SQL Server

Servidor:1

DESKTOP-MSJO6R8\MSSQLSERVER01

PabloK

SQ23PabloL

Servidor: 2

DESKTOP-MSJO6R8

Nombre de usuario: sa (¿?)

Constraseña: LQS23PaLosPibes

Desde SQL Management si olvidaste nombre de usuario y/o contraseña: te autenticás con Windows. Luego vas a seguridad=>inicios de sesión.

Ingresa a VS con continuar sin código, ahí va a explorador de servidores.

Se hace click y pone la contraseña.

Las 3 tildes se sacan. En roles se marca todo y luego conceder y habilitado.

Luego se hace lo propio en base de datos (arriba de todo, donde dice DESKTOP)propiedades=>seguridad)=> modo de autenticación de Windows y SQL server.

No arranco desde explorador de servidores porque se ortiva, si desde explorador de objetos de SQL Server. Ahí creo una base de datos nueva. Ahí creo una nueva tabla (por fin!!!)

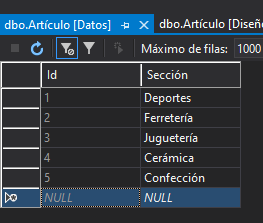
<http://blog.hadsonpar.com/2021/11/agregar-sql-server-visual-studio_0382317599.html>

Esa me ayudó bastante

Al campo Id. Que es numérico e incrementable. Se va a propiedades y se toca: especificación de idendidad=>es una identidad=>true.

*Clase 89*

Dbo.Articulos=> Ver datos (o mostrar datos de tabla)



Y le da algo de contenido

Cómo crear un proyecto que utilice la base de datos?

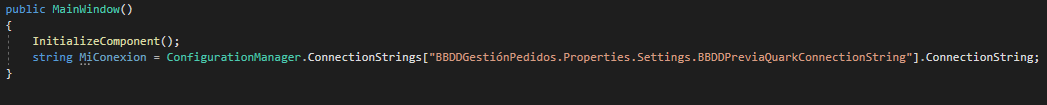
Archivo=>nuevo proyecto=>WPF

Per se, así no se conecta con la BBDD. Todavía hay que hacerlo=>Agregar nuevo origen de datos=>Base de datos=>conjunto de datos=>conexión.

BBDDPreviaQuarkConnectionString Aconseja copiar dicha data al portapapeles.

Se marca la única tabla que hay.

También hay que ir al archivo de C# y en el método main se pone:



Falta agregar una clase: Referencias=>agregar referencia system.configuration => agregar también el mismo using.

BBDDGestiónPedidos es el nombre del proyecto, coincide con el namespace. Tras el último punto es lo que copié al portapapeles.

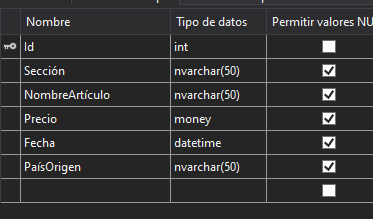
*Clase 90*



Esas son las relaciones. Lo muestra desde Access no se bien por qué.

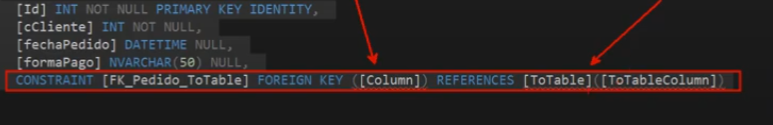
Clientes se relaciona de uno a muchos con pedidos. Pedidos se relaciona de uno a muchos con productos-pedidos. Pero pedidos se relaciona de muchos a uno con productos.

Qué clientes han pagado con tarjeta. Que productos compró tal cliente. Que clientes compraron mas de 6 artículos. Son consultas que se pueden saber porqué las tablas están relacionadas.



Crea luego table de Cliente.

Otra de pedidos, la cual tiene una “Clave foránea” para poder relacionar la tabla de clientes con la tabla de pedido. A Cliente le da a claves externas=>agregarnueva.

