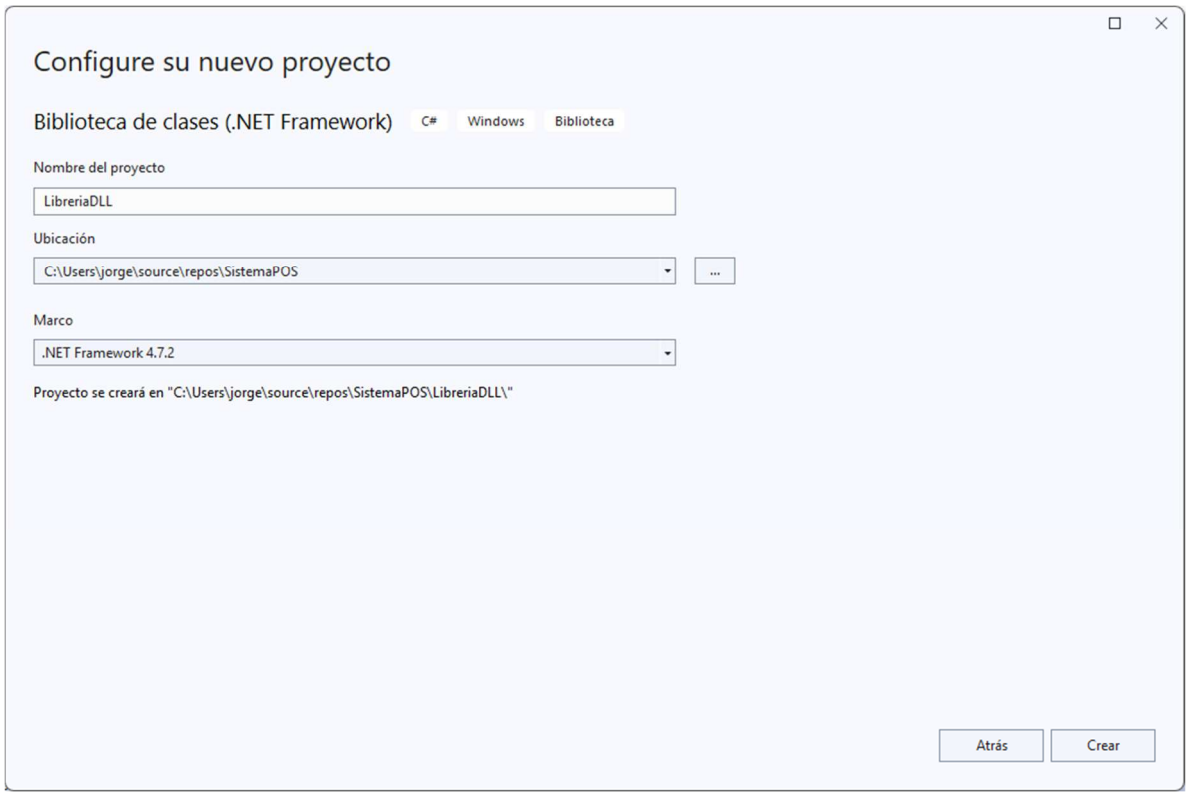


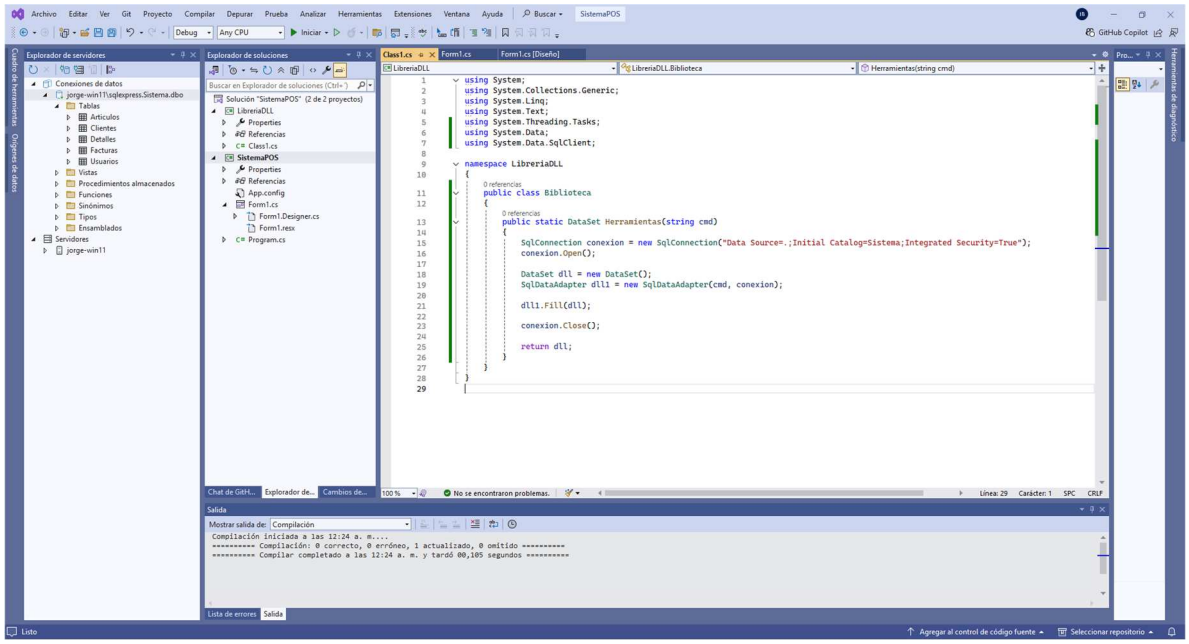
Creando libreria .dll _C_ avanzado_

Agregar nuevo proyecto


Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda
SistemaPOS
Explorador de servidores
Explorador de soluciones
Form1.cs
Form1.cs [Diseño]
Propiedades
Agregar
Nuevo proyecto...
Proyecto existente...
Sitio web existente...
Nuevo elemento... Ctrl+Mayús.+A
Elemento existente... Mayús.+Alt+A
Nueva carpeta de soluciones
New EditorConfig
Salida
Listo


Agregar un nuevo proyecto
Buscar plantillas (Alt+S)
Borrar todo
Plantillas de proyecto recientes
Aplicación de Windows Forms (.NET Framework) C#
Aplicación web de ASP.NET Core C#
C#
Windows
Biblioteca
Proyecto para crear una biblioteca de clases de Razor destinada a .NET
C# Linux macOS Windows Biblioteca Web
Biblioteca de clases de WPF
Proyecto para crear una biblioteca de clases destinada a una aplicación WPF de .NET.
C# Windows Escritorio Biblioteca
Biblioteca de control personalizada de WPF
Proyecto para crear una biblioteca de control personalizada para aplicaciones WPF de .NET.
C# Windows Escritorio Biblioteca
Biblioteca de controles de usuario de WPF
Proyecto para crear una biblioteca de controles de usuario para aplicaciones WPF de .NET.
C# Windows Escritorio Biblioteca
Biblioteca de clases (.NET Framework)
Proyecto para crear una biblioteca de clases de C# (.dll).
C# Windows Biblioteca
Biblioteca de control personalizada de WPF (.NET Framework)
Biblioteca de controles personalizados de Windows Presentation Foundation.
Siguiente

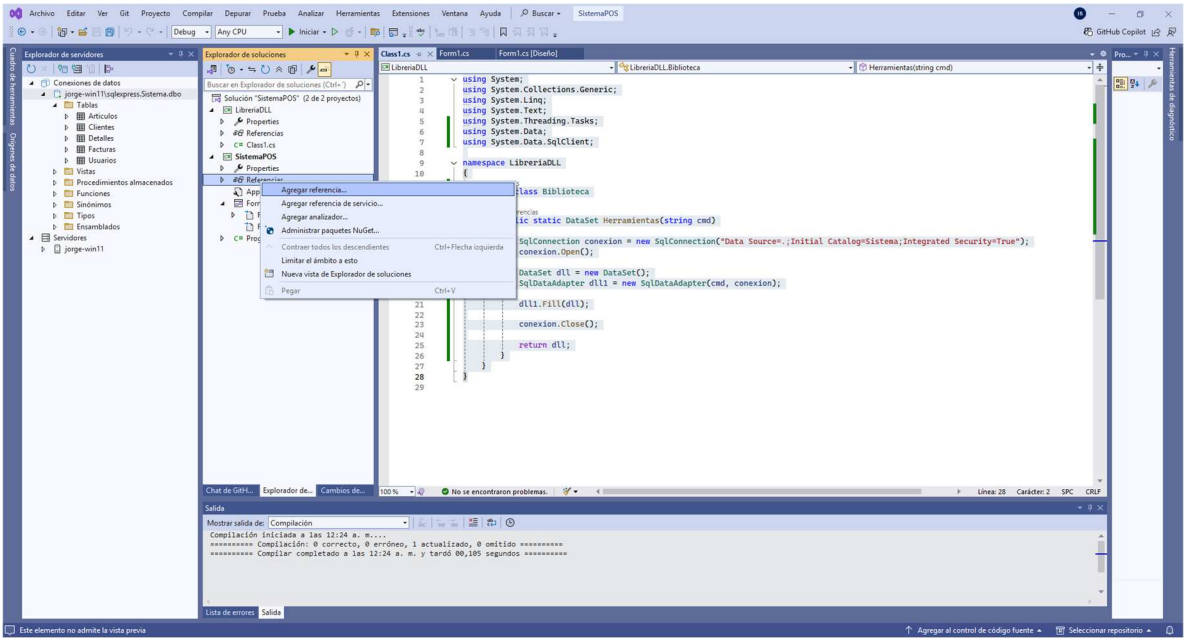


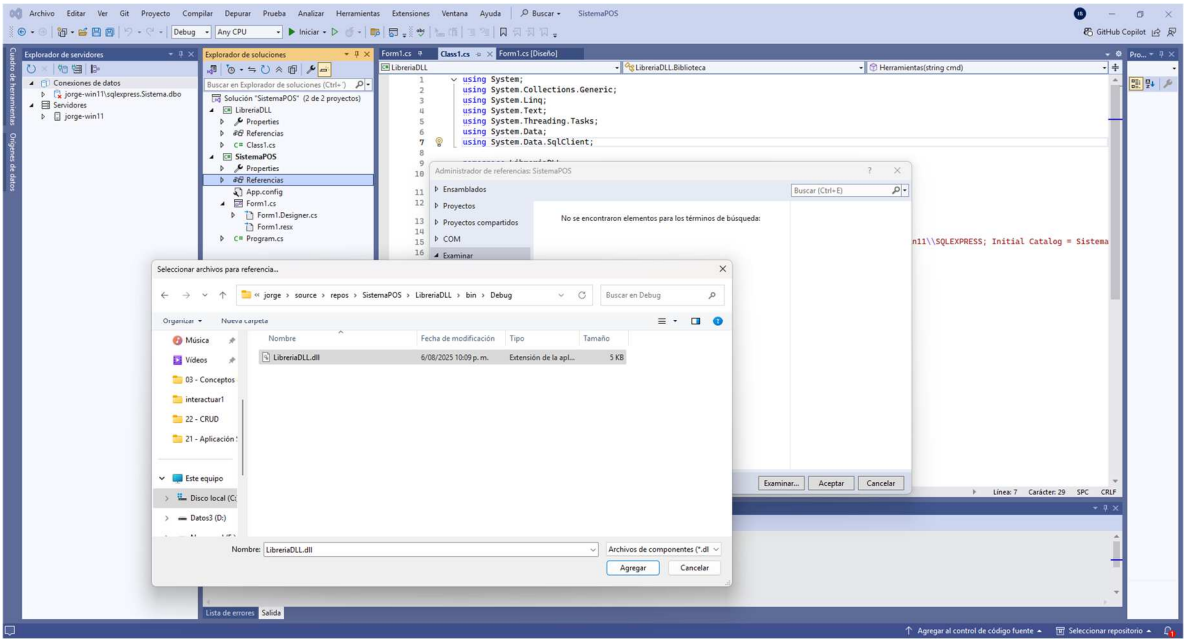


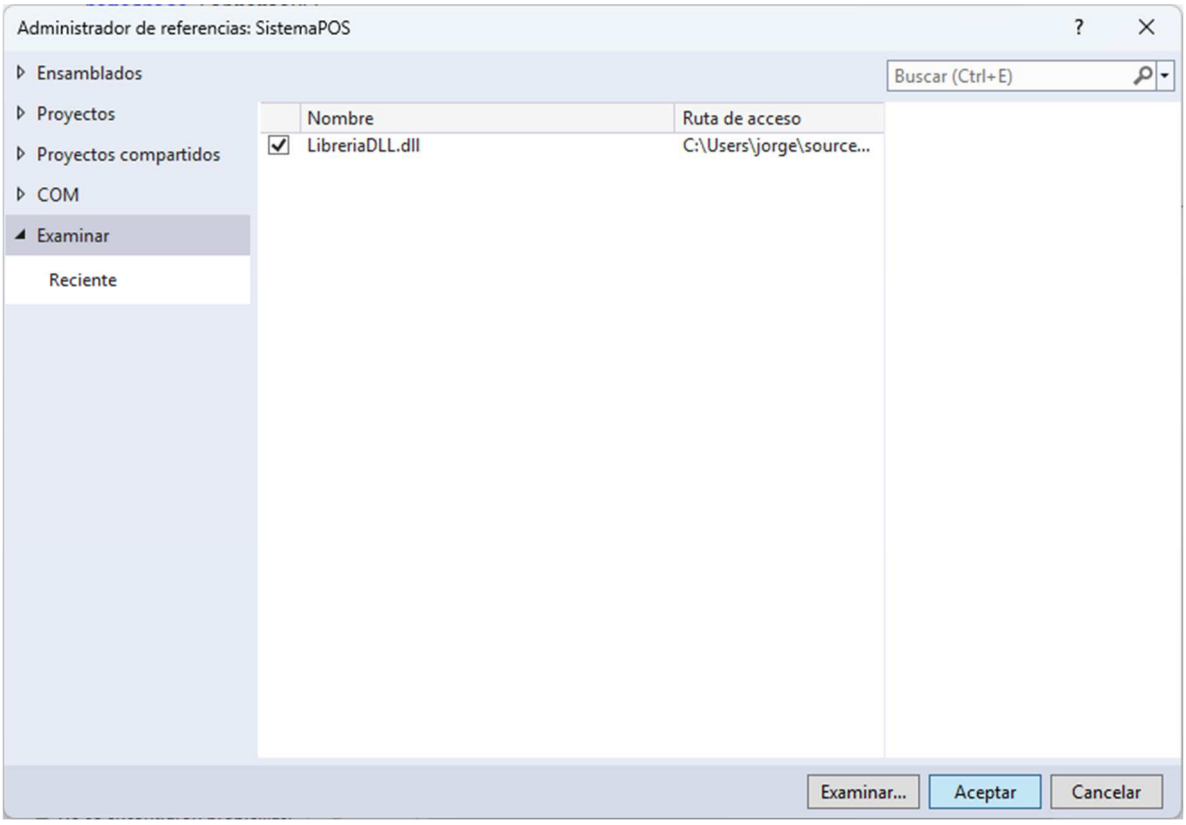

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
```

```
namespace LibreriaDLL
{
   public class Biblioteca
   {
      public static DataSet Herramientas(string cmd)
      {
         SqlConnection conexion = new SqlConnection("Data Source=.;Initial Catalog=Sistema;Integrated
Security=True");
         conexion.Open();

         DataSet dll = new DataSet();
         SqlDataAdapter dll1 = new SqlDataAdapter(cmd, conexion);

         dll1.Fill(dll);

         conexion.Close();

