

Clase 8 - No Presencial. Por Guillermo Guastavino

Modificando la Base y Intro a ADO .NET

Hola a todos!.

Primero vamos a hacer unos cambios en la base de datos Northwind, para aprender cómo resolver un tema que se da seguido, especialmente cuando heredás una base de datos de un cliente o vas a trabajar a otra empresa. La solución es avanzadita, así que si no llevás bien claro y presentes las dos clases de SQL, por favor reléelas, especialmente la Clase 7 Extendida en la sección de Apuntes. Como todas las veces, doy por sabido lo anterior y desde allí partimos.

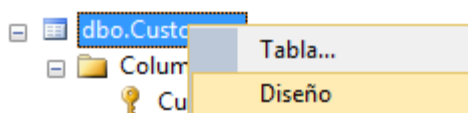
Te acordás que la tabla Customers, tiene un CustomerID que sería la clave principal de Customers, y que conviene ampliamente que sea numérica, y que se incremente automáticamente. El problema es que es de texto... el famoso ALFKI, un desastre para nosotros y el sistema que queremos armar. Para complicarla, otras tablas usan el campo CustomerID como referencia (Orders o sea las facturas). Entonces si cambiamos a numérico, perderíamos la relación que existe con Orders, tendríamos números de un lado (a ALFKI le podría tocar cualquiera), y del otro ALFKI para las facturas. Tranquilamente, podríamos tirar todo y hacer nuestro diseño (algo que van a tener que hacer ustedes para el trabajo final, y para quedarse con un buen ejemplo), pero sigamos en el escenario en el cual heredamos la base, o peor, entramos a trabajar a una empresa en donde está en funcionamiento esta base.

Vamos a realizar una serie de operaciones para poder reemplazar el horrible ALFKI por números, en Customers y en cualquier tabla que se relacione con Customers. Lo más loco es que Employees SI usa una clave numérica... entonces porqué no Customers... en fin... hay que usar lo que hay.

Primero, vamos a modificar Customers, agregando un campo ID de tipo int. Se puede hacer con una instrucción ALTER TABLE:

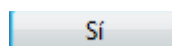
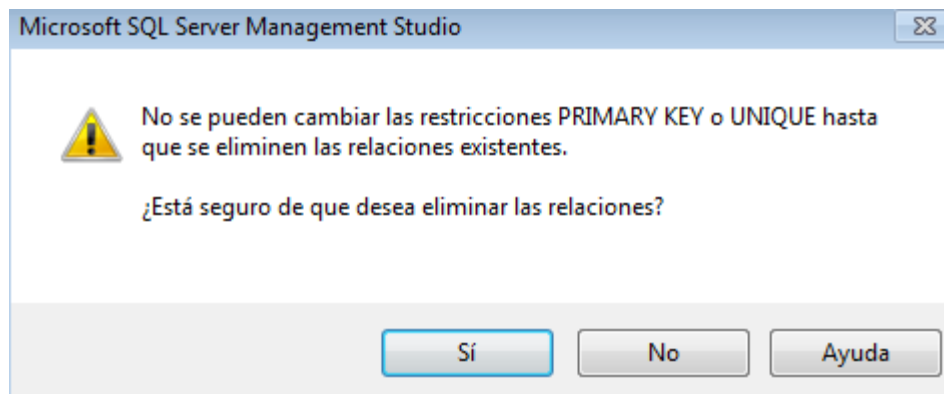
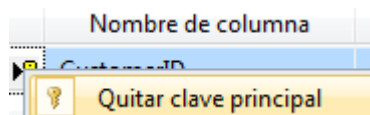
```
ALTER TABLE Customers add ID int
```

Pero vamos a usar el Diseñador:

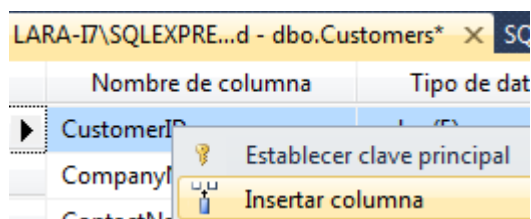


LARA-I7\SQLXPRESS...d - dbo.Customers			SQLQuery4.sql - LA...-I7\G
	Nombre de columna	Tipo de datos	Permitir val...
▶	CustomerID	nchar(5)	<input type="checkbox"/>
	CompanyName	nvarchar(40)	<input type="checkbox"/>
	ContactName	nvarchar(30)	<input checked="" type="checkbox"/>
	ContactTitle	nvarchar(30)	<input checked="" type="checkbox"/>
	Address	nvarchar(60)	<input checked="" type="checkbox"/>
	City	nvarchar(15)	<input checked="" type="checkbox"/>
	Region	nvarchar(15)	<input checked="" type="checkbox"/>
	PostalCode	nvarchar(10)	<input checked="" type="checkbox"/>
	Country	nvarchar(15)	<input checked="" type="checkbox"/>
	Phone	nvarchar(24)	<input checked="" type="checkbox"/>
	Fax	nvarchar(24)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Primero le quitamos la clave principal a CustomerID



Ahora agregamos un campo ID de tipo int:



LARA-I7\SQLXPRES...d - dbo.Customers* X SQLQ	
Nombre de columna	Tipo de datos
CustomerID	nchar(5)

LARA-I7\SQLXPRES...d - dbo.Cust	
Nombre de columna	
ID	

LARA-I7\SQLXPRES...d - dbo.Customers* X SQLQuery4	
Nombre de columna	Tipo de datos
ID	nchar(10)
CustomerID	geography
CompanyName	geometry
ContactName	hierarchyid
ContactTitle	image
	int

Abajo están las propiedades

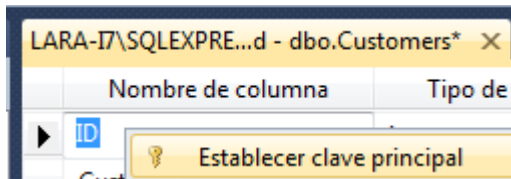
Propiedades de columna		
<div> <div> <div></div> <div>A Z</div> <div></div> </div> <div></div> </div>		
<div> <div></div> <div>(General)</div> </div>		
(Nombre)		ID
Permitir valores NULL		Sí
Tipo de datos		int
Valor o enlace predeterminado		
<div> <div></div> <div>Diseñador de tablas</div> </div>		


En las propiedades buscamos:

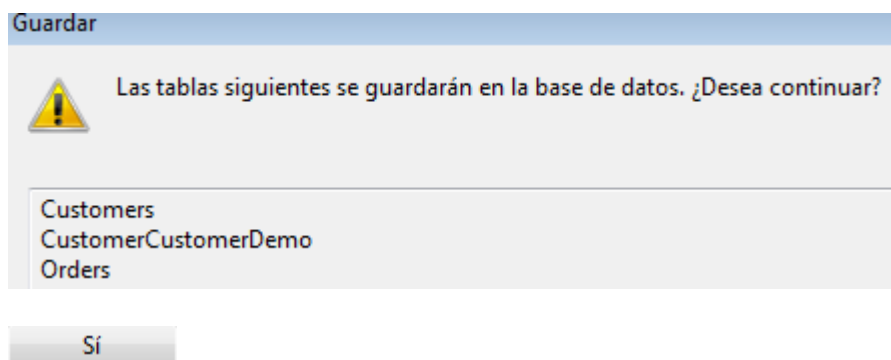
Propiedades de columna		
<div> <div> <div></div> <div>A Z</div> <div></div> </div> <div></div> </div>		
Disperso		No
<div> <div></div> <div>Especificación de columna calculada</div> </div>		
<div> <div></div> <div>Especificación de identidad</div> </div>		No
(Identidad)		No
Incremento de identidad		Sí
Inicialización de identidad		No
Incremento de identidad		1
Inicialización de identidad		1

Es desde donde empieza (1) y cada cuanto incrementa (1). Con esto se genera un número para cada fila, y una nueva fila tendrá el siguiente que corresponda.