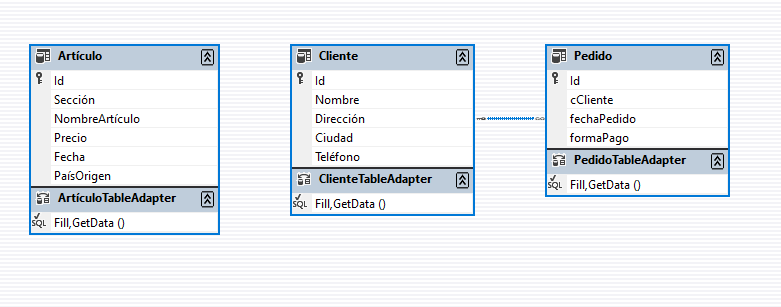


Así queda:

CONSTRAINT [FK\_Pedido\_ToTable] FOREIGN KEY ([cCliente]) REFERENCES [Cliente]([Id])

Luego, actualizer origen de datos. Probablemente sea necesario utilizar el asistente.

Si luego se toca editar dataset con el diseñador (dentro de origen de datos). Aparece esto:



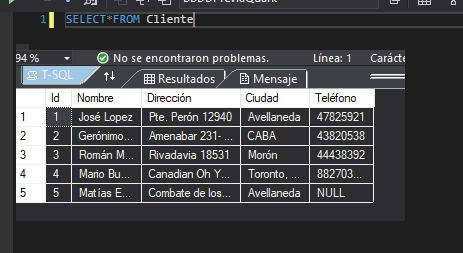
Donde se ve la relación entre 2 tablas.

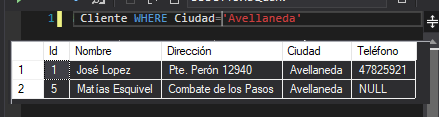
*Clase 91*

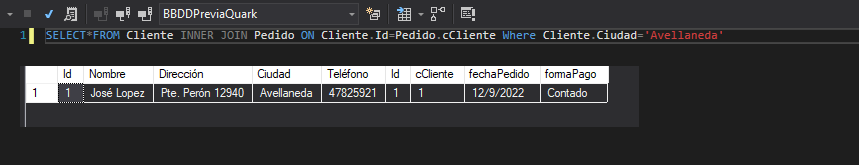
Se llenan datos de la tabla pedido y cliente.

Click derecho sobre la base de datos=>Nueva consulta.

((TIENE QUE ESTAR LA LUCECITA VERDE EN EL EXPLORADOR DE SERVIDORES))







Ahí busca clientes que hayan hecho pedidos en Avellaneda. Cliente.Id=Pedido.cCliente con el inner join implica que las tablas están relacionadas entre sí.

Interfaces gráficas

Menu ver=>Cuadro de herramientas. Agrega label y listbox (o listview)

using System.Data.SqlClient; Agrega eso en c#.

public partial class MainWindow : Window

{

SqlConnection MiConexionSQL;

//Ahí declara un objeto de la clase SqlConnection, posibilitada por la librería "using System.Data.SqlClient;" Esta linea podría ir al ppio.

public MainWindow()

{

InitializeComponent();

string MiConexion = ConfigurationManager.ConnectionStrings["BBDDGestiónPedidos.Properties.Settings.BBDDPreviaQuarkConnectionString"].ConnectionString;

MiConexionSQL = new SqlConnection(MiConexion); //Acá instancia el objeto. Pasa ese string como parámetro al constructor.

}

}

Ahí ya estaría todo conectado.

Luego crea un método para mostrar los clientes.

*Clase 92*

public partial class MainWindow : Window

{

SqlConnection MiConexionSQL;

//Ahí declara un objeto de la clase SqlConnection, posibilitada por la librería "using System.Data.SqlClient;" Esta linea podría ir al ppio.

public MainWindow()

{

InitializeComponent();

string MiConexion = ConfigurationManager.ConnectionStrings["BBDDGestiónPedidos.Properties.Settings.BBDDPreviaQuarkConnectionString"].ConnectionString;

MiConexionSQL = new SqlConnection(MiConexion); //Acá instancia el objeto. Pasa ese string como parámetro al constructor.

MostrarClientes();

}

private void MostrarClientes()

{

string consulta = "SELECT \* FROM CLIENTE";

//Ahora falta donde almacenar toda la info de la tabla. Ese string sería solo una linea.

SqlDataAdapter MiAdaptadorSql = new SqlDataAdapter(consulta, MiConexionSQL);

using (MiAdaptadorSql)

{

DataTable clientesTabla = new DataTable();

//Datatable requiere de la libreria system.data. Con acciones rapidas y refactorizaciones se hace rápido

//ahí se almacena toda la data.

MiAdaptadorSql.Fill(clientesTabla);

//.Fill es un método.

