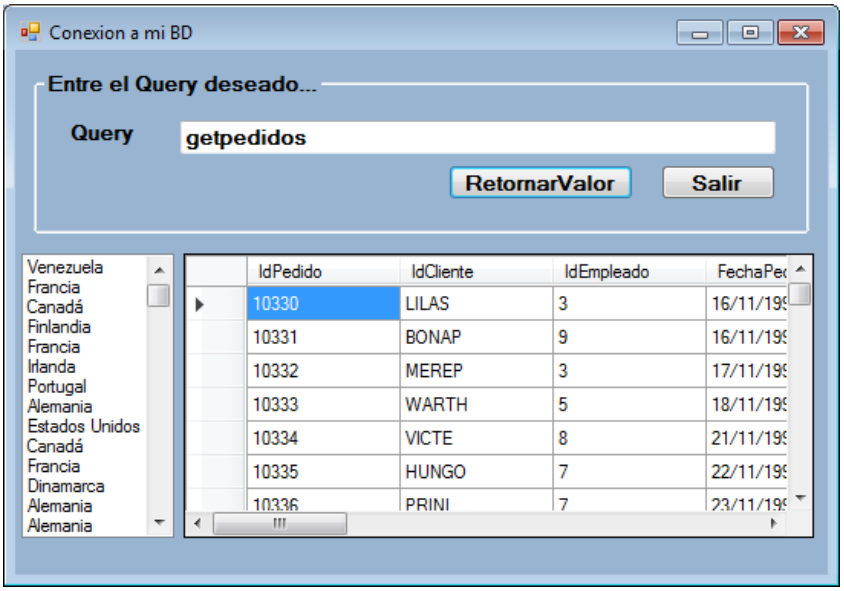
UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA INDUSTRIAL

GUIA DE LABORATORIO:

DISEÑO DE FORMULARIOS EN C# y USO DE ADO.Net



CURSO: INGENIERIA DE SOFTWARE
BASE DE DATOS Y PROGRAMACION VISUAL
INFORMATICA

PROFESOR: Ing. PONCE BENITES Wiler Arturo

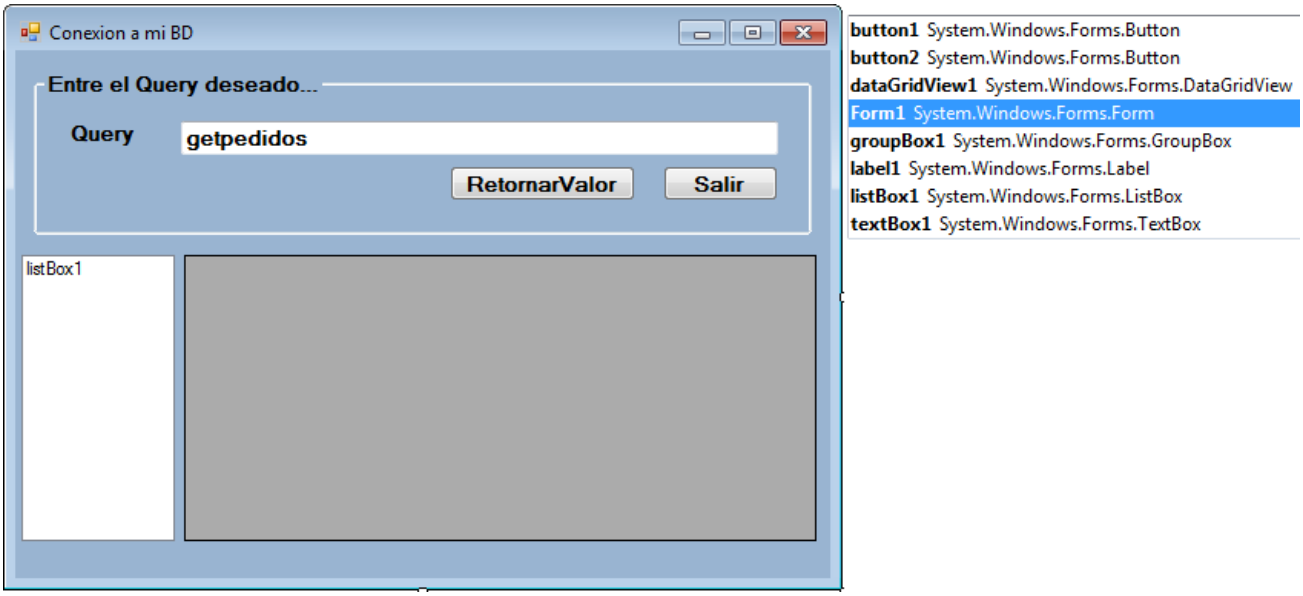
Correo: ingeniería.software@industrial.unmsm.pe
base.datos@industrial.unmsm.pe
informatica@industrial.unmsm.pe

