

# Clase 9 - No Presencial. Por Guillermo Guastavino

## ABM en ADO .NET y trabajo práctico

Hola a todos!.

Esta clase, como les comenté antes, ya nos metemos de lleno en ADO .NET y en aprender otras cosas que necesitamos para el proyecto.

Les adjunté un rar con un proyecto (ABM 1), un script de sql (abm.sql) y un archivo txt (ip.txt).

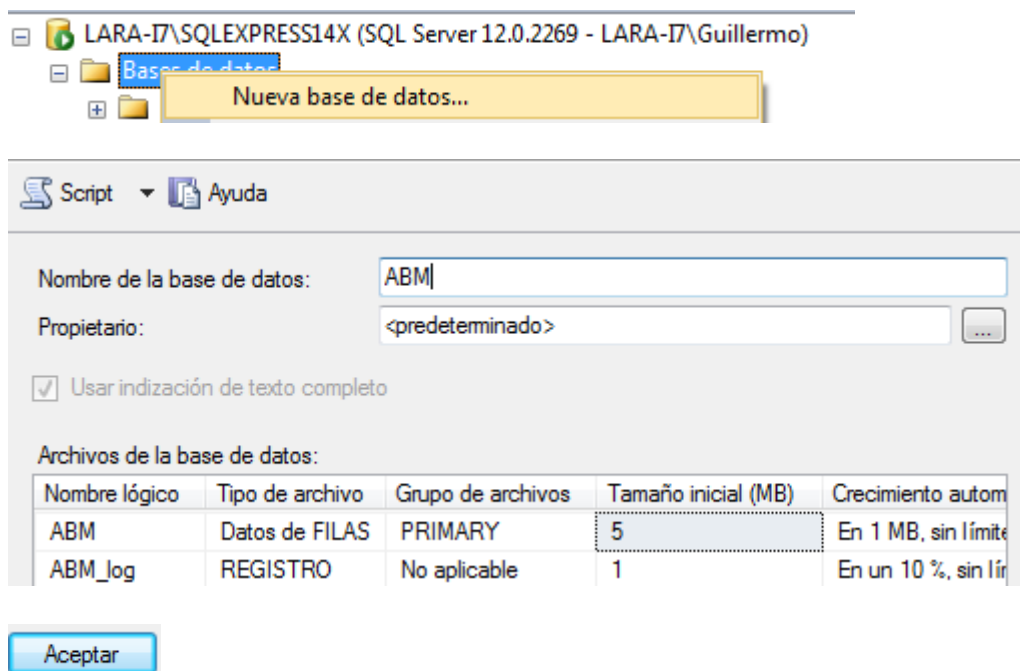
Les había prometido enseñarles lo mismo que yo uso y que fui desarrollando. Cuando entiendan lo que les dí y lo hagan “suyo”, me alcanzaron, y ojalá que me superen, tienen todo el potencial y toda la voluntad para ser muy exitosos.

El proyecto es mínimo, pero hace muchas tareas y contiene algunas de las rutinas copiadas de las que hice para mis sistemas.

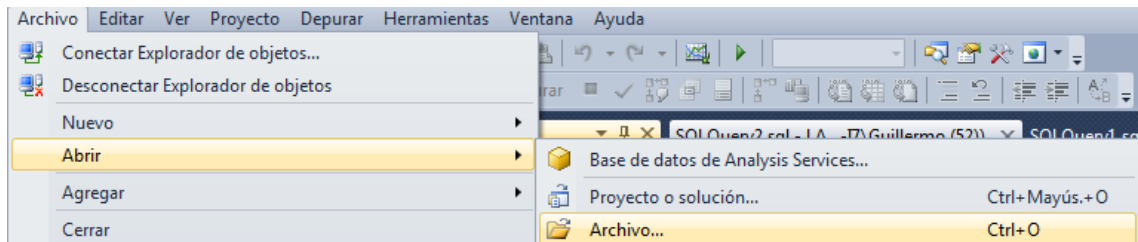
Es un ABM de profesores, sacado de un sistema mío para un instituto. Muy adrede es de profesores y no de clientes, y menos con la estructura de Customers de Northwind. La idea es que lo entiendan y que luego lo adapten todo, para poder cargar clientes desde Customers en vez de profesores. Si pueden cambiar de objeto (profesores a clientes) y de base de datos, van a poder entonces, utilizar éste ABM en cualquier proyecto de ustedes y adaptarlo a cualquier objeto: clientes, productos, proveedores etc.

POR FAVOR: como en todas las clases, entiendo que tienen clara las clases anteriores... si tenés dudas, revé la clase 8 de ADO, y si es necesario las anteriores de SQL.

Primero creé la base ABM en SQL Server 2014:



Ahora abrí el archivo abm.sql:

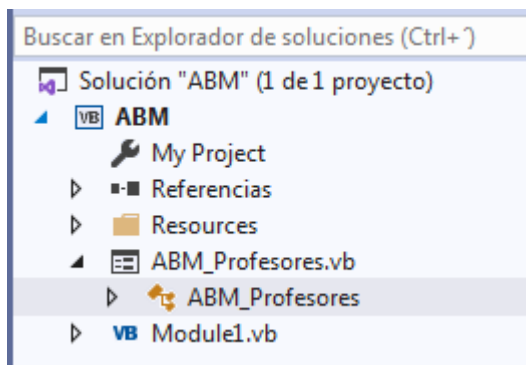


Y busqué la ruta en donde descomprimiste el rar que te pasé, y abrí abm.sql

El script, crea la tabla Profesores y una vista que permite que al dar el número del profesor (ustedes deberán luego que funcione para el ID que agregaron a Customers la clase pasada), devuelve el apellido, una coma y el nombre del profesor.

Ejecutá el script. Ahora tenés una tabla Profesores en ABM, que está vacía.

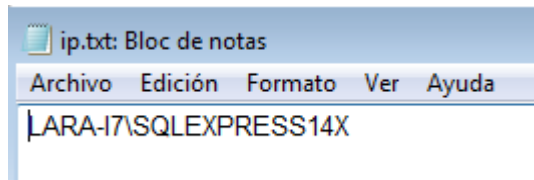
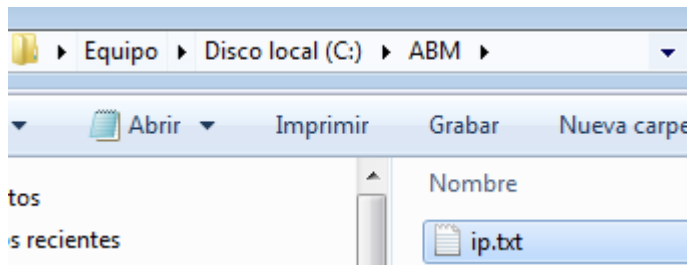
Abrió VS 2019, y abrí el proyecto que te pasé.



Es un ABM, un formulario que permite dar Altas, Bajas y Modificaciones. La visión clásica es un formulario para las Altas, otro para las Bajas y otro para las Modificaciones. Yo uso todo en uno, que es mucho más simple e intuitivo, y para replicar un ABM en otra cosa (como vas a hacer para transformarlo en un ABM de clientes sobre Customers de Northwind), es muchísimo más fácil usar 1 formulario que 3.

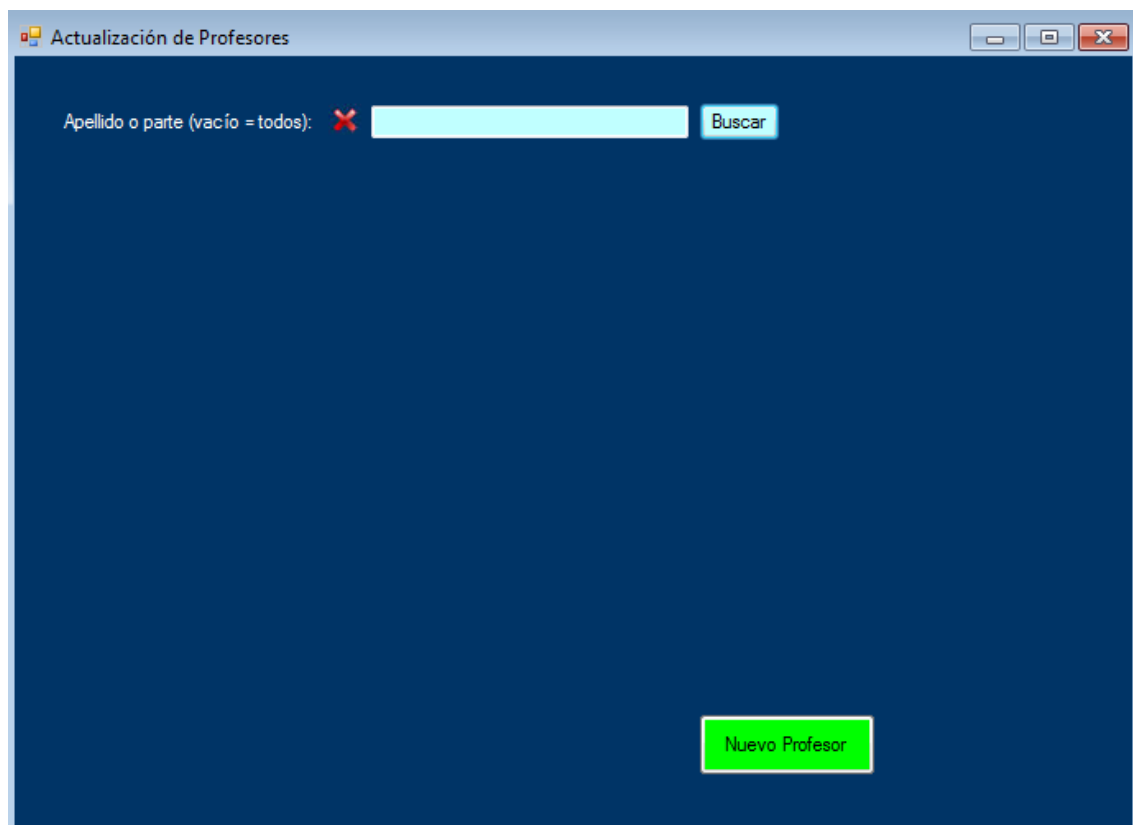
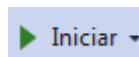
Hay que hacer algo más. Para que sea más modular, para usarlo en diferentes programas, y también para que si un cliente cambia el servidor, no tengas que recompilarle el programa porque cambió la ruta en la connection string; el programa la lee de la ruta C:\ABM\ip.txt