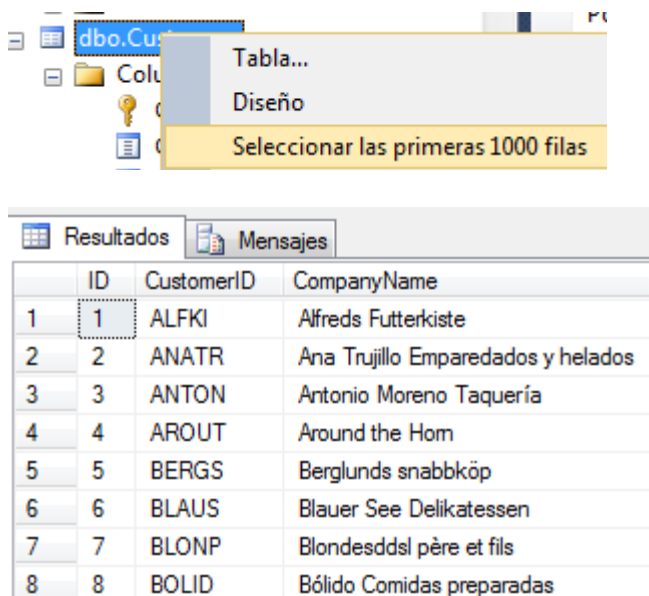
Ahora le damos clave principal a la columna ID:



Ya podemos guardar 





Veamos cómo quedó



Ahora en Orders, agregá ID int debajo de CustomerID:

LARA-17\SQLEXPRES...wind - dbo.Orders* X SQLQuer

	Nombre de columna	Tipo de datos
	OrderID	int
	CustomerID	nchar(5)
	ID	int
	EmployeeID	int


En las propiedades, ponemos 0 en valor predeterminado, para que los nuevos registros tengan 0 en vez de null:

Propiedades de columna

Permitir valores NULL	Sí
Tipo de datos	int
Valor o enlace predeterminado	0

Guardamos: 

Guardar

 Las tablas siguientes se guardarán en la base de datos. ¿Desea continuar?

Shippers
Employees
Orders
Order Details

Sí

Cuando lo vemos, vas a notar que ahora hay un campo ID con 0 para todos

	OrderID	CustomerID	ID
1	10248	VINET	0
2	10249	TOMSP	0
3	10250	HANAR	0
4	10251	VICTE	0
5	10252	SUPRD	0
6	10253	HANAR	0
7	10254	CHOPS	0
8	10255	RICSU	0


Ahora debemos lograr que en Orders.ID, se carguen los Customers.ID, utilizando la relación existente del famoso ALFKI

Necesitamos aprender UPDATE

(No tiene nada que ver con lo que queremos, es sólo un ejemplo)

Esto pone ID a 1 para todas las filas de Orders en donde EmployeeID es 4:

```
update orders set ID=1 where EmployeeID=4
```

 Ejecutar

(156 filas afectadas)

Vemos como quedó:

	OrderID	CustomerID	ID	EmployeeID
1	10248	VINET	0	5
2	10249	TOMSP	0	6
3	10250	HANAR	1	4
4	10251	VICTE	0	3
5	10252	SUPRD	1	4
6	10253	HANAR	0	3
7	10254	CHOPS	0	5
8	10255	RICSU	0	9
9	10256	WELLI	0	3
10	10257	HILAA	1	4
11	10258	ERNSH	0	1
12	10259	CENTC	1	4
13	10260	OTTIK	1	4
14	10261	QUEDE	1	4

Si quiero cambiar más columnas, agrego coma y la próxima asignación, por ejemplo: (no lo ejecutes)

```
update orders set ID=1, EmployeeID=3 where EmployeeID=4
```

