

Universidad Politécnica de Pénjamo (UPP)



Nombre del Alumno: Brandon Iván Márquez Morales.

Matrícula: 220030140.

Carrera: Ingeniería en Software.

Cuatrimestre: 2°.

Grupo: "A".

Materia: Programación Orientada a Objetos.

Tema: Reporte Proyecto Final Programación Orientada a Objetos.

Docente: Miguel Ángel Saldaña Cabeza.

20/04/2021

Contenido

Introducción.....	3
Marco teórico.....	4
¿Qué es la POO?	4
¿Qué es una clase en POO?	4
¿Qué es la herencia en POO?	4
¿Qué es un atributo en POO?	4
¿Qué es el encapsulamiento en POO?.....	5
¿Qué es un método en POO?	5
¿Qué es una sentencia en POO?	5
¿Qué es un evento en POO?	5
Desarrollo.....	6
Conclusión.....	34
Bibliografía	35

Introducción.

En el presente reporte se dará a conocer la administración de la universidad politécnica de Pénjamo (UPP).

Este proyecto se desarrolló en el lenguaje C# en el IDE de programación Visual Studio 2012, Además cuenta con su conexión a base de datos desarrollado con SQL Server Management Studio 2012.

Principalmente se dio a la tarea de desarrollar entender y comprender el diagrama UML que esto nos sirve para modelar correctamente cada una de las clases en código y tablas en base de datos con sus correspondientes atributos y tipos de datos que le corresponden a cada uno.

¿Qué es lo que se administra?

Según el UML desarrollado se administran las siguientes tablas o clases: Persona (Clase que se usa para heredar a: Profesor, Estudiante y Directivo), Profesor, Estudiante, Directivo, Materia, Grupo, Aula y Carrera.

Dicho esto, las funcionalidades de cada tabla o clase tienen las funciones hacer una selección, hacer una inserción, hacer una actualización y hacer una eliminación.

Cada consulta que se haga en el programa también se verá afectado en la base de datos.

Marco teórico.

¿Qué es la POO?

La Programación Orientada a Objetos (POO) es un paradigma de programación, es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos. (profile, 2021)

¿Qué es una clase en POO?

Dentro de la programación orientada a objetos, las clases son un pilar fundamental. Una clase es la descripción de un conjunto de objetos similares; consta de métodos y de datos que resumen las características comunes de dicho conjunto. (masTeclas, 2021)

¿Qué es la herencia en POO?

La herencia permite que se puedan definir nuevas clases basadas de unas ya existentes a fin de reutilizar el código, generando así una jerarquía de clases dentro de una aplicación. Si una clase deriva de otra, esta hereda sus atributos y métodos y puede añadir nuevos atributos, métodos o redefinir los heredados. (if(geek), 2021)

¿Qué es un atributo en POO?

Los atributos son las características individuales que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades. Los atributos se guardan en variables denominadas de instancia, y cada objeto particular puede tener valores distintos para estas variables. (conceptosBasicosdePOO, 2021)

¿Qué es el encapsulamiento en POO?

En Programación modular, y más específicamente en programación orientada a objetos, se denomina encapsulamiento al ocultamiento del estado, es decir, de los datos miembro, de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto. (anonymus, 2021)

¿Qué es un método en POO?

En la programación, un método es una subrutina cuyo código es definido en una clase y puede pertenecer tanto a una clase, como es el caso de los métodos de clase o estáticos, como a un objeto, como es el caso de los métodos de instancia. (Wikipedia, 2021)

¿Qué es una sentencia en POO?

Las sentencias son los elementos básicos en los que se divide el código en un lenguaje de programación. Al fin y al cabo, un programa no es más que un conjunto de sentencias que se ejecutan para realizar una cierta tarea. Además, como ya habrás visto, en Pascal el signo que las separa es el punto y coma. (?, 2021)

¿Qué es un evento en POO?

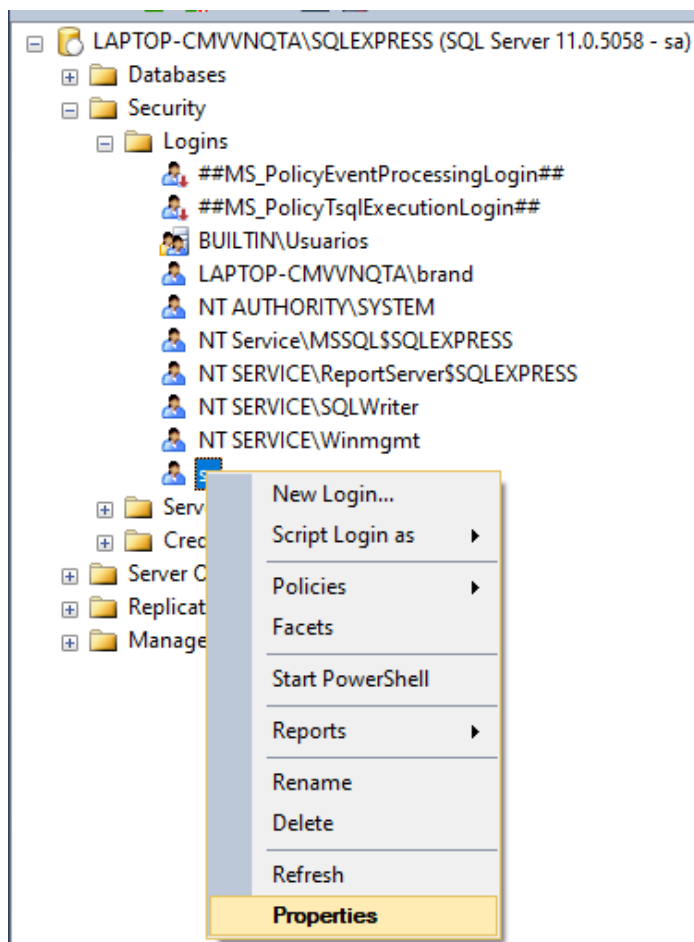
Los Eventos son las acciones sobre el programa, (explorativos, 2021) como, por ejemplo: Clic sobre un botón. Doble clic sobre el nombre de un fichero para abrirlo. ... Pulsar una tecla o una combinación de teclas. (explorativos, 2021)

Desarrollo.

Cabe recalcar que para poder utilizar este proyecto es necesario habilitar el usuario “sa” en SQL Server Management Studio, Esto se hizo más que nada por manejar y practicar un poco lo que es el área de la seguridad.

Entonces para ello se muestra a continuación un pequeño manual de cómo es que se habilita esta configuración:

1. Nos dirigimos a la siguiente dirección: Abrimos servidor local, Security, Logins, sa y damos clic derecho y seleccionamos “Properties”



2.- Aquí vamos a establecer el usuario por “sa” y contraseña por “sa” ya que es así como se tiene en el código de la conexión y deshabilitamos la opción de “Enforce password policy” y damos clic en “OK”

Login Properties - sa

Select a page

- General
- Server Roles
- User Mapping
- Status

Script Help

Login name: Search...

☐ Windows authentication

☒ SQL Server authentication

Password:

Confirm password:

☐ Specify old password

Old password:

☐ Enforce password policy

☐ Enforce password expiration

☐ User must change password at next login

☐ Mapped to certificate

☐ Mapped to asymmetric key

☐ Map to Credential

Add

Mapped Credentials

Credential	Provider
------------	----------

Remove

Default database:

Default language:

OK Cancel

Connection

Server: LAPTOP-CMVVNQTA\SQLEXP

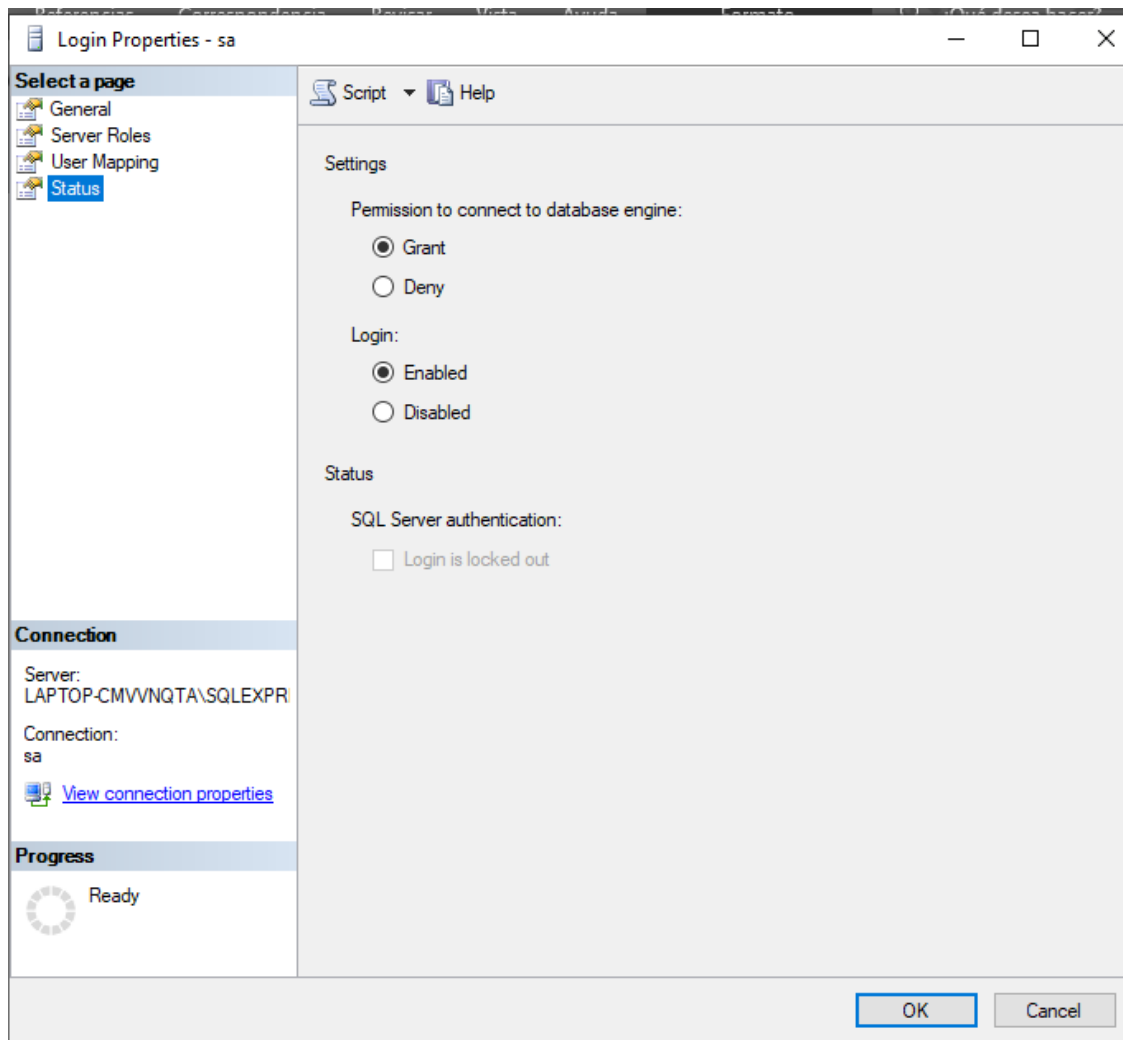
Connection: sa

[View connection properties](#)