Así que creé la carpeta ABM en la raíz del C: y copié adentro el archivo ip.txt que te pasé. Editá ip.txt, y reemplazá mi ruta de servidor por la tuya (es lo mismo que aparece al abrir SQL y lo que pusiste en el proyecto ADO 1 de la clase pasada)



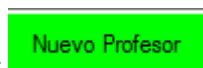
Reemplázalo por tu ruta. Lo mismo vas a hacer cuando armes un proyecto nuevo, o el cliente quiera cambiar el servidor.

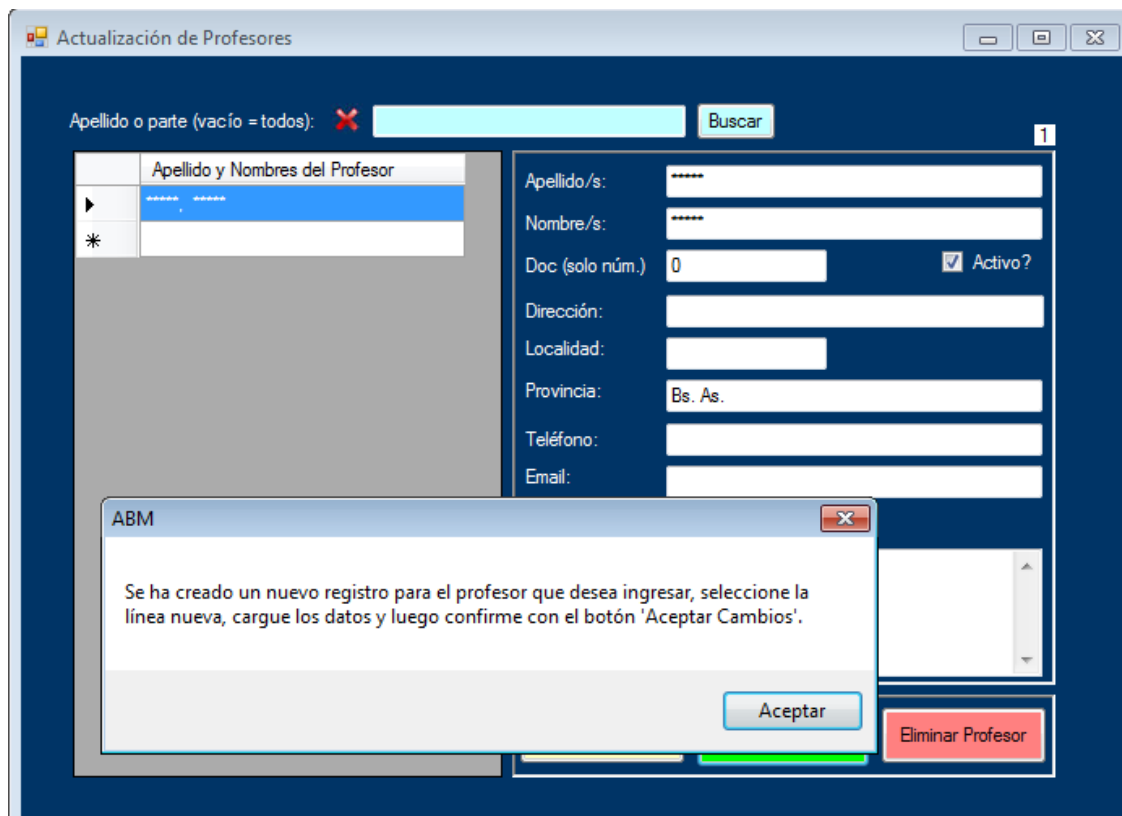
Ejecutá el programa para ver cómo funciona:



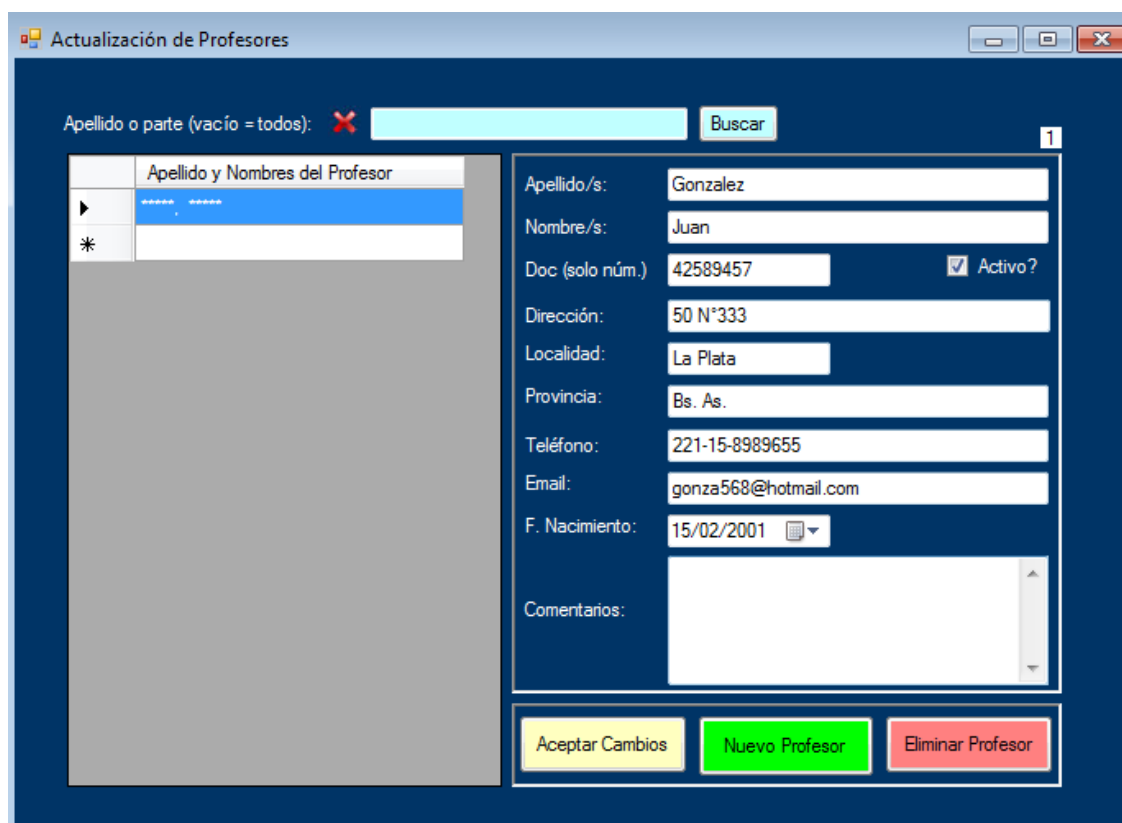
No muestra nada, porque no hay profesores cargados (recién hiciste la tabla).

Lo único posible entonces es apretar



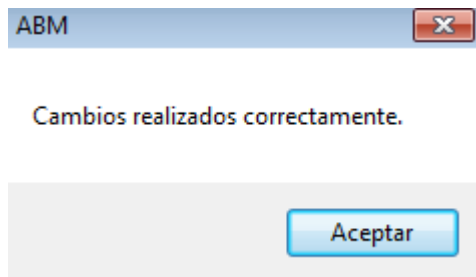


Te creó un registro nuevo en blanco, para que cargues a tu primer profesor



Como cambiamos los datos, hay que apretar

Aceptar Cambios



En la lista de profesores ahora aparece Gonzalez, Juan

NOTA: Abrás notado que los datos que pide para un profesor, son los mismos que pedirías para un cliente o un proveedor... así que para tu propio proyecto, podrías usar esto mismo, cambiando profesor por proveedor, cliente, alumno, paciente o lo que necesites...

Cargá un par más...

Si apretás **Buscar** con el textbox vacío, es “todos”, porque no ponés ningún filtro.

A window titled 'Actualización de Profesores' with a search bar and a 'Buscar' button. Below the search bar is a table with the header 'Apellido y Nombres del Profesor' and three rows: 'GONZALEZ, JUAN', 'LOPEZ, RICARDO', and 'RODRIGUEZ, ALBERTO'. To the right of the table is a form with fields for 'Apellido/s:', 'Nombre/s:', 'Doc (solo núm.)', 'Dirección:', 'Localidad:', 'Provincia:', 'Teléfono:', 'Email:', 'F. Nacimiento:', and 'Comentarios:'. The 'Apellido/s:' field is filled with 'GONZALEZ'. At the bottom right are three buttons: 'Aceptar Cambios', 'Nuevo Profesor', and 'Eliminar Profesor'.