         return dll;
      }
   }
}
```

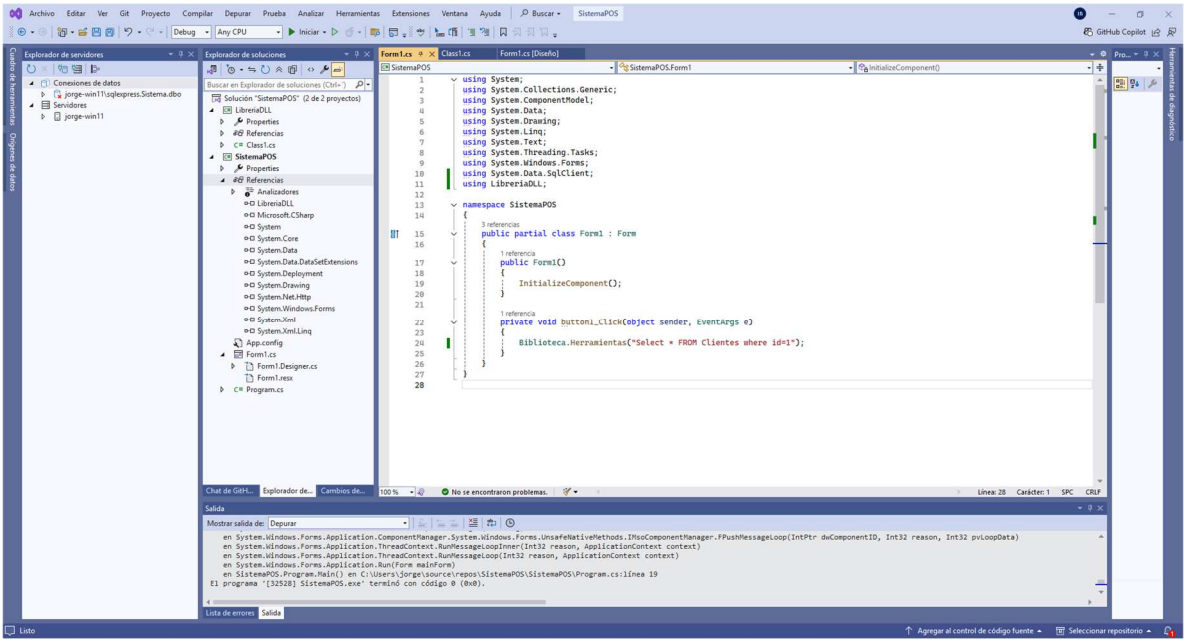
Agregamos el dll como referencia



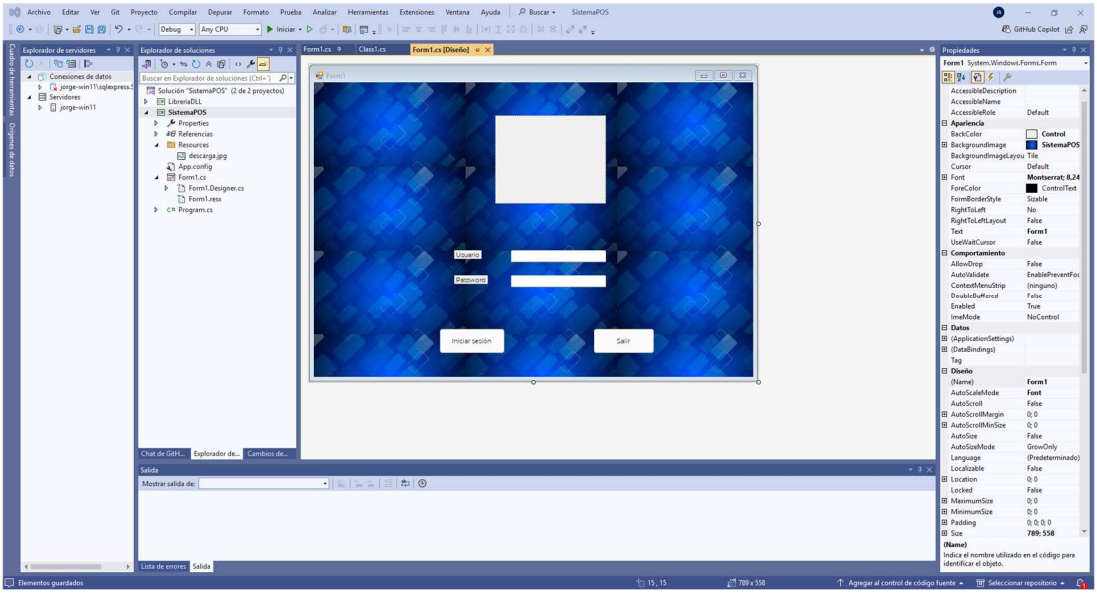




Modificamos Form1.cs



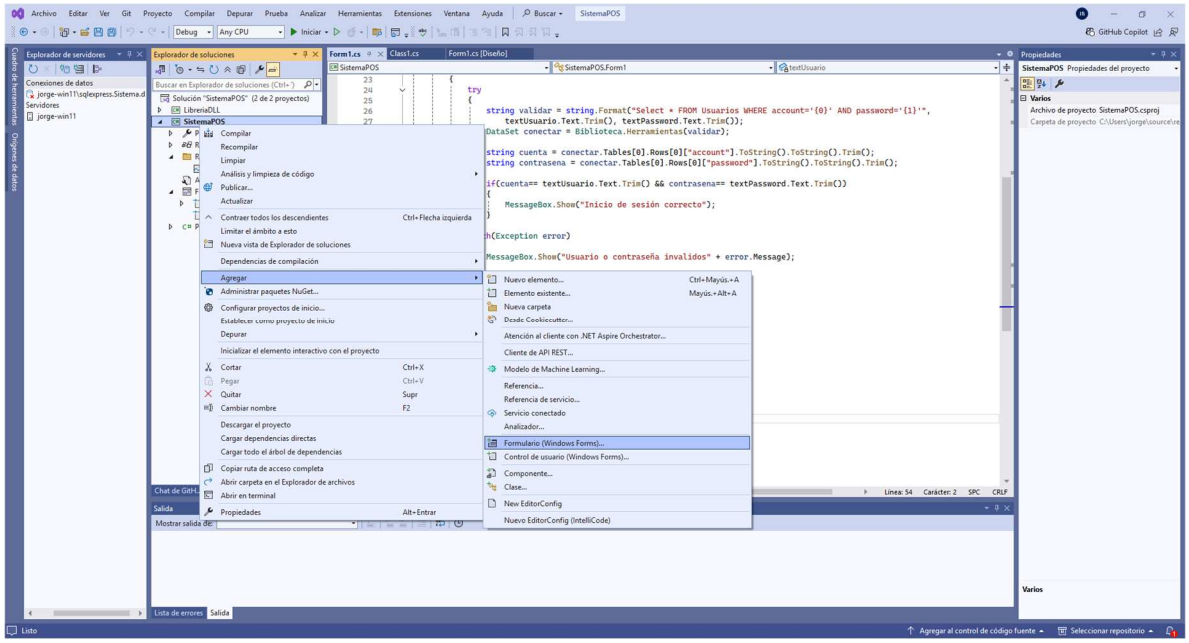
Formulario de inicio se sesión

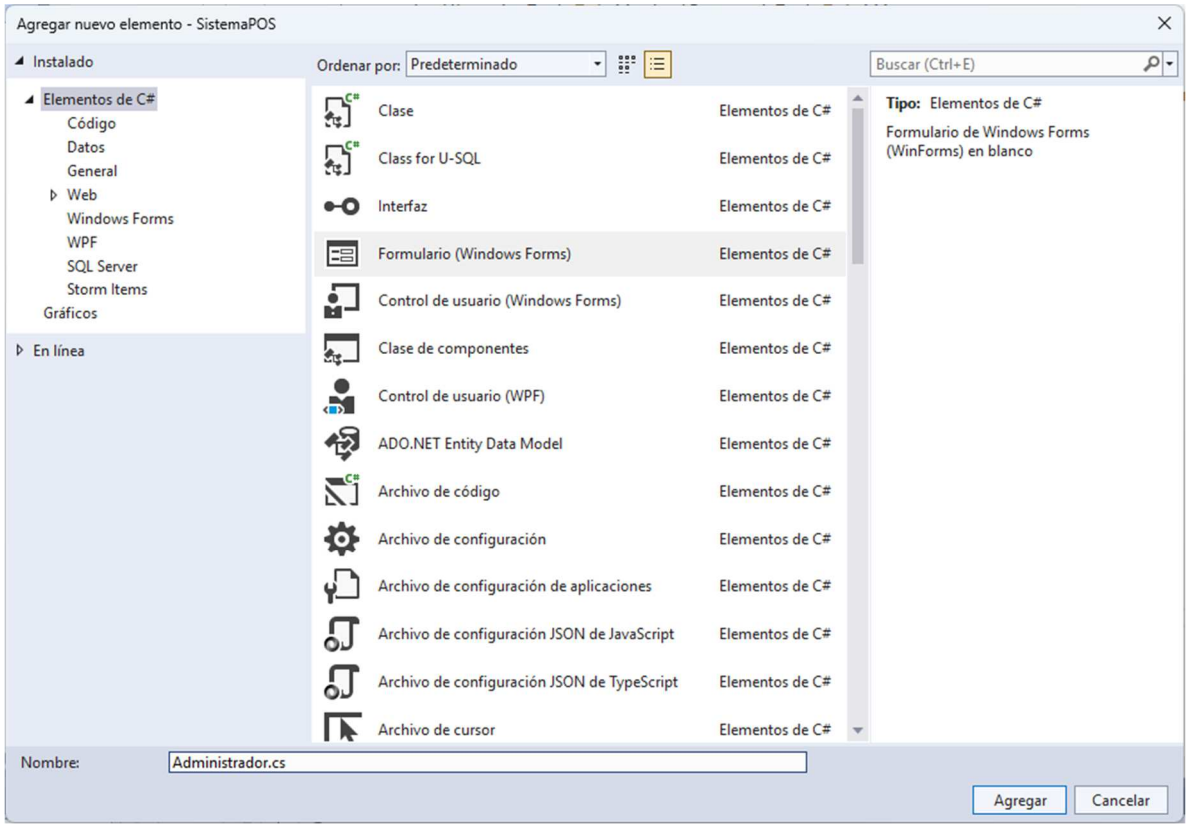


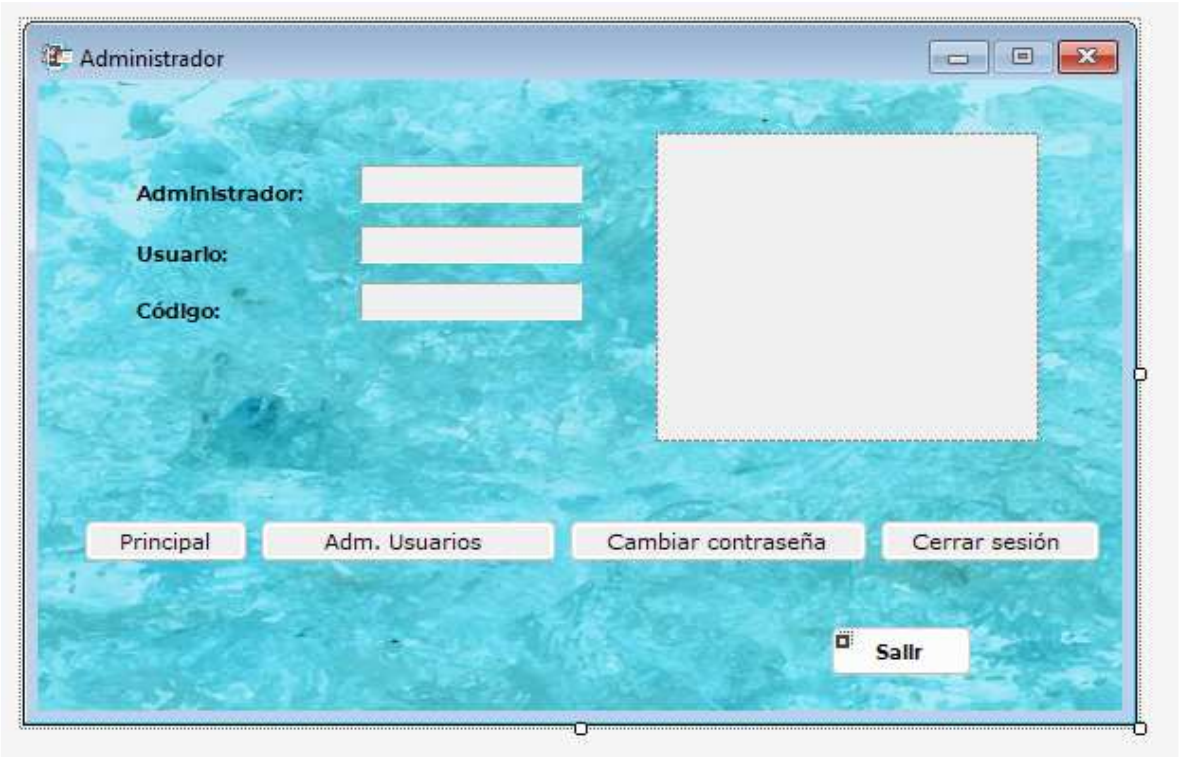

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
using System.Data.SqlClient;
using LibreriaDLL;

namespace SistemaPOS
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         try
         {
            string validar = string.Format("Select * FROM Usuarios WHERE account='{0}' AND password='{1}'",
               textUsuario.Text.Trim(), textPassword.Text.Trim());
            DataSet conectar = Biblioteca.Herramientas(validar);

            string cuenta = conectar.Tables[0].Rows[0]["account"].ToString().ToString().Trim();
            string contrasena = conectar.Tables[0].Rows[0]["password"].ToString().ToString().Trim();

            if(cuenta== textUsuario.Text.Trim() && contrasena== textPassword.Text.Trim())
            {
               MessageBox.Show("Inicio de sesión correcto");
            }
         }
         catch(Exception error)
         {
            MessageBox.Show("Usuario o contraseña invalidos" + error.Message);
         }
      }

      private void button2_Click(object sender, EventArgs e)
      {
         Application.Exit();
      }

      private void label1_Click(object sender, EventArgs e)
      {

      }
   }
}
```

Crear Formularios Usuario y Administrador

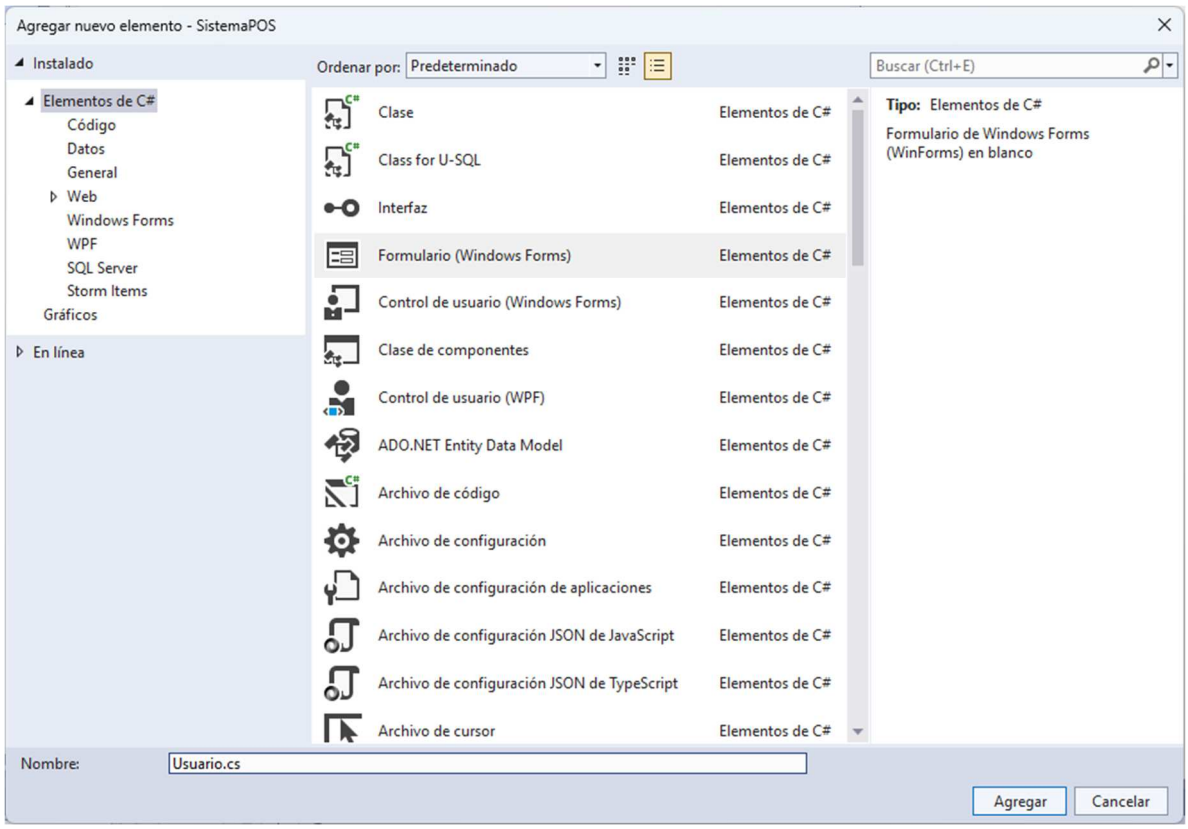

Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda
SistemaPOS
Debug
Any CPU
Iniciar
Explorador de servidores
Explorador de soluciones
Form1.cs
Class1.cs
Form1.cs [Diseño]
Propiedades
Solución "SistemaPOS" (2 de 2 proyectos)
LibreriaDLL
SistemaPOS
Compilar
Recompilar
Limpiar
Publicar...
Actualizar
Agregar
Nuevo elemento... Ctrl+Mayús.+A
Elemento existente... Mayús.+Alt+A
Nueva carpeta
Formulario (Windows Forms)...
Control de usuario (Windows Forms)...
Componente...
Clase...
Cortar Ctrl+X
Pegar Ctrl+V
Quitar Supr
Cambiar nombre F2
Propiedades Alt+Entrar
Lista de errores
Salida
Listo

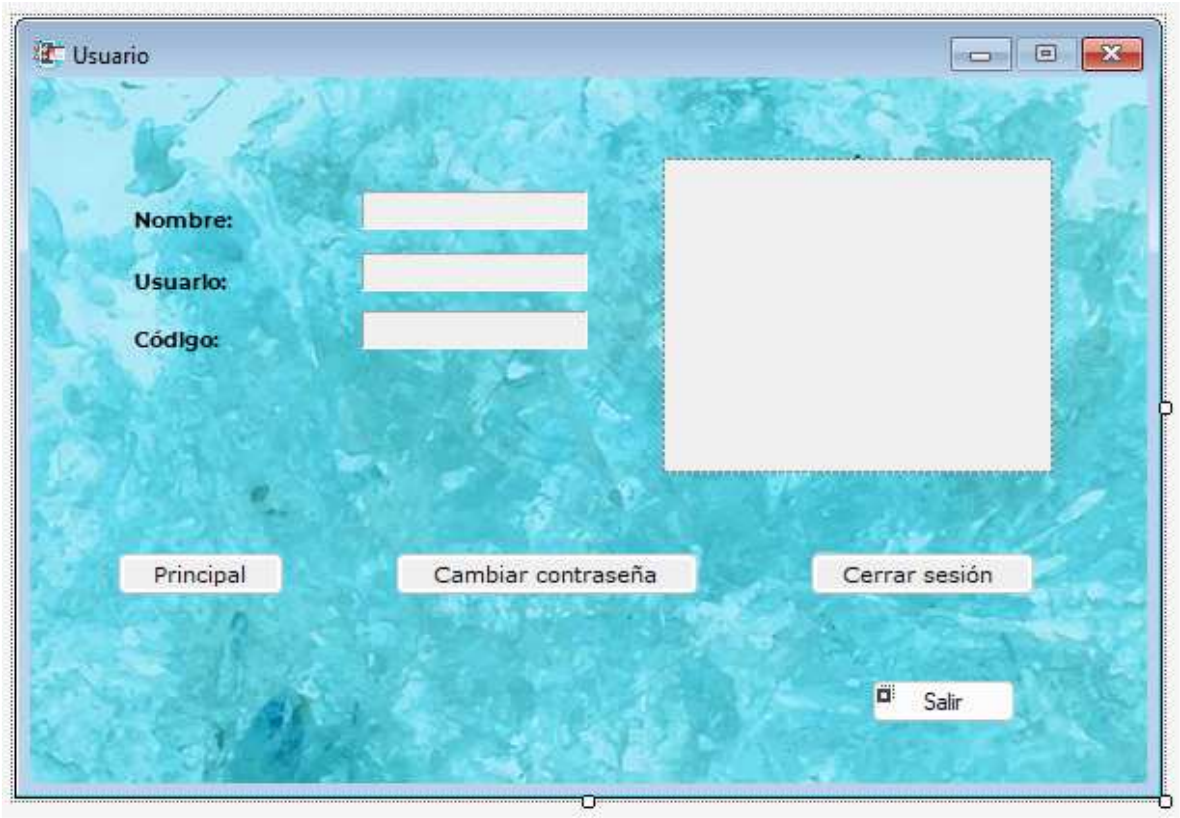

Agregar nuevo elemento - SistemaPOS
Instalado
Ordenar por: Predeterminado
Buscar (Ctrl+E)
Elementos de C#
Código
Datos
General
Web
Windows Forms
WPF
SQL Server
Storm Items
Gráficos
En línea
Clase
Class for U-SQL
Interfaz
Formulario (Windows Forms)
Control de usuario (Windows Forms)
Clase de componentes
Control de usuario (WPF)
ADO.NET Entity Data Model
Archivo de código
Archivo de configuración
Archivo de configuración de aplicaciones
Archivo de configuración JSON de JavaScript
Archivo de configuración JSON de TypeScript
Archivo de cursor
Tipo: Elementos de C#
Formulario de Windows Forms (WinForms) en blanco
Nombre:
Administrador.cs
Agregar
Cancelar




```
using MiConexionDLL;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SistemaPOS
{
   public partial class Administrador : Form
   {
      public Administrador()
      {
         InitializeComponent();
      }

      private void Principal_Click(object sender, EventArgs e)
      {
         //ContenedorPrincipal con_principal = new ContenedorPrincipal();
         //this.Hide();
         //con_principal.Show();
      }

      private void Administrador_FormClosed(object sender, FormClosedEventArgs e)
```

```
        {
            Application.Exit();
        }
    }
}
```






```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SistemaPOS
{
    public partial class Usuario : Form
    {
        public Usuario()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //ContenedorPrincipal con_principal = new ContenedorPrincipal();
            //this.Hide();
            //con_principal.Show();
        }

        private void Usuario_FormClosed(object sender, FormClosedEventArgs e)
```

```
        {
            Application.Exit();
        }
    }
}
```