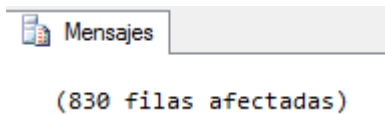
Volvamos a lo nuestro:

Necesitamos que ID de Orders se cargue con ID de Customers

Te acordás cuando vimos como relacionar tablas en un select?, acá es similar:

```
update orders set orders.id = (  
    select customers.id from customers  
    where customers.CustomerID= orders.CustomerID )
```

Ejecutamos



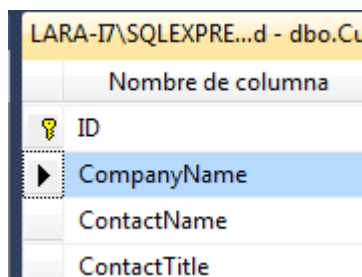
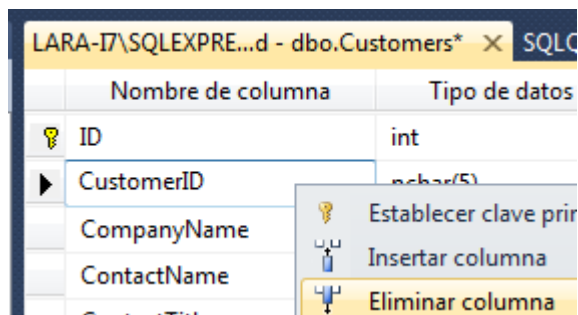
El select relaciona ambas tablas, y devuelve sólo al campo customers.id, que se lo asignamos a orders.id

Veamos cómo quedó:

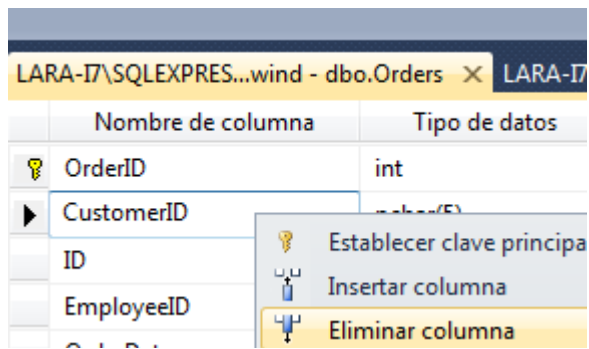
	OrderID	CustomerID	ID
1	10248	VINET	85
2	10249	TOMSP	79
3	10250	HANAR	34
4	10251	VICTE	84
5	10252	SUPRD	76
6	10253	HANAR	34
7	10254	CHOPS	14
8	10255	RICSU	68
9	10256	WELLI	88
10	10257	HILAA	35
11	10258	ERNSH	20
12	10259	CENTC	13
13	10260	OTTIK	56

Relacionó por CustomerID y obtuvimos el ID correspondiente a cada uno

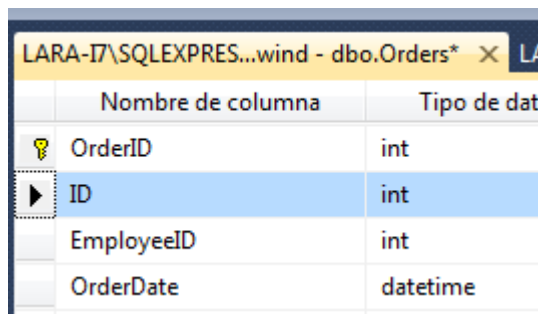
Ahora ya nos sobra el campo CustomerID en ambas tablas, así que los sacamos:



Sacalo también de Orders:



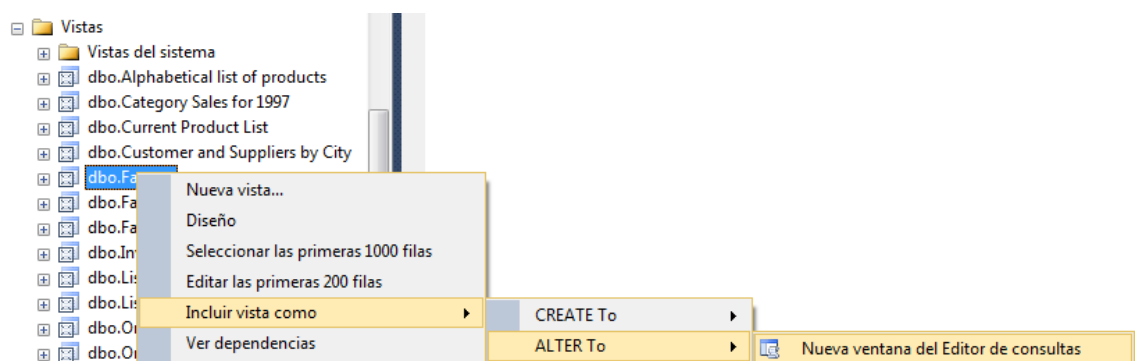
Nombre de columna	Tipo de datos
OrderID	int
CustomerID	int
ID	int
EmployeeID	int
OrderDate	datetime



Nombre de columna	Tipo de datos
OrderID	int
ID	int
EmployeeID	int
OrderDate	datetime



Editemos la consulta Facturas, que hicimos la clase pasada pero usando ALTER TABLE




```

SQLQuery12.sql - LA...-I7\Guillermo (60)) X SQLQuery11.sql - LA...-I7\Guillermo (59)
USE [Northwind]
GO

/***** Object: View [dbo].[Factura]    Script Date: 15/05/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

ALTER VIEW [dbo].[Factura]
AS
SELECT      TOP (100) PERCENT dbo.Customers.CustomerID AS
              dbo.Products.ProductName AS Product
              dbo.[Order Details].UnitPrice * 21
              (dbo.[Order Details].UnitPrice * 21
FROM          dbo.Customers INNER JOIN
              dbo.Orders ON dbo.Customers.CustomerID = dbo.Orders.CustomerID
              dbo.[Order Details] ON dbo.Orders.OrderID = dbo.[Order Details].OrderID
              dbo.Products ON dbo.[Order Details].ProductID = dbo.Products.ProductID
GO

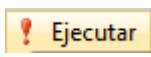
```

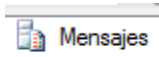
Cambiamos en todos los lugares en donde dice CustomerID, por ID

```

TOP (100) PERCENT dbo.Customers.ID AS NCLi,
-----
dbo.Orders ON dbo.Customers.ID = dbo.Orders.ID INNER JOIN

```

 Ejecutar para cambiar la vista. No necesitamos guardar el script

 Mensajes
Comandos completados correctamente.

Hacemos un select de la factura 10643 como en la clase 7:

```
select * from factura where nfact=10643
```

	NCLi	Ciente	NFact	FFac	Producto	Precio	IVA	PconIVA	Cant	Subtotal
1	1	Alfreds Futterkiste	10643	1997-08-25 00:00:00.000	Rössle Sauerkraut	45,60	9,576	55,176	15	827,64
2	1	Alfreds Futterkiste	10643	1997-08-25 00:00:00.000	Chartreuse verte	18,00	3,78	21,78	21	457,38
3	1	Alfreds Futterkiste	10643	1997-08-25 00:00:00.000	Spegesild	12,00	2,52	14,52	2	29,04

Funcionó igual que en la clase 7, pero sabemos que ya no existe un campo clave con ALFKI, así que nuestra relación de reemplazo funcionó perfectamente.

Intro a ADO .NET

ADO .NET es un conjunto de clases que te brindan todas las herramientas que necesitás para acceder a bases de datos relacionales como SQL Server, Oracle Server, MySQL, también Access, y no relacionales con DBase, una planilla de Excel, Word o hasta archivos txt. Básicamente puede acceder a cualquier cosa que tenga un formato, inclusive si no es soportado por Microsoft, ya que el proveedor de la base de datos “Provider”, provee las clases que se necesitan para acceder a su formato y listo. Lo más interesante de ADO es que si escribis todo un sistema para Access, y luego querés pasarte a SQL Server o Oracle, el programa es prácticamente el mismo, cambia el provider y la connection string.