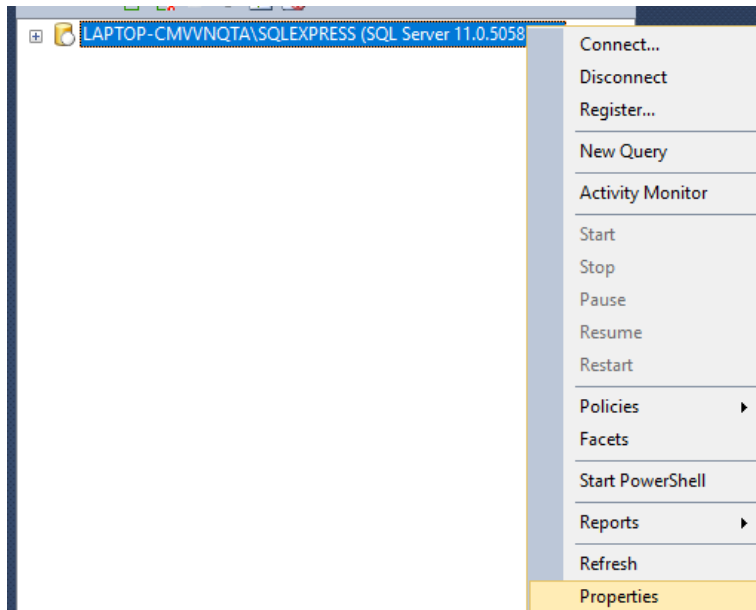
Progress

Ready

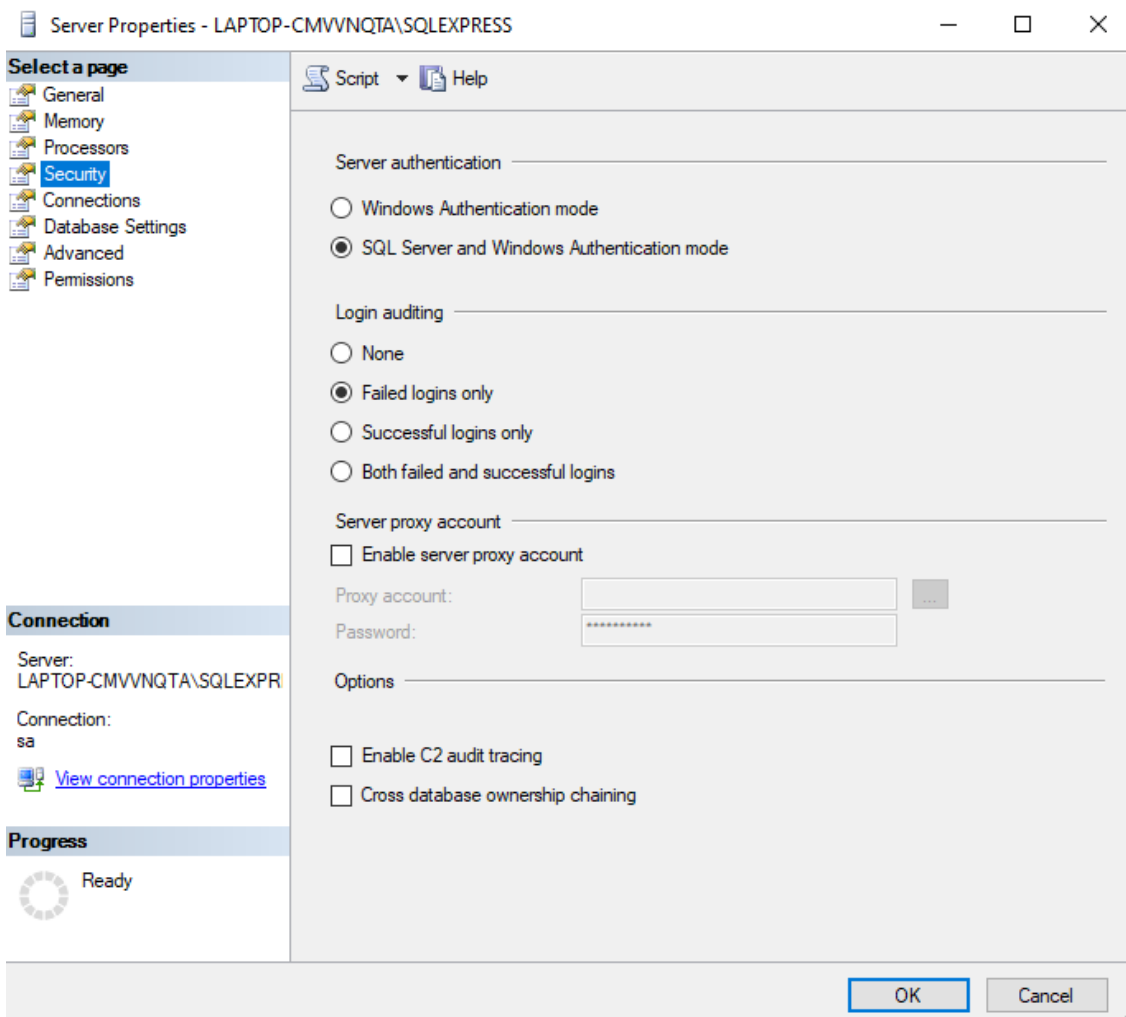
3.- En el apartado de “Status” vamos habilitar el Login dando clic en “Enabled” y damos clic en “OK”:



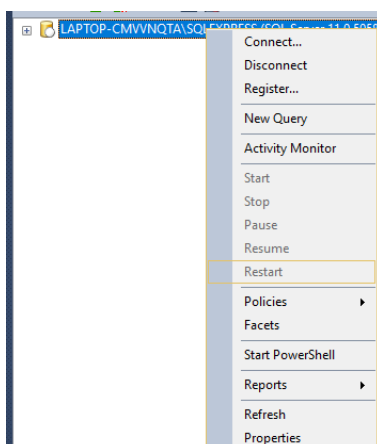
4.- Para poder conectarnos mediante la autenticación SQL Server debemos dar clic derecho en el nombre del servidor de Windows y seleccionar la opción Properties:



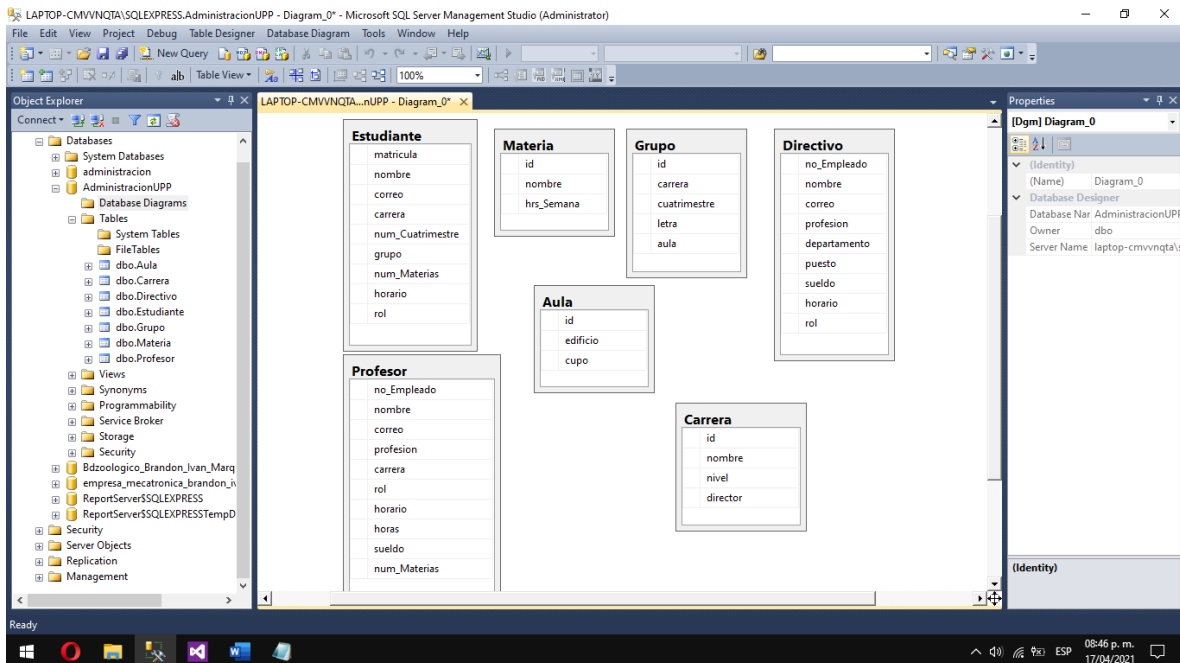
5.- Después nos dirigimos al apartado de “Security” y en el apartado de “Server authentication” seleccionamos “SQL Server and Windows Authentication mode” y damos clic en “OK”



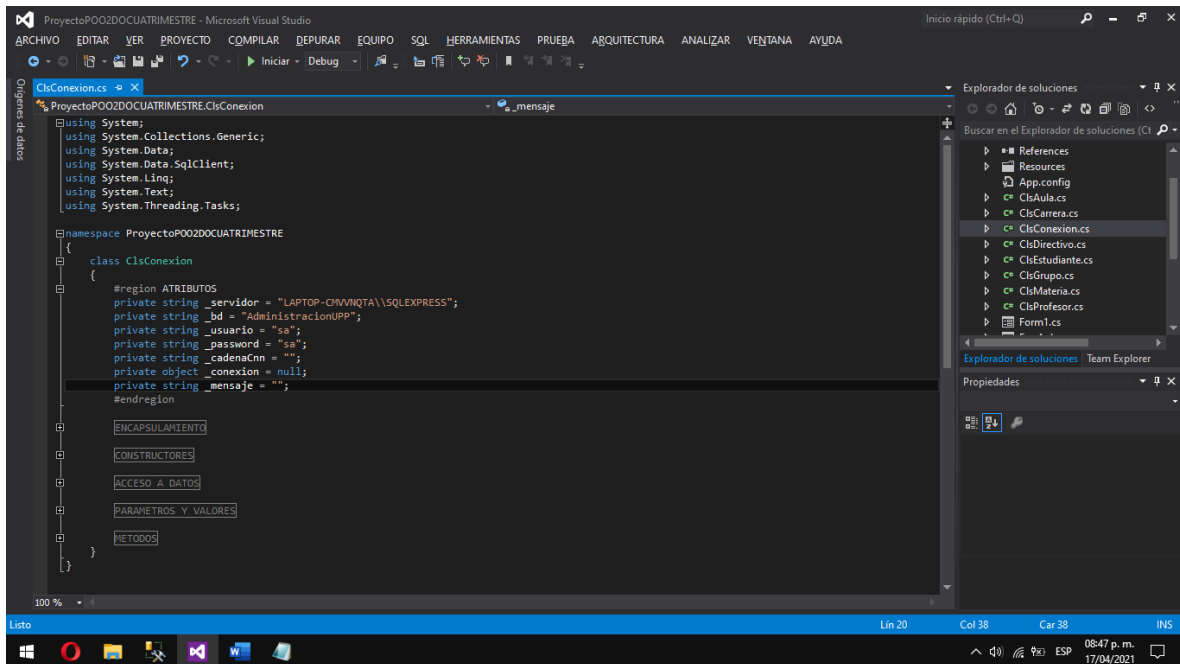
6.- Finalmente reiniciamos el Servidor:



Principalmente se crea la base de datos para posteriormente crear la conexión en c#.



Ahora esta clase de la conexión a la bd en C#, En la primera región es de atributos que se requieren:



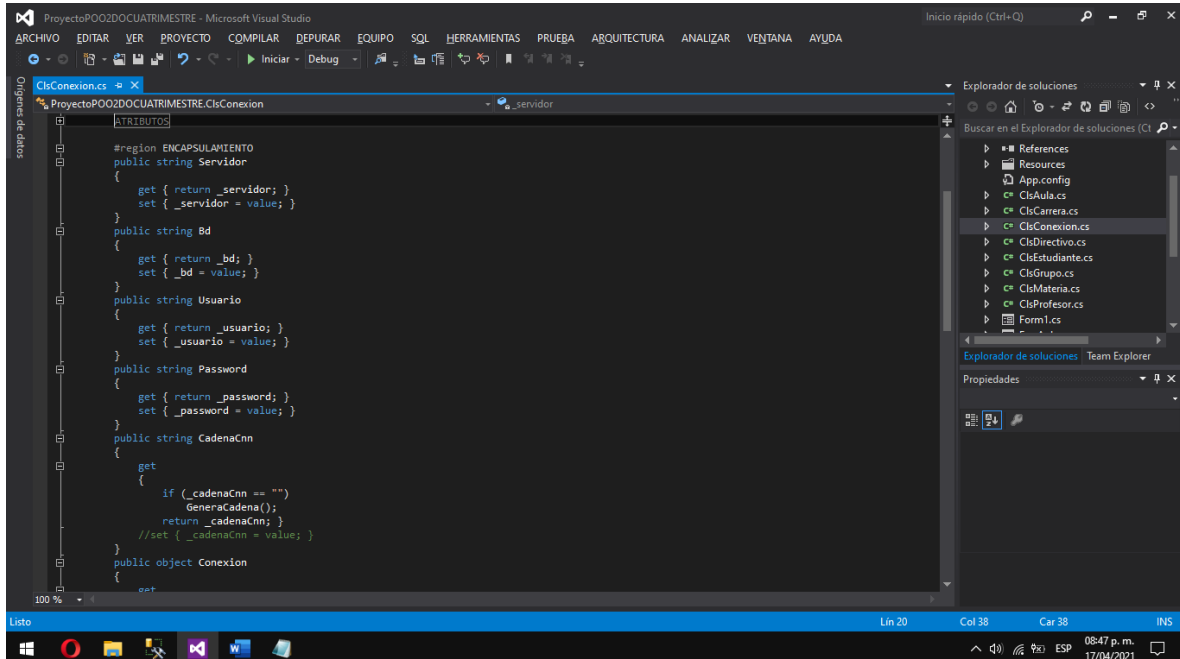
The screenshot shows the Visual Studio IDE with the file `ClsConexion.cs` open. The code is in the `namespace ProyectoPOO2DOCUATRIMESTRE` and defines a `class ClsConexion`. Inside the class, there is a region named `ATRIBUTOS` containing several private fields: `_servidor`, `_bd`, `_usuario`, `_password`, `_cadenaCnn`, `_conexion`, and `_mensaje`. The code also includes using statements for `System`, `System.Collections.Generic`, `System.Data`, `System.Data.SqlClient`, `System.Linq`, `System.Text`, and `System.Threading.Tasks`. The right-hand side of the IDE shows the `Explorador de soluciones` (Solution Explorer) with a tree view of the project files, including `ClsConexion.cs`.