Cambio en el código del formulario Login.cs

```
using MiConexionDLL;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SistemaPOS
{
    public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();
        }

        public static String Codigo = "";

        private void button2_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                ConexionSQL conexion = new ConexionSQL();
                using (SqlConnection conn = conexion.ObtenerConexion())
                {
                    conn.Open();
                    MessageBox.Show("Conexión exitosa a SQL Server 2022 Verificada");

                    // Ejemplo de consulta
                    string validar = string.Format("Select * FROM Usuarios WHERE account='{0}' AND password='{1}'",
                    textUsuario.Text.Trim(), textPassword.Text.Trim());
                    SqlCommand cmd = new SqlCommand(validar, conn);
                    SqlDataAdapter da = new SqlDataAdapter(cmd);
                    DataSet conectar = new DataSet();
```

```
            //DataTable dt = new DataTable();

            da.Fill(conectar);

            Codigo = conectar.Tables[0].Rows[0]["id_usuario"].ToString().Trim();
            string cuenta = conectar.Tables[0].Rows[0]["account"].ToString().ToString().Trim();
            string contrasena = conectar.Tables[0].Rows[0]["password"].ToString().ToString().Trim();

            if (cuenta == textUsuario.Text.Trim() && contrasena == textPassword.Text.Trim())
            {
               MessageBox.Show("Inicio de sesión correcto");
            }

            if (cuenta == textUsuario.Text.Trim() && contrasena == textPassword.Text.Trim())
            {
               if (Convert.ToBoolean(conectar.Tables[0].Rows[0]["validar_admin"].ToString().Trim()) == true)
               {
                  Administrador Admin = new Administrador();
                  this.Hide();
                  Admin.Show();
               }
               else
               {
                  Usuario User = new Usuario();
                  this.Hide();
                  User.Show();
               }
            }
            else
            {
               MessageBox.Show("Usuario incorrecto o contraseña incorrectos");
            }
         }
      }
      catch (Exception ex)
      {
         MessageBox.Show("Error: " + ex.Message);
      }

   }

   private void Login_FormClosed(object sender, FormClosedEventArgs e)
   {
      Application.Exit();
   }
  }
}
```

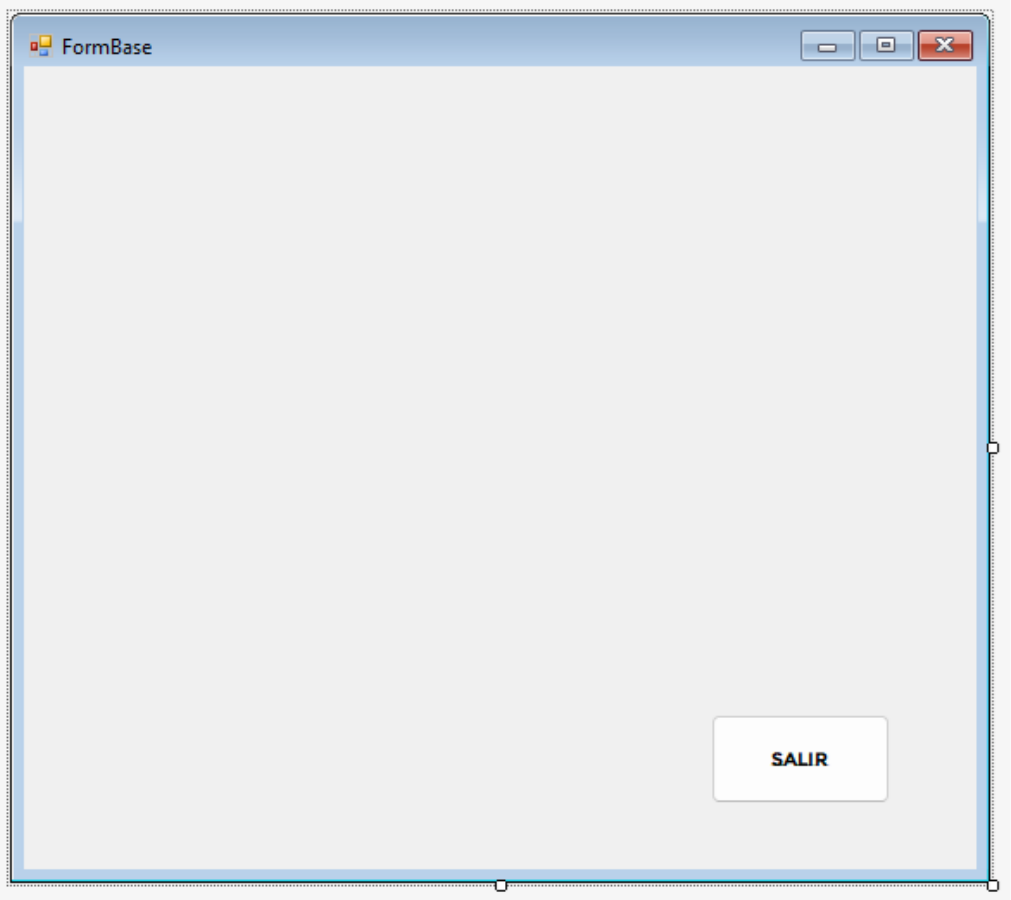
Nota: probar la ejecución del proyecto. Despliegue de ventanas Administrador y Usuario

Programar Formularios Usuario y Administrador

```
private void Administrador_Load(object sender, EventArgs e)
{
    ConexionSQL conexion = new ConexionSQL();
    using (SqlConnection conn = conexion.ObtenerConexion())
    {
        conn.Open();
        string consulta = "SELECT * FROM Usuarios WHERE id_usuario=" + Login.Codigo;
        SqlCommand cmd = new SqlCommand(consulta, conn);
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        DataSet conectar = new DataSet();
        da.Fill(conectar);

        lAdmin.Text = conectar.Tables[0].Rows[0]["username"].ToString();
        lAdminUser.Text = conectar.Tables[0].Rows[0]["account"].ToString();
        lAdminCodigo.Text = conectar.Tables[0].Rows[0]["id_usuario"].ToString();

        string imagen = conectar.Tables[0].Rows[0]["imagen"].ToString();
        pictureBox1.Image = Image.FromFile(imagen);
    }
}


private void Usuario_Load(object sender, EventArgs e)
{
    ConexionSQL conexion = new ConexionSQL();
    using (SqlConnection conn = conexion.ObtenerConexion())
    {
        conn.Open();
        string consulta = "SELECT * FROM Usuarios WHERE id_usuario=" + Login.Codigo;
        SqlCommand cmd = new SqlCommand(consulta, conn);
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        DataSet conectar = new DataSet();
        da.Fill(conectar);

        lNombre.Text = conectar.Tables[0].Rows[0]["username"].ToString();
        lUsuario.Text = conectar.Tables[0].Rows[0]["account"].ToString();
        lCodigo.Text = conectar.Tables[0].Rows[0]["id_usuario"].ToString();

        string imagen = conectar.Tables[0].Rows[0]["imagen"].ToString();
        pictureBox1.Image = Image.FromFile(imagen);
    }
}
```

Herencia en ventanas

Creamos formulario FormBase



Propiedad Modifier de botón a public y cambiamos la herencia de los formularios a FormBase.

Programamos el botón salir

```
private void btnSalir_Click(object sender, EventArgs e)
{
   if (MessageBox.Show("Desea salir? ", "Aviso", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes);
   {
      this.Close();
   }
}
```

Y creamos cuatro métodos virtuales para reutilizar.

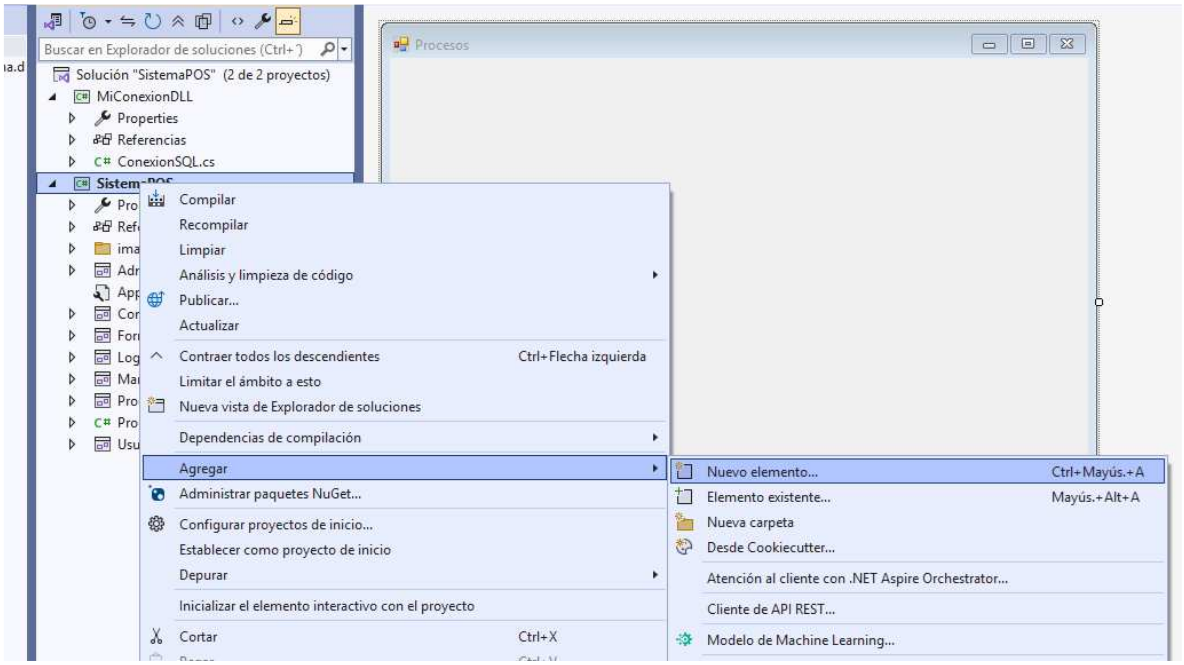
```
public virtual void Eliminar()
{

}

public virtual void Nuevo()
{
```

```
}

public virtual void Consultar()
{

}
public virtual Boolean Guardar()
{
   return false;
}
```

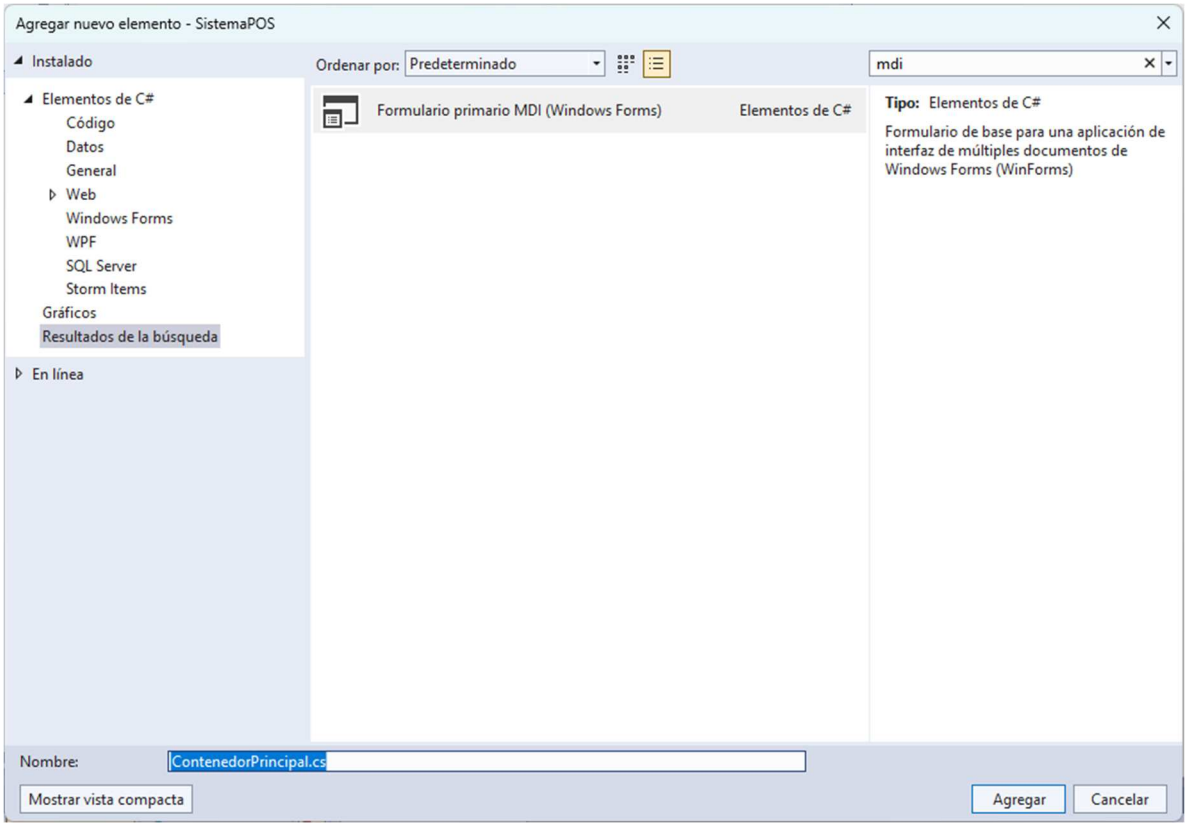
Contenedor Principal

- Crear Formularios para: Mantenimiento, Consultas y Procesos que hereden de FormBase

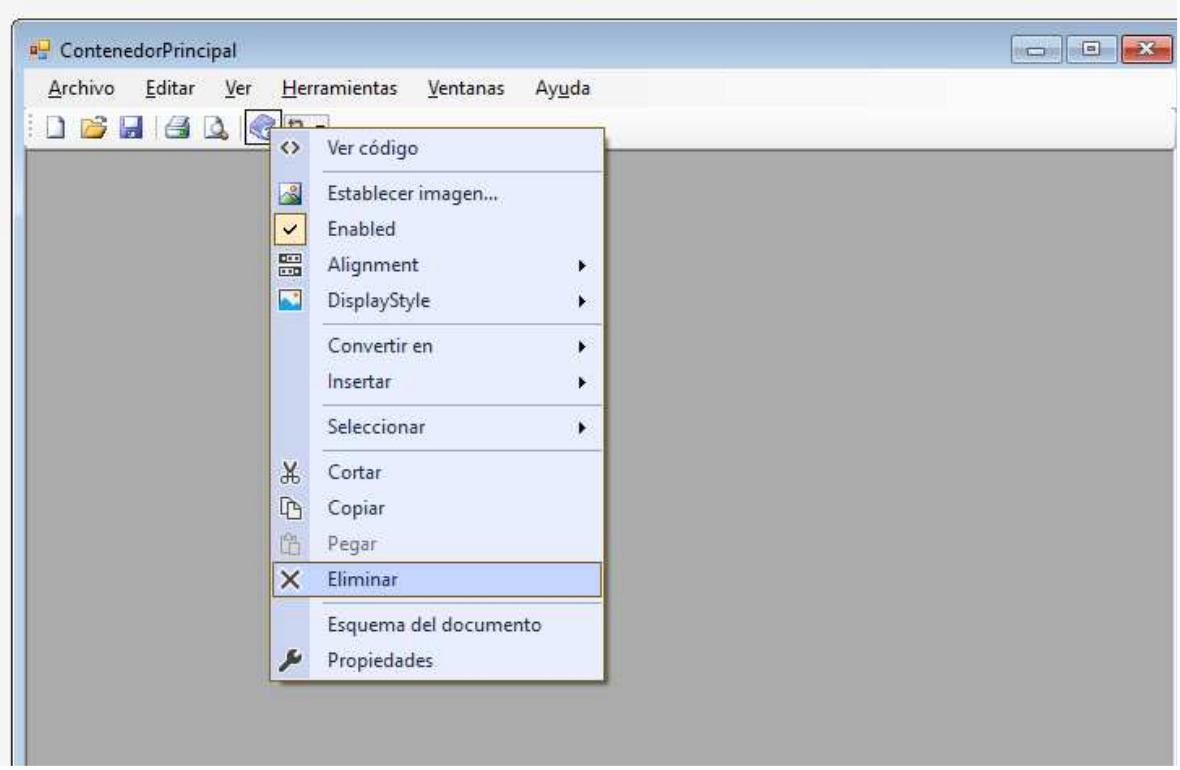
- Agregar Nuevo Elemento:



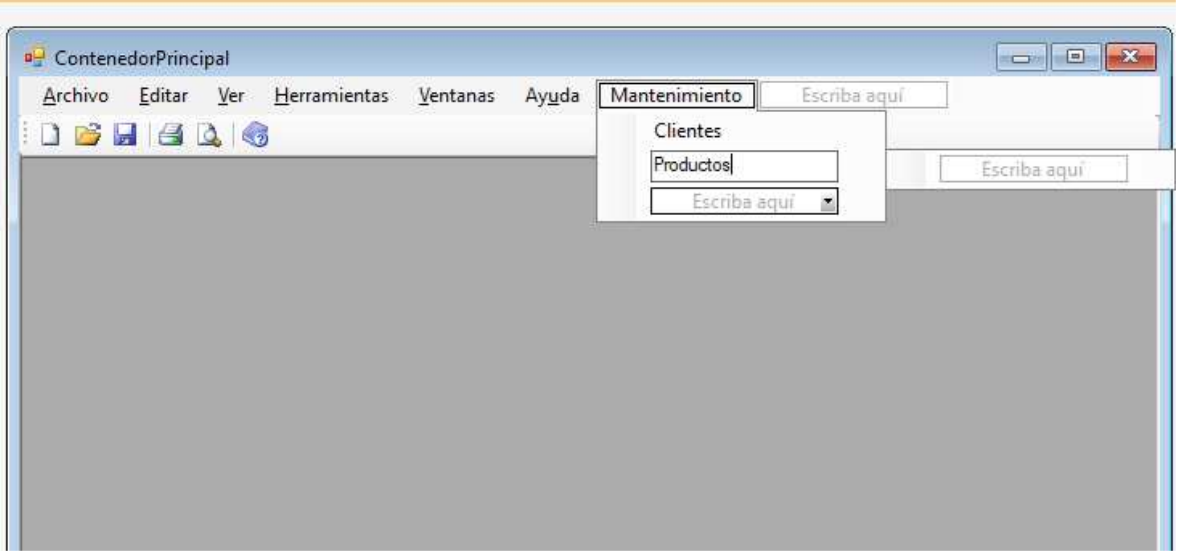
- En buscar escribimos mdi:

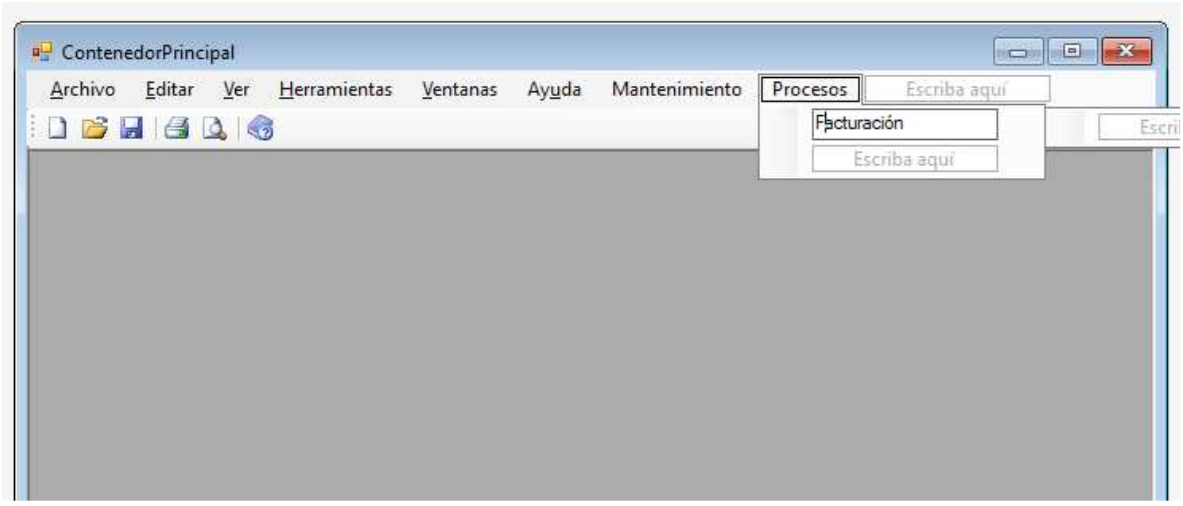


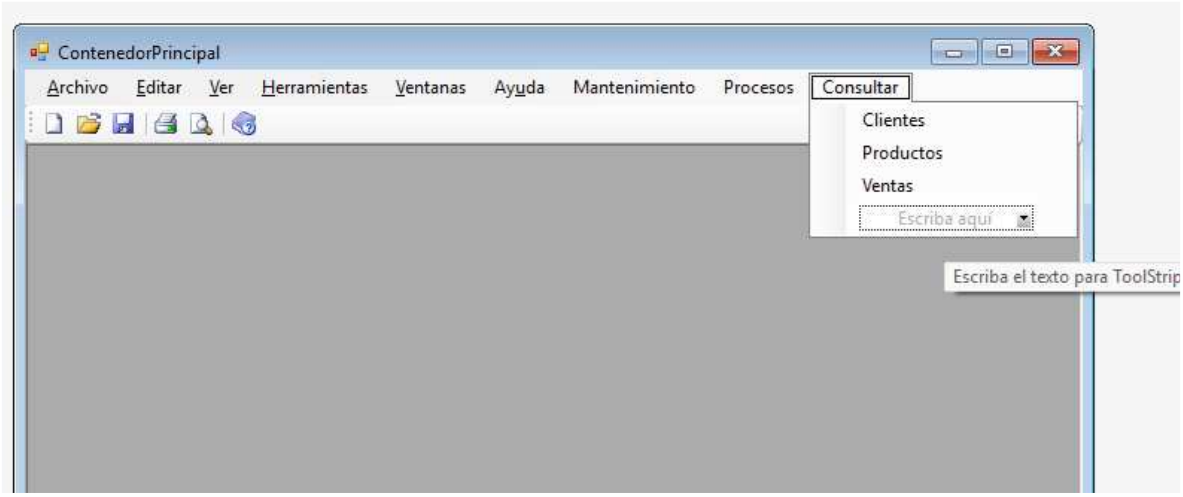
- En el contenedorPrincipal eliminamos el ultimo boton



- En el contenedorPrincipal agregamos elementos al menú superior asi:





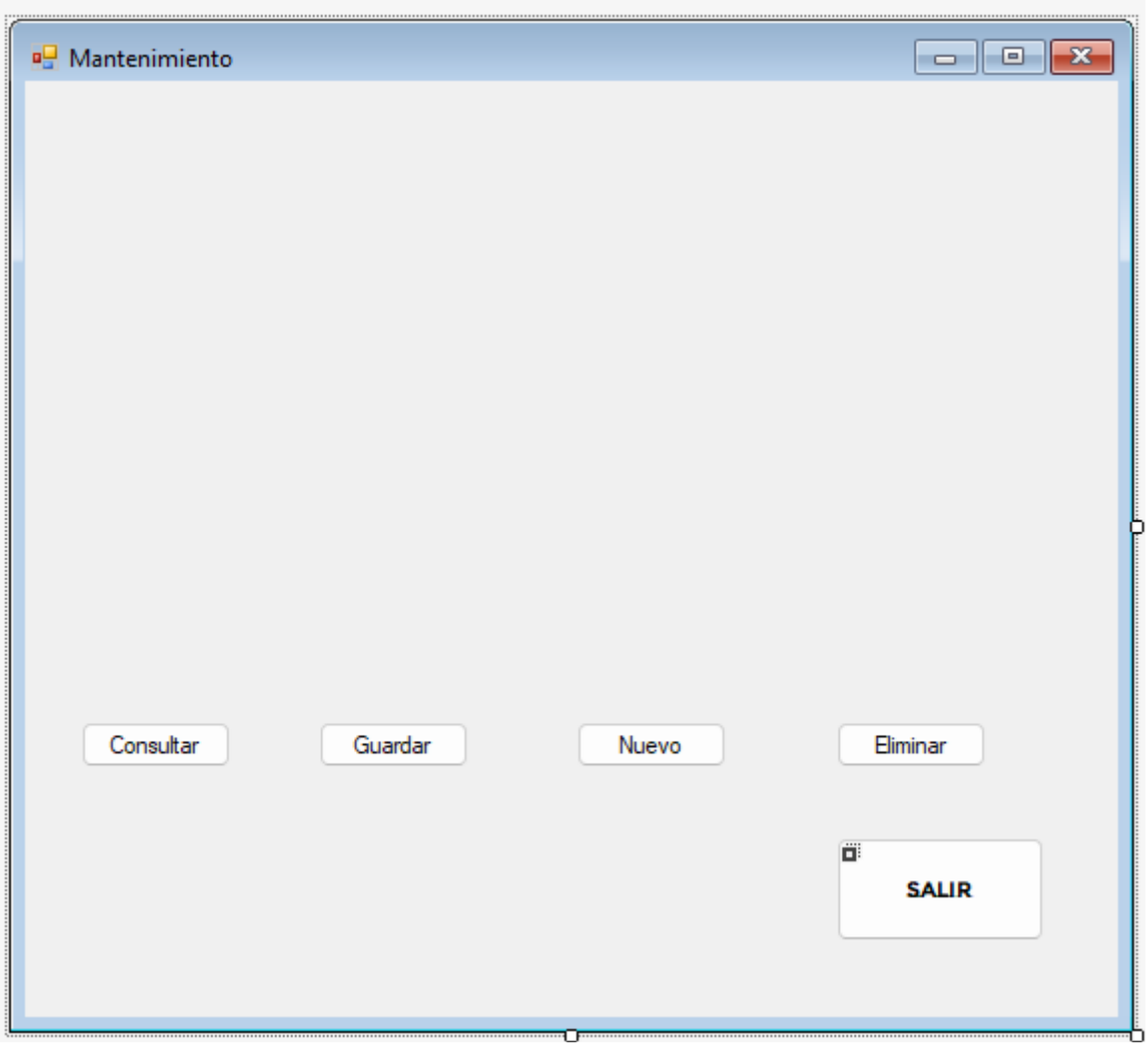


- En la ventana del Administrador y Usuarios programamos el botón Principal y probamos el llamado al Contenedor principal.

```
private void Principal_Click(object sender, EventArgs e)
{
   ContenedorPrincipal con_principal = new ContenedorPrincipal();
   this.Hide();
   con_principal.Show();
}
```

- Creamos un formulario MantenimientoClientes que debe heredar de Mantenimiento

- En Mantenimiento agregamos 4 Botones y cambiamos la propiedad modifier a public:

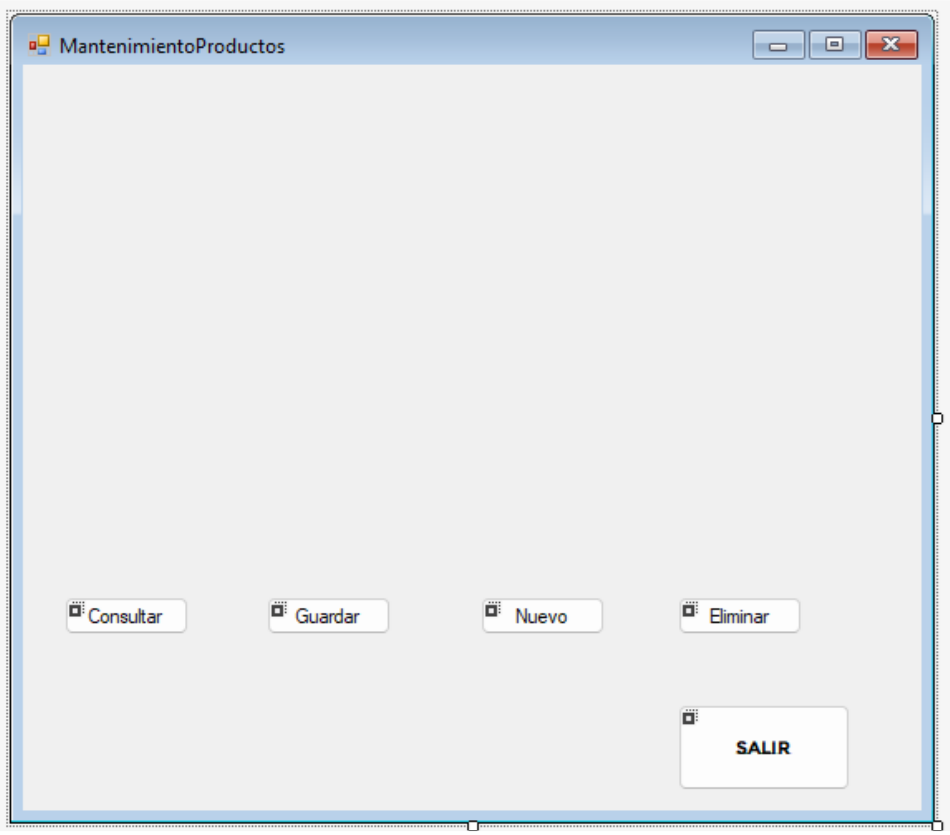
- 

- Programamos los botones para que llamen su respectivos métodos virtuales, creados en Formbase.

```
private void button1_Click(object sender, EventArgs e)
{
   Consultar();
}

private void button2_Click(object sender, EventArgs e)
{
   Guardar();
}

private void button3_Click(object sender, EventArgs e)
{
   Nuevo();
}
```

```
private void button4_Click(object sender, EventArgs e)
{
   Eliminar();
}
```

- Creamos un formulario MantenimientoProductos que debe heredar de Mantenimiento



- Vamos a llamar las ventanas dentro del contenedor Principal. Doble Click sobre Mantenimiento->Clientes

```
private void clientesToolStripMenuItem_Click(object sender, EventArgs e)
{
   MantenimientoClientes ManCli = new MantenimientoClientes();
   ManCli.MdiParent = this;
   ManCli.Show();
}
```

Compilamos y probamos el llamado.

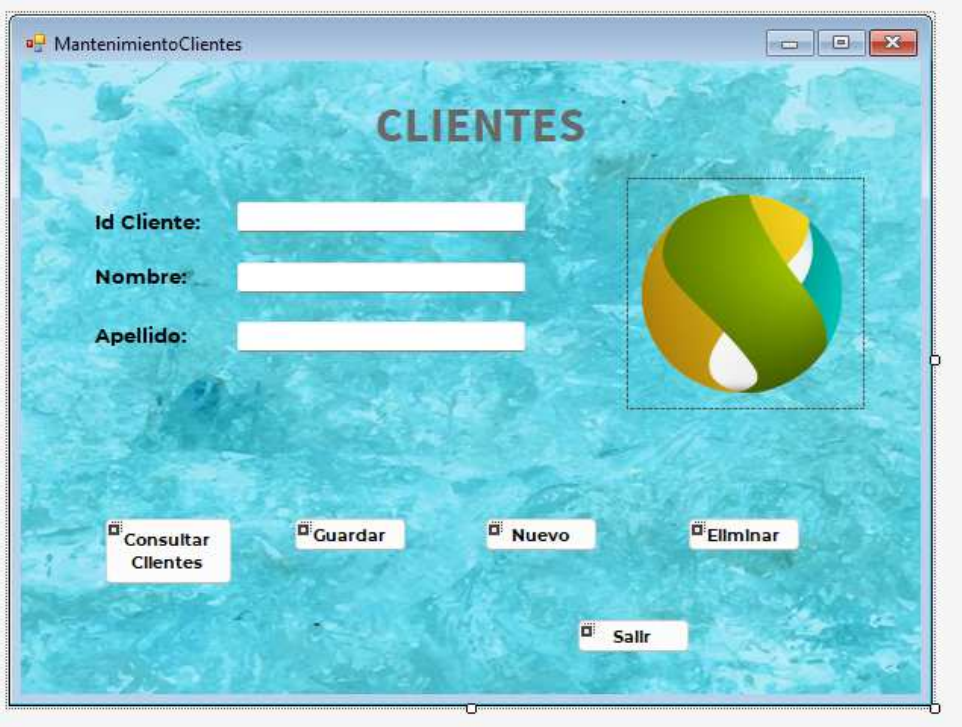
- Vamos a llamar las ventanas dentro del contenedor Principal. Doble Click sobre Mantenimiento->Productos

```
private void productosToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
   MantenimientoProductos ManPro = new MantenimientoProductos();
   ManPro.MdiParent = this;
   ManPro.Show();
}
```

Compilamos y probamos el llamado.

- Verificamos los procedimientos almacenados en SQL server.
- Modificamos la ventana MantenimientoClientes quedando asi:



- Modificamos la ventana MantenimientoProductos quedando asi:

Guardar eliminar clientes y productos

- Dentro de Mantenimiento Producto vamos crear métodos para el llamado a los procedimientos almacenados

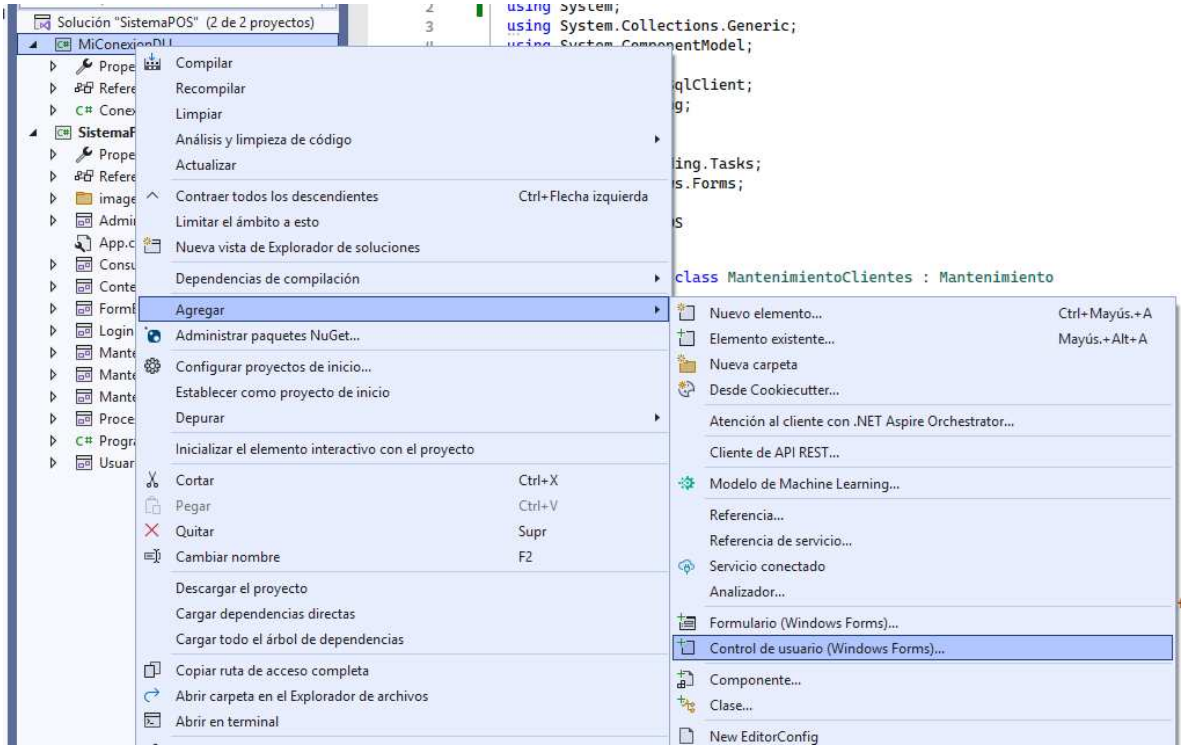
```
public override Boolean Guardar()
{
   try
   {
      ConexionSQL conexion = new ConexionSQL();
      using (SqlConnection conn = conexion.ObtenerConexion())
      {
         conn.Open();
         MessageBox.Show("Conexión exitosa a SQL Server 2022 Verificada");

         // Ejemplo de consulta
         string insertar = string.Format("EXEC ActualizarProductos '{0}','{1}','{2}'", textId_Producto.Text.Trim(),
textDescripcion.Text.Trim(), textPrecio.Text.Trim());
         SqlCommand cmd = new SqlCommand(insertar, conn);
         SqlDataAdapter da = new SqlDataAdapter(cmd);
         DataSet conectar = new DataSet();
         //da.Fill(conectar);
         MessageBox.Show("Producto guardado correctamente");
         return true;
      }
```