La Connection string es un string que brinda a ADO la información para poder conectarse con la base de datos. Entonces, cambiando la connection string, te vas a conectar con otra base sin casi cambiar nada más. Si siempre te vas a “una base superior”, no vas a tener problemas.

Realmente ADO .NET anda con cualquier cosa y es muy fácil de usar.

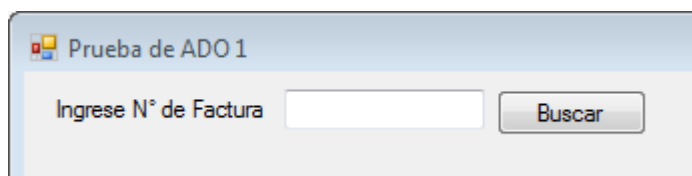
Vamos a crear una pequeña aplicación para buscar facturas en Northwind, usando la consulta Factura que preparamos antes.

Creá un nuevo proyecto llamado ADO1

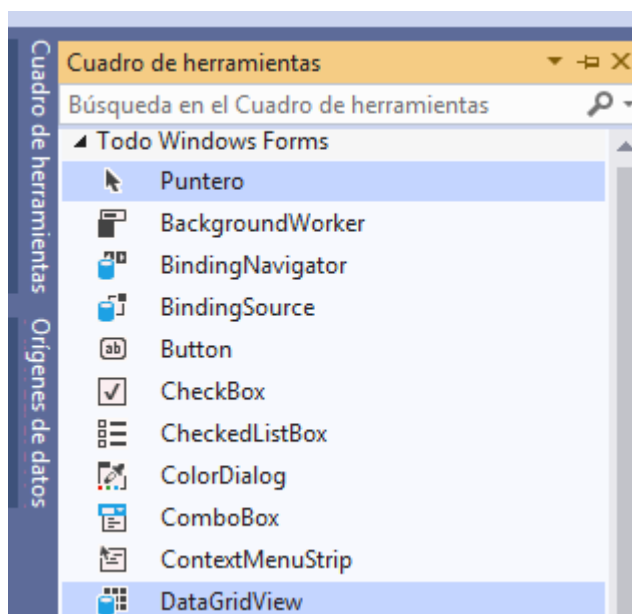
Agregar una label con text “Ingrese N° de Factura”

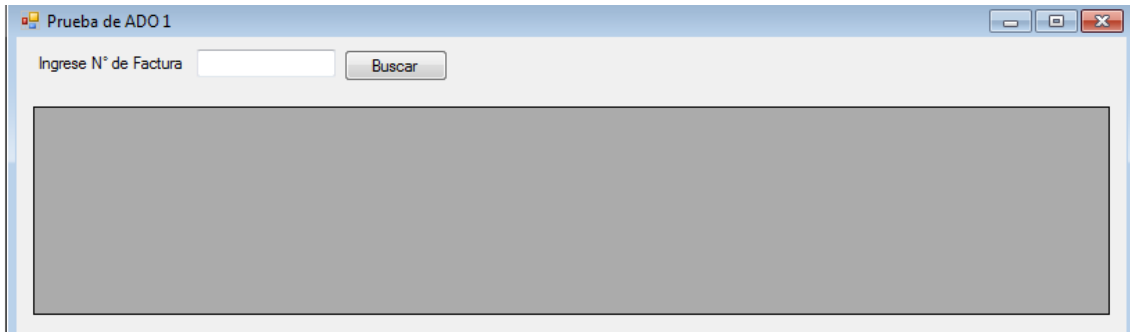
Un textbox NFact con MaxLength=5

Un botón bBuscar con text Buscar

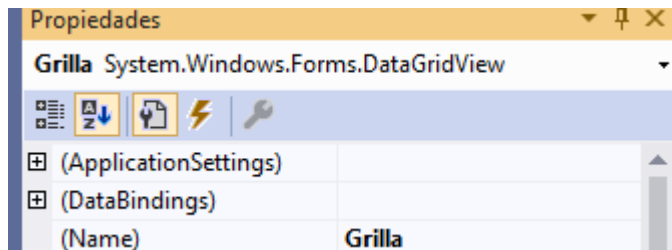


Agregamos un DataGridView, que es una grilla en dónde vamos a ver los datos





Al DataGridView1 lo llamamos Grilla para que sea más fácil.



Pasamos al código

Ponemos un ENTER delante de Public Class Form1 para poder escribir en la línea de arriba

Ahora indicamos que namespaces necesitamos. En VB se utiliza Imports y en C# se usa Include

```
Imports System.Data.SqlClient
1 referencia
Public Class Form1
```

Con eso, ahora tenemos acceso a la clase SqlConnection que es una clase de ADO específica para SQL Server

Ahora creamos un objeto SqlConnection, que tiene dentro a la connection string, mediante la cual le decimos a ADO donde está el servidor y el nombre de la base, entre muchas cosas posibles. De todas esas cosas, elegí lo mínimo indispensable y que brinde seguridad.

```
Imports System.Data.SqlClient
1 referencia
Public Class Form1
    Dim con As New SqlConnection("data source=LARA-I7\SQLEXPRESS14X; initial catalog=Northwind; integrated security=true")
End Class
```

Con (o el nombre que quieras, será un objeto de tipo SqlConnection, con todas las propiedades, métodos y eventos de la clase, ya que es una instancia de la clase SqlConnection

```
Dim con As New SqlConnection("data source=LARA-I7\SQLEXPRESS14X; initial catalog=Northwind; integrated security=true")
```

“data source=” es la ruta, IP o instancia de la base de datos. Acá tenés que poner el nombre de tu instancia de SQL Server. Si no te la acordás, andá a SQL, y te la copiás del Nombre del Servidor. En mi caso:



“initial catalog” indica el nombre de la base de datos que vas a usar entre las que tengas en la instancia que indicaste. Usamos Northwind

Integrated security=true, activa la seguridad y la encriptación de los paquetes de datos.

Ahora que el objeto Con apunta al servidor y a la base, debemos crear un objeto SqlDataAdapter, en el cual indicaremos una consulta, o la llamada a un Sp (store procedure de SQL)

Para poder accederla desde distintas partes, vamos a poner a los objetos siguientes en una Sub Buscar:

Sub Buscar()

```
Dim da As New SqlDataAdapter("select * from factura where nfact=10643", con)  
End Sub
```

Al objeto lo llamé “da” por dataadapter. Adentro le puse la misma consulta que usamos para probar la vista. Para que sepa en dónde tiene que buscar a la vista, le paso al final el objeto con, que tiene todos los datos de la instancia y la base. Voy enganchando un objeto con otro.

Para que no busque sólo a la factura 10643, vamos a concatenarlo con el textbox. Usaremos la función Val (la próxima les paso funciones hechas por mi y que uso en todos mis sistemas), que devuelve el valor numérico de un string. “25 A 67” devuelve 25, porque lo que sigue ya no es un número. “A25” devuelve 0, “perro” devuelve 0. Le aplico también INT (redondea a entero), por si al usuario se le ocurre la locurita de poner un número de factura con decimales. Finalmente CSTR convierte el número a

string y trim le quita espacios de más adelante y atrás si los tuviera. Todo esto es para validar un poquito al número de la factura en el textbox

```
Dim da As New SqlDataAdapter("select * from factura where nfact=" & CStr(Int(Val(Nfact.Text))).Trim, con)
```

El objeto Da, “apunta” a la vista, pero no lee ni hace nada. para recibir los registros que vuelvan de la vista, necesito otro objeto, un Data Set. El DataSet es un lugar para almacenar a los registros que se leen. Imaginate un placard con cajones. Todo el placard es el DataSet, pero si te pido que guardes algo allí, me vas a preguntar en cual de los cajones querés que guarde las cosas... Estos “cajones” son los DataTables. En cada Datatable, podría guardar un paquete de datos diferente. Por ejemplo en uno a los clientes, en otro a las facturas y en otro a los productos.