Al hacer click en un profesor de la lista, se posiciona en ése

Actualización de Profesores

Apellido o parte (vacío = todos):  Buscar 3

	Apellido y Nombres del Profesor
	GONZALEZ, JUAN
▶	LOPEZ, RICARDO
	RODRIGUEZ, ALBERTO
*	

Apellido/s:   
 Nombre/s:   
 Doc (solo núm.):  ☒ Activo?  
 Dirección:   
 Localidad:   
 Provincia:   
 Teléfono:   
 Email:   
 F. Nacimiento:    
 Comentarios:

Si en el textbox para buscar, pongo G, me va a mostrar a todos los que empiezan con G, si pongo GON, sólo los que empiezan con Gon etc. al apretar Buscar.

Como hay uno sólo que empieza con Gon, me muestra a Gonzalez:

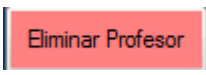
Actualización de Profesores

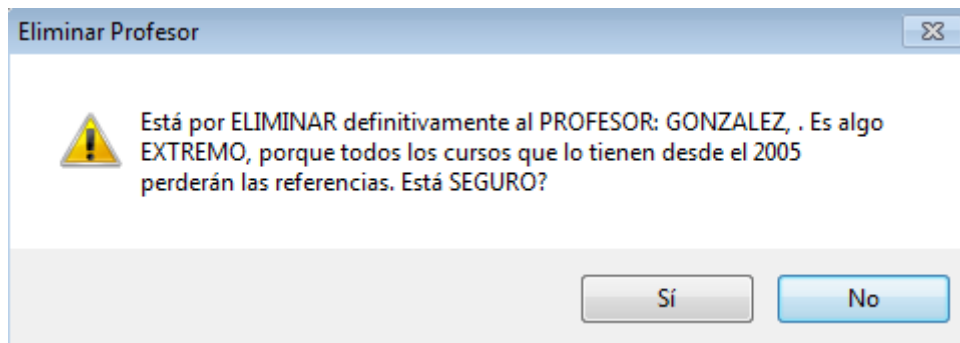
Apellido o parte (vacío = todos):  Buscar 1

	Apellido y Nombres del Profesor
▶	GONZALEZ, JUAN
*	

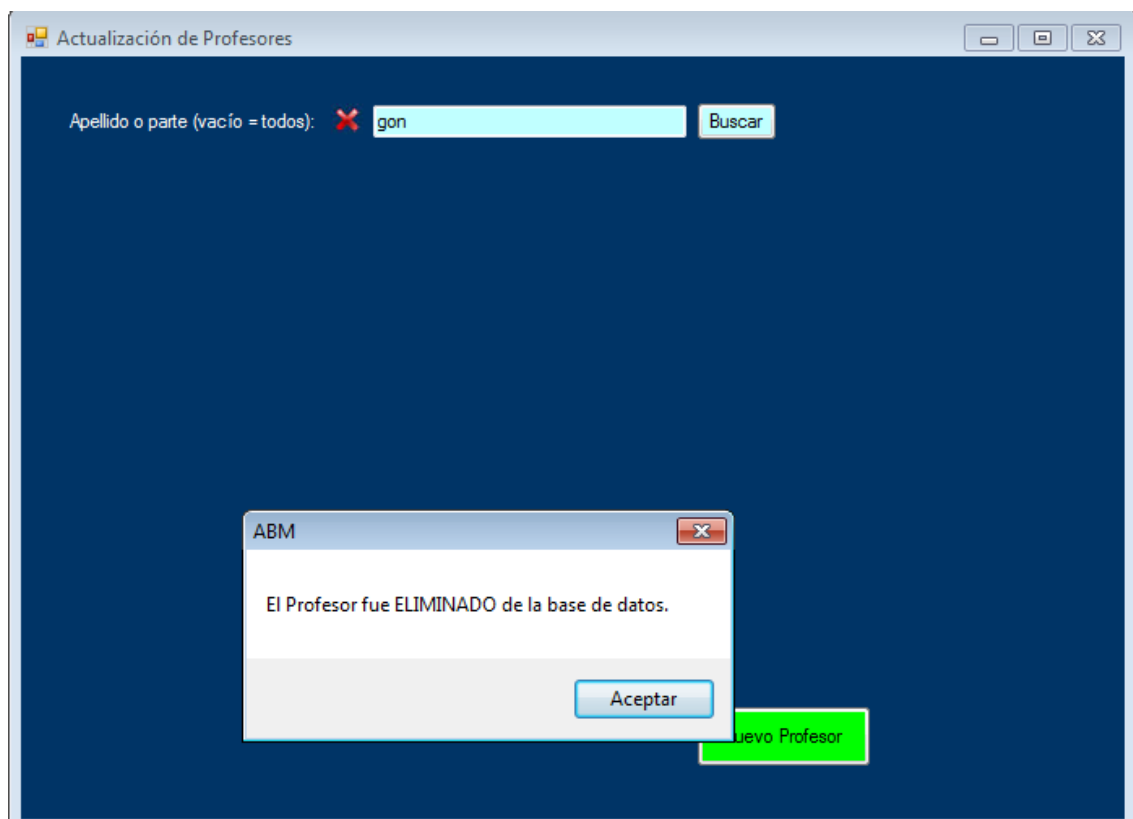
Apellido/s:   
 Nombre/s:   
 Doc (solo núm.):  ☒ Activo?  
 Dirección:   
 Localidad:   
 Provincia:   
 Teléfono:   
 Email:   
 F. Nacimiento:    
 Comentarios:

Esto me permite estar seguro que es al que quiero Eliminar (de la base...)

Apretá 



Lo eliminamos sin remordimientos...



Y mostraría a los otros que empiezan con gon (que no tenemos ninguno más)

Al buscar nada, o tocar la X para borrar lo que está escrito en el filtro, y buscar, me muestra los que quedan en orden alfabético (la lista se llena con la vista que hicimos en SQL)

Actualización de Profesores

Apellido o parte (vacío = todos): ✖

	Apellido y Nombres del Profesor
▶	LOPEZ, RICARDO
	RODRIGUEZ, ALBERTO
*	

Apellido/s:

Nombre/s:

Doc (solo núm.)  ☒ Activo?

Dirección:

Localidad:

Provincia:

Teléfono:

Email:

F. Nacimiento:

Comentarios:

Cerramos y vamos a ver el código

El código es lo más simple posible, hay mucho para mejorar y desarrollar por ustedes, pero lo poco que tiene es, como vieron, sólido

Abrás notado que hay un módulo llamado Module1 (o sea con el nombre por defecto, podrías llamarlo Rutinas), en el que guardo las funciones y procedimientos generales, que me sirven para éste y otros proyectos. Cuando voy a armar un proyecto nuevo, puedo copiar el archivo en el nuevo proyecto y agregarlo. También podría crear una clase, en donde todas las funciones y subs son los métodos de la clase (o varias clases...), compilarlo como DLL y usarlo ya compilado en todos los proyectos.

Agregué unas pocas rutinas para éste proyecto y las cosas que le vamos a ir agregando al “grande”.

VB

```
Imports System.IO
Imports System.Data.SqlClient
Imports System.Net.Mail
```

C#



```

using System;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;

```

Ya conocemos el data.sqlClient, que me permite acceder a las clases de ADO. System.IO son las clases de Input Output, o sea las de entradas y salidas, para manejo de archivos. Me habilita lo que necesito para la función Leerarchivo, y poder leer el archivo ip.txt con la ruta del servidor.

System.Net.Mail, son clases para enviar y recibir mails. No agregue ahora funciones que la usen.

VB

```

Function Leerarchivo(ByVal archivo As String) As String
    'en ip.txt poner la instancia de SQL p. ej: LARA-I7\SQL20122
    If File.Exists("c:\ABM\ip.txt") = True Then
        Dim SR As StreamReader = File.OpenText("c:\ABM\ip.txt")
        Dim Line As String = SR.ReadLine()
        SR.Close()
        Return Line
    End If
End Function

```

C#

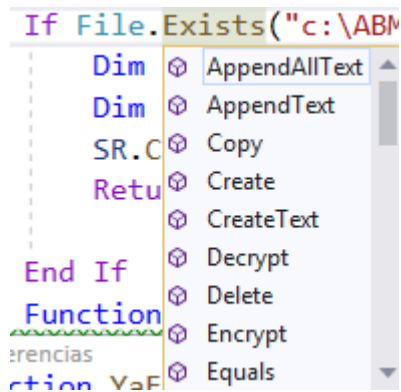
```

public string LeerHostDb()
{
    // En le archivo hostDb.txt poner la instancia de SQL p. ej: BAUBYTE-NOTE\SQLEXPRESS14
    if (File.Exists(@"C:\Connections\hostDb.txt") == true)
    {
        // Abrimos el Archivo y Guardamos el contenido en SR
        StreamReader SR = File.OpenText(@"C:\Connections\hostDb.txt");
        // Leemos las líneas hasta el enter y guardamos en Line
        string Line = SR.ReadLine();
        // Cerramos el archivo
        SR.Close();
        // Devolvemos la línea que leida
        return Line;
    }

    return default;
}

```

La clase File (exploré lo que te muestra al poner el .:



Tiene muchos métodos más para manejo de archivos... Exists y una ruta, devuelve verdadero, si el archivo de la ruta existe. Como ip.txt existe porque lo pegamos hace un rato, entonces devuelve verdadero y entra en el IF para leerlo.

```
Dim SR As StreamReader = File.OpenText("c:\ABM\ip.txt")
Dim Line As String = SR.ReadLine()
SR.Close()
```

Un stream, es un string potenciado... porque puede albergar texto, un word, un video, un archivo de audio...