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

namespace ProyectoPOO2DOCUATRIMESTRE
{
    class ClsConexion
    {
        #region ATRIBUTOS
        private string _servidor = "LAPTOP-CHVNIQTA\\SQLEXPRESS";
        private string _bd = "AdministracionUPP";
        private string _usuario = "sa";
        private string _password = "sa";
        private string _cadenaCnn = "";
        private object _conexion = null;
        private string _mensaje = "";
        #endregion

        #region ENCAPSULAMIENTO
        #region CONSTRUCTORES
        #region ACCESO A DATOS
        #region PARAMETROS Y VALORES
        #region METODOS
    }
}
```

La siguiente región es el encapsulamiento de los atributos:

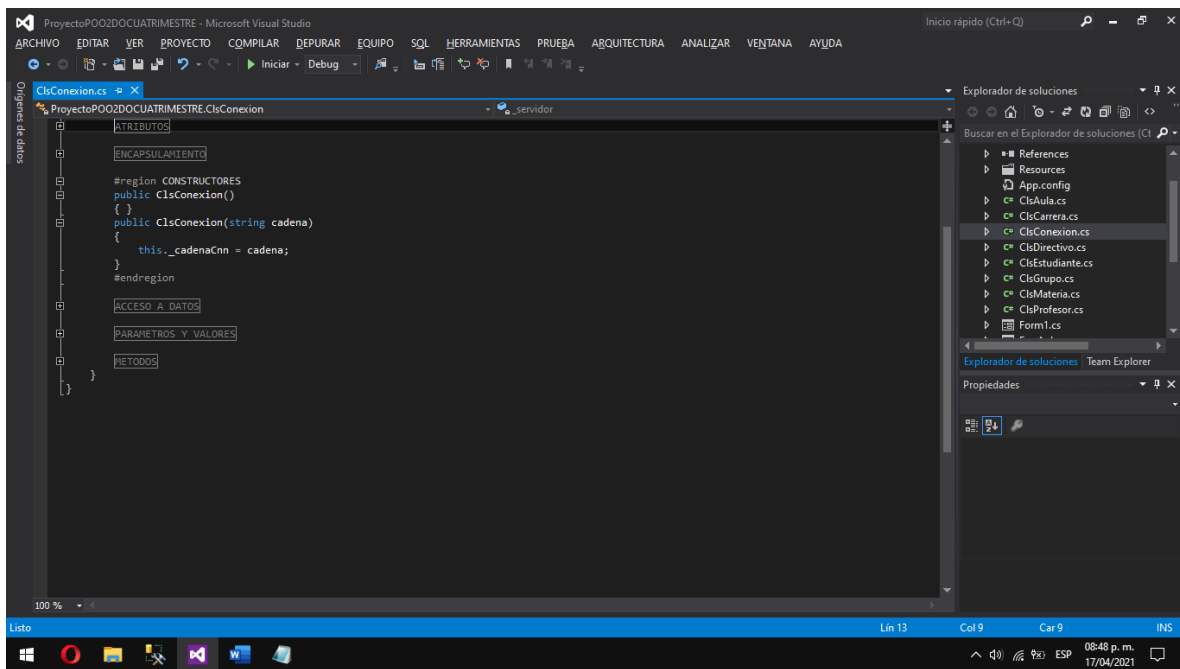


The screenshot shows the same Visual Studio IDE with the `ClsConexion.cs` file. The `ATRIBUTOS` region has been replaced by an `ENCAPSULAMIENTO` region. This region contains public properties for `Servidor`, `Bd`, `Usuario`, `Password`, `CadenaCnn`, and `Conexion`. Each property has a `get` method to return the value and a `set` method to assign the value. The `CadenaCnn` property has a `get` method that checks if the connection string is empty and calls `GeneraCadena()` if it is. The `Conexion` property is a public object. The right-hand side of the IDE shows the `Explorador de soluciones` (Solution Explorer) with the same tree view of the project files.

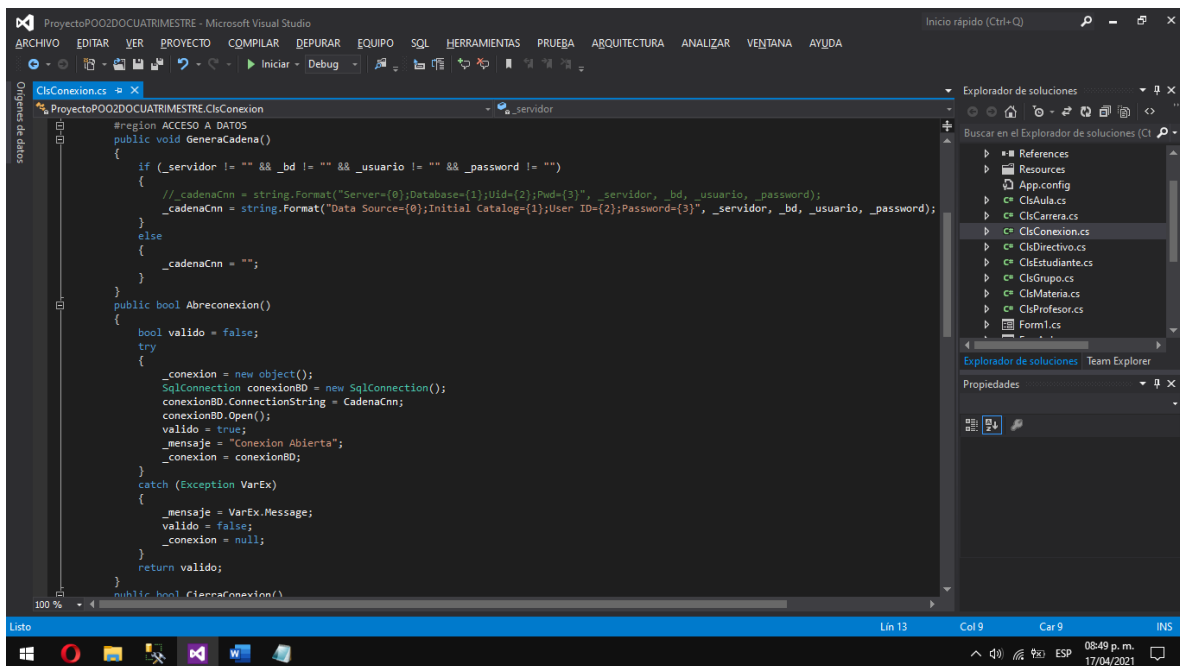
```
#region ENCAPSULAMIENTO
public string Servidor
{
    get { return _servidor; }
    set { _servidor = value; }
}
public string Bd
{
    get { return _bd; }
    set { _bd = value; }
}
public string Usuario
{
    get { return _usuario; }
    set { _usuario = value; }
}
public string Password
{
    get { return _password; }
    set { _password = value; }
}
public string CadenaCnn
{
    get
    {
        if (_cadenaCnn == "")
            GeneraCadena();
        return _cadenaCnn;
    }
    //set { _cadenaCnn = value; }
}
public object Conexion
{