Campus Virtual: http://campus.industrial.unmsm.edu.pe

Web: http://industrial.unmsm.edu.pe

2016

Form1 (Diseño)



Form1 (Programación)

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

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

        #region Ejecutando button1
        private void button1_Click(object sender, EventArgs e)
        {
            LlenandoListBox(textBox1.Text);
            dataGridView1.DataSource = EjecutarQuery(textBox1.Text);
        }
        #endregion

        #region finalizar el sistema
        private void button2_Click(object sender, EventArgs e)
        {
            Close();
        }
        #endregion
```

```
        #region Ejecutando El Query
        public DataTable EjecutarQuery(string miquery)
        {
            try
            {
                Conn adoconn=new Conn();
                SqlConnection miconnection = new
SqlConnection(adoconn.getconnectionstring("Northwind"));
                // creando una tablaen memoria para guardar el resultado
                DataTable midatatable = new DataTable();
                SqlCommand micomando = new SqlCommand(miquery, miconnection);

                SqlDataAdapter miadactador = new SqlDataAdapter();
                miadactador.SelectCommand=micomando;
                miadactador.Fill(midatatable);

                return midatatable;
            }
            catch (Exception ex)
            {
                throw new Exception("Error ejecutando el query: "+ ex.Message);
            }
        }
        #endregion

        #region Llenando la lista
        public void LlenandoListBox(string miquery)
        {
            Conn adoconn = new Conn();
            SqlConnection miconecction = new
SqlConnection(adoconn.getconnectionstring("Northwind"));
            SqlCommand micomando = new SqlCommand(miquery,miconecction);
            miconecction.Open();
            SqlDataReader reader = micomando.ExecuteReader();

            while (reader.Read())
            {
                string resultado = reader["PaísDestinatario"].ToString();
                listBox1.Items.Add(resultado);
            }

        }
        #endregion
    }
}
```

Clase: Conn.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Configuration;

namespace pyAdoNet
{
    class Conn
    {
        #region Declara-variables
        string SqlServer = "", SqlUser = "", SqlPassword = "", SqlDataBase = "", NTaut = "";
        #endregion

        #region MetodoConst-Conn
        private string connectionstringcustom()
        {
            string StrConn;
            try
            {
                if(NTaut.ToLower() == "false")

StrConn="server="+SqlServer+";uid="+SqlUser+";pwd="+SqlPassword+";database="+SqlDataBase;
                else
                    StrConn="Data Source="+SqlServer+";Integrated Security=SSPI"+";Initial
Catalog="+SqlDataBase;
                return StrConn;
            }
            catch(Exception ex)
            {
                throw new Exception("SQLConnection Error: " + ex.Message);
            }
        }
        #endregion

        #region RetornarConn
        public string getconnectionstring(string DB)
        {
            try
            {
            SqlServer=ConfigurationManager.AppSettings["SqlServer"].ToString();
            SqlUser=ConfigurationManager.AppSettings["SqlUser"].ToString();
            SqlPassword=ConfigurationManager.AppSettings["SqlPassword"].ToString();
            SqlDataBase=DB;
            NTaut=ConfigurationManager.AppSettings["NTAuthentication"].ToString();
            return connectionstringcustom();
            }
            catch (Exception ex)
            {
                throw new Exception("Error en el metodo connection string: " + ex.Message);
            }
        }
        #endregion
    }
}
```