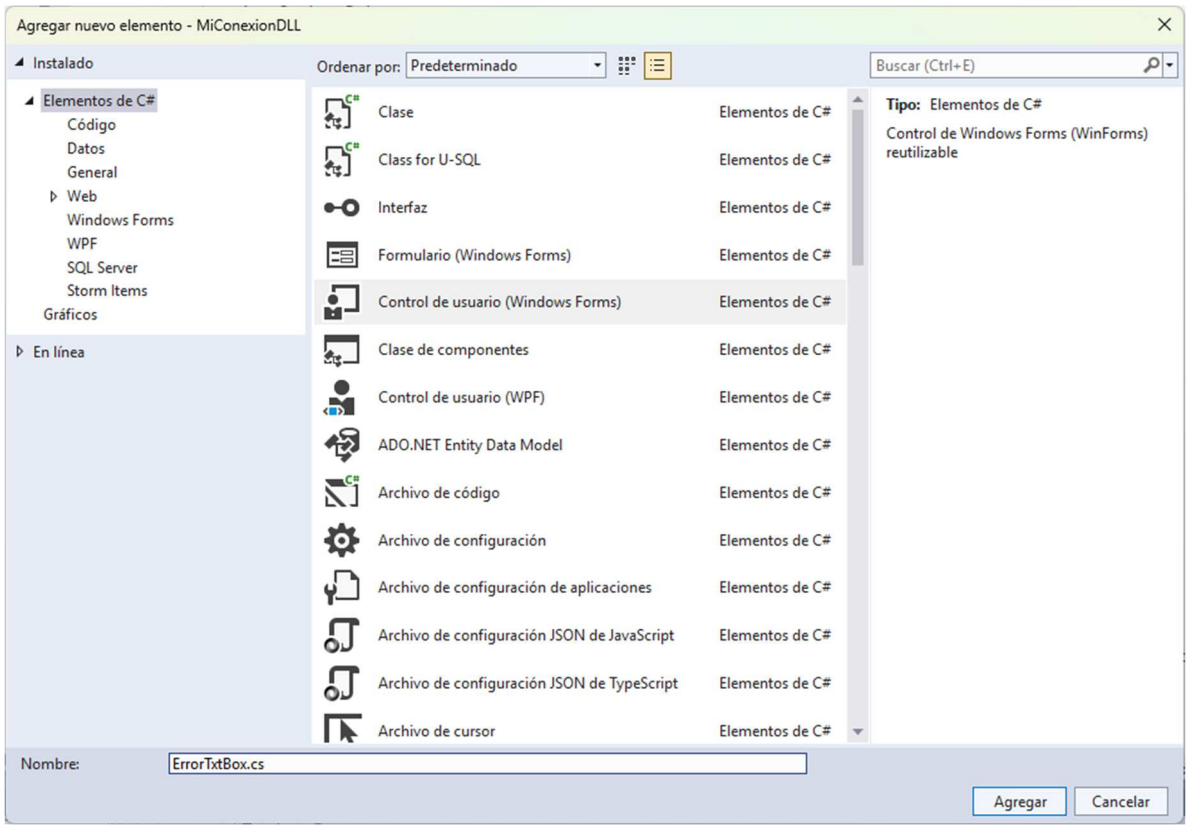
```
    }
    catch (Exception error)
    {
        MessageBox.Show("Ha ocurrido un error: " + error);
        return false;

    }

}


public override void Eliminar()
{
    try
    {
        ConexionSQL conexion = new ConexionSQL();
        using (SqlConnection conn = conexion.ObtenerConexion())
        {
            conn.Open();
            MessageBox.Show("Conexión exitosa a SQL Server 2022 Verificada");
            string eliminar = string.Format("EXEC EliminarProductos '{0}'", textId_Producto.Text.Trim());
            SqlCommand cmd = new SqlCommand(eliminar, conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataSet conectar = new DataSet();
            da.Fill(conectar);
            MessageBox.Show("El producto se ha eliminado correctamente");
        }
    }
    catch (Exception error)
    {
        MessageBox.Show("Ha ocurrido un error: " + error);
    }
}
```

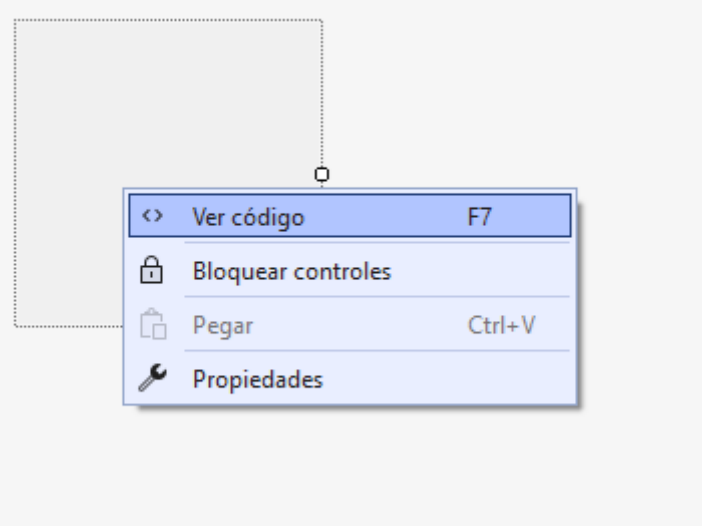
- Dentro de Mantenimiento Clientes vamos crear los mismos métodos para el llamado a los procedimientos almacenados

Validaciones

- Agregamos Control de usuario(Windows Forms)


Solución "SistemaPOS" (2 de 2 proyectos)
MiConexionDLL
Compilar
Recompilar
Limpiar
Análisis y limpieza de código
Actualizar
Contraer todos los descendientes Ctrl+Flecha izquierda
Limitar el ámbito a esto
Nueva vista de Explorador de soluciones
Dependencias de compilación
Agregar
Administrar paquetes NuGet...
Configurar proyectos de inicio...
Establecer como proyecto de inicio
Depurar
Inicializar el elemento interactivo con el proyecto
Cortar Ctrl+X
Pegar Ctrl+V
Quitar Supr
Cambiar nombre F2
Descargar el proyecto
Cargar dependencias directas
Cargar todo el árbol de dependencias
Copiar ruta de acceso completa
Abrir carpeta en el Explorador de archivos
Abrir en terminal
Nuevo elemento... Ctrl+Mayús.+A
Elemento existente... Mayús.+Alt+A
Nueva carpeta
Desde Cookiecutter...
Atención al cliente con .NET Aspire Orchestrator...
Cliente de API REST...
Modelo de Machine Learning...
Referencia...
Referencia de servicio...
Servicio conectado
Analizador...
Formulario (Windows Forms)...
Control de usuario (Windows Forms)...
Componente...
Clase...
New EditorConfig
using System;
using System.Collections.Generic;
class MantenimientoClientes : Mantenimiento


Agregar nuevo elemento - MiConexionDLL
Instalado
Elementos de C#
Código
Datos
General
Web
Windows Forms
WPF
SQL Server
Storm Items
Gráficos
En línea
Ordenar por: Predeterminado
Buscar (Ctrl+E)
Clase Elementos de C#
Class for U-SQL Elementos de C#
Interfaz Elementos de C#
Formulario (Windows Forms) Elementos de C#
Control de usuario (Windows Forms) Elementos de C#
Clase de componentes Elementos de C#
Control de usuario (WPF) Elementos de C#
ADO.NET Entity Data Model Elementos de C#
Archivo de código Elementos de C#
Archivo de configuración Elementos de C#
Archivo de configuración de aplicaciones Elementos de C#
Archivo de configuración JSON de JavaScript Elementos de C#
Archivo de configuración JSON de TypeScript Elementos de C#
Archivo de cursor Elementos de C#
Tipo: Elementos de C#
Control de Windows Forms (WinForms) reutilizable
Nombre: ErrorTxtBox.cs
Agregar
Cancelar




```
namespace MiConexionDLL
{
    2 referencias
    public partial class ErrorTxtBox : TextBox
    {
        0 referencias
        public ErrorTxtBox()
        {
            InitializeComponent();
        }
    }
}
```

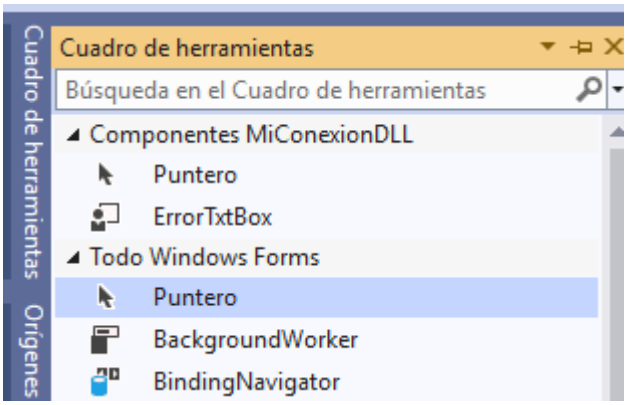
- Adicionamos en ErrorTxtBox el método Validar

```
public Boolean Validar
{
   set;
   get;
}
```

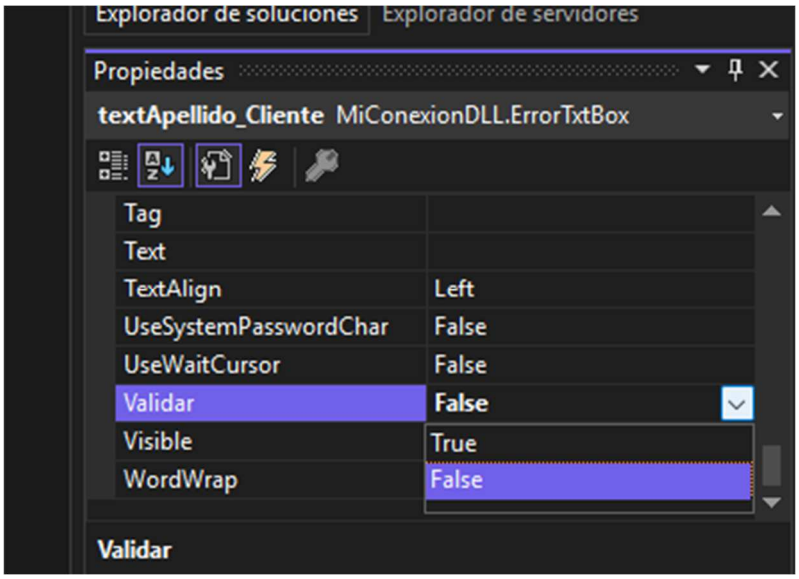
- Volvemos al diseño y Corregimos error comentando la línea así:

```
        private void InitializeComponent()
        {
            components = new System.ComponentModel.Container();
            //this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        }

        #endregion
    }
}
```

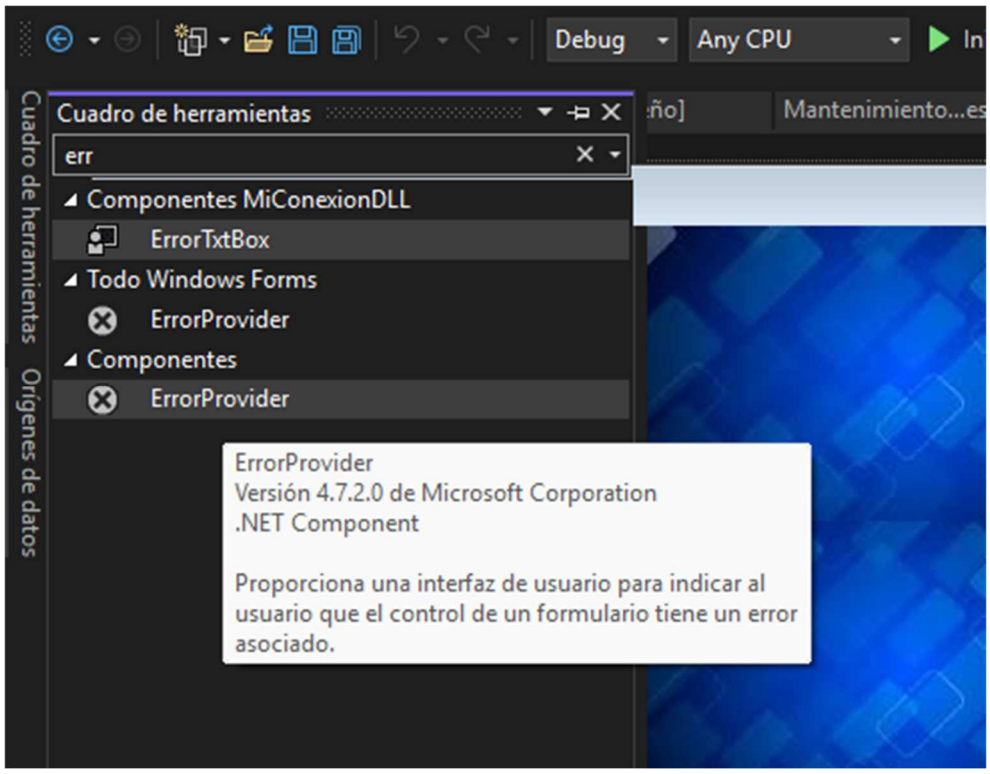
- Compilamos la librería dll y verificamos en el cuadro de controles que aparece un nuevo control.

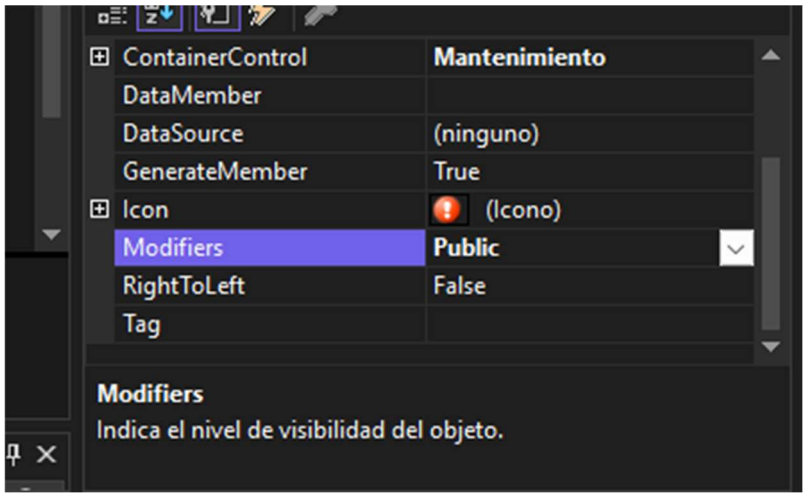


- Reemplazamos los TextBox por el TextBox especial creado.
- Modificamos la propiedad Validar de los textBox a True



- En la ventana Mantenimiento adicionamos ErrorProvider y modificamos la propiedad Modifier a Public.





- Modificamos el archivo ConexionSQL.cs

```
using System.Windows.Forms;
.
.
.
.
public static Boolean ValidarFormulario(Control ObjetoError, ErrorProvider ErrorProvider)
{
   Boolean SiError = false;
   foreach (Control campo in ObjetoError.Controls)
```

```
    {
        if (campo is ErrorTxtBox)
        {
            ErrorTxtBox objeto = (ErrorTxtBox)campo;
            if (objeto.Validar == true)
            {
                if (string.IsNullOrEmpty(objeto.Text.Trim()))
                {
                    ErrorProvider.SetError(objeto, "Los campos no pueden estar vacios");
                    SiError = true;
                }
            }
            else
            {
                ErrorProvider.SetError(objeto, "");
            }
        }
    }
    return SiError;
}
```

- Modificamos MantenimientoClientes.cs en el método Guardar()

```
public override Boolean Guardar()
{
    if(ConexionSQL.ValidarFormulario(this, errorProvider1)== false)
    {
        try
        {
            ConexionSQL conexion = new ConexionSQL();
            using (SqlConnection conn = conexion.ObtenerConexion())
            {
                conn.Open();
                MessageBox.Show("Conexión exitosa a SQL Server 2022 Verificada");

                // Ejemplo de consulta
                string insertar = string.Format("EXEC ActualizarClientes '{0}','{1}','{2}'", textId_Cliente.Text.Trim(),
textNombre_Cliente.Text.Trim(), textApellido_Cliente.Text.Trim());
                SqlCommand cmd = new SqlCommand(insertar, conn);
                SqlDataAdapter da = new SqlDataAdapter(cmd);
                DataSet conectar = new DataSet();
                da.Fill(conectar);
                MessageBox.Show("Cliente guardado correctamente");
                return true;
            }

        }
        catch (Exception error)
        {
            MessageBox.Show("Ha ocurrido un error: " + error);
            return false;
```

```
    }
  }
  else
  {
    return false;
  }


}
```

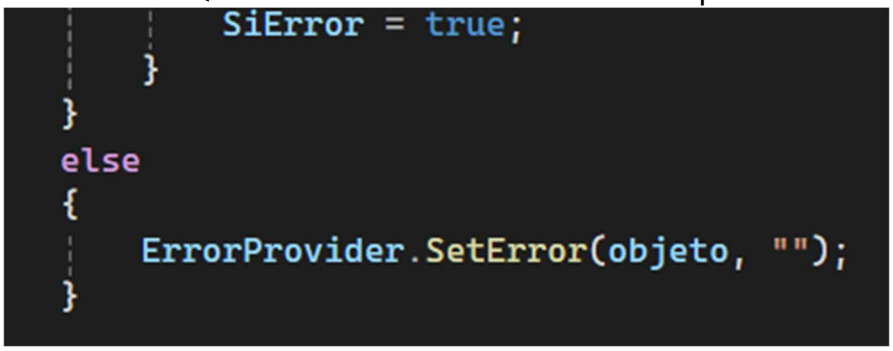
- En la ventana MantenimientoCliente le damos dobleclick a los textBox e incluimos el siguiente código:

```
errorProvider1.Clear();
```

- Modificar ErrorTxtBox.cs y agregar el método ValidarNumeros

```
public Boolean ValidarNumeros
{
  set;
  get;
}
```

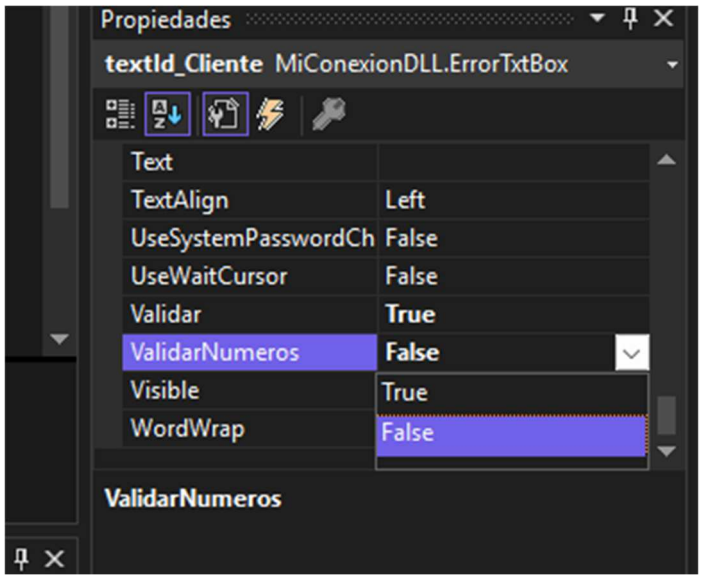
- En ConexionSQL.cs modificamos ValidarFormulario quitamos el else:




```
if (objeto.ValidarNumeros == true)
{
  int contador = 0, EncontrarLetras = 0;
  foreach (char letra in objeto.Text.Trim())
  {
    if (char.IsLetter(objeto.Text.Trim(), contador){
      EncontrarLetras++;
    }
    contador++;
  }
  if (EncontrarLetras != 0)
  {
    SiError = true;
    ErrorProvider.SetError(objeto, "Solo se aceptan numeros");
  }
```

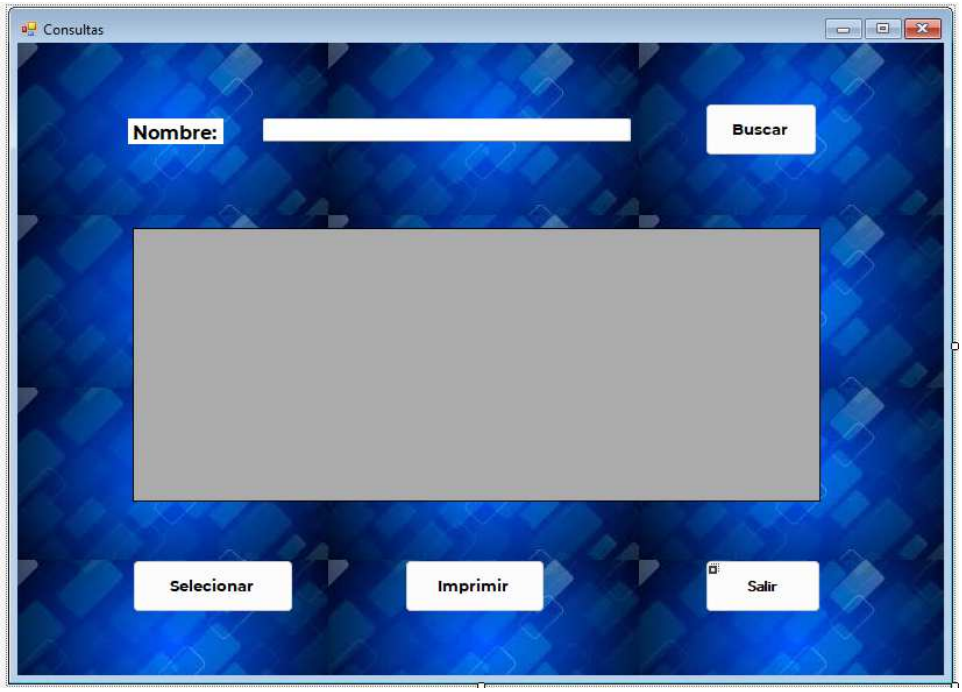
}

- Modificamos la propiedad ValidarNumeros de los textBox a True



Consultas Dinámicas

- Crear Formulario Consultas que hereda de FormBase.