Creo un objeto SR del tipo StreamReader, que va a contener el texto que esté del otro lado del igual, sólo que no hay texto, sino otra vez la clase File, con el método OpenText, que abre (no lee) al ip.txt.

El string Line, va a leer una línea (hasta el ENTER o el final de archivo). SR apuntaba al archivo abierto ip.txt, entonces el método ReadLine, leerá una sólo línea del archivo (la única que hay, que tiene la ruta del servidor)

SR que apunta la archivo, lo cierra usando el método CLOSE.

Finalmente la función devuelve Line con la ruta del servidor en Return Line.

## VB

```
Function YaExisteSQL(ByVal sql As String) As Boolean
    'sele pasa un select de sql que en teoria busca algo, por ejemplo un numero de cheque, si hay registros la fun devuelve
    'true, estilo "el cheque ya existe", si no esta, devuelve false
    Dim ar As String
    Dim con As New SqlConnection("data source=" & CStr(Leerarchivo(ar)) & "; initial catalog=abm; integrated security=true")
    Dim da1 As New SqlDataAdapter(sql, con)
    Dim ds1 As New DataSet
    da1.Fill(ds1, "afidesc")
    If ds1.Tables("afidesc").Rows.Count < 1 Then
        Return False
    Else
        Return True
    End If
End Function
```

## C#

```
// Para comprobar si existe en la DB
0 referencias
public bool YaExisteSql(string sql)
{
    // Le pasamos un SELECT de SQL que en teoria busca algo, por ejemplo un numero de cheque, si hay registros la Funcion devuelve
    // True, estilo "el cheque ya existe", si no esta, devuelve False
    SqlDataAdapter dataAdapter = new SqlDataAdapter(sql, Conectar());
    DataSet dataSet = new DataSet();
    dataAdapter.Fill(dataSet, "existe");
    return dataSet.Tables["existe"].Rows.Count >= 1;
}
```

De la clase pasada pueden entender casi todo lo de ésta función. Lo que hace es tomar una sentencia Select, que debería apuntar a un único registro que pudiera existir o no, por ejemplo “Select Ncli from Customers where Ncli=100”. Entonces ejecuta el select, y si hay un Ncli con número 100 (o sea... me devolvió 1 registro la consulta, en vez de nada), es que el cliente 100 está en la tabla Customers.

Para saber cuantos registros me devolvió, miro el método Rows (filas). Count (cantidad) del Dataset Ds1. (Si no recordás qué es un dataset, o un sql dataadapter etc., por favor mirá ya mismo la clase 8.)

Mirá que en la conection string, en vez de la ruta del servidor como vimos en la clase pasada, invoca llerarchivo, que la devuelve.

## VB

```
Function VNum(ByVal NTexto As String) As Decimal
    'transforma un texto con algo, que puede ser un numero con coma o punto o perro, y devuelve un valor decimal siempre
    Return CDec(Val(NTexto.Trim.Replace(",", ".")))
End Function
```

## C#

```
// Para convertir los String o Num
3 referencias
public decimal Vnum(string NTexto)
{
    // transforma un texto con algo, que puede ser un numero con coma o punto o perro, y devuelve un valor decimal siempre
    return System.Convert.ToDecimal(Conversion.Val(NTexto.Trim().Replace(",", ".")));
}
```

VNUM es una función que desarrollé hace muchos muchos años, inclusive para otros lenguajes, y hay ex alumnos profesionales desde hace muchos años que aún la usan y me lo recuerdan. Convierte lo que venga en Ntexto (generalmente desde un textbox en el que pudiste escribir un número pero también cualquier cosa), en un número. También unifica el símbolo decimal, pudiendo usarse , o .

El resultado lo convierte a decimal (porque VAL devuelve el valor de un texto, pero en formato doble). Sirve para validar un textbox o un string, en donde se supone que hay un número entero o decimal, y con los decimales después de una coma o de un punto.

## VB

```
Function SQL_Accion(ByVal Sql_de_accion As String) As Boolean
    'Ejecuta la consulta de accion 'sql_de_accion' abriendo la conexion automaticamente
    'se da cuenta si es de insert, update o delete, porque mira dentro de la sentencia que se le pasa
    'devuelve true si se pudo hacer, y false si hubo algún error
    Dim ar As String
    ' Dim conn As New SqlConnection(IIf(Leerarchivo(ar).indexOf("ULTRA") >= 0, "data source=ULTRABOOK\SQL2000;user id=sa;passw
    Dim conn As New SqlConnection("data source=" & CStr(Leerarchivo(ar)) & "; initial catalog=abm; integrated security=true")

    Dim adapter As New SqlDataAdapter, salida As Boolean = True

    Try
        conn.Open()
        If Sql_de_accion.ToUpper.IndexOf("INSERT") Then
            'MsgBox(Sql_de_accion)
            adapter.InsertCommand = New SqlCommand(Sql_de_accion, conn)
            adapter.InsertCommand.ExecuteNonQuery()
        Else
            If Sql_de_accion.ToUpper.IndexOf("UPDATE") Then
                adapter.UpdateCommand = New SqlCommand(Sql_de_accion, conn)
                adapter.UpdateCommand.ExecuteNonQuery()
            Else
                If Sql_de_accion.ToUpper.IndexOf("DELETE") Then
                    adapter.DeleteCommand = New SqlCommand(Sql_de_accion, conn)
                    adapter.DeleteCommand.ExecuteNonQuery()
                Else
                    'esta mal la sintaxis porque no hay ni insert, ni delete ni update
                    salida = False
                End If
            End If
        End If
    Catch ex As Exception
        MsgBox(ex.Message)
        salida = False
    End Try
    conn.Close()
    Return salida
End Function
```

## C#

```

public bool SqlAccion(string SqlDeAccion)
{
    // Ejecuta la consulta de accion 'SqlDeAccion' abriendo la conexion automaticamente
    // se da cuenta si es de insert, update o delete, porque mira dentro de la sentencia que se le pasa
    // devuelve true si se pudo hacer, y false si hubo algún error
    // Para despues idicarle las consultas o un store procedure de SQL
    SqlConnection connection = Conectar();
    SqlDataAdapter dataAdapter = new SqlDataAdapter();
    // Para ver si hubo o no error en la ejecucion
    bool salida = true;
    try
    {
        connection.Open();
        // Buscamos en el contenido con IndexOf si tiene algunas de las acciones
        if ((SqlDeAccion.ToString().ToUpper().IndexOf("INSERT")) != -1)
        {
            dataAdapter.InsertCommand = new SqlCommand(SqlDeAccion, connection);
            dataAdapter.InsertCommand.ExecuteNonQuery();
        }
        else if ((SqlDeAccion.ToUpper().IndexOf("UPDATE")) != -1)
        {
            dataAdapter.UpdateCommand = new SqlCommand(SqlDeAccion, connection);
            dataAdapter.UpdateCommand.ExecuteNonQuery();
        }
        else if ((SqlDeAccion.ToUpper().IndexOf("DELETE")) != -1)
        {
            dataAdapter.DeleteCommand = new SqlCommand(SqlDeAccion, connection);
            dataAdapter.DeleteCommand.ExecuteNonQuery();
        }
        else
        {
            // esta mal la sintaxis porque no hay ni insert, ni delete ni update
            salida = false;
        }
    }
    // Capturamos el Error
    catch (Exception ex)
    {
        // Mostaremos el Error y Devolvemos False en la salida es decir que todo salio mal
        Interaction.MsgBox(ex.Message);
        salida = false;
    }

    // Cerramos la Conexion y Devolvemos True en la salida es decir que todo salio bien
    connection.Close();
    return salida;
}
}

```

Es una función a la cual le paso una Consulta de Acción en transact SQL, y me devuelve true si se hizo bien, o false si hubo un error. Las consultas de acción, son las que hacen algo... UPDATE modifica un registro o fila, INSERT inserta un nuevo registro, y DELETE lo borra.

Así que los botones del formulario, llamarán a esta función para insertar, modificar o eliminar profesores.

Para cada acción, hay un método específico de la clase SqlDataAdapter (clase 8). Instanciamos adapter como del tipo SqlDataAdapter, entonces:

Adapter.Insertcommand va a hacer un INSERT  
 Adapter.Updatecommand va a hacer un UPDATE  
 Y Adapter.Deletecommand va a hacer un DELETE

El ExecuteNonQuery, indica que estoy ejecutando un comando en ves de una Query (una consulta).

Entonces, el if toma la sentencia que le pasaste, la convierte a mayúsculas (.toupper), y luego busca si tiene adentro (con .indexof) la palabra “INSERT”, “UPDATE” o “DELETE”, para ejecutar el comando que corresponda.

Si falla, la variable salida se pone a false y es lo que se devuelve para indicar que hubo error.

VB

```
Function NumSQL(ByVal numero As String) As String
    'Recibe un número desde un textbox por ejemplo, lo verifica como número válido,
    'y luego le cambia la coma por punto para que sea válido en una sentencia de sql,
    'luego lo devuelve
    Return CStr(VNum(numero)).Trim.Replace(",", ".")
End Function
```

---

C#

```
public string NumSql(string numero)
{
    // Recibe un número desde un textbox por ejemplo, lo verifica como número válido,
    // y luego le cambia la coma por punto para que sea válido en una sentencia de sql,
    // luego lo devuelve
    return System.Convert.ToString(Vnum(numero)).Trim().Replace(",", ".");
}
```

Los números con decimales, en una sentencia por ejemplo de UPDATE, deben tener punto en vez de coma decimal.

NumSQL convierte un NUMero al formato de SQL.

Primero lo valido y corrijo con Vnum, luego lo convierto a string con CStr (convertir a string, primeras clases) y le reemplazo la coma por un punto.

Devuelvo el número convertido a string y formateado para poder meterlo directamente en una sentencia INSERT o UPDATE sin ningún problema.

VB

```
Function RellenaNum(ByVal numero As Integer, ByVal cantidad As Integer) As String
    'Rellena con 0s adelante el numero. Ideal para dias y meses:
    'RellenaNum(3,2)---> "03" RellenaNum(3,4)--->"0003"
    Dim snum As String = CStr(numero).Trim
    Return "00000000000000000000".Substring(0, cantidad - snum.Length) & snum
End Function
```

C#

```
// Para Rellenar los Numeros con la cantidad que necesitamos
3 referencias
public string RellenaNum(int numero, int cantidad)
{
    // Rellena con 0s adelante el numero. Ideal para dias y meses:
    // RellenaNum(3,2)---> "03" RellenaNum(3,4)--->"0003"
    string snum = System.Convert.ToString(numero).Trim();
    return "00000000000000000000".Substring(0, cantidad - snum.Length) + snum;
}
```

El ejemplo en verde (que son recordatorios que me escribí como comentarios), dice todo lo que se necesita. Sirve por ejemplo para armar el número de una factura. Le paso 235 y cantidad 8 y me devuelve "00000235"

Substring, obtiene un pedacito de un string, empezando en la posición 0 (la primera porque es una colección), y con la cantidad de caracteres que indica su segundo parámetro. Acá le pego adelante un monton de ceros, y recorto la cantidad elegida desde la derecha (.length devuelve la longitud o cantidad de caracteres de un string). Entonces como 235 tiene 3 caracteres, recorto 5 ceros para ponerle adelante.

## VB

```
Function FechaSQL(ByVal fecha As Date) As String
    'Devuelve fecha en el formato 'AAAAMMDD'
    Return "" & RellenaNum(Year(fecha), 4) & RellenaNum(Month(fecha), 2) & RellenaNum(fecha.Day, 2) & ""
End Function
```

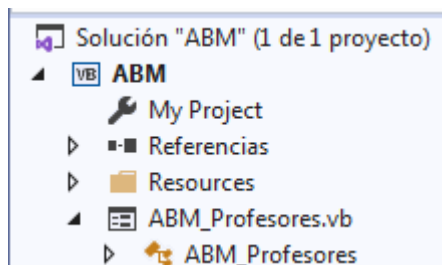
## C#

```
// Para Formatear las Fechas en el formato correcto para SQL
0 referencias
public string FechaSql(DateTime fecha)
{
    // Devuelve fecha en el formato 'AAAAMMDD'
    return "" + RellenaNum(fecha.Year, 4) + RellenaNum(fecha.Month, 2) + RellenaNum(fecha.Day, 2) + "";
}
```

Parecido a NunSQL, respecto al formato que deben tener, en este caso las fechas, al ser agregadas a un INSERT o un UPDATE. El formato americano es MM-DD-AAAA, mes primero, pero si lo armo al revés es AAAAMMDD, con el mes en el medio... entonces lo convierto a AAAAMMDD y listo.

Year devuelve el año, Month el mes y Day el día de una fecha. Usando rellenarum, les pongo el cero adelante si el mes o el día es de un solo dígito.

Vayamos ahora a ABM\_Profesores.vb



```
Dim con As New SqlConnection("data source=" & CStr(Leerarchivo(ar)) & "; initial catalog=abm; integrated security=true")
```

Con (clase 8) apunta al servidor y a la base, para todo el formulario

Lo que sigue en verde, es el SQL que ejecutaste, que lo metí también como comentario en el formulario, para tenerlo a mano como referencia.

```

Sub buscar(ByVal condicion As String)
    Dim da As New SqlDataAdapter("SELECT TOP (100) PERCENT nprof,apeynom from profesores_búsqueda where " & condicion & " order by apeynom", con)
    Dim ds As New DataSet
    da.Fill(ds, "Alumnos")
    If ds.Tables("Alumnos").Rows.Count = 0 Then

        DataGridView1.Visible = False

        pBotones.Visible = False
        pCampos.Visible = False
        lLegajo.Visible = False
    Else

        DataGridView1.DataSource = ds.Tables("Alumnos")
        DataGridView1.Refresh()
        DataGridView1.Visible = True

        lLegajo.Visible = True
    End If
End Sub

```

C#

```

public void buscar(string condicion)
{
    SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT TOP (100) PERCENT ID As id, CompanyName AS cliente," +
        " ContactName AS contacto FROM Customers WHERE " + condicion + " ORDER BY cliente", routines.Conectar());
    DataSet dataSet = new DataSet();
    dataAdapter.Fill(dataSet, "clientes");
    if (dataSet.Tables["clientes"].Rows.Count == 0)
    {
        gridClientes.Visible = false;
        pBotones.Visible = false;
        pCampos.Visible = false;
        lIdCliente.Visible = false;
    }
    else
    {
        gridClientes.DataSource = dataSet.Tables["clientes"];
        gridClientes.Refresh();
        gridClientes.Visible = true;
        lIdCliente.Visible = true;
    }
}

```

La función buscar, es casi igual a la que vimos en la clase 8, solo que usa Rows.Count para ver si devuelve o no registros. Si devuelve 0, pone la grilla a visible=false, para que no se vea si no tiene nada, y oculta a los botones menos al de dar de alta a un nuevo profesor.

La condición del where, es una variable que se le pasa por parámetro, para poder armarla afuera de la función.

```

Private Sub ABM_Profesores_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    buscar(" apeynom like '" & tApellido.Text & '%" ")
End Sub

```

C#

```

private void abmCustomers_Load(object sender, EventArgs e)
{
    buscar(" CompanyName LIKE '" + tBuscar.Text + '%" ');
}

```

Habíamos visto que el Load del formulario, es lo primero que se ejecuta al cargar el formulario. En éste caso, lo primero que hace es ejecutar la sub buscar para que haya algo. El like (de las clases de SQL), toma todo lo que empieza con lo que escriba antes de %, así “like ‘gon%’” va a ser todo lo que empieza con gon. Como al principio tApellido.Text está vacío, like ‘%’ equivale a “todo”, y muestra a todos los profesores.



```

Private Sub DataGridView1_RowEnter(ByVal sender
    FilaClick(e)
End Sub

```

---

Salta cuando te posicionás en una fila de la grilla (eligiendo a un profesor). Llama a FilaClick, pasándole e, que es un objeto que tiene información sobre la fila en la que hiciste el click al posicionarte.

```

Sub FilaClick(ByVal e As Object)
    Dim fila As Integer = e.RowIndex
    Dim tfila As String

    If IsNothing(DataGridView1.Rows(fila).Cells(0).Value) Then
        lLegajo.Text = "0"
        pBotones.Visible = False
        pCampos.Visible = False
        Exit Sub
    Else
        tfila = DataGridView1.Rows(fila).Cells(0).Value
        lLegajo.Text = tfila.ToString()
        CargarCamposAlumnos()
    End If
End Sub

```

---

C#

```

/**Para Obtener el numero de la fila seleccionada al ir pasando con las flechas*/
1 referencia
private void gridClientes_CellEnter(object sender, DataGridViewCellEventArgs e)
{
    FilaClick(e.RowIndex);
}
//Filtro de la Fila Seleccionada
2 referencias
public void FilaClick(int fila)
{
    string tfila;
    if (Information.IsNothing(gridClientes.Rows[fila].Cells[0].Value))
    {
        lIdCliente.Text = "0";
        pBotones.Visible = false;
        pCampos.Visible = false;
        return;
    }
    else
    {
        tfila = gridClientes.Rows[fila].Cells[0].Value.ToString();
        lIdCliente.Text = tfila.ToString();
        CargarCamposClientes();
    }
}

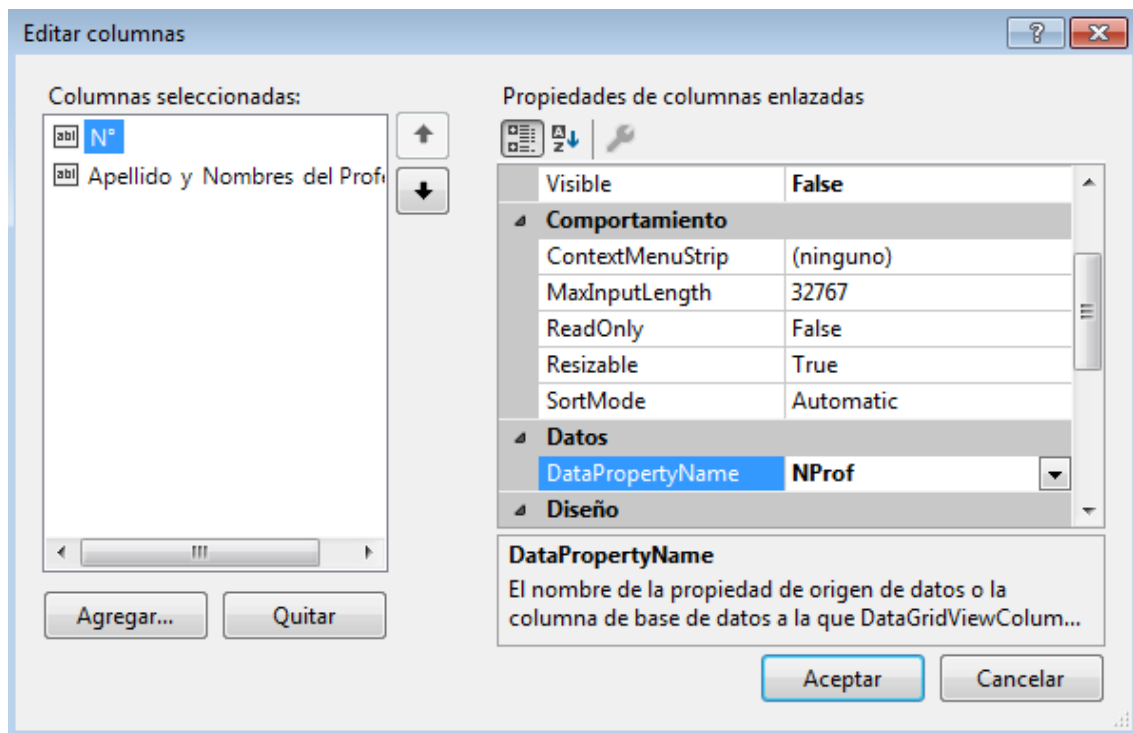
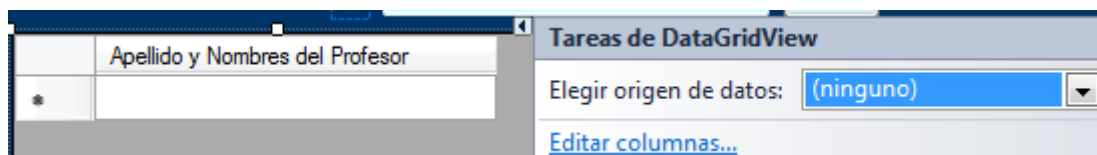
```

Entonces esta función se llama cada vez que apretás en una línea, cambiando de profesor. El objeto e, dijimos, que tenía toda la información de la línea, y precisamente, debemos leer esa información para saber el ID del profesor. Con el ID vamos a poder ir a buscar todos los datos de ése profesor y llenar los campos de la derecha con esos datos.

```
Dim fila As Integer = e.RowIndex  
Dim tfila As String
```

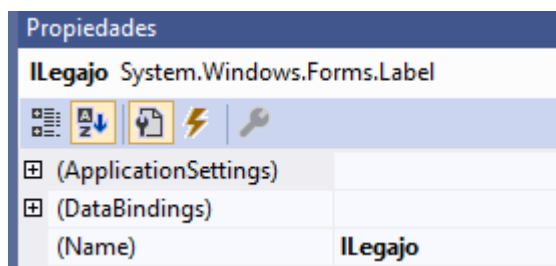
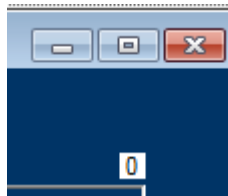
Una propiedad del objeto e que hemos obtenido, es RowIndex (Índice de la Fila o Número de fila). Ya te conté que el paquete de datos que se carga en un dataset, se asemeja a una especie de planilla de Excell, que empieza en la fila 0, y su primera columna, también es 0. Entonces la primera celda es la 0,0, que es en dónde se guarda el id del profesor. Si bien en la grilla sólo mostramos el apellido y nombre, acordate que leámos desde la vista profesores\_búsqueda, a nprof y apeunom, entonces nprof es la celda 0,0 y la que nos interesa.

Sobre la grilla apretá:



Mirá que la grilla también tiene esos dos campos, siendo el primero Nprof (está seleccionado en la foto). El primer dato de la foto es Visible false, o sea que nprof es la celda 0,0, pero no se ve, y ApeyNom está en la 0,1 y si se ve. Como estamos trabajando sobre una sólo fila, la fila seleccionada, ésta va a ser siempre la 0 (la única), mientras que la columna, apartir de 0, va apuntando a cada campo que agregamos.

Ya que estás mirando al formulario, fijate en el 0 que está arriba a la izquierda



Es una label llamada lLegajo, en la que guardaremos el nprof del profesor, cada vez que se invoque FilaClick

```
If IsNothing(DataGridView1.Rows(fila).Cells(0).Value) Then
```

Como fila tiene la fila seleccionada, Rows(fila).Cells(0).value devuelve el valor o dato que hay en la fila elegida, celda o columna 0. Como puede ser que apuntemos a nada si no hay nada en la grilla, preguntamos usando la funcion IsNothing (que devuelve true si “es nada” lo que hay adentro).

Si es “nada”, pone 0 en lLegajo, porque no se corresponde con ningún profesor,

```
If IsNothing(DataGridView1.Rows(fila).Cells(0).Value) Then
    lLegajo.Text = "0"
    pBotones.Visible = False
    pCampos.Visible = False
Exit Sub
```

Y oculta los botones. Exit Sub fuerza para salir de la sub, porque ya no hay más nada que hacer.

Por el else:

```

tfila = DataGridView1.Rows(fila).Cells(0).Value
lLegajo.Text = tfila.ToString()
CargarCamposAlumnos()
End If

```

Carga en la variable tfila el valor de la celda fila,0, o sea de Nprof. Convertido a string con .ToString, lo muestra en lLegajo, y llama a la función CargarCamposAlumnos(), que en realidad debería ser CargarCamposProfesor....

```

Sub CargarCamposAlumnos()
    If Val(lLegajo.Text) = 0 Then
        pBotones.Visible = False
        pCampos.Visible = False

        Exit Sub
    Else
        pBotones.Visible = True
        pCampos.Visible = True
        Dim da As New SqlDataAdapter("SELECT upper(ltrim(rtrim(isnull(apellidoprof,'****'))
        Dim ds As New DataSet
        da.Fill(ds, "Alumnos")
        TextBox1.Text = ds.Tables("Alumnos").Rows(0)("apellido")
        TextBox2.Text = ds.Tables("Alumnos").Rows(0)("nombres")
        TextBox3.Text = ds.Tables("Alumnos").Rows(0)("doc")

        TextBox4.Text = ds.Tables("Alumnos").Rows(0)("Dirección")
        TextBox5.Text = ds.Tables("Alumnos").Rows(0)("localidad")
        TextBox8.Text = ds.Tables("Alumnos").Rows(0)("provincia")
        TextBox6.Text = ds.Tables("Alumnos").Rows(0)("teléfonos")
        TextBox7.Text = ds.Tables("Alumnos").Rows(0)("email")
        CheckBox1.Checked = IIf(ds.Tables("Alumnos").Rows(0)("estado") = 0, False, True)

        TextBox12.Text = ds.Tables("Alumnos").Rows(0)("comentarios")

        DateTimePicker1.Value = ds.Tables("Alumnos").Rows(0)("fechanacimiento")
    End If
End Sub

```

C#

```

/**Carga los Datos del Cliente seleccionado en los campos*/
1 referencia
public void CargarCamposClientes()
{
    if (Conversion.Val(1IdCliente.Text) == 0)
    {
        pBotones.Visible = false;
        pCampos.Visible = false;
        return;
    }
    else
    {
        pBotones.Visible = true;
        pCampos.Visible = true;
        SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT UPPER(LTRIM(RTRIM(ISNULL(CompanyName,'*****'))))" +
            " AS cliente, UPPER(LTRIM(RTRIM(ISNULL(ContactName,'*****')))) AS contacto, LTRIM(RTRIM(ISNULL(ContactTitle,''))) AS cargo " +
            ", LTRIM(RTRIM(ISNULL(Address,''))) AS direccion, LTRIM(RTRIM(ISNULL(City,''))) AS ciudad, LTRIM(RTRIM(ISNULL(Region,''))) AS " +
            " AS localidad , LTRIM(RTRIM(ISNULL(PostalCode,''))) AS cp, LTRIM(RTRIM(ISNULL(Country,''))) AS pais, " +
            "LTRIM(RTRIM(ISNULL(Phone,''))) AS telefono, LTRIM(RTRIM(ISNULL(Fax,''))) AS fax FROM Customers WHERE ID=" +
            + Conversion.Val(1IdCliente.Text), routines.Conectar());

        DataSet dataSet = new DataSet();
        dataAdapter.Fill(dataSet, "clientes");
        tCliente.Text = dataSet.Tables["clientes"].Rows[0]["cliente"].ToString();
        tContacto.Text = dataSet.Tables["clientes"].Rows[0]["contacto"].ToString();
        tCargo.Text = dataSet.Tables["clientes"].Rows[0]["cargo"].ToString();
        tDireccion.Text = dataSet.Tables["clientes"].Rows[0]["direccion"].ToString();
        tLocalidad.Text = dataSet.Tables["clientes"].Rows[0]["localidad"].ToString();
        tCP.Text = dataSet.Tables["clientes"].Rows[0]["cp"].ToString();
        tCiudad.Text = dataSet.Tables["clientes"].Rows[0]["ciudad"].ToString();
        tPais.Text = dataSet.Tables["clientes"].Rows[0]["pais"].ToString();
        tTelefono.Text = dataSet.Tables["clientes"].Rows[0]["telefono"].ToString();
        tFax.Text = dataSet.Tables["clientes"].Rows[0]["fax"].ToString();
    }
}

```

Acá usamos el número del profesor, guardado en lLegajo.text, para ir a buscarlo y luego llenar todos los textboxes de la derecha con los distintos datos.

Primero pregunta si es 0 (acordate que si era Nothing lo ponía a 0), entonces si es 0, oculta todo y se va. Si no es 0, es porque tenemos a un profesor y hay que ir a buscar sus datos.

```

Dim da As New SqlDataAdapter("SELECT
upper(ltrim(rtrim(isnull(apellidoprof,'*****')))) as apellido,
upper(ltrim(rtrim(isnull(nombreprof,'*****')))) as nombres,isnull([documento-
Prof],0) as doc, ltrim(rtrim(isnull(domicilioprof,''))) as
dirección,ltrim(rtrim(isnull(localidadprof,''))) as
localidad,ltrim(rtrim(isnull(provinciaprof,''))) as
provincia,ltrim(rtrim(isnull(teléfonosProf,''))) as
teléfonos,FechanacimientoProf as
fechanacimiento,ltrim(rtrim(isnull(comentariosProf,''))) as
comentarios,ltrim(rtrim(isnull([E-Mail-Prof],''))) as email, isnull(estado,0)
as Estado from profesores where nprof=" & Val(1Legajo.Text), con)

```

C#

```

SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT UPPER(LTRIM(RTRIM(ISNULL(CompanyName,'*****'))))" +
    " AS cliente, UPPER(LTRIM(RTRIM(ISNULL(ContactName,'*****')))) AS contacto, LTRIM(RTRIM(ISNULL(ContactTitle,''))) AS cargo " +
    ", LTRIM(RTRIM(ISNULL(Address,''))) AS direccion, LTRIM(RTRIM(ISNULL(City,''))) AS ciudad, LTRIM(RTRIM(ISNULL(Region,''))) AS " +
    " AS localidad , LTRIM(RTRIM(ISNULL(PostalCode,''))) AS cp, LTRIM(RTRIM(ISNULL(Country,''))) AS pais, " +
    "LTRIM(RTRIM(ISNULL(Phone,''))) AS telefono, LTRIM(RTRIM(ISNULL(Fax,''))) AS fax FROM Customers WHERE ID=" +
    + Conversion.Val(1IdCliente.Text), routines.Conectar());

```

Acá armamos un select similar al de la vista o al que usamos la clase pasada. Lo que te va a llamar la atención, es que por ejemplo, en vez de poner simplemente apellido, pone:

```
upper(ltrim(rtrim(isnull(apellidoprof,'*****')))) as apellido
```

eso es una expresión un poco más compleja de SQL para validar el dato que entra, evitando que sea null (nulo) con isnull, que si es nulo apellidoprof, lo reemplaza con “\*\*\*\*\*”. Luego rtrim le quita los espacios de la derecha (r por right), si fuera por ejemplo “Peres”, queda “Peres”. Ltrim (l por left), le quita los espacios de la izquierda, por ejemplo “ Peres” queda “Peres”. El uso en conjunto de “ltrim(rtrim(“ de SQL es equivalente a trim de VS, borrando los espacios de

adelante y de atrás de un texto. Finalmente la función upper, pone a mayúsculas el apellido: por ejemplo, si era “Peres”, devuelve “PERES”. Toda esas conversiones se realizan directamente en SQL en el mismo select, con lo cual garantizamos que “apellido” nos viene limpio y listo para mostrar en un textbox.

Ya hablamos varias veces, que lo que viene en un dataset, en este caso el dataset ds, datatable “Alumnos” (que podría ser “Profesores” pero da lo mismo ahora); es como una planilla de Excell, con la primera fila a 0, y la primera columna a 0 también; o, MUCHO MEJOR que usar números para las columnas (que si agregás o quitás una se te destartala todo...) es usar el alias que le diste en el select. Entonces:

```
TextBox1.Text = ds.Tables("Alumnos").Rows(0)("apellido")
```

La Fila 0, columna “Apellido”, apunta al apellido del profesor, y lo guardamos en el textbox1. Hacemos lo mismo con los demás campos.

```
CheckBox1.Checked = IIf(ds.Tables("Alumnos").Rows(0)("estado") = 0, False, True)
```

CheckBox1 es un checkbox (el cuadradito para tildar...). La propiedad Checked está en true, cuando el cuadradito está tildado, y a false cuando no lo está. El tema es que el campo “estado” es bit, 0 o 1, no true o false; entonces con un iif (ver clases anteriores), si es 1 lo convierto en true para tildar el checkbox, y si es 0 lo paso a false para destildarlo. Así el checkbox representará el valor guardado en “estado”.

```
DateTimePicker1.Value = ds.Tables("Alumnos").Rows(0)("fechanacimiento")
```



ese es un DateTimePicker, que permite elegir una fecha, escribiendo su día, mes o año, o tocando la flechita y eligiéndola de un calendario. Mirale las propiedades, y vas a ver que hay bastantes cosas para tocar, como por ejemplo, limitar las fechas a un rango, con un mínimo y un máximo, entre muchas más.

La propiedad value del DateTimePicker, me permite leer o cambiar la fecha del mismo, así que directamente le paso la celda del dataset con la fecha de nacimiento.

En nuestro sistema, van a ser muy importantes las fechas, como fecha de factura, vencimiento, compra, venta etc. Lamentablemente en ASP .NET (el próximo curso y el más importante de los dos), no hay de estos objetos... disfrútalos en este curso bajo Windows...

```
Private Sub bBuscar_Click(ByVal sender As System.Object  
    buscar(" apeynom like '" & tApellido.Text & '%" ")  
End Sub
```

Al apretar el botón bBuscar, llama a buscar, como ya vimos en el Load.

```

Private Sub PictureBox8_Click(ByVal sender As System.Object)
    tApellido.Text = ""
    buscar(" apeynom like '" & tApellido.Text & "%' ")
End Sub

```

Esta es la X que borra el cuadro de texto. Entonces primero borra lo que hay, poniendo a "" (vacío) la propiedad text, y luego ejecuta buscar directamente.

```

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    If MessageBox.Show("Está por ELIMINAR definitivamente al PROFESOR: " & TextBox1.Text.Trim.

    If SQL_Accion("delete from profesores where nprof=" & Val(llegajo.Text)) = False Then
        MsgBox("Hubo un error al intentar borrar al Profesor, reintente, y si el error persist
    Else

        buscar(" nprof=" & Val(llegajo.Text))
        MsgBox("El Profesor fue ELIMINADO de la base de datos.")

    End If
End Sub

```

## C#

```

private void bEliminar_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Está por ELIMINAR definitivamente el Cliente: " + tCliente.Text.Trim().ToUpper() +
        ",. Es algo EXTREMO. Está SEGURO?", "Eliminar Cliente", MessageBoxButtons.YesNo, MessageBoxIcon.Warning,
        MessageBoxDefaultButton.Button2) == System.Windows.Forms.DialogResult.No)
        return;
    // Ejecutamos el Delete
    if (routines.SqlAccion("DELETE FROM Customers WHERE ID=" + Conversion.Val(lIdCliente.Text)) == false)
    {
        MessageBox.Show("Hubo un Error al intentar Borrar el Cliente, Reintente, y Si el Error Persiste," +
            " Anote Todos los Datos que Quizo Ingresar y Comuníquese con el Programador (Otra Vez).", "Eliminar Cliente",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        buscar(" id=" + Conversion.Val(lIdCliente.Text));
        MessageBox.Show("El Cliente fue ELIMINADO de la Base de Datos.");
    }
}

```

Este es el botón “Eliminar”. Primero pregunta si querés hacerlo (para eso vimos clases atrás el MessageBox), y si querés borrarlo, ejecutamos la función SQL:Accion (que ya vimos más arriba), pasándole una sentencia de DELETE. Si la función devuelve false, es porque algo salió mal, así que pone un cartel indicando que hubo un error.

Si no hay error, el profesor fue eliminado, así que invoca buscar, pasando al número del profesor (que ya no está), entonces no muestra nada y avisa. Podrías cambiar para buscar a “todos”, así ya te llena la lista con los sobrevivientes...

```

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Dim errores As String = "", en As String = vbCrLf
    If TextBox1.Text.Trim.Length < 3 Then
        errores &= "Debe completar el Apellido del profesor." & en
    End If
    If TextBox2.Text.Trim.Length < 3 Then
        errores &= "Debe completar el o los nombres del profesor." & en
    End If
    TextBox3.Text = Val(TextBox3.Text.Trim.Replace(".", "").Replace(" ", "").Replace(", ", ""))
    If TextBox3.Text.Trim.Length < 4 Or TextBox3.Text.IndexOf("11111") > -1 Or TextBox3.Text.
        errores &= "Debe completar CORRECTAMENTE el documento del profesor." & en
    End If
    If TextBox7.Text.Trim.Length <> 0 And (TextBox7.Text.IndexOf("@") < 0 Or TextBox7.Text.In
        errores &= "Verifique por favor el email del profesor. No es obligatorio, pero si lo
    End If
    If errores.Length > 0 Then
        MsgBox("Hubo errores, por favor verifique y corrija antes de intentar de nuevo:" & en
        Exit Sub
    End If
    ' TextBox12.Text = "update alumnos set apellidos='" & TextBox1.Text.Trim.ToUpper.Replace(
    If SQL_Accion("update profesores set estado=" & IIf(CheckBox1.Checked, 1, 0) & ", apellid
        MsgBox("Cambios realizados correctamente.")

        buscar(" nprof=" & VNum(lLegajo.Text))
    Else
        MsgBox("Se produjo un error al querer guardar los datos del profesor, reintente, y si
    End If
End Sub

```

## C#

```

private void bGuardar_Click(object sender, EventArgs e)
{
    // Para Guardar los Errores que Surjan
    string errores = "";
    // Guardamos el caracter del enter
    string enter = Constants.vbCrLf;
    if (tCliente.Text.Trim().Length < 3)
    {
        errores += "Debe completar el Cliente." + enter;
    }

    if (tContacto.Text.Trim().Length < 3)
    {
        errores += "Debe Completar el o los Nombres de Contacto." + enter;
    }

    tTelefono.Text = tTelefono.Text.Trim().Replace(".", "").Replace(" ", "").Replace(", ", "").Replace("-", "");
    if (tTelefono.Text.Trim().Length < 4 | tTelefono.Text.IndexOf("11111") > -1 | tTelefono.Text.IndexOf("12345") > -1 | tTelefono.Text.IndexOf("000000") > -1)
    {
        errores += "Debe completar CORRECTAMENTE el Numero de Telefono." + enter;
    }

    tFax.Text = tFax.Text.Trim().Replace(".", "").Replace(" ", "").Replace(", ", "").Replace("-", "");
    if (tFax.Text.Trim().Length < 4 | tFax.Text.IndexOf("11111") > -1 | tFax.Text.IndexOf("12345") > -1 | tFax.Text.IndexOf("000000") > -1)
    {
        errores += "Debe completar CORRECTAMENTE el Numero de Fax." + enter;
    }

    if (errores.Length > 0)
    {
        MessageBox.Show("Hubo errores, Por Favor Verifique y Corrija Antes de Intentar de Nuevo:" + enter + enter + errores);
        return;
    }
}

```



```

// Ejecutamos el Update
if (routines.SqlAccion("UPDATE Customers SET CompanyName='" + tCliente.Text.Trim().ToUpper().Replace("'", "''") +
    "', ContactName='" + tContacto.Text.Trim().ToUpper().Replace("'", "''") + "', ContactTitle='" +
    tCargo.Text.Trim().ToUpper().Replace("'", "''") + "', Address='" + tDireccion.Text.Trim().ToUpper().Replace("'", "''") +
    "', City='" + tCiudad.Text.Trim().ToUpper().Replace("'", "''") + "', Region='" +
    tLocalidad.Text.Trim().ToUpper().Replace("'", "''") + "', PostalCode='" + tCP.Text.Trim().ToUpper().Replace("'", "''") +
    "', Country='" + tPais.Text.Trim().ToUpper().Replace("'", "''") + "', Phone=" +
    Conversion.Val(tTelefono.Text.Trim().Replace(".", "").Replace(" ", "").Replace(",", ""))
    + ", Fax=" + Conversion.Val(tFax.Text.Trim().Replace(".", "").Replace(" ", "").Replace(",", ""))
    + " WHERE ID=" + routines.Vnum(IIIdCliente.Text)) == true)
{
    MessageBox.Show("Cambios Realizados Correctamente.", "Editar Cliente", MessageBoxButtons.OK, MessageBoxIcon.Information);
    buscar(" ID=" + routines.Vnum(IIIdCliente.Text));
}
else
{
    MessageBox.Show("Se Produjo un Error al Querer Guardar los Datos del Cliente, Reintente, y si el Error Persiste," +
        " Añote Todos los Datos que Quiso Ingresar y Comuníquese con el Programador (Otra Vez).",
        "Editar Cliente", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
}
}

```

Este es el botón de Actualizar, y el más importante de los 3, porque debés validar todos los datos que te cargó el usuario antes de guardarlos.. La mayoría de lo que hay allí ya lo conocemos...

Algunas cosas nuevas:

**en As String = vbCrLf**

Guarda en la variable “en” (por ENTER), a vbCrLf que equivale a CHR(13) & CHR(10) que no es otra cosa que precisamente el carácter ENTER o Retorno de Carro, que como podés notar, no es un solo byte sino 2...

**Val(TextBox3.Text.Trim.Replace(".", "").Replace(" ", "").Replace(",", ""))**

Replace reemplaza en un string, un carácter o varios (substring) por otro/s. En este caso, el primer replace le reemplaza a textbox3.text, un punto por nada, o sea elimina los puntos; luego elimina los espacios reemplazándolos por nada; luego las comas, y finalmente le quita caracteres no numéricos. Hubiera sido mejor haber usado VNum en vez de Val.

En cada IF, valida algo. Errores estaba a “” (vacío), pero cuando un if falla porque hay error, le concatena el error a “errores”, entonces si errores.length >0 (o sea la longitud o cantidad de caracteres de errores) es mayor que 0, es porque juntamos un error o varios al pasar por los IF.

```

If errores.Length > 0 Then
    MsgBox("Hubo errores, por favor verifique y corrija antes de intentar de nuevo:" & en & en & errores)
    Exit Sub
End If

```

Y se nos va de la función indicando que hubo errores. **& en & en & errores**, le está concatenando antes de errores a dos ENTERs, o sea va a meter dos líneas en blanco y luego la de los errores.

```

If SQL_Accion("update profesores set estado=" & IIf(CheckBox1.Checked, 1,
0) & ", apellidoprof='" & TextBox1.Text.Trim.ToUpper.Replace("'", "''") & "',
nombreprof='" & TextBox2.Text.Trim.ToUpper.Replace("'", "''") & "', [documento-
prof]='" & Val(TextBox3.Text.Trim.Replace(".", "").Replace(" ", "").Replace(",",
"")) & ", domicilioprof='" & TextBox4.Text.Trim.ToUpper.Replace("'", "''") & "',
localidadprof='" & TextBox5.Text.Trim.ToUpper.Replace("'", "''") & "',
provinciaprof='" & TextBox8.Text.Trim.ToUpper.Replace("'", "''") & "',

```

```

teléfonosprof=' ' & TextBox6.Text.Trim.ToUpper.Replace("'", "´") & "', [e-mail-
prof]=' ' & TextBox7.Text.Trim.ToUpper.Replace("'", "´") & "',
fechanacimientoprof=" & FechaSQL(DateTimePicker1.Value) & ", comentariosprof=" &
TextBox12.Text.Trim.ToUpper.Replace("'", "´") & "' where nprof=" &
VNum(11legajo.Text)) = True Then
    MsgBox("Cambios realizados correctamente.")

```

Agarrate fuerte con ese UPDATE!... no... no es tanto... miralo despacito y con cuidado, lo que hace es la inversa de cuando validamos en SQL lo que leyó, ahora validamos lo que se va a guardar con el UPDATE. Todo lo que está allí ya lo vimos, y sino preguntame lo que necesites.

```

Private Sub bNuevoAlumno_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles bNuevoAlumno.Click
    If SQL_Accion("insert into profesores (apellidoprof, nombreprof, [documento-prof],domicilioprof, localidadprof, teléfonosprof, fecha
        buscar(" apeynom like '****%' ")
        MsgBox("Se ha creado un nuevo registro para el profesor que desea ingresar, seleccione la línea nueva, cargue los datos y luego
    End If
End Sub

```

Este es el botón de “Nuevo Profesor”. Llama a un insert que va a llenar los campos del nuevo registro con los valores por defecto (mirá el código de SQL y vas a ver que varios campos tienen “ DEFAULT “ y luego un valor, que es el por defecto. Los de apellido tienen \*\*\*\*, entonces, luego de agregar, busca los que tengan \*\*\*\* para entonces seleccionar el que acaba de crear, para que lo cargues.

Como habrás notado, TODO lo que vimos antes está acá... más muchas cosas nuevas. Todo lo que vimos, fue con el objetivo de ir aprendiendo lo que íbamos a necesitar para armar un sistema, que por razones de tiempo (coronavirus...), vamos a explorar completamente en el próximo curso.

HAY MUCHO para analizar y aprender en esta clase, y MUCHO para repasar de las clases anteriores, si o si, le tenés que prestar la atención que merece durante toda la semana.

Como Trabajos Prácticos obligatorios (es porque si no los hacés, no practicás y no aprendés...), tenemos....

- 1) Create otro proyecto igual a este, y luego andá cambiando todo para que use Customers de Northwind de las clases anteriores, en vez de Profesores
- 2) Create otro proyecto Sistema, en donde vas a modificar el script para crear la tabla Clientes basada en Profesores (es lo mismo, pero te faltarían los campos CUIT de 30, Usuario de 10 y Clave de 10). Luego con Clientes (menos Usuario y Clave), hacés Proveedores que es exactamente igual. También creás las vistas de cada uno. Luego armás un formulario para ABM de clientes y otro para ABM de proveedores. El formulario de inicio, será un menú (son su salida y todo lo que ya usaste) y con botones y menú desplegable, para acceder a Clientes y Proveedores. Estos 3 formularios van a ser la semilla del sistema que vamos a armar, y seguiremos trabajando sobre este. En ASP .NET vamos a usar las mismas tablas y vistas que creaste.

Sobre el punto 2, crearemos el Sistema para el TP final, y para que te lo lleves de modelo; que luego extenderemos en Prog IV.

Tenés mucho para aprender, entender y hacer, Te avisé que en la 9 “nos metíamos con todo...”

Mándame lo que hagas para cumplir con la clase, pero más que nada para que pueda seguirte y ayudarte para lo que necesites.

Espero que te sirva la clase y que la disfrutes (no que la padezcas...esto es tu Sistema, y tu inicio para programar comercialmente, yo USO lo que te pasé...).

Nos leemos en el foro!

Guillermo Guastavino