App.config (Valores asignados según tutorial)

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="SqlServer" value="Su maquina"/>
    <add key="SqlUser" value="sa"/>
    <add key="SqlPassword" value="pass"/>
    <add key="NtAuthentication" value="False"/>
  </appSettings>
</configuration>
```

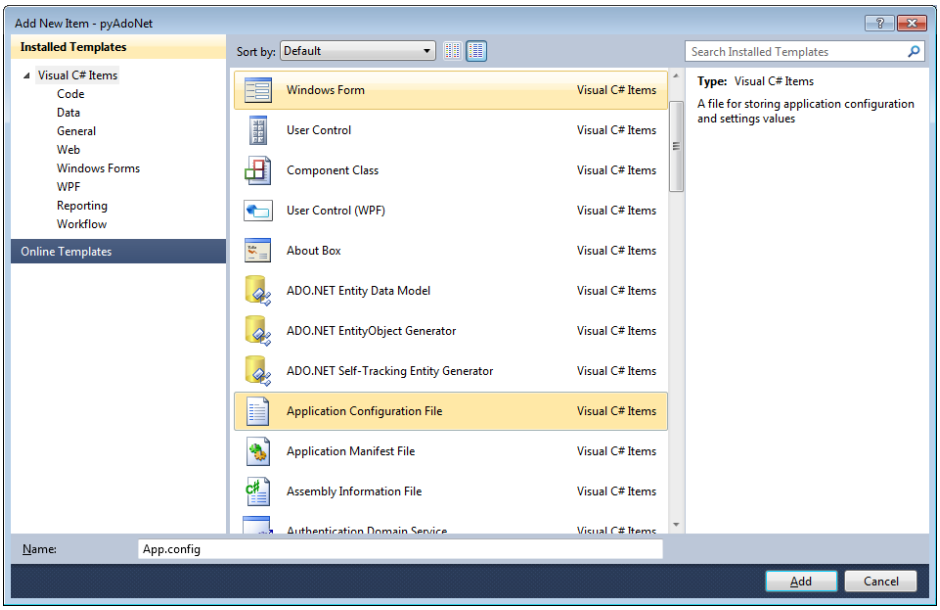
App.config (Valores asignados según la máquina de laboratorio, para autenticación Windows)

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="SqlServer" value="IND007S"/>
    <add key="SqlUser" value="Carlos"/>
    <add key="SqlPassword" value="JB2468"/>
    <add key="NtAuthentication" value="True"/>
  </appSettings>
</configuration>
```

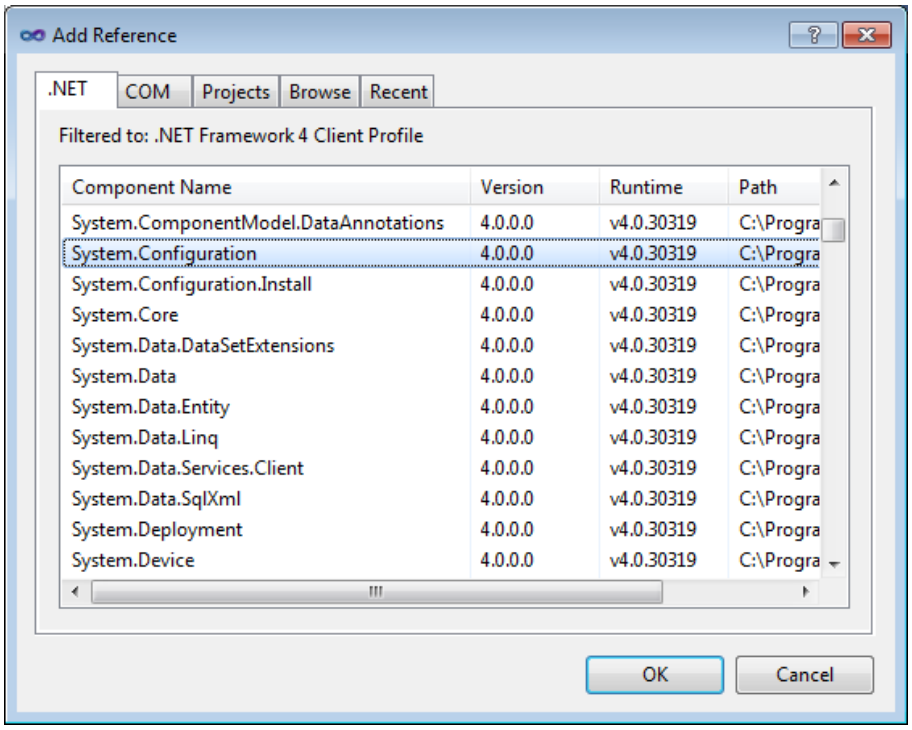
App.config (Valores asignados según la máquina de laboratorio, para autenticación SQL Server)