- Modificar el código del formulario Consultas, crear método MostrarInfoDG.

```
using MiConexionDLL;
.
.
.
public DataSet MostrarInfoDG(string tabla)
{
   DataSet DS = new DataSet();
   ConexionSQL conexion = new ConexionSQL();
   using (SqlConnection conn = conexion.ObtenerConexion())
   {
      conn.Open();
      MessageBox.Show("Conexión exitosa a SQL Server 2022 Verificada");
      // Ejemplo de consulta
      string cmd1 = string.Format("Select * FROM " + tabla);
      SqlCommand cmd = new SqlCommand(cmd1, conn);
      SqlDataAdapter da = new SqlDataAdapter(cmd);
      da.Fill(DS);
   }
   return DS;
}
```

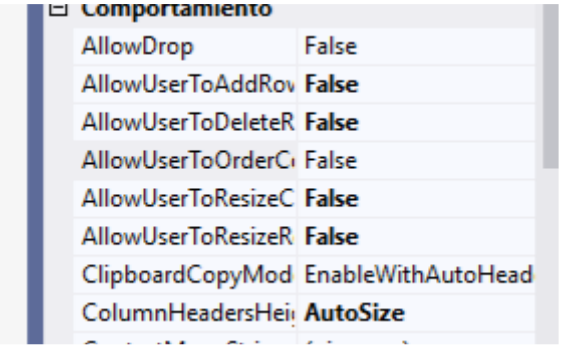
- Programamos el botón Seleccionar

```
private void button1_Click(object sender, EventArgs e)
 {
    if(dataGridView1.Rows.Count == 0)
    {
       return;
    }
    else
    {
       DialogResult = DialogResult.OK;
       Close();
    }
 }
```

- Creamos Formulario ConsultarCliente que herede de Consultas

- Creamos Formulario ConsultarProductos que herede de Consultas

- Compilamos para que herede todos los elementos de Consultas
- En el formulario Consultas cambiamos las propiedades del dataGridView de Modifiers a public, AllowUserToAddRows a false, AllowUserToDeleteRows a false, AllowUserToOrderColumns a false, AllowUserToResizeColumns a false, AllowUserToResizeRows a false, AutoSizeColumnsMode a Fill



- En el formulario ConsultarProductos programamos el método Load

```
private void ConsultarProductos_Load(object sender, EventArgs e)
 {
    dataGridView1.DataSource = MostrarInfoDG("Articulos").Tables[0];
 }
```

- En el formulario ConsultarClientes programamos el método Load

```
private void ConsultarCliente_Load(object sender, EventArgs e)
{
   dataGridView1.DataSource = MostrarInfoDG("Clientes").Tables[0];
}
```

- En el formulario ContenedorPrincipal programamos los botones Consultas/Productos y Consultas/Clientes

```
private void productosToolStripMenuItem1_Click(object sender, EventArgs e)
{
   ConsultarProductos ConsulPro = new ConsultarProductos();
   ConsulPro.MdiParent = this;
   ConsulPro.Show();
}

private void clientesToolStripMenuItem1_Click(object sender, EventArgs e)
{
   ConsultarCliente ConsulClien = new ConsultarCliente();
   ConsulClien.MdiParent = this;
   ConsulClien.Show();
}
```

- En el formulario Consultas modificamos la propiedad Modifiers a public en todos los botones y en el textBox.

- En el formulario ConsultarProductos programamos el botón Buscar

```
private void button3_Click(object sender, EventArgs e)
{
```

```
   if (string.IsNullOrEmpty(textBox1.Text.Trim()) == false)
   {
      try
      {
         DataSet DS = new DataSet();
         ConexionSQL conexion = new ConexionSQL();
         using (SqlConnection conn = conexion.ObtenerConexion())
         {
            conn.Open();
            string buscar = "Select * from Articulos WHERE Nombre_Producto LIKE ('%" + textBox1.Text.Trim() +
"%')";
            SqlCommand cmd = new SqlCommand(buscar, conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            da.Fill(DS);
            dataGridView1.DataSource = DS.Tables[0];
         }
      }
      catch (Exception error)
      {
         MessageBox.Show("No se puede conectar, Error: ", error.Message);
      }
   }
}
```

- En el formulario ConsultarCliente programamos el botón Buscar

```
private void button3_Click(object sender, EventArgs e)
{
   if (string.IsNullOrEmpty(textBox1.Text.Trim()) == false)
   {
      try
      {
         DataSet DS = new DataSet();
         ConexionSQL conexion = new ConexionSQL();
         using (SqlConnection conn = conexion.ObtenerConexion())
         {
            conn.Open();
            string buscar = "Select * from Clientes WHERE Nombre_cliente LIKE ('%" + textBox1.Text.Trim() +
"%')";
            SqlCommand cmd = new SqlCommand(buscar, conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            da.Fill(DS);
            dataGridView1.DataSource = DS.Tables[0];
         }
      }
      catch (Exception error)
      {
         MessageBox.Show("No se puede conectar, Error: ", error.Message);
      }
   }
}
```

- En el formulario MantenimientoProductos programamos el textBox de ID Producto. Y modificamos la propiedad Validar en true y la propiedad de ValidarNumero en true

```
private void textId_Producto_TextChanged(object sender, EventArgs e)
{
   errorProvider1.Clear();
}
```

- En el formulario MantenimientoProductos modificamos el método Guardar, agregando todo el código dentro de un if

```
if (ConexionSQL.ValidarFormulario(this, errorProvider1) == false)
 {
//Nota: Agregar el contenido aqui
 }
 else
 {
    return false;
 }
```

- En el formulario MantenimientoProductos programamos el botón Nuevo

```
private void button3_Click(object sender, EventArgs e)
{
   if (string.IsNullOrEmpty(textId_Producto.Text.Trim())==false &&
      string.IsNullOrEmpty(textDescripcion.Text.Trim())==false &&
      string.IsNullOrEmpty(textPrecio.Text.Trim()) == false)
   {
      textId_Producto.Text = "";
      textDescripcion.Text = "";
      textPrecio.Text = "";
   }
}
```

- En el formulario MantenimientoClientes programamos el botón Nuevo

```
private void button3_Click(object sender, EventArgs e)
{
   if (string.IsNullOrEmpty(textId_Cliente.Text.Trim())==false &&
      string.IsNullOrEmpty(textNombre_Cliente.Text.Trim())==false &&
      string.IsNullOrEmpty(textApellido_Cliente.Text.Trim()) == false){

      textId_Cliente.Text = "";
      textNombre_Cliente.Text = "";
      textApellido_Cliente.Text = "";
   }
}
```

- En el formulario MantenimientoProductos programamos el botón Consultar

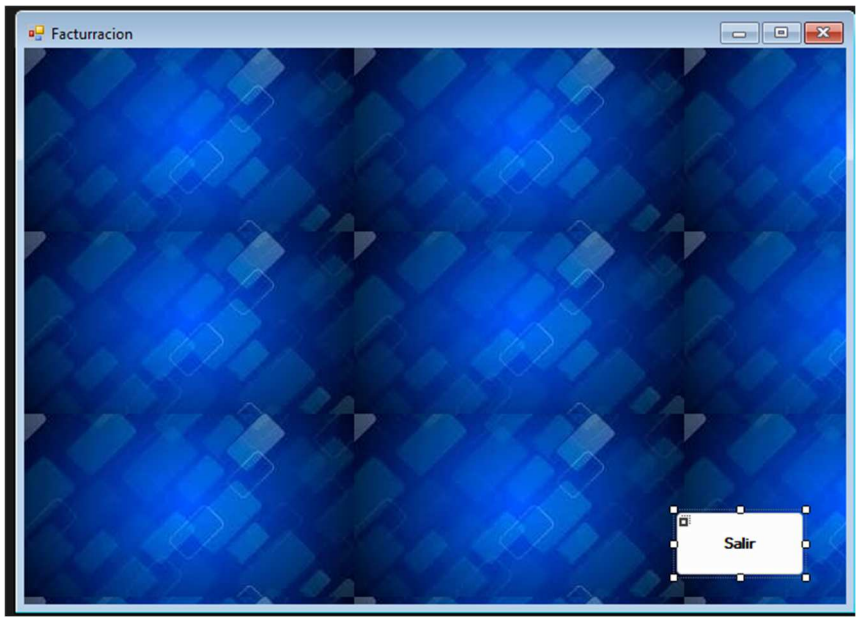
```
private void button1_Click(object sender, EventArgs e)
{
   ConsultarProductos ConsulPro = new ConsultarProductos();
   ConsulPro.Show();
}
```

- En el formulario MantenimientoClientes programamos el botón Consultar

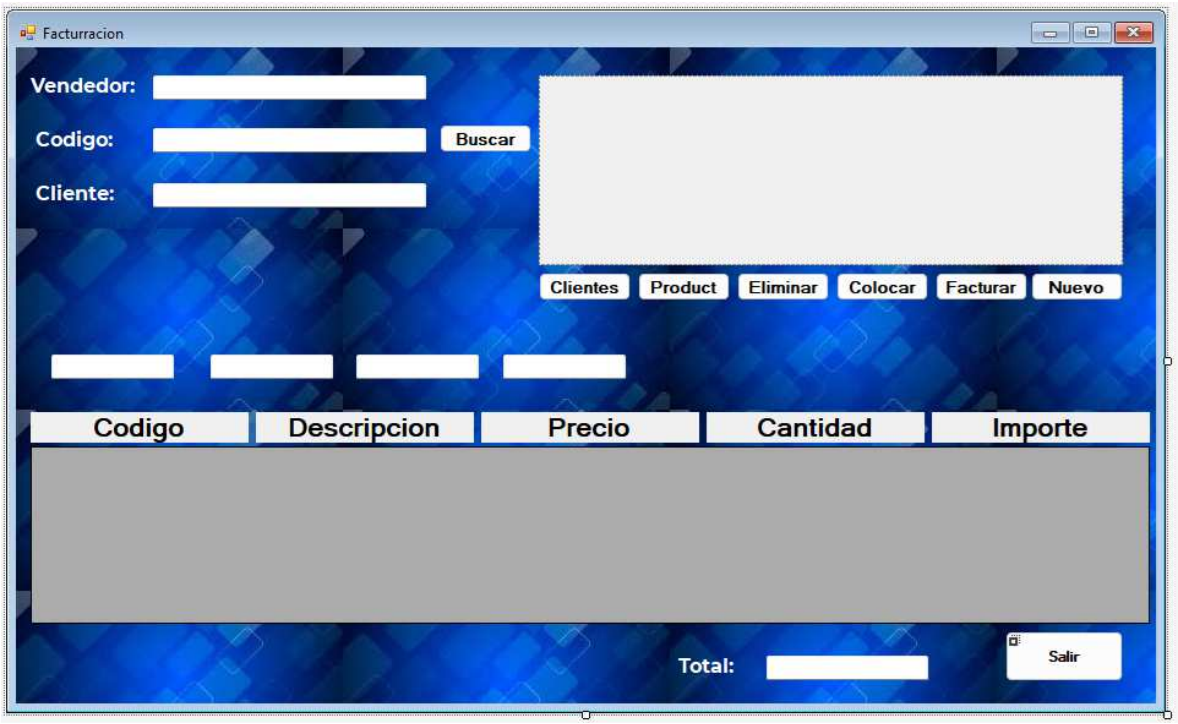
```
private void button1_Click(object sender, EventArgs e)
{
   ConsultarCliente ConsulClien = new ConsultarCliente();
   ConsulClien.Show();
}
```

Módulo de Facturación

- Creamos un formulario Facturacion que herede de Procesos



- Modificamos el diseño del formulario Facturacion



- En el Formulario Facturacion cambiamos el textBox del Vendedor por un Label llamado lbVendedor y cambiamos los nombres internos. De los demás textBox: txtCodigoCliente, txtCliente, txtCodigoProducto, txtDescripcion, txtPrecio, txtCantidad, lbTotal.

- En el Formulario Facturacion cambiamos los nombres internos de los botones: btClientes, btProductos, btEliminar, btColocar, btFacturar, btNuevo, btBuscar
- Llamar el Formulario Facturacion desde el ContenedorPrincipal.

```
private void facturaciónToolStripMenuItem_Click(object sender, EventArgs e)
{
   Facturacion Factu = new Facturacion();
   Factu.MdiParent = this;
   Factu.Show();
}
```

- En el Formulario Facturacion programamos el método Load

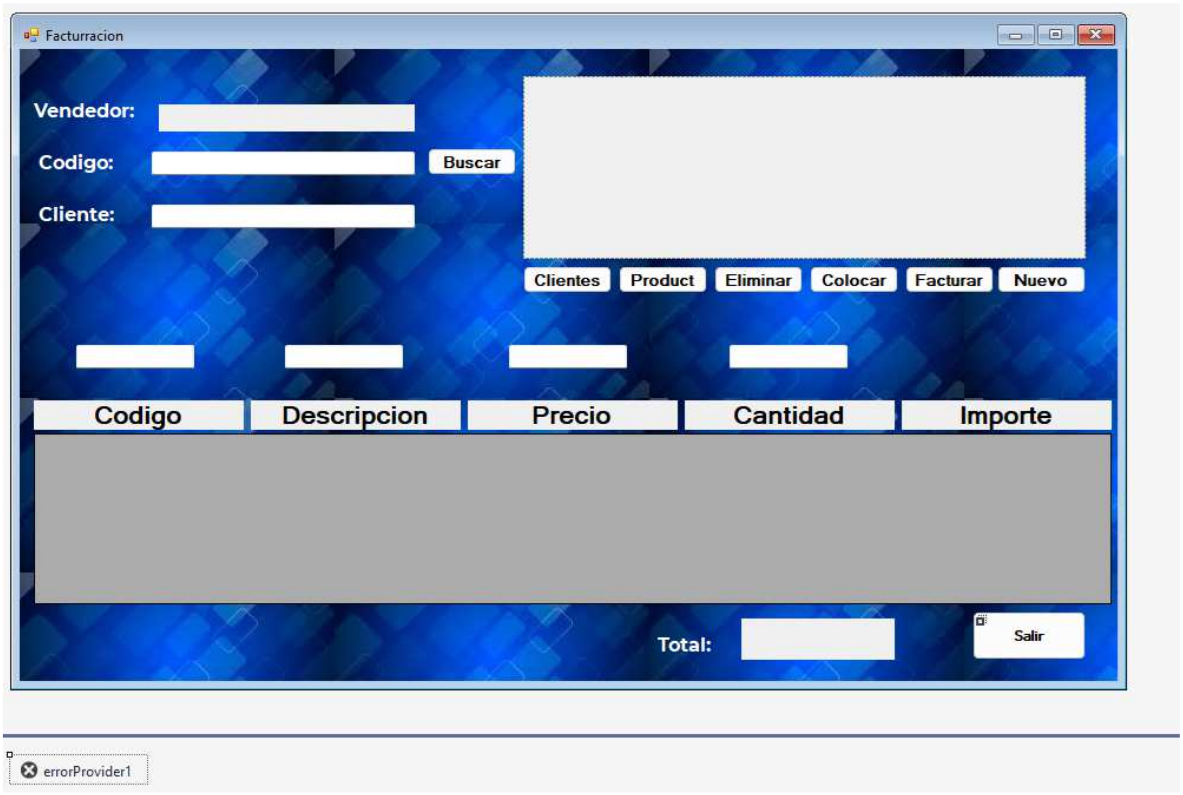
```
private void Facturacion_Load(object sender, EventArgs e)
{
   DataSet DS = new DataSet();
   ConexionSQL conexion = new ConexionSQL();
   using (SqlConnection conn = conexion.ObtenerConexion())
   {
      conn.Open();
      string vendedor = "Select * From Usuarios Where id_usuario=" + Login.Codigo;
      SqlCommand cmd = new SqlCommand(vendedor, conn);
```

```
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        da.Fill(DS);
        lbVendedor.Text = DS.Tables[0].Rows[0]["username"].ToString().Trim();
    }

}
```

- En el Formulario ContenedorPrincipal programamos el botón Procesos/Facturacion

```
private void facturaciónToolStripMenuItem_Click(object sender, EventArgs e)
{
    Facturacion Factu = new Facturacion();
    Factu.MdiParent = this;
    Factu.Show();
}
```

- En el formulario Facturacion le cambiamos la propiedad Validar a True a todos los texBox y agregamos el errorProvider al formulario.



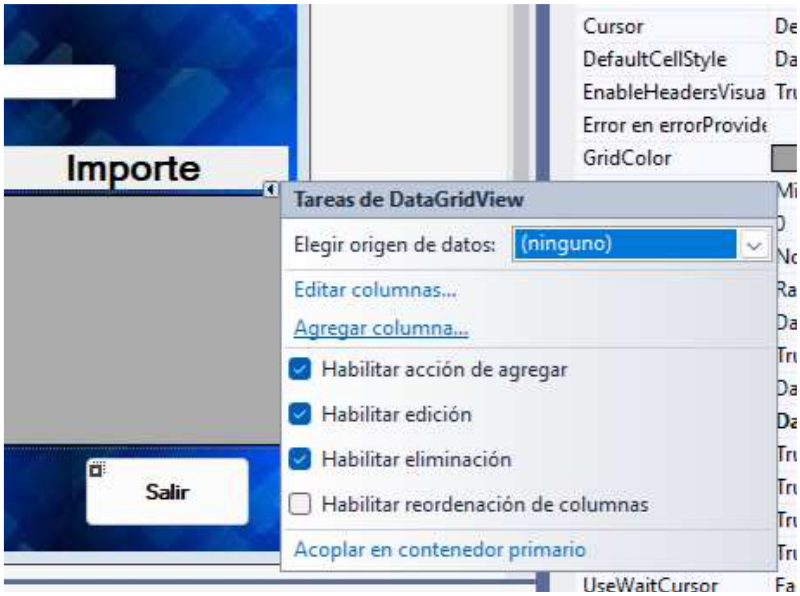
- En el formulario Facturacion programamos el botón Buscar

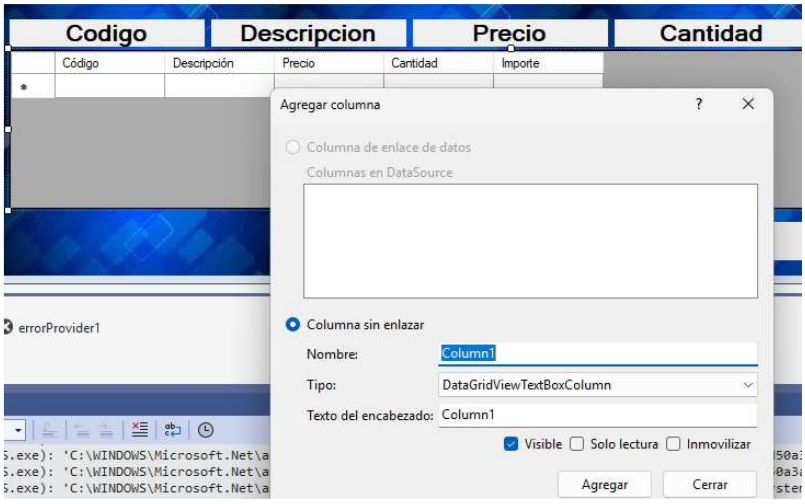
```
private void btBuscar_Click(object sender, EventArgs e)
{
    try
    {
        if (string.IsNullOrEmpty(txtCodigoCliente.Text.Trim()) == false)
```

```
        {
            DataSet DS = new DataSet();
            ConexionSQL conexion = new ConexionSQL();
            using (SqlConnection conn = conexion.ObtenerConexion())
            {
                conn.Open();
                string cmd1 = string.Format("Select Nombre_cliente from Clientes Where id_clientes = '{0}'",
txtCodigoCliente.Text.Trim());
                SqlCommand cmd = new SqlCommand(cmd1, conn);
                SqlDataAdapter da = new SqlDataAdapter(cmd);
                da.Fill(DS);
                txtCliente.Text = DS.Tables[0].Rows[0]["Nombre_cliente"].ToString().Trim();

                txtCodigoProducto.Focus();
            }
        }
    }
    catch (Exception error)
    {
        MessageBox.Show("Ha ocurrido un erroro: " + error.Message);
    }

}
```

- Modificamos el diseño del formulario Facturacion

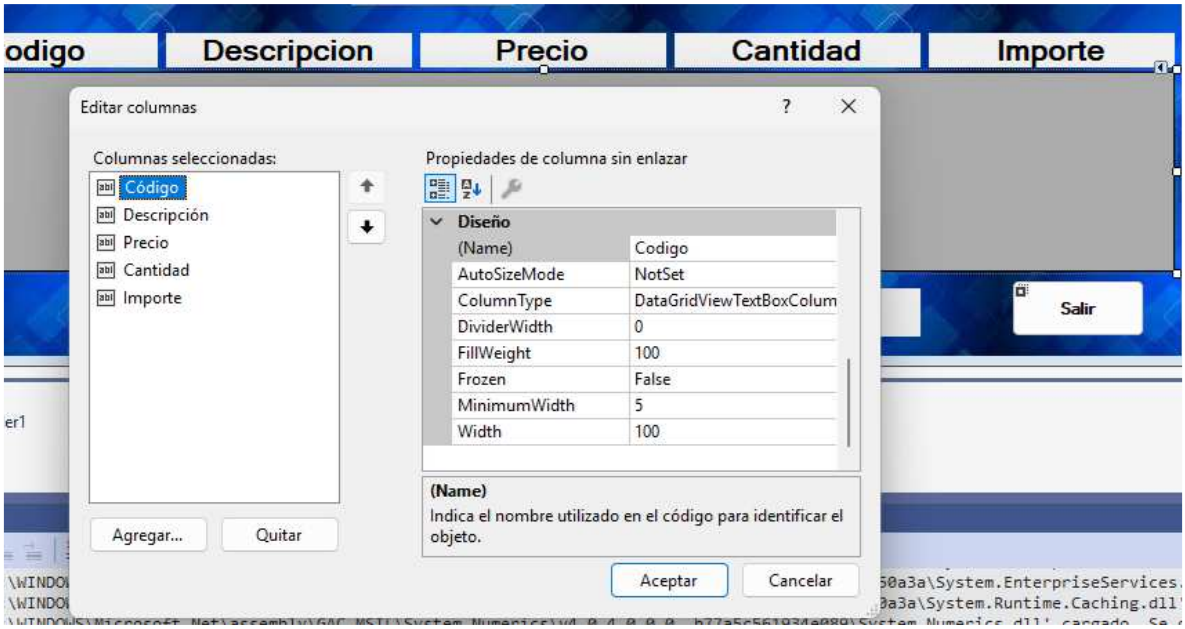
- En el DataGridView del Formulario Facturacion agregamos columnas:





- En el DataGridView del Formulario Facturacion modificamos las propiedades: RowHeadersVisible a False, ColumnHeadersVisible a False y AllowUserToAddRows a False

- En el DataGridView del Formulario Facturacion editamos las columnas en la propiedad Width le colocamos un tamaño acorde al tamaño de las columnas



- En el formulario Facturacion programamos el botón Colocar

```
private void btColocar_Click(object sender, EventArgs e)
{
   if (ConexionSQL.ValidarFormulario(this, errorProvider1) == false)
   {
      bool existe = false;
      int numeroFila = 0;

      if (contadorFila == 0)
      {
         dataGridView1.Rows.Add(txtCodigoProducto.Text, txtDescripcion.Text, txtPrecio.Text, txtCantidad.Text);
         double importe = Convert.ToDouble(dataGridView1.Rows[contadorFila].Cells[2].Value) *
Convert.ToDouble(dataGridView1.Rows[contadorFila].Cells[3].Value);
         dataGridView1.Rows[contadorFila].Cells[4].Value = importe;

         contadorFila++;
      }
      else
      {
         foreach (DataGridViewRow Fila in dataGridView1.Rows)
         {
            if (Fila.Cells[0].Value.ToString() == txtCodigoProducto.Text)
            {

               existe = true;
               numeroFila = Fila.Index;

            }
         }
         if (existe == true)
         {
```

```
            dataGridView1.Rows[numeroFila].Cells[3].Value = (Convert.ToDouble(txtCantidad.Text) +
Convert.ToDouble(dataGridView1.Rows[numeroFila].Cells[3].Value)).ToString();
            double importe = Convert.ToDouble(dataGridView1.Rows[numeroFila].Cells[2].Value) *
Convert.ToDouble(dataGridView1.Rows[numeroFila].Cells[3].Value);
            dataGridView1.Rows[numeroFila].Cells[4].Value = importe;

          }
          else
          {
            dataGridView1.Rows.Add(txtCodigoProducto.Text, txtDescripcion.Text, txtPrecio.Text,
txtCantidad.Text);
            double importe = Convert.ToDouble(dataGridView1.Rows[contadorFila].Cells[2].Value) *
Convert.ToDouble(dataGridView1.Rows[contadorFila].Cells[3].Value);
            dataGridView1.Rows[contadorFila].Cells[4].Value = importe;

            contadorFila++;
          }
        }
    }
    total = 0;
    foreach (DataGridViewRow Fila in dataGridView1.Rows)
    {
       total += Convert.ToDouble(Fila.Cells[4].Value);
    }
    lbTotal.Text = "USD$ " + total.ToString();
}
```

- En el formulario Facturacion programamos el botón Eliminar

```
private void btEliminar_Click(object sender, EventArgs e)
 {
    if(contadorFila > 0)
    {
       total = total- (Convert.ToDouble(dataGridView1.Rows[dataGridView1.CurrentRow.Index]));
       lbTotal.Text = "USD$ " + total.ToString();

       dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index);
       contadorFila--;
    }
 }
```

- Realizamos cambios en el formulario Consultas, en las propiedades del botón Seleccionar le cambiamos el nombre a btnSeleccionar.

- En el formulario Facturacion programamos el botón Clientes

```
private void btClientes_Click(object sender, EventArgs e)
{
   ConsultarCliente ConsulClien = new ConsultarCliente();
   ConsulClien.ShowDialog();
```

```
   if(ConsulClien.DialogResult == DialogResult.OK)
   {
      txtCodigoCliente.Text =
ConsulClien.dataGridView1.Rows[ConsulClien.dataGridView1.CurrentRow.Index].Cells[0].Value.ToString();
      txtCliente.Text =
ConsulClien.dataGridView1.Rows[ConsulClien.dataGridView1.CurrentRow.Index].Cells[1].Value.ToString();

      txtCodigoProducto.Focus();
   }
}
```

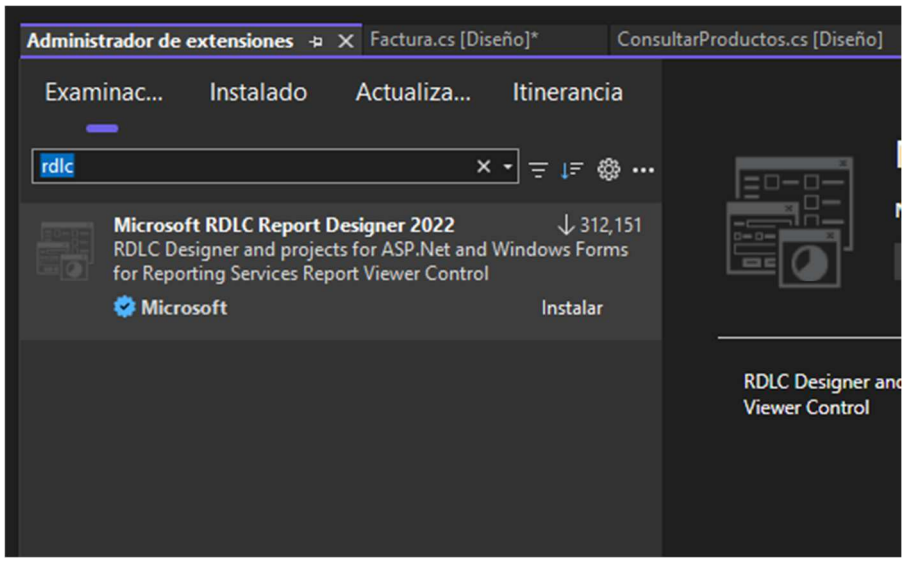
- En el formulario Facturacion programamos el botón Productos

```
private void btProductos_Click(object sender, EventArgs e)
{
   ConsultarProductos ConsulPro = new ConsultarProductos();
   ConsulPro.ShowDialog();

   if(ConsulPro.DialogResult == DialogResult.OK)
   {
      txtCodigoProducto.Text =
ConsulPro.dataGridView1.Rows[ConsulPro.dataGridView1.CurrentRow.Index].Cells[0].Value.ToString();
      txtDescripcion.Text =
ConsulPro.dataGridView1.Rows[ConsulPro.dataGridView1.CurrentRow.Index].Cells[1].Value.ToString();
      txtPrecio.Text =
ConsulPro.dataGridView1.Rows[ConsulPro.dataGridView1.CurrentRow.Index].Cells[2].Value.ToString();
      txtCantidad.Focus();
   }
}
```

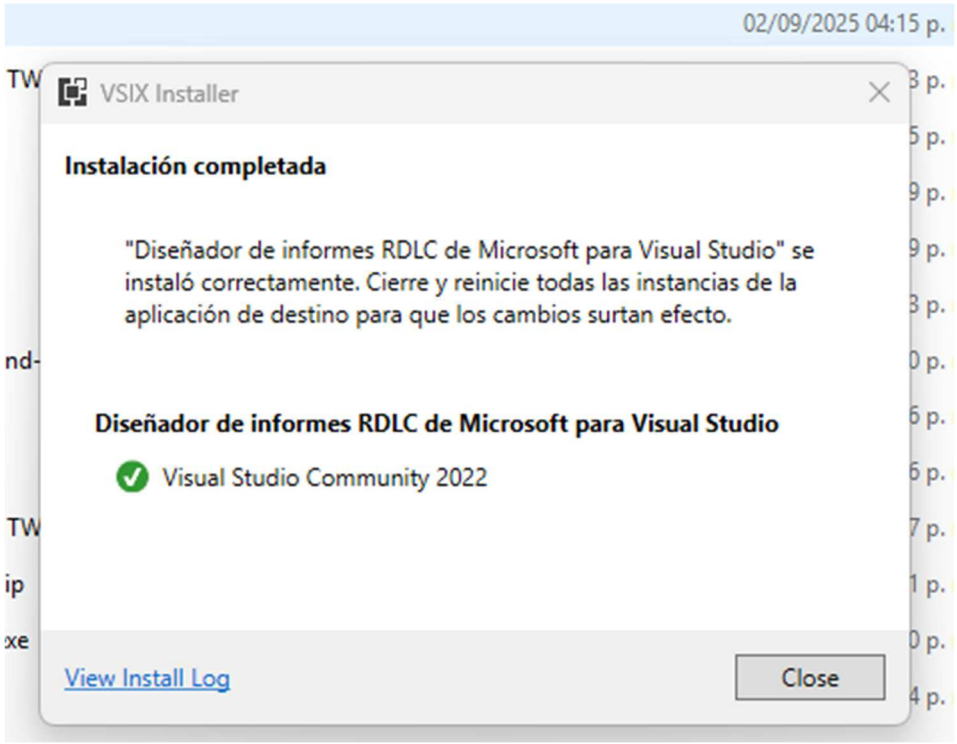
- En el formulario Facturacion programamos el botón Nuevo y creamos un método Nuevo.

```
private void btNuevo_Click(object sender, EventArgs e)
{
   Nuevo();
}

public override void Nuevo()
{
   txtCodigoCliente.Text = "";
   txtCliente.Text = "";
   txtCodigoProducto.Text = "";
   txtDescripcion.Text = "";
   txtPrecio.Text = "";
   txtCantidad.Text = "";
   lbTotal.Text = "USD$ 0";
   dataGridView1.Rows.Clear();
   contadorFila = 0;
   total = 0;
   txtCodigoCliente.Focus();
}
```

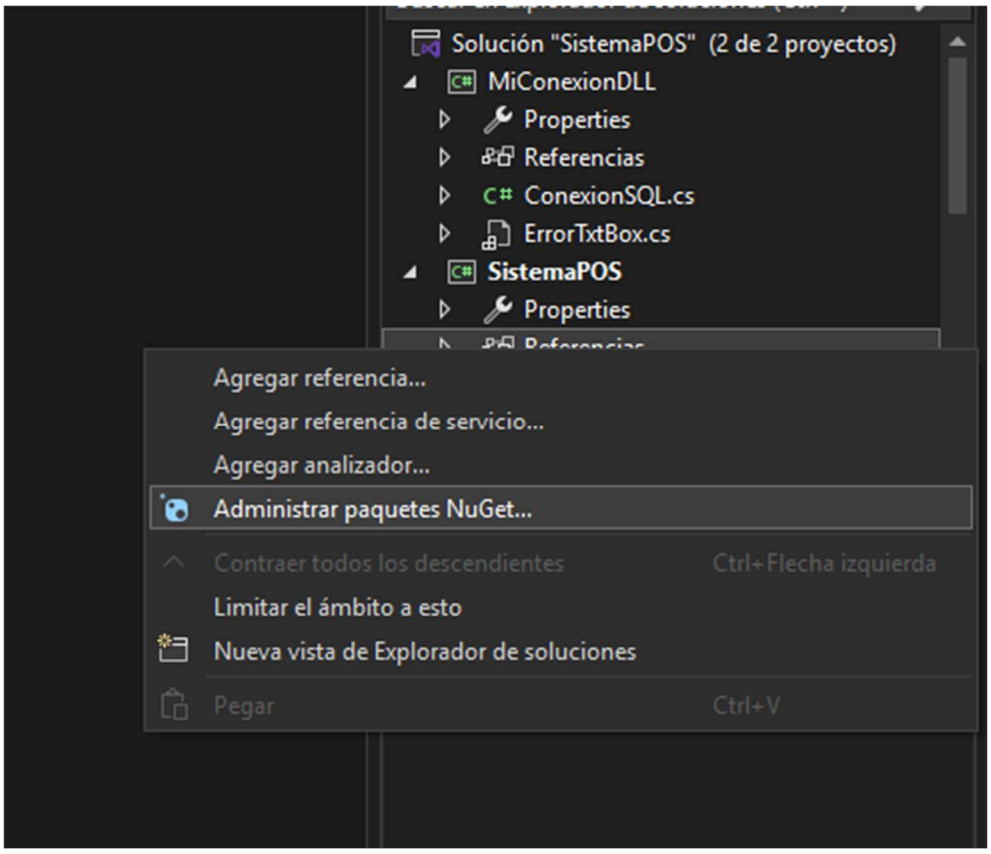
- Agregamos formulario nuevo llamado Factura para el reporte
- Agregamos herramienta reportViewer al formulario Factura.
    - Extensiones/Administrador de Extensiones
    - Escribimos en buscar rdlc para instalarlo.