    get
    {

```

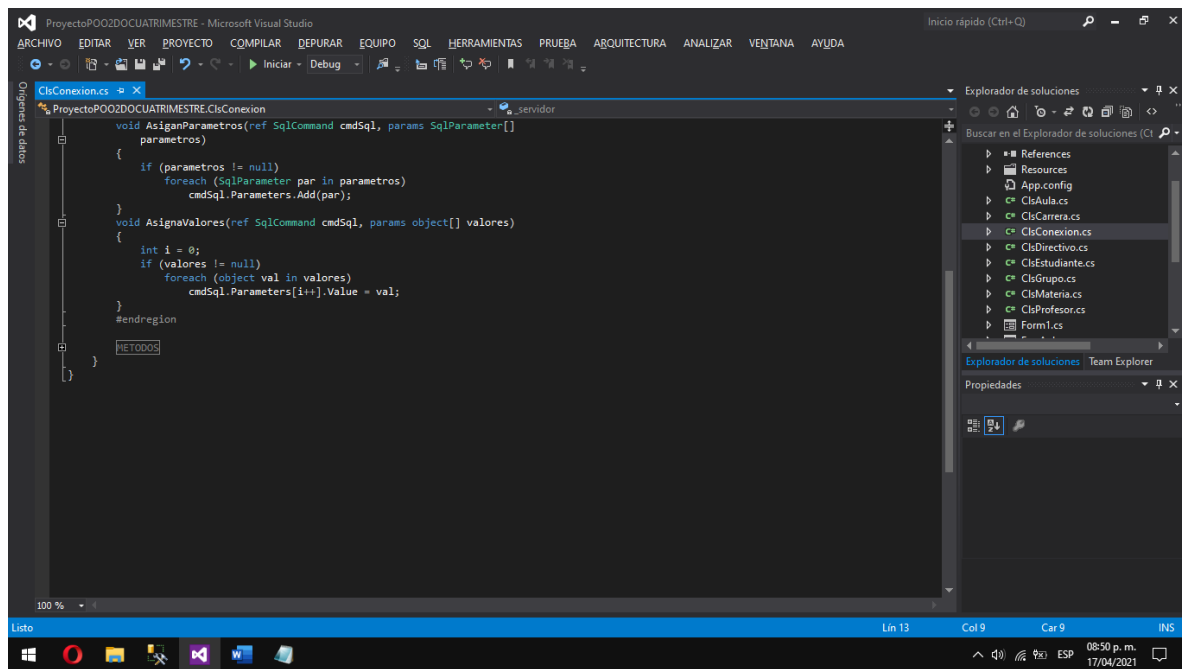
Tenemos el constructor de la clase:



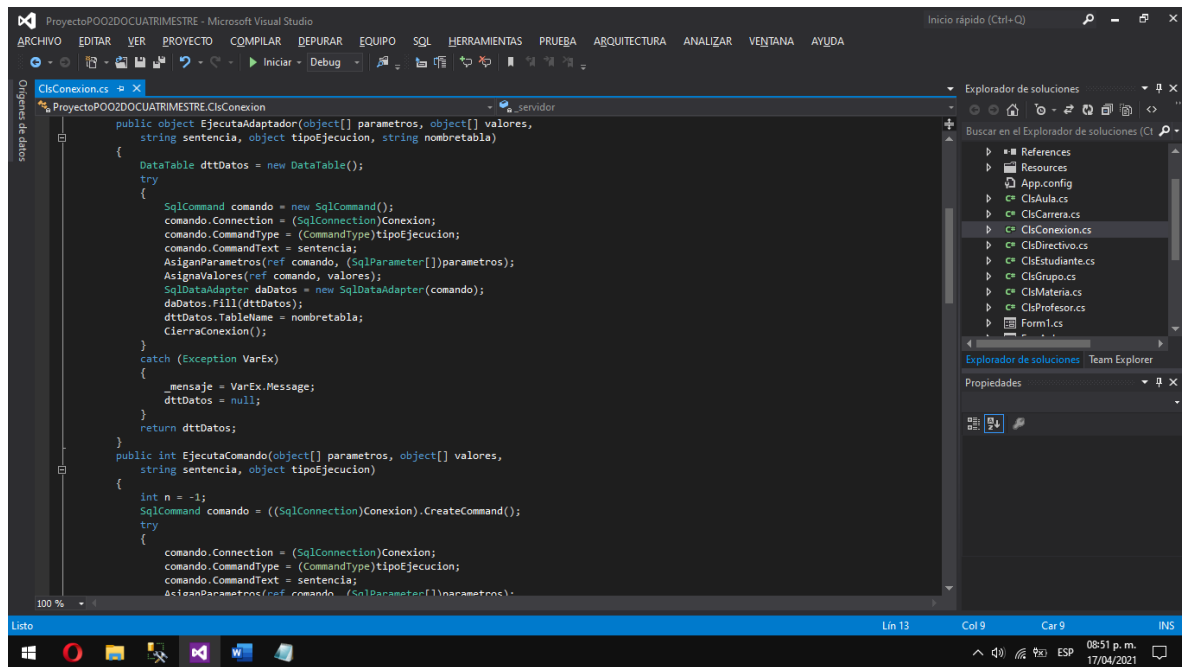
La siguiente región es para poder acceder a los datos:

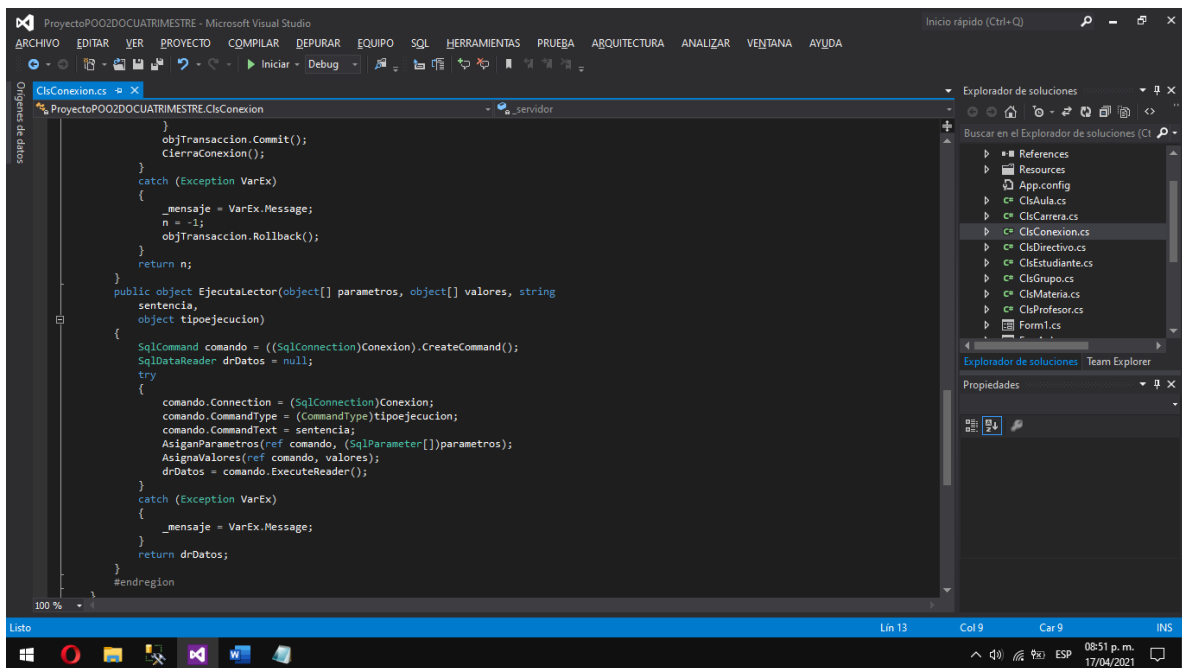
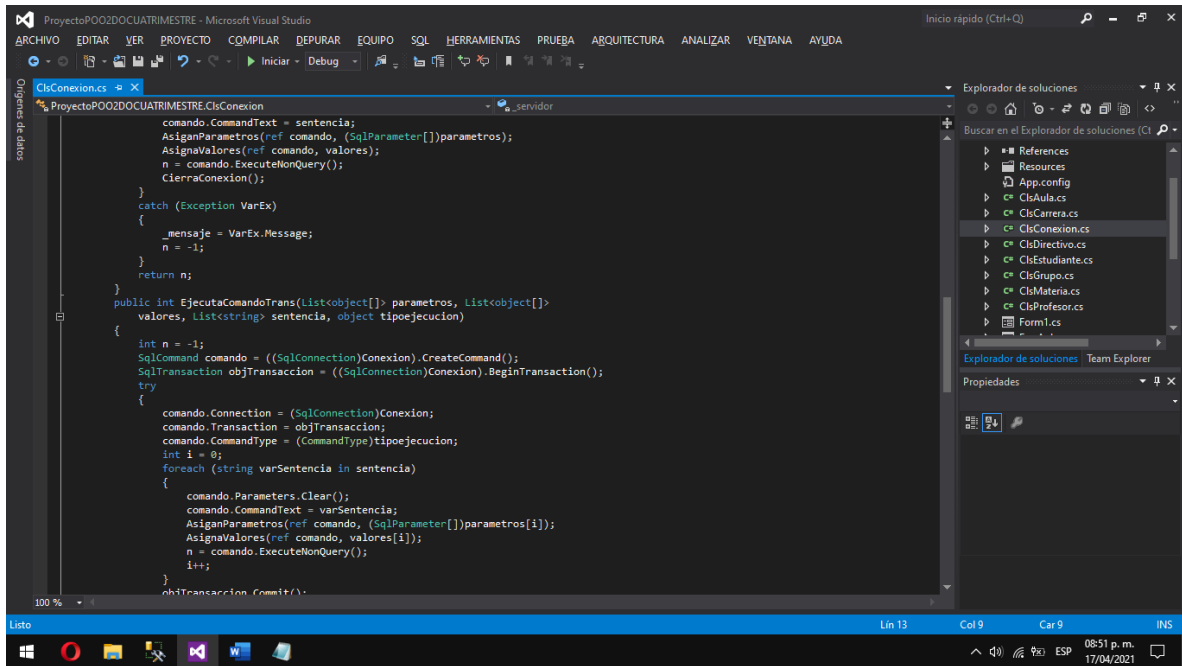


Esta región es para los parámetros y valores:



Y por último se tiene la región para los métodos:

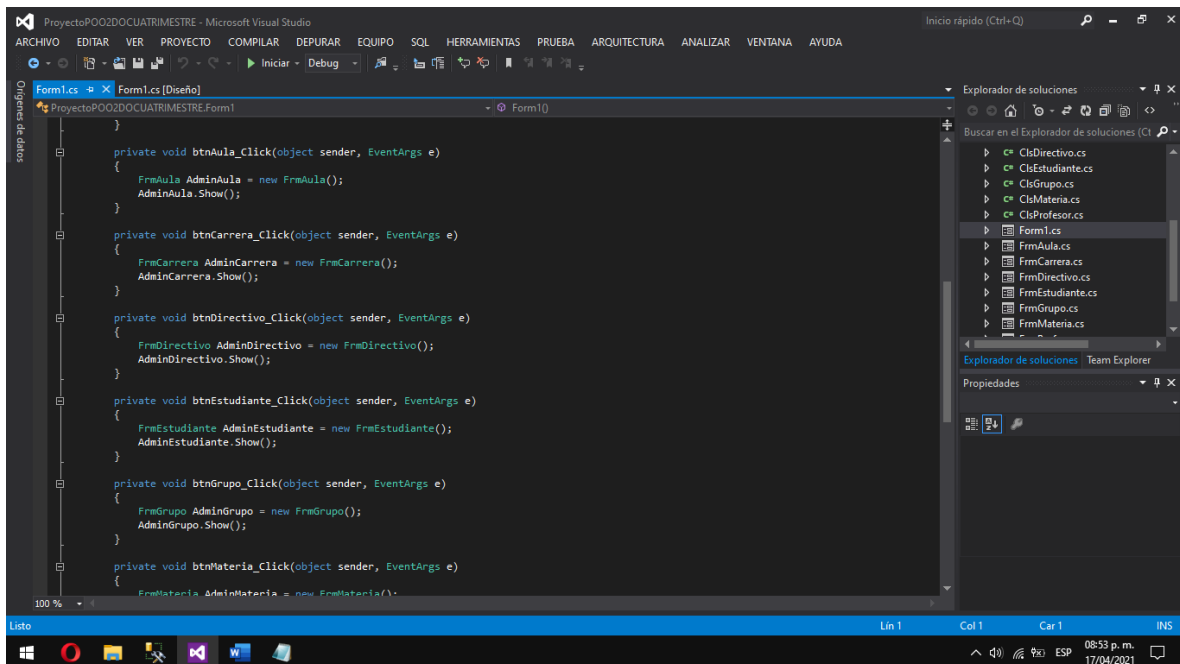




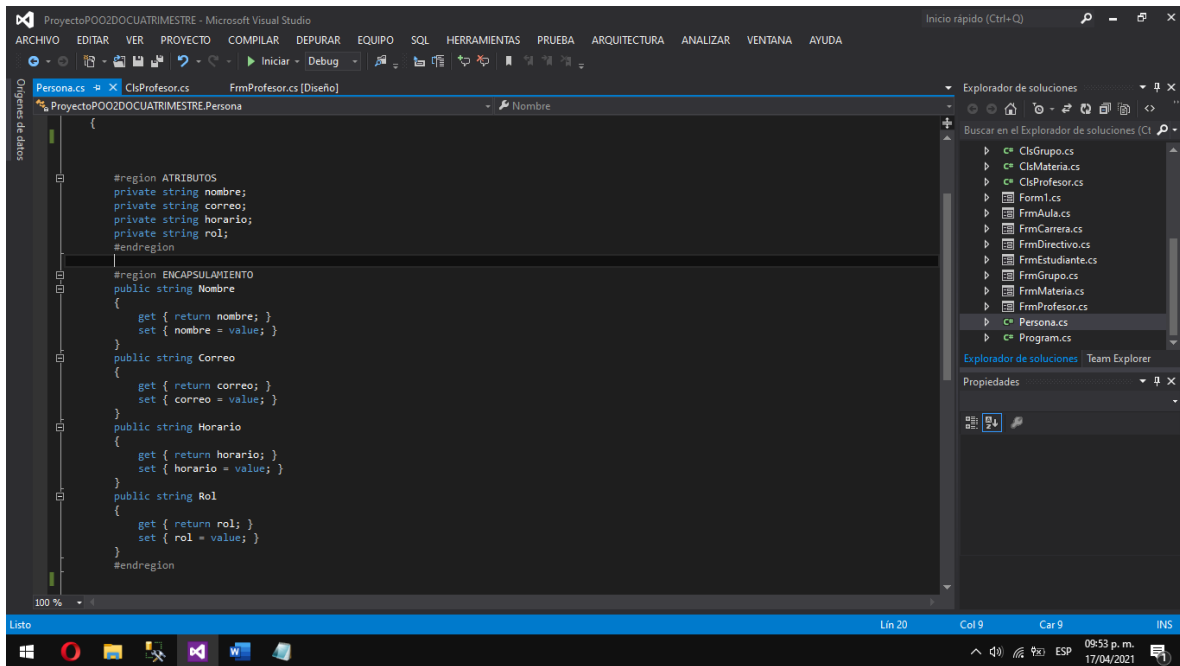
Una vez terminada la conexión se creó el formulario para el menú para elegir la tabla a administrar:



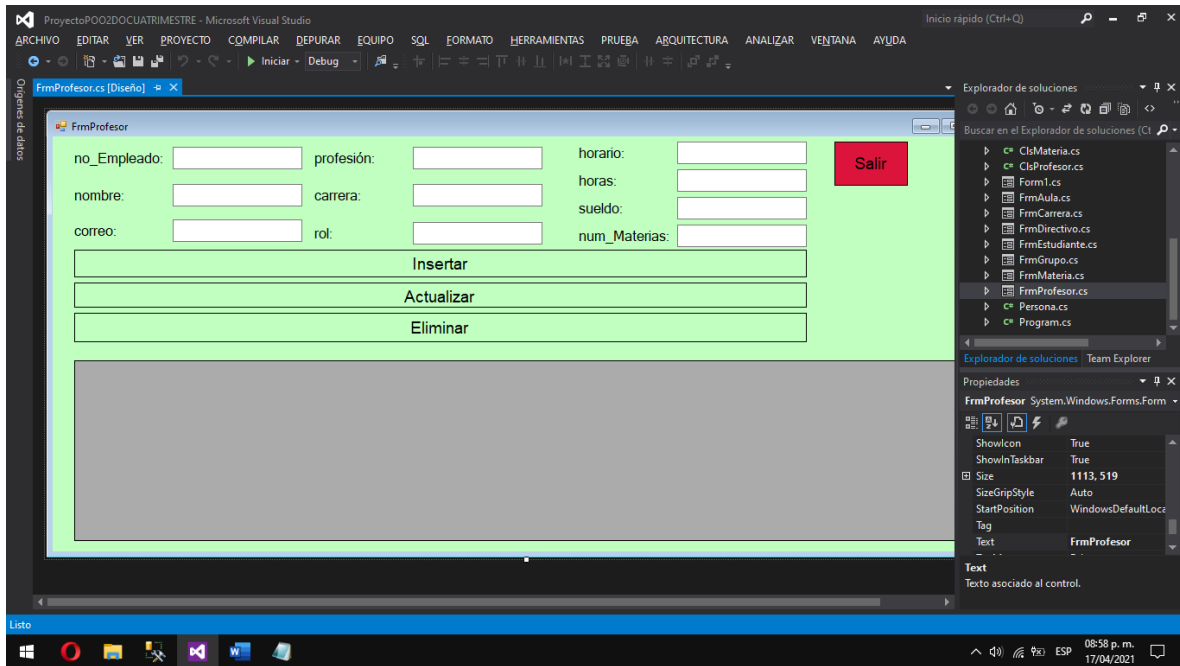
Posteriormente para el código de cada botón se creó la instancia para acceder a cada formulario correspondiente de cada tabla:



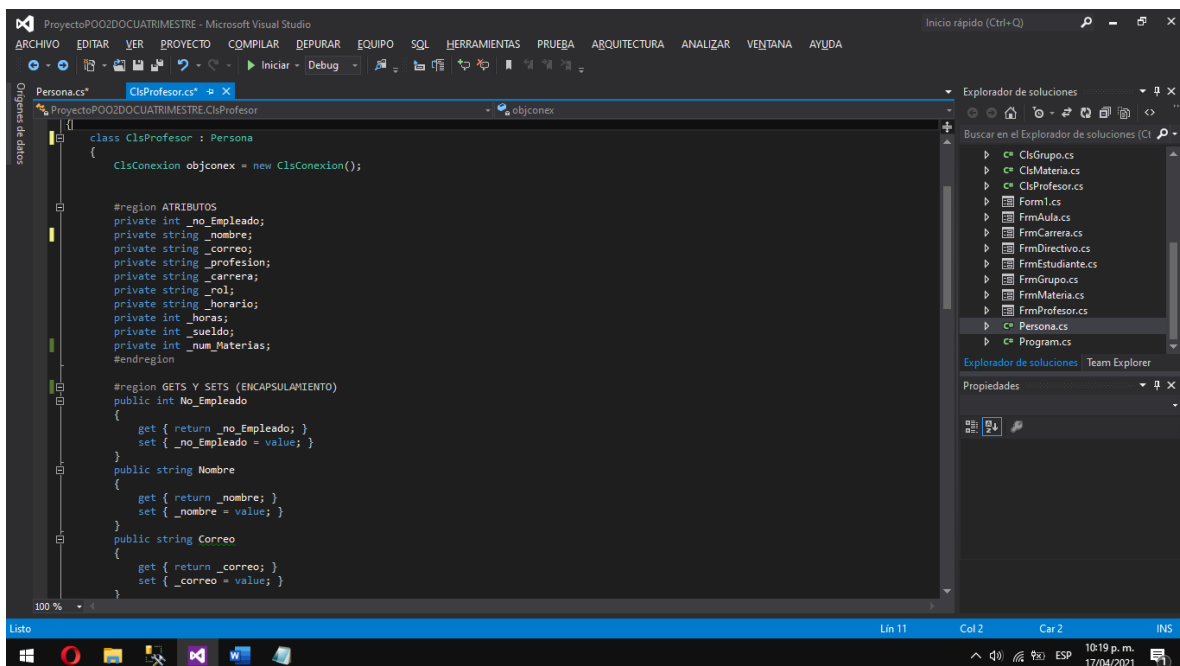
Se crea una clase llamada persona que esa se encargará de heredar atributos:



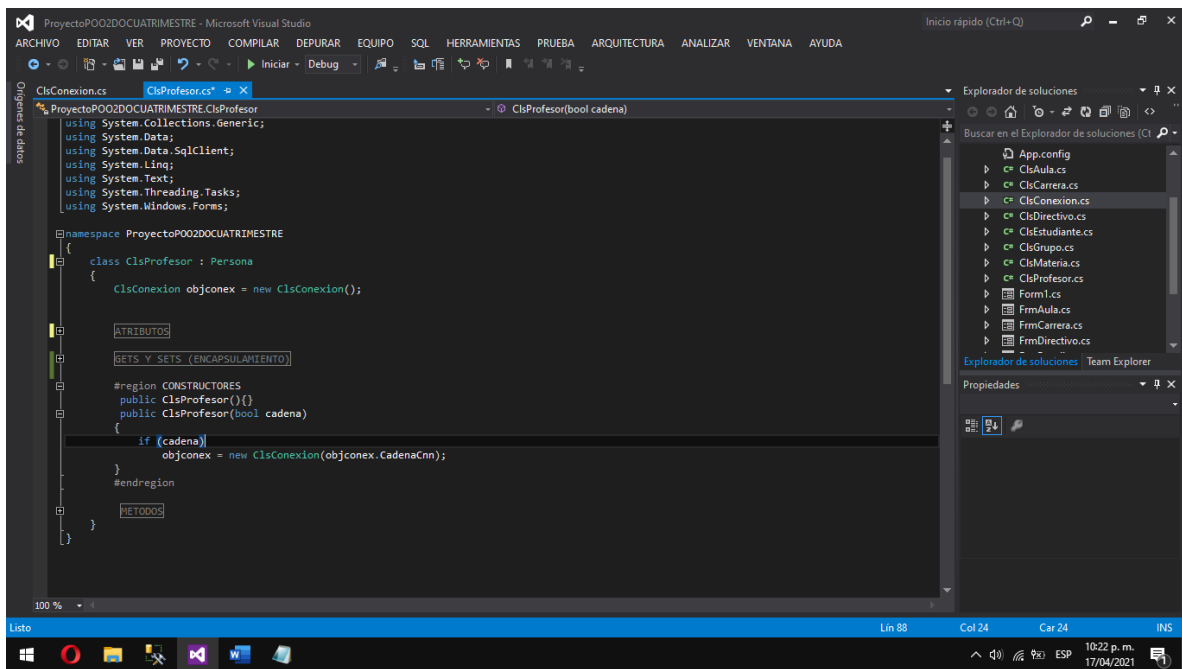
Ahora se mostrará solo un formulario y clase de una tabla ya que en los demás formularios es casi lo mismo lo único que cambia son los atributos y también el código de dicho formulario, los controles que se utilizaron en este formulario son: label, textbox, button y datagriewview



Ahora se mostrará la clase de dicho formulario, principalmente se crea la instancia de la clase ClsConexión y se muestra la declaración y encapsulamiento de sus respectivos atributos:



Ahora se muestra la región de los constructores para generar la cadena:



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

namespace ProyectoPOO2DOCUATRIMESTRE
{
    class ClsProfesor : Persona
    {
        ClsConexion objconex = new ClsConexion();

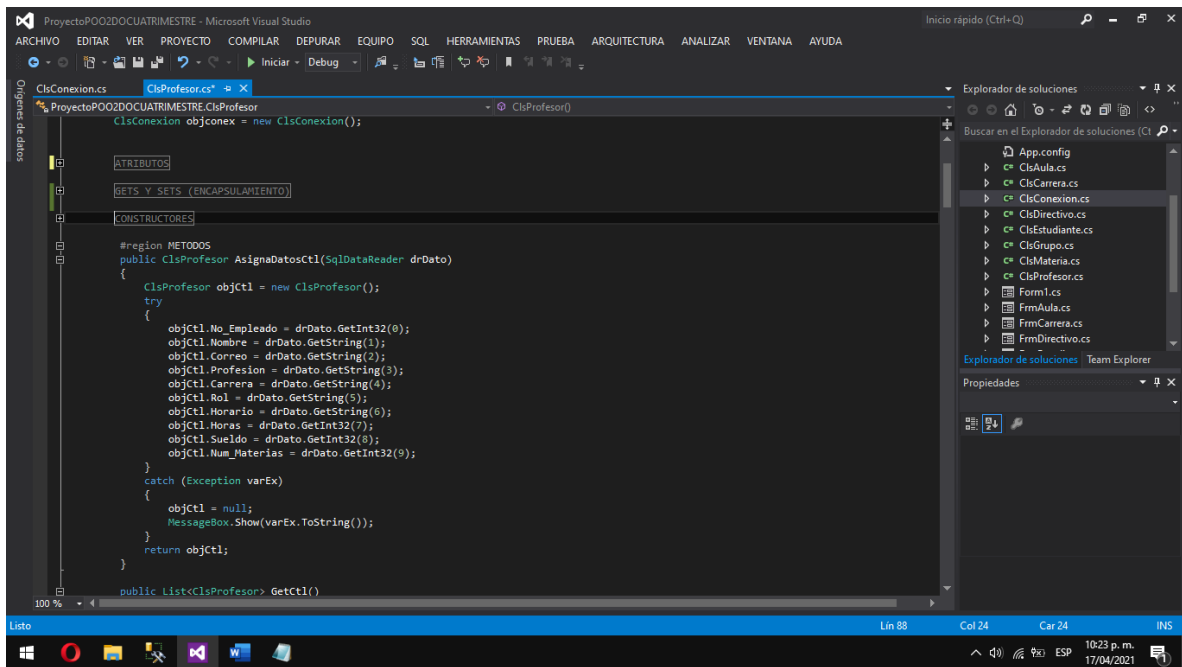
        ATRIBUTOS

        GETS Y SETS (ENCAPSULAMIENTO)

        #region CONSTRUCTORES
        public ClsProfesor(){}
        public ClsProfesor(bool cadena)
        {
            if (cadena)
            {
                objconex = new ClsConexion(objconex.CadenaCnn);
            }
        }
        #endregion

        METODOS
    }
}
```

Ahora se muestra el método para asignar los datos a la tabla:



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

namespace ProyectoPOO2DOCUATRIMESTRE
{
    class ClsProfesor : Persona
    {
        ClsConexion objconex = new ClsConexion();

        ATRIBUTOS

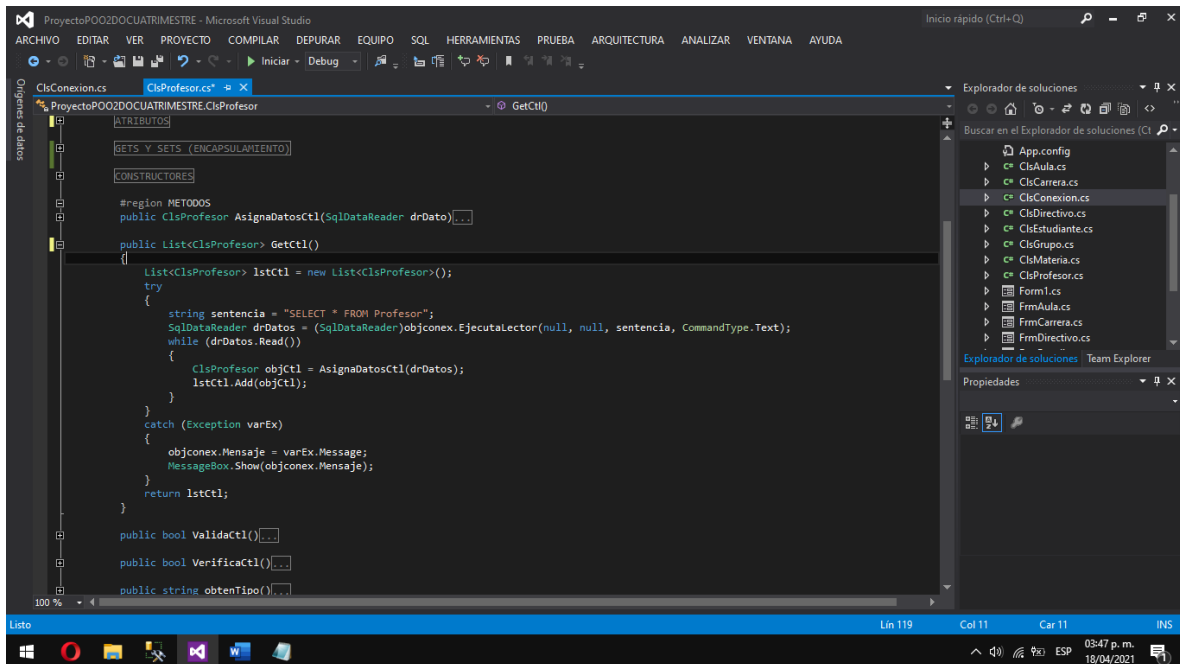
        GETS Y SETS (ENCAPSULAMIENTO)

        CONSTRUCTORES

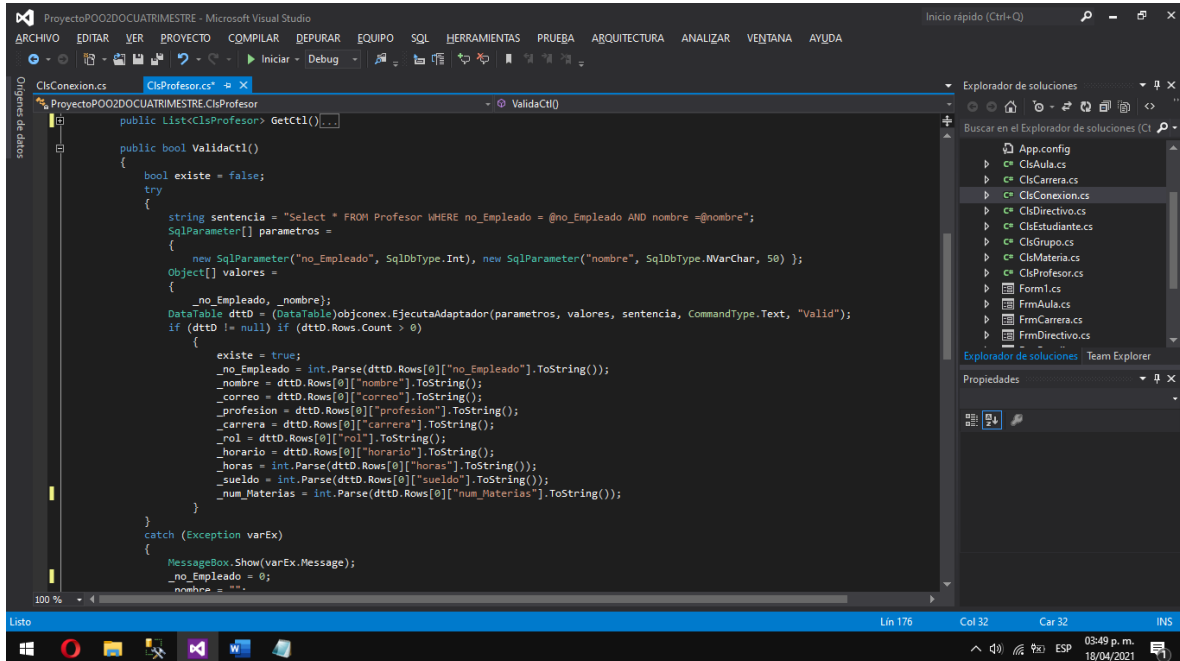
        #region METODOS
        public ClsProfesor AssignaDatosCtl(SqlDataReader drDato)
        {
            ClsProfesor objCtl = new ClsProfesor();
            try
            {
                objCtl.No_Empleado = drDato.GetInt32(0);
                objCtl.Nombre = drDato.GetString(1);
                objCtl.Correo = drDato.GetString(2);
                objCtl.Profesion = drDato.GetString(3);
                objCtl.Carrera = drDato.GetString(4);
                objCtl.Rol = drDato.GetString(5);
                objCtl.Morario = drDato.GetString(6);
                objCtl.Horas = drDato.GetInt32(7);
                objCtl.Sueldo = drDato.GetInt32(8);
                objCtl.Num_Materias = drDato.GetInt32(9);
            }
            catch (Exception varEx)
            {
                objCtl = null;
                MessageBox.Show(varEx.ToString());
            }
            return objCtl;
        }

        public List<ClsProfesor> GetCtl()
        {
            return null;
        }
        #endregion
    }
}
```

Se crea un método de tipo lista de la misma clase para obtener los datos de la tabla utilizando la sentencia “SELECT”



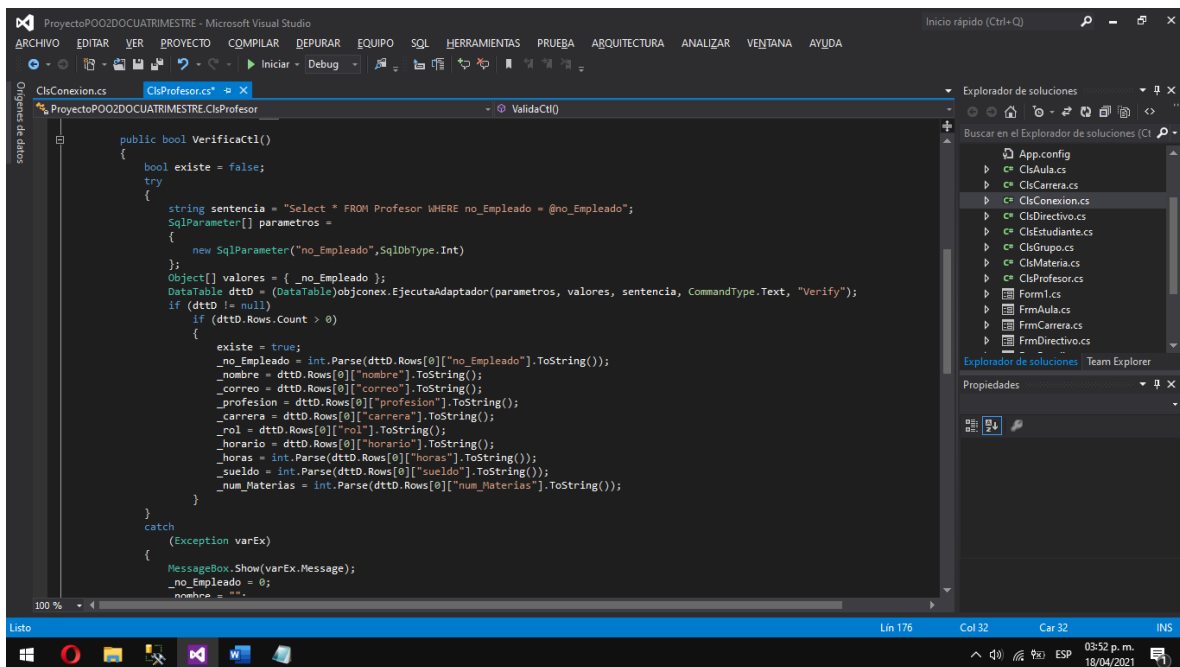
Método para validar el dato de la tabla con la sentencia “SELECT”



Si se detecta algún error se devuelve vacío para cadenas y 0 para enteros:

```
    }  
    catch (Exception varEx)  
    {  
        MessageBox.Show(varEx.Message);  
        _no_Empleado = 0;  
        _nombre = "";  
        _correo = "";  
        _profesion = "";  
        _carrera = "";  
        _rol = "";  
        _horario = "";  
        _horas = 0;  
        _sueldo = 0;  
        _num_Materias = 0;  
        existe = false;  
    }  
    return existe;  
}
```

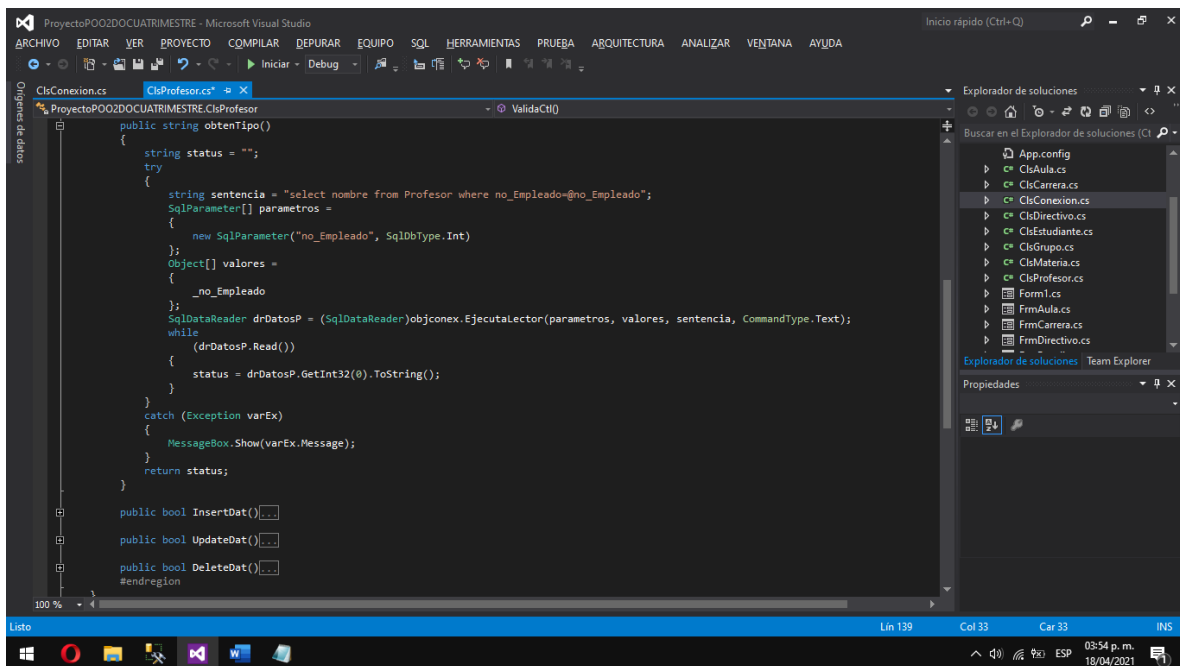
Método para verificar datos de la tabla:



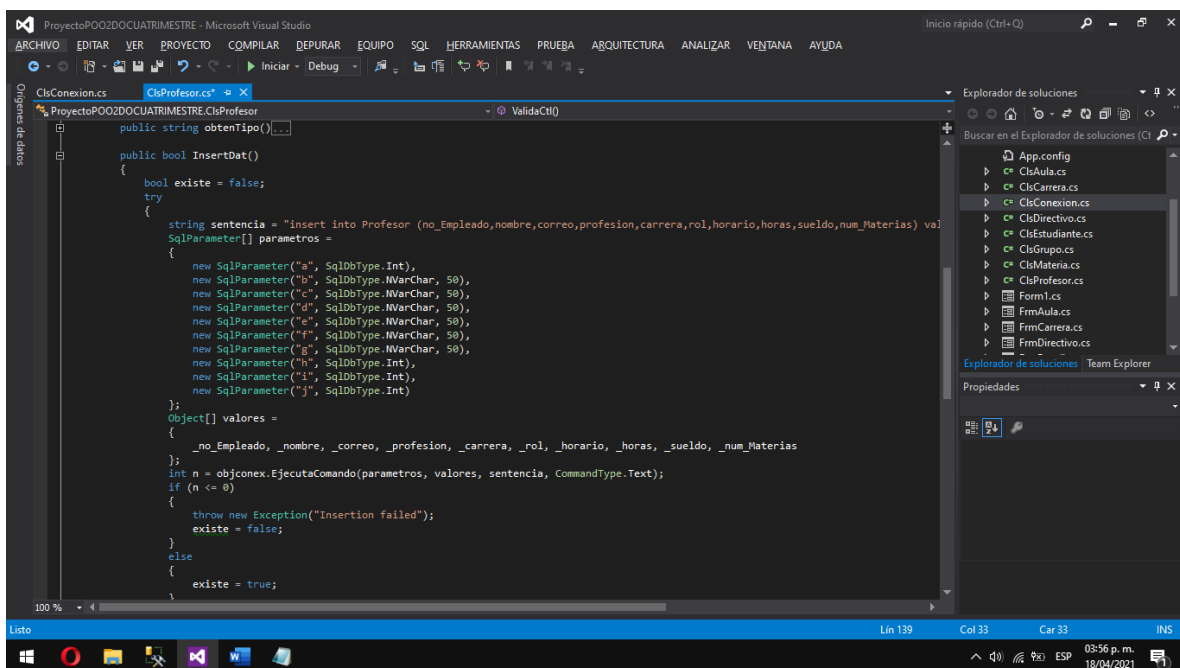
Si se detecta algún error se devuelve vacío para cadenas y 0 para enteros:

```
}
catch (Exception varEx)
{
    MessageBox.Show(varEx.Message);
    _no_Empleado = 0;
    _nombre = "";
    _correo = "";
    _profesion = "";
    _carrera = "";
    _rol = "";
    _horario = "";
    _horas = 0;
    _sueldo = 0;
    _num_Materias = 0;
    existe = false;
}
return existe;
}
```

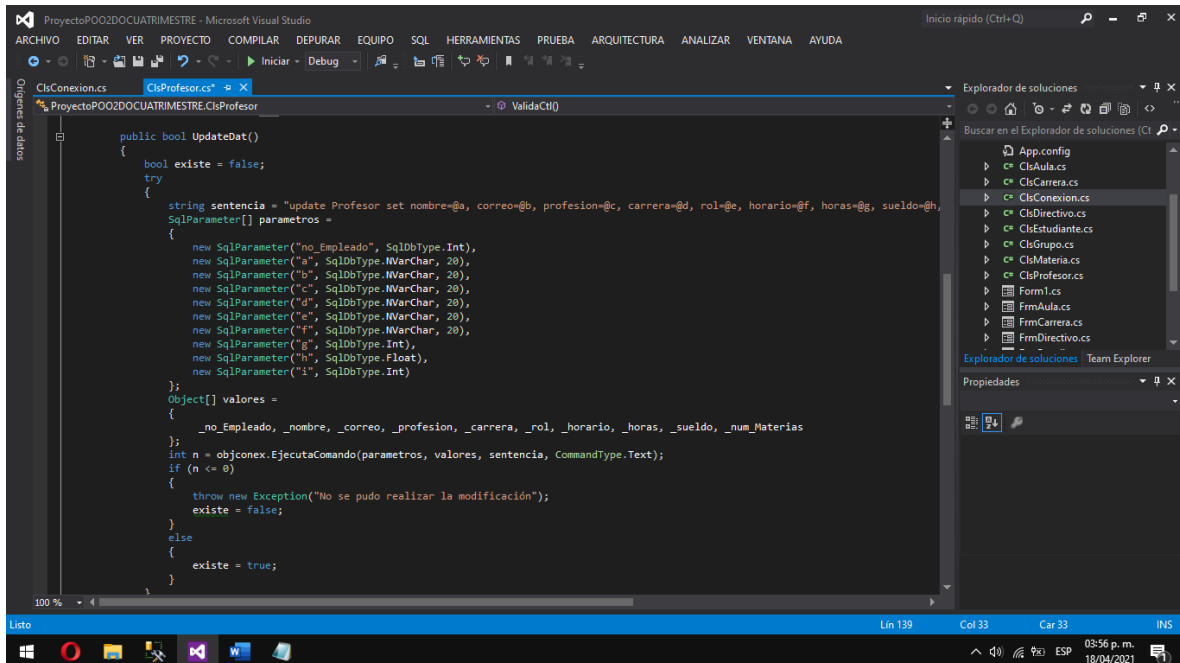
Método para obtener el tipo de los datos de la tabla:



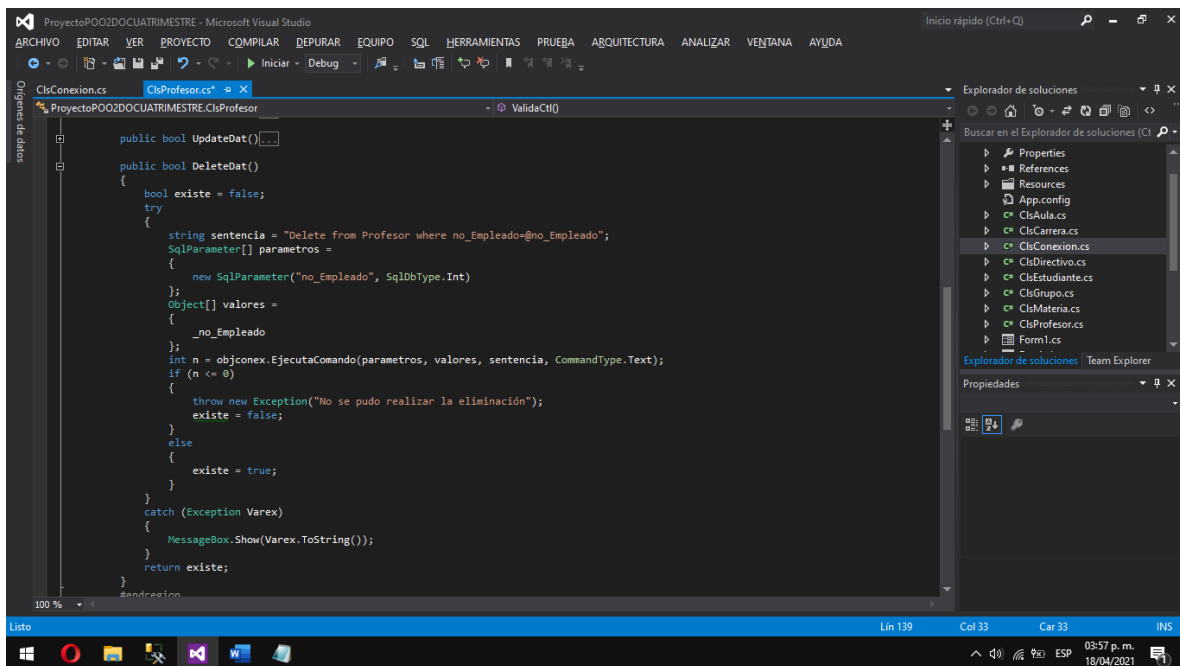
Método para insertar datos en la tabla utilizando la sentencia “INSERT”
Utilizando parámetros para cada uno de los atributos:



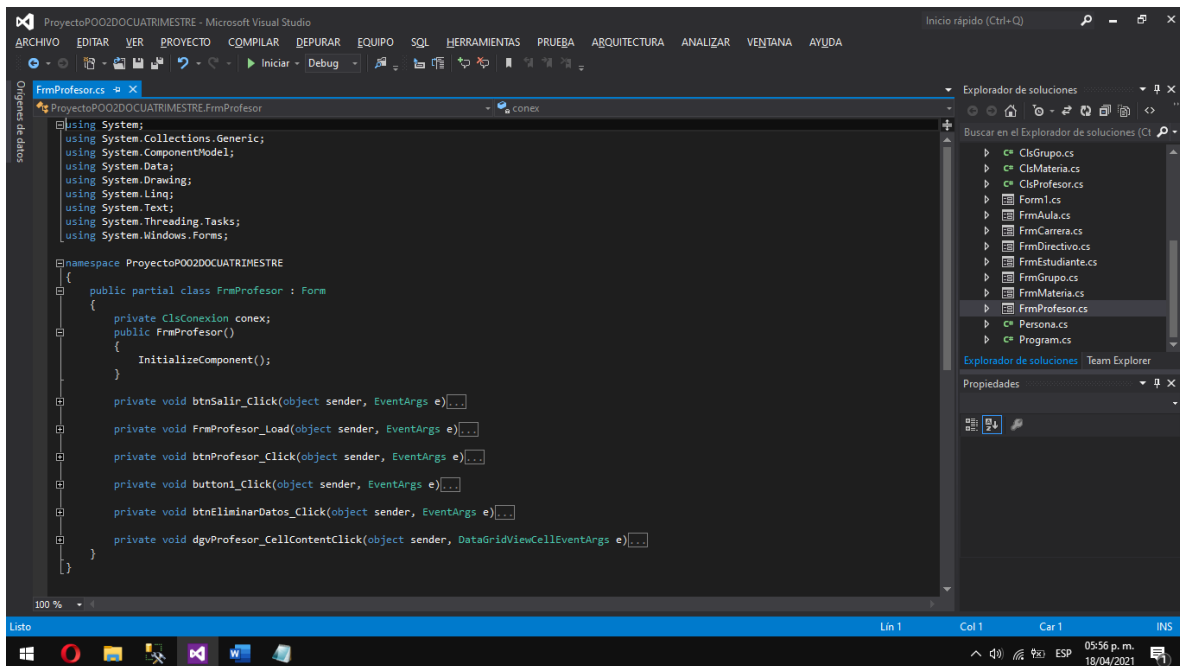
Método para actualizar datos en la tabla utilizando la sentencia “UPDATE” Utilizando parámetros para cada uno de los atributos:



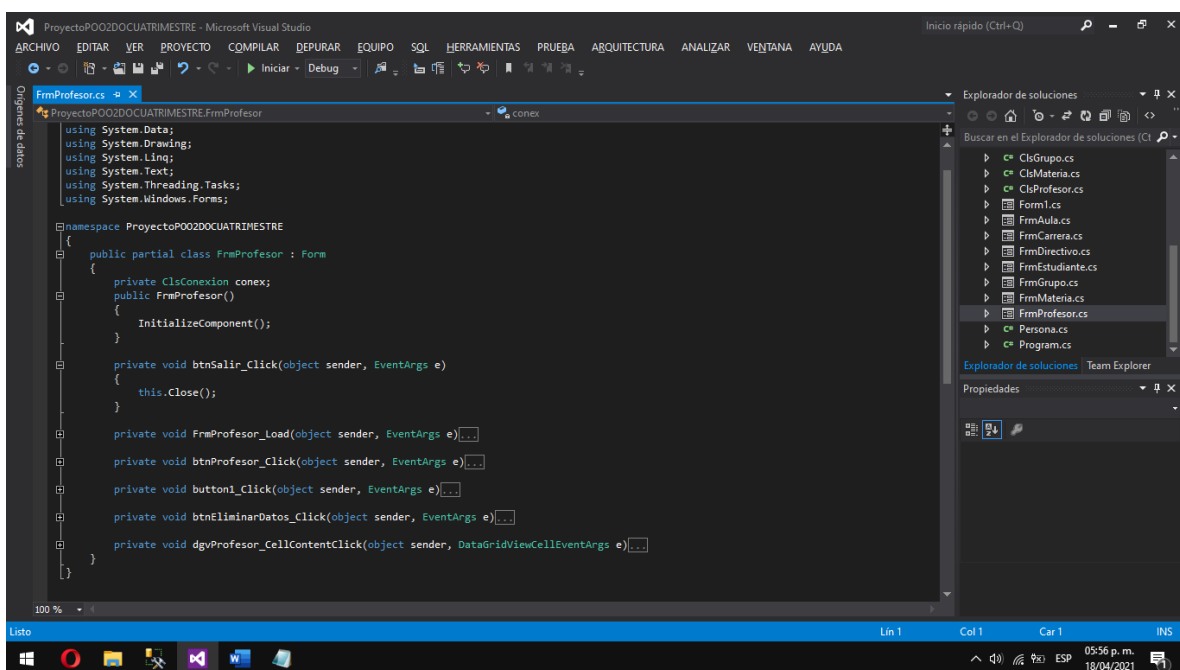
Método para eliminar datos de la tabla con la sentencia “DELETE”:



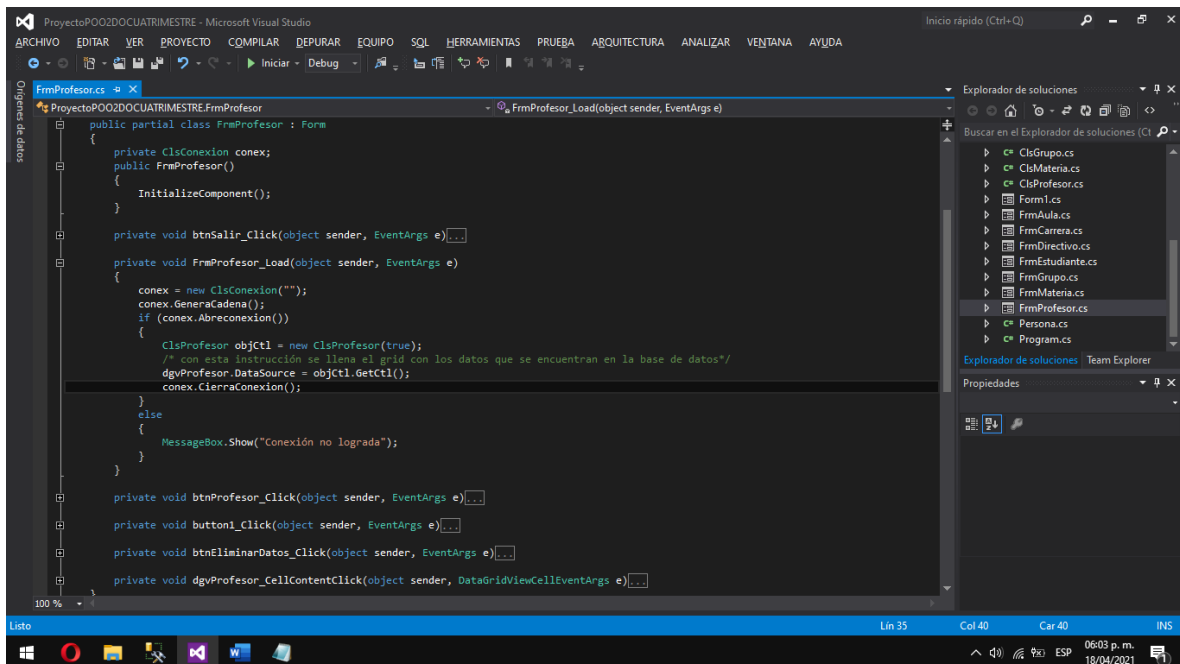
Ahora se mostrará el código del formulario de una tabla, principalmente se hace una llamada a la clase de la conexión privadamente creando un objeto llamado “conex”, posteriormente se muestra los eventos:



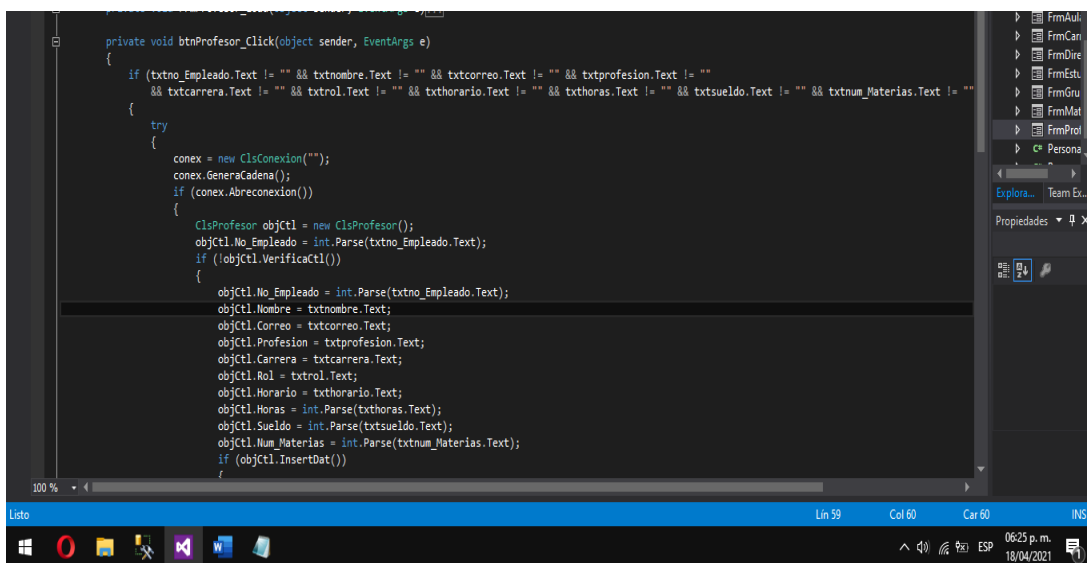
El primer evento es el de salir de tal formulario dando click un botón llamado salir:



El segundo evento es el evento Load que sirve para cargar cosas al inicializar tal formulario, en este caso lo que se hace es hacer una instancia de la clase profesor para que a nuestro DataGridView concatenado con el objeto DataSource que sirve para buscar datos igualándolo a objeto de la tabla es decir los datos, además de concatenarlo con get de la tabla (que es un método que se creó en la clase) y esto sirve para obtener los datos, posteriormente se cierra la conexión, caso contrario se imprime mensaje que no se logró la conexión.



El siguiente método es para insertar datos a la tabla principalmente se verifica si cada campo correspondiente de cada atributo es diferente de vacío, caso contrario se tienen que llenar todos los campos, después se genera la cadena de la conexión, entonces si se abre la conexión se crea la instancia de la clase Profesor, después se comprueba si el objeto de la tabla o datos son diferentes del método verificar dato, de ser así con el objeto creado se concatena con cada uno de los sets correspondientes de cada atributo y se va igualar a cada una de las cajas de texto, si se requiere conversión se hace en este caso con los enteros.



```
private void btnProfesor_Click(object sender, EventArgs e)
{
    if (txtno_Empleado.Text != "" && txtnombre.Text != "" && txtcorreo.Text != "" && txtprofesion.Text != ""
        && txtcarrera.Text != "" && txtrol.Text != "" && txthorario.Text != "" && txthoras.Text != "" && txtsuelo.Text != "" && txtnum_Materias.Text != "")
    {
        try
        {
            conex = new ClsConexion("");
            conex.GeneraCadena();
            if (conex.Abreconexion())
            {
                ClsProfesor objCtl = new ClsProfesor();
                objCtl.No_Empleado = int.Parse(txtno_Empleado.Text);
                if (!objCtl.VerificaCtl())
                {
                    objCtl.No_Empleado = int.Parse(txtno_Empleado.Text);
                    objCtl.Nombre = txtnombre.Text;
                    objCtl.Correo = txtcorreo.Text;
                    objCtl.Profesion = txtprofesion.Text;
                    objCtl.Carrera = txtcarrera.Text;
                    objCtl.Rol = txtrol.Text;
                    objCtl.Horario = txthorario.Text;
                    objCtl.Horas = int.Parse(txthoras.Text);
                    objCtl.Suelo = int.Parse(txtsuelo.Text);
                    objCtl.Num_Materias = int.Parse(txtnum_Materias.Text);
                    if (objCtl.InsertDat())
                    {

```

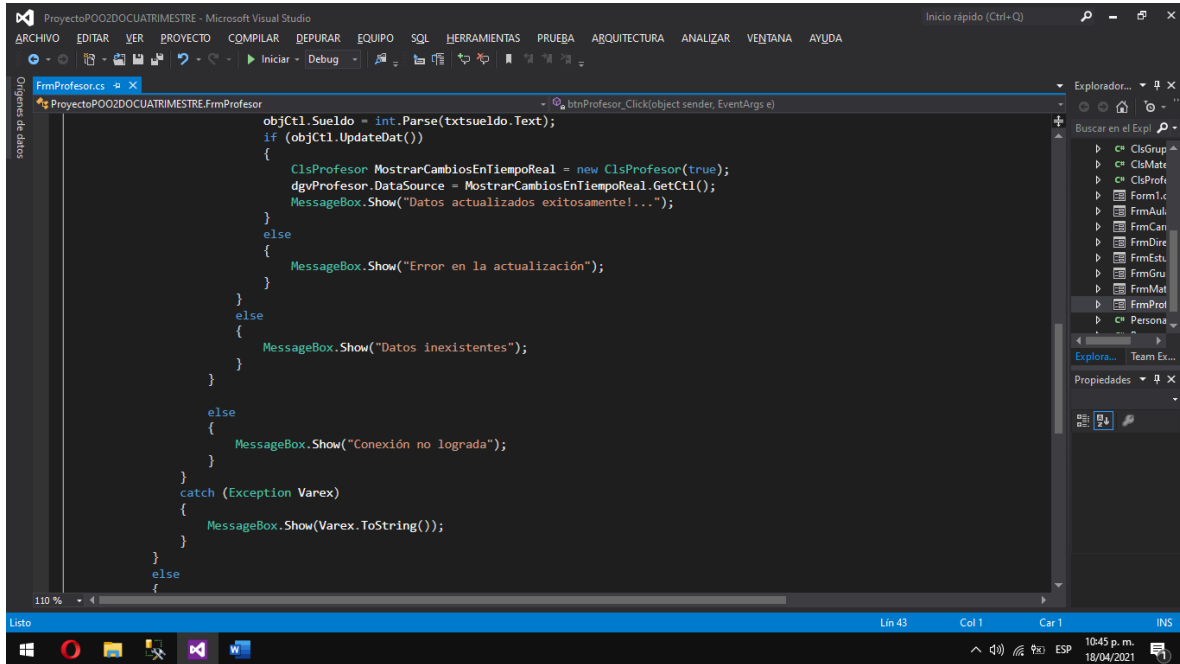
Ahora si se insertaron los datos correctamente se hace una instancia para mostrar los cambios en tiempo real y se muestra ventana de inserción exitosa. Casos contrarios se muestran los siguientes mensajes:

```
objCtl.Num_Materias = int.Parse(txtnum_Materias.Text);
if (objCtl.InsertDat())
{
    ClsProfesor MostrarCambiosEnTiempoReal = new ClsProfesor(true);
    dgvProfesor.DataSource = MostrarCambiosEnTiempoReal.GetCtl();
    MessageBox.Show("Datos insertados exitosamemnte!...");
}
else
{
    MessageBox.Show("Error al insertar");
}
}
else
{
    MessageBox.Show("Datos ya existentes");
}
}
else
{
    MessageBox.Show("Conexión no lograda");
}
}
catch (Exception Varex)
{
    MessageBox.Show(Varex.ToString());
}
}
```

El siguiente método es para actualizar datos a la tabla principalmente se verifica si cada campo correspondiente de cada atributo es diferente de vacío, caso contrario se tienen que llenar todos los campos, después se genera la cadena de la conexión, entonces si se abre la conexión se crea la instancia de la clase Profesor, después se comprueba si el objeto de la tabla o datos son diferentes del método verificar dato, de ser así con el objeto creado se concatena con cada uno de los sets correspondientes de cada atributo y se va igualar a cada una de las cajas de texto, si se requiere conversión se hace en este caso con los enteros.

```
private void button1_Click(object sender, EventArgs e)
{
    if (txtno_Empleado.Text != "" && txtnombre.Text != "" && txtcorreo.Text != "" && txtprofesion.Text != ""
        && txtcarrera.Text != "" && txtrol.Text != "" && txthorario.Text != "" && lblhoras.Text != "" && txtsuelo.Text != "" && txtn
    {
        try
        {
            conex = new ClsConexion("");
            conex.GeneraCadena();
            if (conex.Abreconexion())
            {
                ClsProfesor objCtl = new ClsProfesor();
                objCtl.No_Empleado = int.Parse(txtno_Empleado.Text);
                if (objCtl.VerificaCtl())
                {
                    objCtl.No_Empleado = int.Parse(txtno_Empleado.Text);
                    objCtl.Nombre = txtnombre.Text;
                    objCtl.Correo = txtcorreo.Text;
                    objCtl.Profesion = txtprofesion.Text;
                    objCtl.Carrera = txtcarrera.Text;
                    objCtl.Rol = txtrol.Text;
                    objCtl.Horario = txthorario.Text;
                    objCtl.Horas = int.Parse(txthoras.Text);
                    objCtl.Suelo = int.Parse(txtsuelo.Text);
                    if (objCtl.UpdateDat())
                    {
                        ClsProfesor MostrarCambiosEnTiempoReal = new ClsProfesor(true);
                        dgvProfesor.DataSource = MostrarCambiosEnTiempoReal.GetCtl();
                        MessageBox.Show("Datos actualizados exitosamente! ");
                    }
                }
            }
        }
        catch { }
    }
}
```

Ahora si se actualizaron los datos correctamente se hace una instancia para mostrar los cambios en tiempo real y se muestra ventana de actualización exitosa. Casos contrarios se muestran los siguientes mensajes:



```
ProjectoPOO2DOCUATRIMESTRE - Microsoft Visual Studio
Inicio rápido (Ctrl+Q)

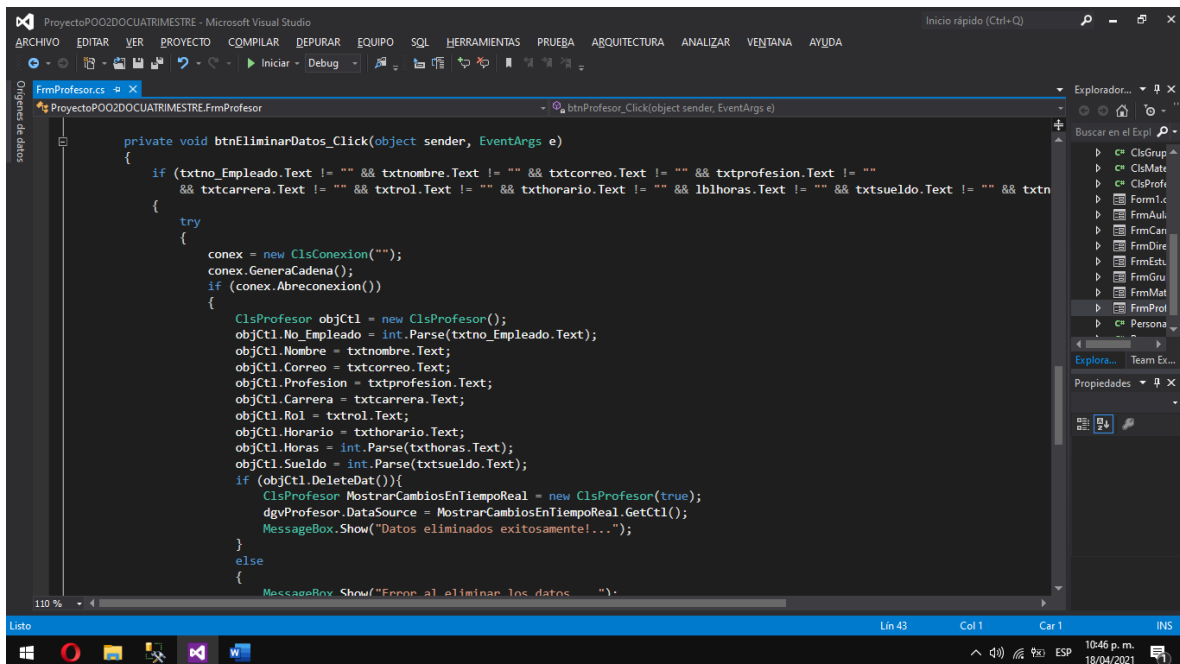
FrmProfesor.cs
ProjectoPOO2DOCUATRIMESTRE.FrmProfesor
btnProfesor_Click(object sender, EventArgs e)

objCtl.Sueldo = int.Parse(txtsueldo.Text);
if (objCtl.UpdateDat())
{
    ClsProfesor MostrarCambiosEnTiempoReal = new ClsProfesor(true);
    dgvProfesor.DataSource = MostrarCambiosEnTiempoReal.GetCtl();
    MessageBox.Show("Datos actualizados exitosamente!");
}
else
{
    MessageBox.Show("Error en la actualización");
}
else
{
    MessageBox.Show("Datos inexistentes");
}
else
{
    MessageBox.Show("Conexión no lograda");
}
catch (Exception Varex)
{
    MessageBox.Show(Varex.ToString());
}
else
{
}
```

Explorador...
Buscar en el Expl...
ClsGrup
ClsMate
ClsProfe
Form1.c
FrmAul
FrmCan
FrmDire
FrmEstu
FrmGru
FrmMat
FrmProfi
Persona
Explora... Team Ex...
Propiedades

Lin 43 Col 1 Car 1 INS
Listo
10:45 p. m.
18/04/2021

El siguiente método es para eliminar datos a la tabla principalmente se verifica si cada campo correspondiente de cada atributo es diferente de vacío, caso contrario se tienen que llenar todos los campos, después se genera la cadena de la conexión, entonces si se abre la conexión se crea la instancia de la clase Profesor, después se comprueba si abre la conexión, de ser así con el objeto creado se concatena con cada uno de los sets correspondientes de cada atributo y se va igualar a cada una de las cajas de texto, si se requiere conversión se hace en este caso con los enteros.



```
private void btnEliminarDatos_Click(object sender, EventArgs e)
{
    if (txtno_Empleado.Text != "" && txtnombre.Text != "" && txtcorreo.Text != "" && txtprofesion.Text != ""
        && txtcarrera.Text != "" && txtrol.Text != "" && txthorario.Text != "" && lblhoras.Text != "" && txtsueldo.Text != "" && txtno)
    {
        try
        {
            conex = new ClsConexion("");
            conex.GeneraCadena();
            if (conex.Abreconexion())
            {
                ClsProfesor objCtl = new ClsProfesor();
                objCtl.No_Empleado = int.Parse(txtno_Empleado.Text);
                objCtl.Nombre = txtnombre.Text;
                objCtl.Correo = txtcorreo.Text;
                objCtl.Profesion = txtprofesion.Text;
                objCtl.Carrera = txtcarrera.Text;
                objCtl.Rol = txtrol.Text;
                objCtl.Horario = txthorario.Text;
                objCtl.Horas = int.Parse(txthoras.Text);
                objCtl.Sueldo = int.Parse(txtsueldo.Text);
                if (objCtl.DeleteDat()){
                    ClsProfesor MostrarCambiosEnTiempoReal = new ClsProfesor(true);
                    dgvProfesor.DataSource = MostrarCambiosEnTiempoReal.GetCtl();
                    MessageBox.Show("Datos eliminados exitosamente!");
                }
                else
                {
                    MessageBox.Show("Error al eliminar los datos.");
                }
            }
        }
        catch { }
    }
}
```

De ser así se muestran los cambios en tiempo real, casos contrarios se muestran los posibles errores:

```
        objCtl.Sueldo = int.Parse(txtSueldo.Text);  
        if (objCtl.DeleteDat()){  
            ClsProfesor MostrarCambiosEnTiempoReal = new ClsProfesor(true);  
            dgvProfesor.DataSource = MostrarCambiosEnTiempoReal.GetCtl();  
            MessageBox.Show("Datos eliminados exitosamente!...");  
        }  
        else  
        {  
            MessageBox.Show("Error al eliminar los datos....");  
        }  
    }  
    else  
    {  
        MessageBox.Show("Conexión no lograda");  
    }  
}  
catch (Exception Varex)  
{  
    MessageBox.Show(Varex.ToString());  
}  
}  
else  
{  
    MessageBox.Show("Hay un campo para llenar");  
}  
}
```


Este último evento (**CellContentClick**) sirve para seleccionar una fila del DataGridView y después dar click en cualquier campo de los atributos y automáticamente cada dato de cada campo se igualará a cada caja de texto correspondientemente. Primero le decimos a cada de texto a que va a ser igual, primero se indica el nombre del DataGridView (dgvProfesor), Lo concatenamos con CurrentRow (Indica que es un registro), Lo concatenamos con Cells (Esto indica la celda) posteriormente entre corchtes [] le indicamos el número de celda, Lo concatenamos con Value (Indica que es un valor) y finalmente lo convertimos a cadena con .ToString();

```
private void dgvProfesor_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    txtno_Empleado.Text = dgvProfesor.CurrentRow.Cells[0].Value.ToString();
    txtnombre.Text = dgvProfesor.CurrentRow.Cells[1].Value.ToString();
    txtcorreo.Text = dgvProfesor.CurrentRow.Cells[2].Value.ToString();
    txtprofesion.Text = dgvProfesor.CurrentRow.Cells[3].Value.ToString();
    txtcarrera.Text = dgvProfesor.CurrentRow.Cells[4].Value.ToString();
    txtrol.Text = dgvProfesor.CurrentRow.Cells[5].Value.ToString();
    txthorario.Text = dgvProfesor.CurrentRow.Cells[6].Value.ToString();
    txthoras.Text = dgvProfesor.CurrentRow.Cells[7].Value.ToString();
    txtsueldo.Text = dgvProfesor.CurrentRow.Cells[8].Value.ToString();
    txtnum_Materias.Text = dgvProfesor.CurrentRow.Cells[9].Value.ToString();
}
}
```

Conclusión.

Me llevé una muy buena experiencia al desarrollar este proyecto ya que me gusto la forma de trabajo que planeé, Además de seguir llevando en práctica todo lo que se compone de la programación orientada a objetos como las clases, herencia, atributos, métodos y encapsulamientos.

Una de las mejores partes en las que trabaje fue en la creación de la base de datos ya que se me da y me gusta mucho el trabajar con las bases de datos, Además de trabajar en el lenguaje C# ya que me gusta más que Python a la hora de crear interfaces gráficas ya que tenemos la librería de WindowsForms y nos da una gran variedad de controles para solo arrastrarlos nuestro formulario y para darles diferentes propiedades, lo podemos de manera gráfica y no a código como en Python con la librería Tkinter, ya que la parte de dar diferentes propiedades a un control en Python puede dar error si escribimos mal el código y creo que todo eso es un poco más tardado.

Bibliografía

? (2021). ? Obtenido de ?: Las sentencias son los elementos básicos en los que se divide el código en un lenguaje de programación. Al fin y al cabo, un programa no es más que un conjunto de sentencias que se ejecutan para realizar una cierta tarea. Además, como ya habrás visto, en

anonymus. (2021). *anonymus*. Obtenido de anonymus: <https://www.neumoytoraxpanama.org/cloud/resources/documentos/encapsulamiento-informtica.pdf>

conceptosBasicosdePOO. (2021). *conceptosBasicosdePOO*. Obtenido de [conceptosBasicosdePOO: http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm#:~:text=Los%20atributos%20son%20las%20características,valores%20distintos%20para%20estas%20variables](http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm#:~:text=Los%20atributos%20son%20las%20características,valores%20distintos%20para%20estas%20variables).

explorativos, c. (2021). *curriculos explorativos*. Obtenido de curriculos explorativos: <http://contenidos.sucerman.com/nivel3/dispositivos/unidad1/leccion2.html>

if(geek). (2021). *if(geek)*. Obtenido de if(geek): <https://ifgeekthen.everis.com/es/herencia-en-programacion-orientada-objetos#:~:text=La%20herencia%20permite%20que%20se,métodos%20o%20redefinir%20los%20heredados>.

masTeclas. (2021). *masTeclas*. Obtenido de masTeclas: <https://www.masquetecclas.com/articulo/el-concepto-de-clase/>

profile. (2021). *profile*. Obtenido de profile: <https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/>

Wikipedia. (2021). *Wikipedia*. Obtenido de Wikipedia: [https://es.wikipedia.org/wiki/Método_\(informática\)#:~:text=En%20la%20programación%2C%20un%20método,de%20los%20métodos%20de%20instancia](https://es.wikipedia.org/wiki/Método_(informática)#:~:text=En%20la%20programación%2C%20un%20método,de%20los%20métodos%20de%20instancia).