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="SqlServer" value=" IND007S "/>
    <add key="SqlUser" value=" Carlos "/>
    <add key="SqlPassword" value=" JB2468"/>
    <add key="NtAuthentication" value="False"/>
  </appSettings>
</configuration>
```

Adicionar un Item: **Application Configuration File → App.config**



Adicionar un **Reference: System Configuration**,



Conexión a la base de datos Northwind:

Data Source=.;Initial Catalog=Northwind;User ID=USER01

Ejemplo de ejecución de algunas sentencias SQL:

Abrir la base de datos en SQL Server:

```
use Northwind

select * from Pedidos

select IdPedido, IdCliente, PaísDestinatario from Pedidos

select IdPedido, IdCliente, FechaPedido, FechaEntrega, FechaEnvío,
PaísDestinatario from Pedidos

select FormaEnvío, Cargo, Destinatario, CódPostalDestinatario,
      CiudadDestinatario, PaísDestinatario
      from Pedidos

select * from Productos

select IdProducto, IdProveedor, IdCategoría, NombreProducto,
      CantidadPorUnidad, PrecioUnidad
      from Productos

select IdProducto, NombreProducto, CantidadPorUnidad, PrecioUnidad,
      Suspendido
      from Productos

select IdProducto, NombreProducto, CantidadPorUnidad, PrecioUnidad,
      UnidadesEnExistencia
      from Productos
      where UnidadesEnExistencia = 0

select IdProducto, NombreProducto, CantidadPorUnidad, PrecioUnidad,
      UnidadesEnExistencia
      from Productos
      where UnidadesEnExistencia != 0

select IdProducto, NombreProducto, CantidadPorUnidad, PrecioUnidad,
      Suspendido
      from Productos
      where Suspendido = 0

select IdProducto, NombreProducto, CantidadPorUnidad, PrecioUnidad,
      Suspendido
      from Productos
      where Suspendido = 1

select IdProducto, NombreProducto, CantidadPorUnidad, PrecioUnidad,
      NivelNuevoPedido
      from Productos
      order by NivelNuevoPedido

select IdProducto, NombreProducto, CantidadPorUnidad, PrecioUnidad,
      NivelNuevoPedido
      from Productos
      where NivelNuevoPedido != 0
      order by NivelNuevoPedido