Los Datatables son colecciones, como lo es un string o un listbox. Una colección es un conjunto variable de datos de diferentes tipos, es variable porque puede cambiar la cantidad de elementos que contenga. Como una colección de figuritas, que puedo tener 20, luego conseguir 4 más y más tarde perder 10. Mi primera figurita es la 1, pero en las colecciones de .NET, el primer elemento es el 0. En este caso, voy a mandarlo al DataTable 0 (que sería el “primer cajón”), aunque a los cajones les puedo pegar papelitos indicando que tienen adentro. Los papelitos para los Datatables son los alias, un nombre con el cual trato de explicar que hay adentro, aunque puedo llamarlo de cualquier forma. En nuestro caso vamos a llamarlo Facturas al DataTable, y al “placard”, el DataSet, lo llamaremos Ds.

```
Dim ds As New DataSet  
da.Fill(ds, "Facturas")
```

Primero creo la instancia de la clase DataSet y la llamo ds (o lo que quieras). Luego lleno (Fill) al DataSet ds, en su cajón o Datatable “Facturas” con los registros que vienen del DataAdapter da que definí antes.

En éste momento, ds.tables(“Facturas”), va a tener adentro todos los registros que vengan de la vista, o ninguno si busco una factura que no existe.

Ahora le paso el paquete de registros (que en memoria se asemeja a una planilla de Excell con filas y columnas), a la Grilla para que los muestre:

```
Grilla.DataSource = ds.Tables("Facturas")
```

Con esto le estoy diciendo que la fuente de los datos con que tiene que cargarse la Grilla, la busque en ds y su datatable “facturas”

Me falta sólo decirle a la Grilla que refresque los datos para que pueda mostrarlos:

```
Grilla.Refresh()
```

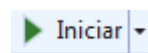
Vamos a llamar a la función buscar, desde el evento click del botón bBuscar.

Todo el programa:

```
Imports System.Data.SqlClient

1 referencia
Public Class Form1
    Dim con As New SqlConnection("data source=LARA-I7\SQLEXPRESS14X; initial catalog=Northwind; integrated security=true")
    1 referencia
    Sub Buscar()
        Dim da As New SqlDataAdapter("select * from factura where nfact=" & CStr(Int(Val(Nfact.Text))).Trim, con)
        Dim ds As New DataSet
        da.Fill(ds, "Facturas")
        Grilla.DataSource = ds.Tables("Facturas")
        Grilla.Refresh()
    End Sub
    0 referencias
    Private Sub bBuscar_Click(sender As Object, e As EventArgs) Handles bBuscar.Click
        Buscar()
    End Sub
End Class
```

Ejecutamos:



Y probamos con 10643 y con otras facturas, y me las va a mostrar en la grilla:

	NCL	Cliente	NFact	FFac	Producto	Precio	IVA	Pcon
▶	1	Alfreds Futterkiste	10643	25/08/1997	Rössle Sauerkraut	45.6000	9.5760	55.1760
	1	Alfreds Futterkiste	10643	25/08/1997	Chartreuse verte	18.0000	3.7800	21.7800
	1	Alfreds Futterkiste	10643	25/08/1997	Spegesild	12.0000	2.5200	14.5200
*								

La clase que viene, nos metemos de cabeza en ADO y ya vamos a empezar a armar nuestro proyecto.

Se me hizo de madrugada escribiendo y empecé a las 9!, trato de subir la clase y nos leemos en la clase!.

Abrazo!

Guillermo Guastavino