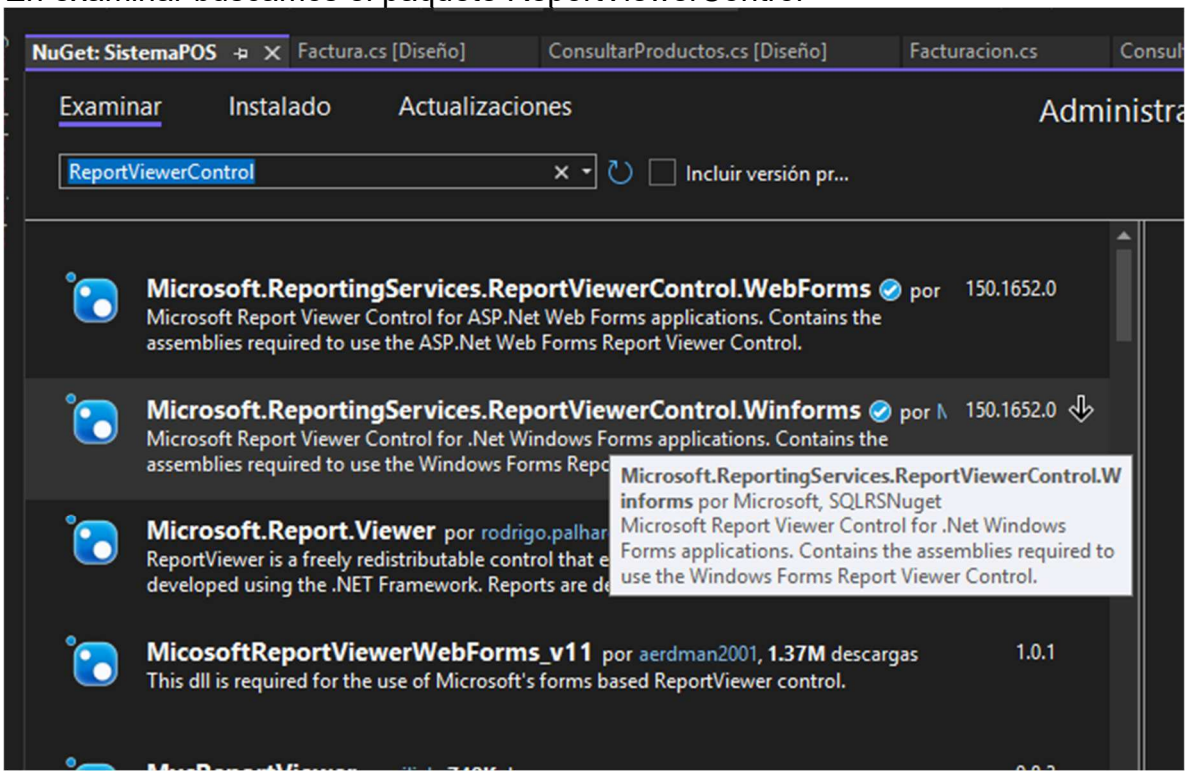
Instalar el paquete descargado.



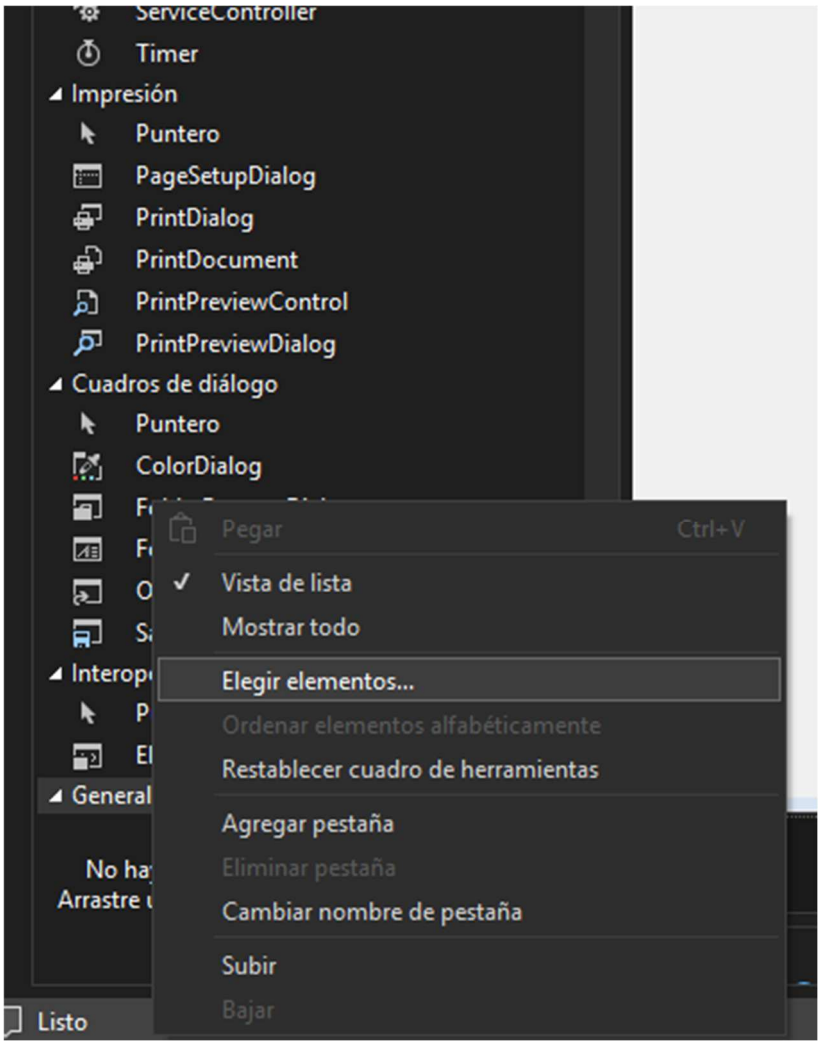
- En referencias



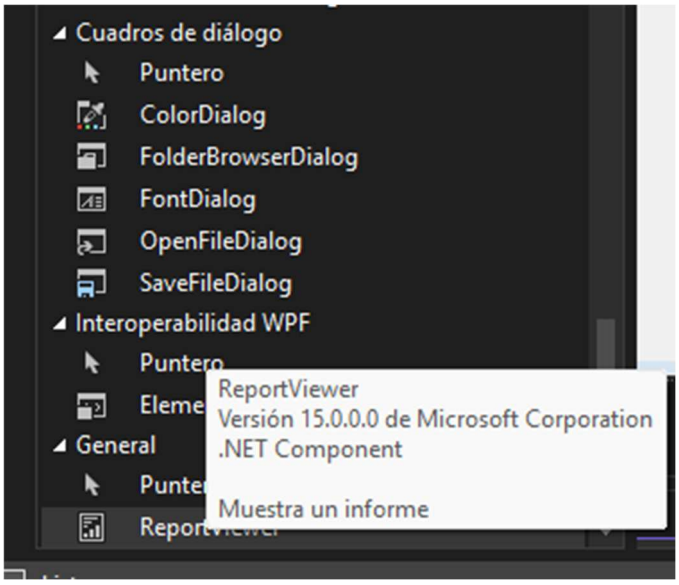
- En examinar buscamos el paquete ReportViewerControl



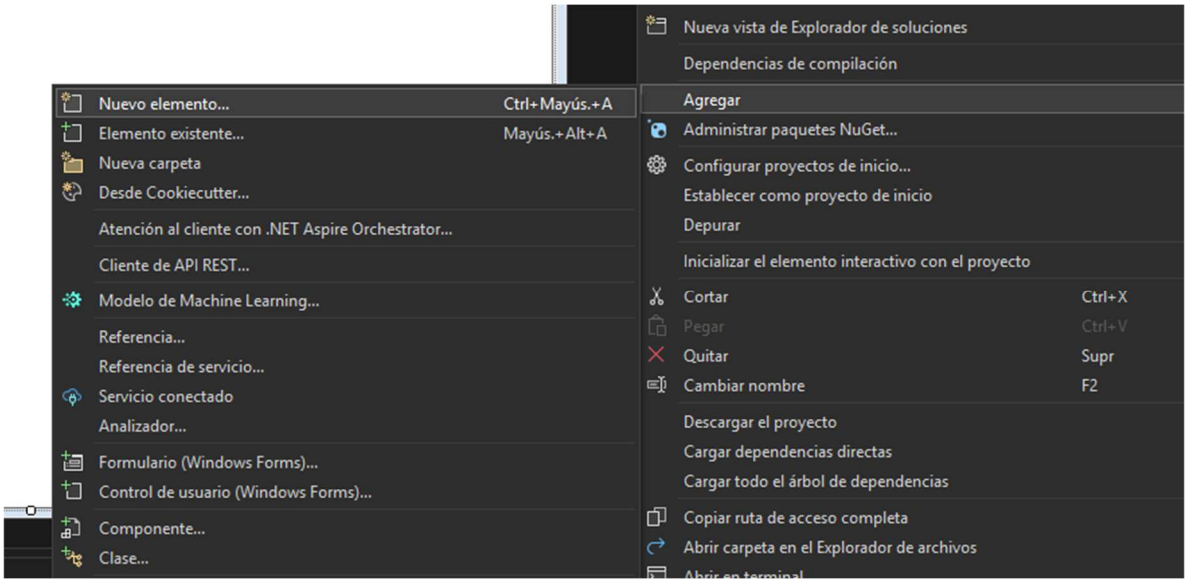
- Y lo instalamos
- En el Cuadro de Herramientas en General seleccionamos Elegir Elementos

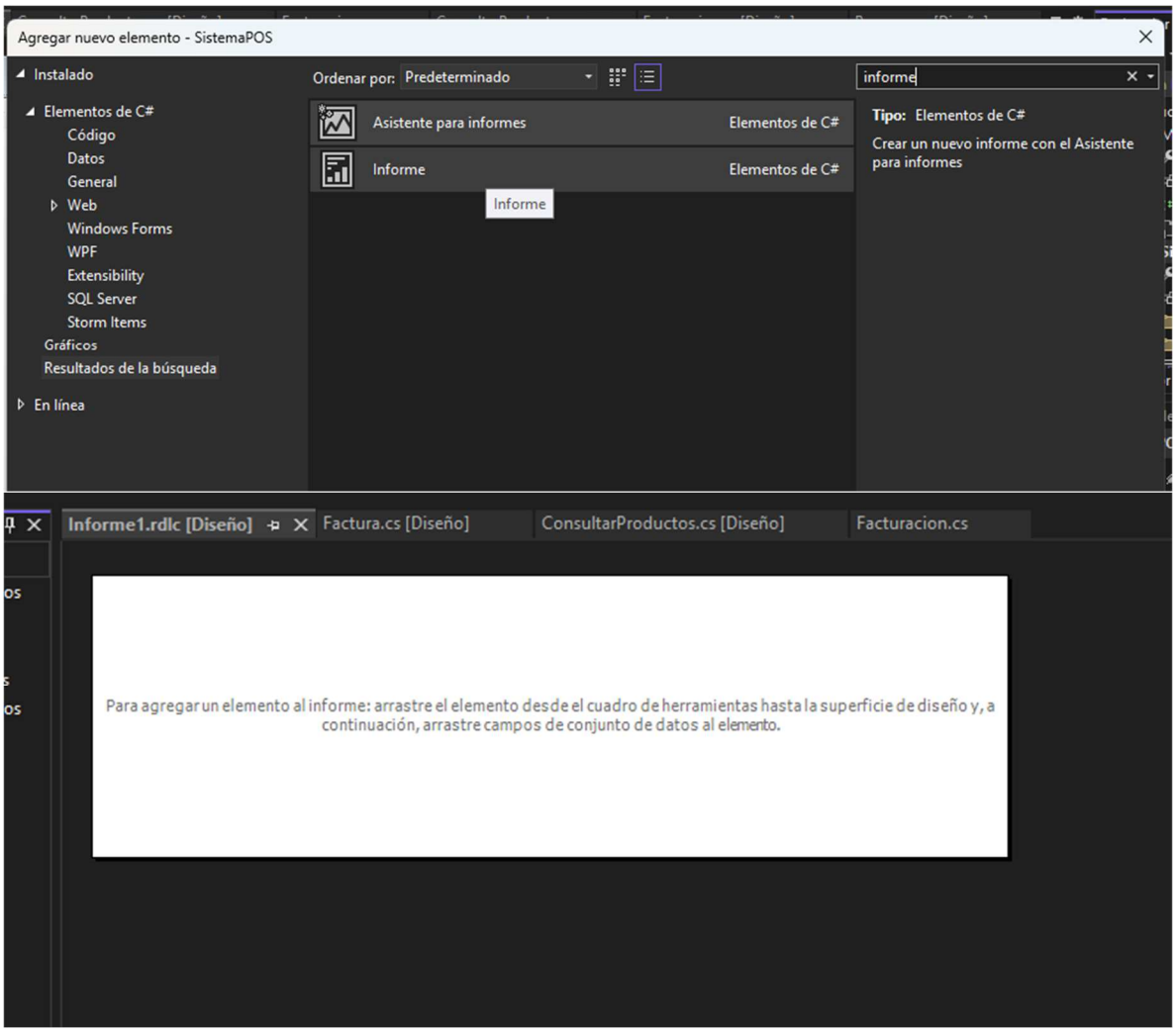


- En examinar, buscamos el proyecto SistemaPOS/packages/Microsoft.Repor…./lib/net40/Microsoft.ReportViewer.Design.dll …Abrir y Aceptar



- Pegamos el elemento ReportViewer al formulario.
- Agregamos Elemento Nuevo.



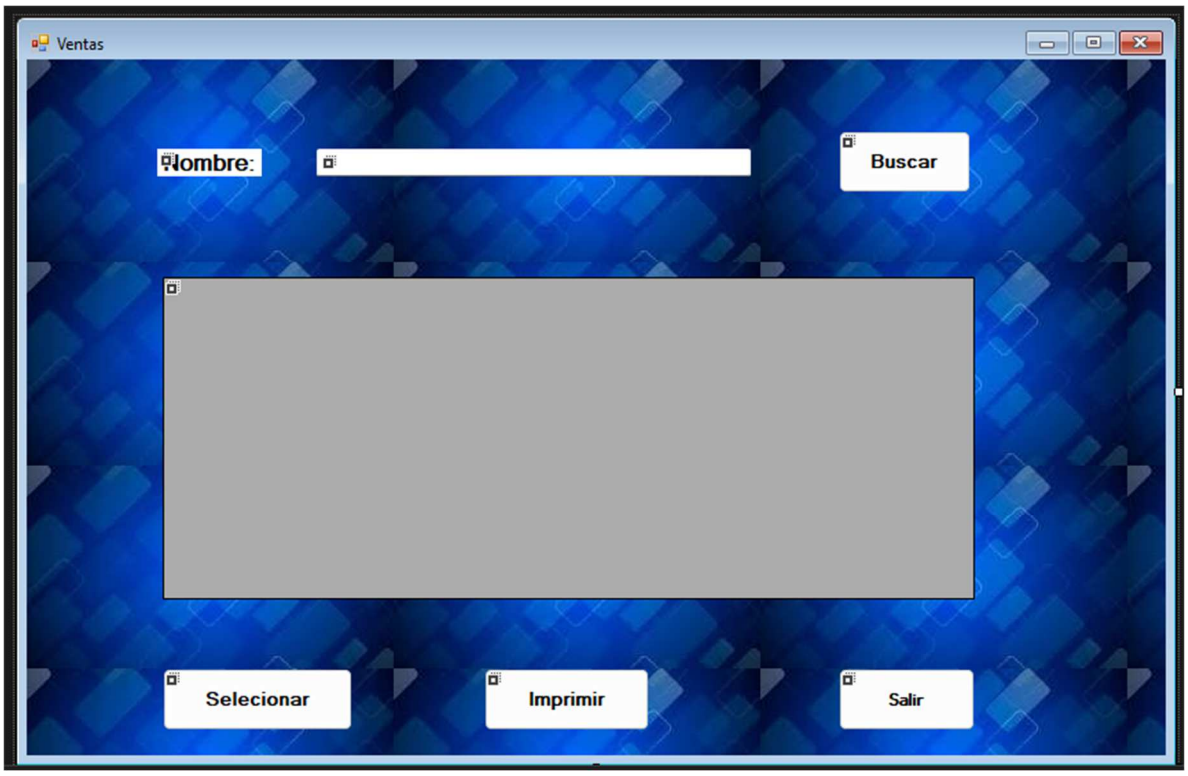


- Crear informe personalizado. Usando video proporcionado por el instructor.
- Programamos el botón Facturar

```
private void btFacturar_Click(object sender, EventArgs e)
{
   if (contadorFila != 0)
   {
      try
      {
         DataSet DS = new DataSet();
         ConexionSQL conexion = new ConexionSQL();
         using (SqlConnection conn = conexion.ObtenerConexion())
         {
            conn.Open();
            string cmd1 = string.Format("Exec ActualizarFaturas '{0}'", txtCodigoCliente.Text.Trim());
            SqlCommand cmd = new SqlCommand(cmd1, conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            da.Fill(DS);
```

```
            string NumeroFactura = DS.Tables[0].Rows[0]["NumeroFactura"].ToString().Trim();

            foreach (DataGridViewRow Fila in dataGridView1.Rows)
            {
               cmd1 = string.Format("Exec ActualizarDetalles '{0}','{1}','{2}','{3}'", NumeroFactura,
Fila.Cells[0].Value.ToString(), Fila.Cells[2].Value.ToString(), Fila.Cells[3].Value.ToString());
               SqlCommand cmd2 = new SqlCommand(cmd1, conn);
               SqlDataAdapter da1 = new SqlDataAdapter(cmd2);
               da1.Fill(DS);
            }
            cmd1 = "Exec DatosFactura " + NumeroFactura;
            SqlCommand cmd3 = new SqlCommand(cmd1, conn);
            SqlDataAdapter da2 = new SqlDataAdapter(cmd3);
            da2.Fill(DS);

            Factura fac = new Factura();
            fac.reportViewer1.LocalReport.DataSources[0].Value = DS.Tables[0];
            fac.ShowDialog();

            Nuevo();
         }
      }
      catch (Exception error)
      {
         MessageBox.Show("Error: " + error.Message);
      }
   }

}
```

- Crear formulación de Ventas que hereda de Consultas



- Adicionamos el método load en la ventana Ventas

```
private void Ventas_Load(object sender, EventArgs e)
{
   dataGridView1.DataSource = MostrarInfoDG("Facturas").Tables[0];
}
```

- En el ContenedorPrincipal programamos el botón Consultar/Ventas

```
private void ventasToolStripMenuItem_Click(object sender, EventArgs e)
{
   Ventas ventas = new Ventas();
   ventas.MdiParent = this;
   ventas.Show();
}
```

- Programamos el Botón Buscar del formulario Ventas

```
private void button3_Click(object sender, EventArgs e)
{
   if (string.IsNullOrEmpty(textBox1.Text.Trim()) == false)
   {
      try
      {
         DataSet DS = new DataSet();
         ConexionSQL conexion = new ConexionSQL();
         using (SqlConnection conn = conexion.ObtenerConexion())
```

```
        {
            conn.Open();
            string buscar = "Select * from Facturas WHERE Numerofactura LIKE ('%" + textBox1.Text.Trim() +
"%')";
            SqlCommand cmd = new SqlCommand(buscar, conn);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            da.Fill(DS);
            dataGridView1.DataSource = DS.Tables[0];
        }
      }
      catch (Exception error)
      {
        MessageBox.Show("No se puede conectar, Error: ", error.Message);
      }
   }
}
```