select IdProducto, NombreProducto, CantidadPorUnidad, PrecioUnidad,
      NivelNuevoPedido
      from Productos
      order by NivelNuevoPedido
```

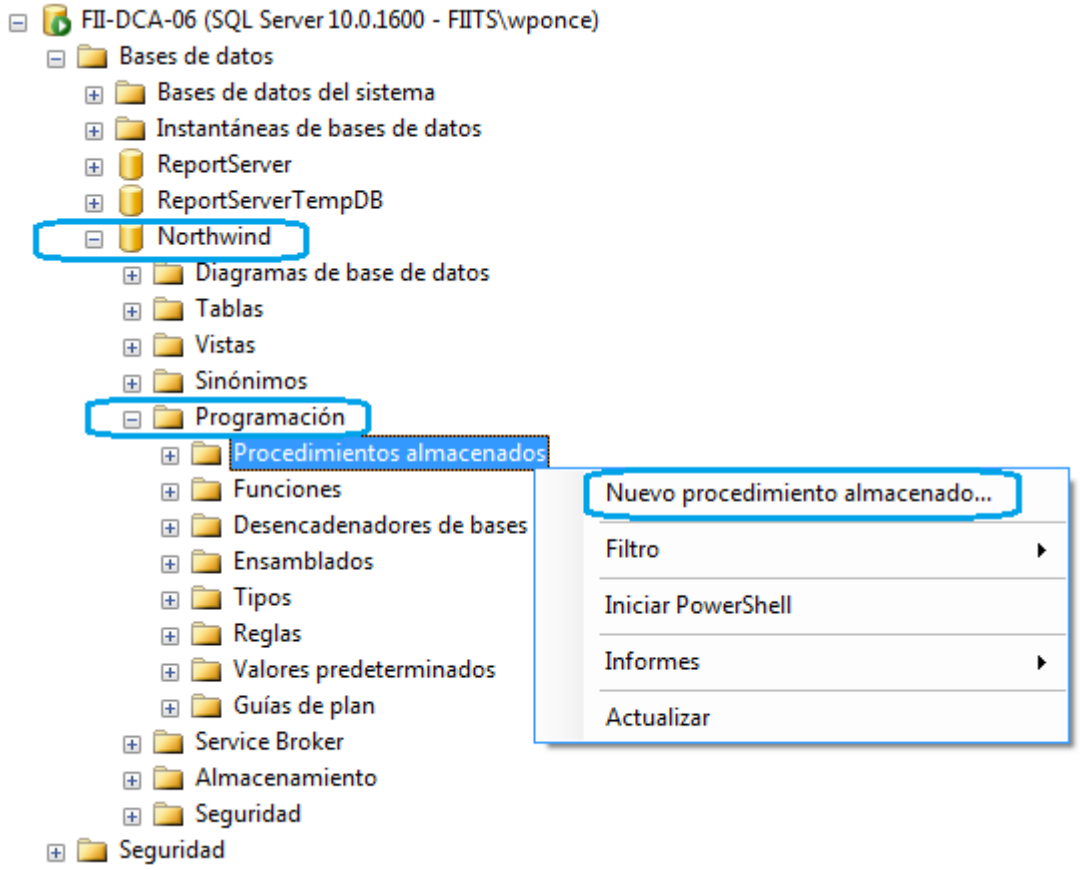
```
select * from [Compañías de envíos]

select Tratamiento,Nombre, Apellidos, Cargo, Jefe from Empleados order by Cargo
```

Por su cuenta realizar otras consultas SQL:

Crear un procedimiento almacenado:

Ir a **SQLServer**, seleccionar la BD: **Northwind** / **Programación** / **Procedimientos almacenados**

Dar **clic derecho** al mouse, seleccionar: **Nuevo procedimiento almacenado**…



Como resultado, SQL Server crea una nueva consulta y nos entrega el siguiente modelo de cómo crear un procedimiento almacenado:

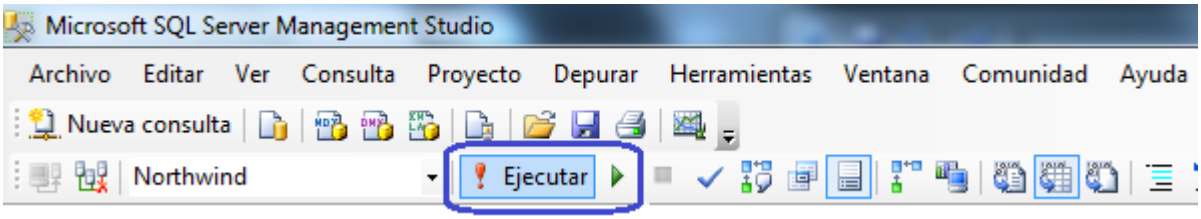
```
-- ================================================
-- Template generated from Template Explorer using:
-- Create Procedure (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
-- ================================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:        <Author,,Name>
-- Create date: <Create Date,,>
-- Description:   <Description,,>
-- =============================================
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
      -- Add the parameters for the stored procedure here
```