ListadoClientes.DisplayMemberPath = "Nombre";

//esta linea señala que de la información de la tabla que "Nombre" es lo que se quiere ver en el listbox.

ListadoClientes.SelectedValuePath = "Id";

//Id se utilizará mucho luego, es un campo clave.

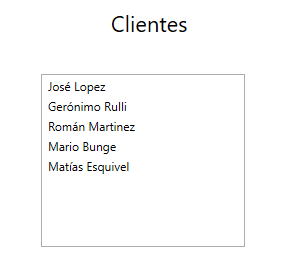
ListadoClientes.ItemsSource = clientesTabla.DefaultView;

//Se especifica de dónde viene toda la información del listbox.

}

}

}



*Clase 93*

private void MuestraPedidos()

{

//Copia código del otro método muestra y luego modifica.

string consulta = "SELECT \* FROM PEDIDO P INNER JOIN CLIENTE C ON C.ID=P.cCLIENTE" + " WHERE C.ID=@CLIENTEID";

//Antes del WHERE dejé un espacio. Sin ese espacio, estalla la App.

SqlCommand sqlComando = new SqlCommand(consulta, MiConexionSQL);

SqlDataAdapter miAdaptadorSql = new SqlDataAdapter(sqlComando);

using (miAdaptadorSql)

{

sqlComando.Parameters.AddWithValue("@CLIENTEID", ListadoClientes.SelectedValue);

//Primero va el parámetro, después va el cliente del listbox sobre el cual se hace click.

DataTable pedidosTabola = new DataTable();

miAdaptadorSql.Fill(pedidosTabola);

PedidosCliente.DisplayMemberPath = "fechaPedido";

PedidosCliente.SelectedValuePath = "Id";

//El campo clave no cambia

PedidosCliente.ItemsSource = pedidosTabola.DefaultView;

}

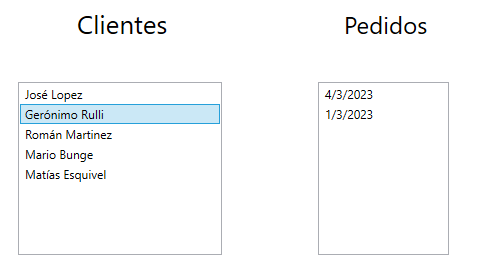
}

private void ListadoClientes\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

MuestraPedidos();

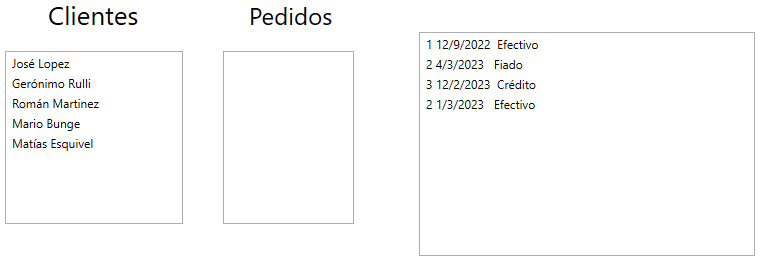
}



*Clase 94*

CRUD

Aplicación de tipo CRUD. Es una aplicación gráfica capaz de Crear, Actualizar y Eliminar (CReate, Update and Delete).



Agrega un listbox mas donde aparecen todos los pedidos.

private void MuestraTodosPedidos()

{

string consulta = "SELECT CONCAT (cCLIENTE, ' ', FECHAPEDIDO, ' ', FORMAPAGO) AS INFOTOTAL FROM PEDIDO";

//\*1 Se crea un campo que no existe, que concatena los 3 campos

//AS DA LUGAR AL ALIAS, EL CAMPO QUE NO EXISTE

SqlDataAdapter miAdaptadorSql = new SqlDataAdapter(consulta, MiConexionSQL);

using (miAdaptadorSql)

{

DataTable pedidosTabla = new DataTable();

//dice que es una especie de tabla virtual.

miAdaptadorSql.Fill(pedidosTabla);

TodosPedidos.DisplayMemberPath = "INFOTOTAL";

//No se pueden concatenar con + o , los campos. \*1

//La novedad es que aquí se quieren mostrar los 3 campos (todos), no uno solo.

TodosPedidos.SelectedValue = "Id";

TodosPedidos.ItemsSource = pedidosTabla.DefaultView;

}

}

Clase 95

Borrar datos.

<https://www.youtube.com/watch?v=HYqGPuTrOJg>

Problemas con el Id, se va 1000 para arriba.