```
      <@Param1, sysname, @p1> <Datatype_For_Param1, , int> =
<Default_Value_For_Param1, , 0>,
      <@Param2, sysname, @p2> <Datatype_For_Param2, , int> =
<Default_Value_For_Param2, , 0>
AS
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON;

    -- Insert statements for procedure here
      SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

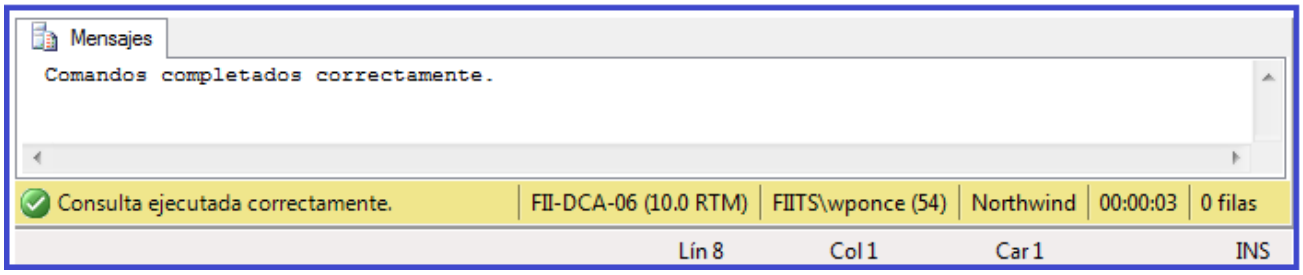
Hacemos las siguientes modificaciones y obtenemos:

```
-- ============================================================
-- Plantilla genera a partir de Explorador de plantillas usando:
-- Crear procedimiento (Nuevo menú) .SQL
--
-- Utilizar las Especificar valores para parámetros de plantilla
-- Comando (Ctrl-Shift-M) para llenar en el parámetro
-- valores por debajo.
--
-- Este bloque de comentarios no se incluirá en
-- La definición del procedimiento.
-- ============================================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================================
-- Autor  : Ing. Wiler Arturo Ponce Benites>
-- Fecha  : 12/06/2014
-- Función: Mostrar todos los reditros de la tabla Pedidos
-- ============================================================
CREATE PROCEDURE getpedidos
AS
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON;
    -- Insert statements for procedure here
      SELECT * FROM Pedidos
END
GO
```

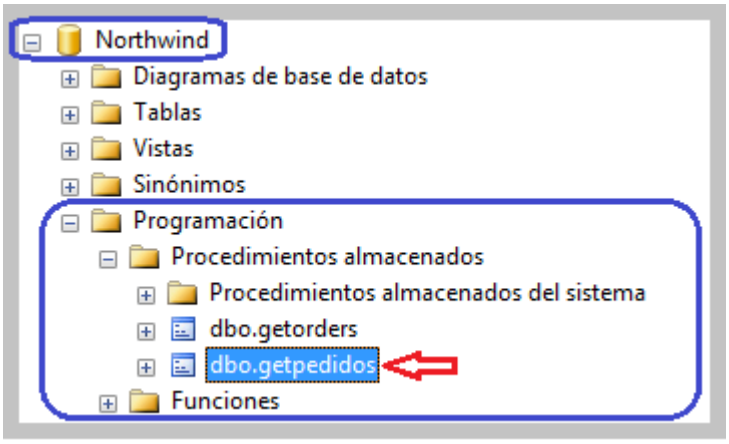
Lo seleccionamos y luego ejecutamos, haciendo clic en el botón Ejecutar:



Nos da el mensaje el siguiente mensaje:



Luego vamos al explorador de objetos, desplegamos, previamente actualizar y podemos observar el nuevo objeto creado:



Con esto ya tenemos terminado el proyecto.

Regresamos a Visual Studio – Visual C# y ejecutamos el proyecto y obtenemos el siguiente resultado: