CIBERTEC

Programación Orientada a Objetos II

ÍNDICE

PRESENTACIÓN

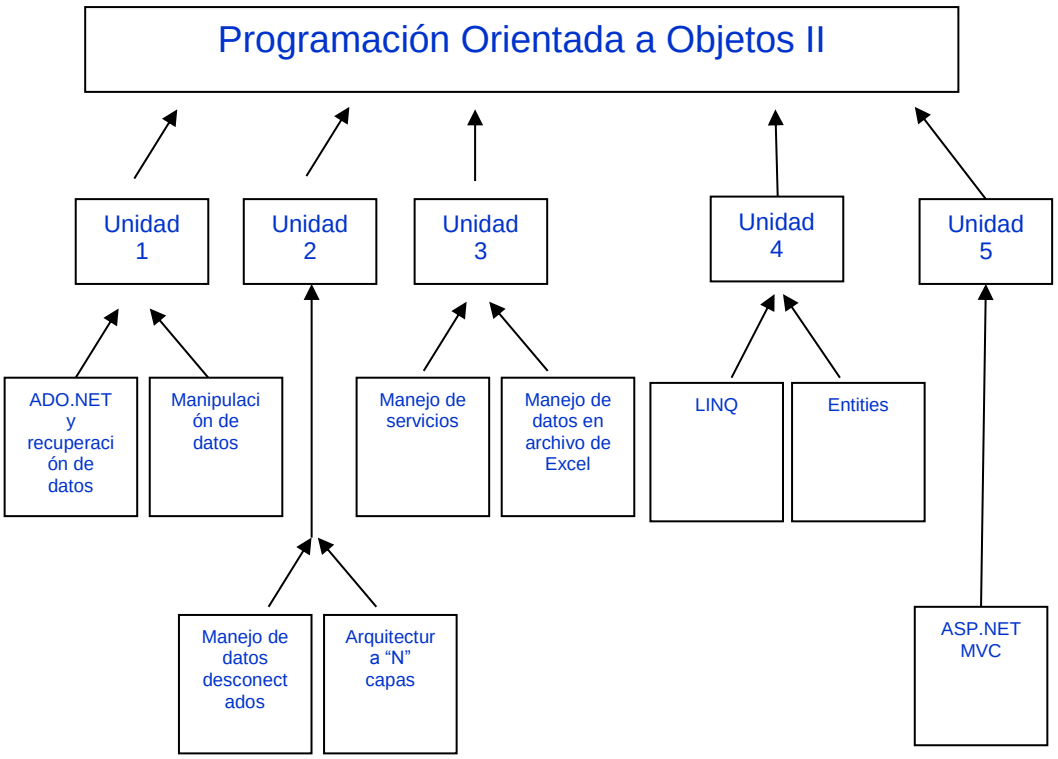
Visual Studio 2012 y su plataforma .NET FrameWork 4.5 es un amplio conjunto de bibliotecas de clases donde se incluye ADO.NET 4.5, que representa las clases para el acceso a datos, el cual viene con importantes patrones en el aumento de la productividad.

Programación Orientada a Objetos II pertenece a la línea de tecnología y se dicta en las carreras de tecnología de la institución. El curso brinda un conjunto de herramientas de programación para trabajar con un origen de datos que permita l alumno realizar operaciones de consulta y actualización de datos en forma eficiente.

El manual para este curso ha sido diseñado bajo la modalidad de unidades de aprendizaje, las que desarrollamos durante semanas determinadas. En cada una de ellas, hallará los logros que se deberá alcanzar al final de la unidad; el tema tratado, el cual será ampliamente desarrollado; y los contenidos, que debe desarrollar. Por último, encontrará las actividades y trabajos prácticos que deberá desarrollar en cada sesión, que le permitirán reforzar lo aprendido en la clase.

El curso es eminentemente práctico: consiste en programación orientada a objetos en Visual C# con base de datos en SQL Server utilizando ADO.NET. La primera parte de este manual nos enseña, mediante ejemplos práticos, a familiarizarnos con los objetos ADO.NET, pararealizar operaciones de consulta y actualización de datos desde un origen de datos, sea en forma conectada o desconectada a la fuente e datos. Aprenderemos a implementar las operaciones de datos utilizamos arquitectura de capas y servicios. Luego vamos a realizar operaciones para generar datos en formato Excel y grabar imágenes en la base de datos. A continuación, implementaremos aplicaciones manejando LINQ y ADO ENTITY para las operaciones de consulta y actualización de datos. Y por ultimo aprenderemos el patrón MVC.

RED DE CONTENIDOS



UNIDAD

1

OPERACIONES CONECTADAS A UN ORIGEN DE DATOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno realiza operaciones de consulta y actualización de datos en el entorno de una aplicación Windows conectado a un origen de datos utilizando la librería ADO.NET

TEMARIO

Tema 1: Introducción a ADO.NET (1 horas)

1. Arquitectura del ADO.NET
2. Manejando una cadena de conexión a un origen de datos utilizando ConfigurationManager

ACTIVIDADES PROPUESTAS

- Los alumnos diseñan formularios que manejan la conexión a una fuente de datos para realizar operaciones de consulta a la fuente de datos.
- Los alumnos desarrollan los laboratorios de esta semana

1. INTRODUCCION ADO.NET

Cuando desarrolle aplicaciones con ADO.NET contará con diferentes requisitos para trabajar con datos. En algunos casos, puede que simplemente desee realizar una consulta de datos en un formulario; en otros casos necesita actualizar los datos.

Independientemente de lo que haga con los datos, hay ciertos conceptos fundamentales que debe de comprender acerca del enfoque de los datos en ADO.NET, los cuales los trataremos en este primer capítulo del manual.

1.1 ARQUITECTURA Y PROVEEDORES DE DATOS

Tradicionalmente, el procesamiento de datos ha dependido principalmente de un modelo de dos niveles basado en una conexión. A medida que aumenta el uso que hace el procesamiento de datos de arquitecturas de varios niveles, los programadores están pasando a un enfoque sin conexión con el fin de proporcionar una mejor escalabilidad a sus aplicaciones.

Arquitectura de ADO.NET

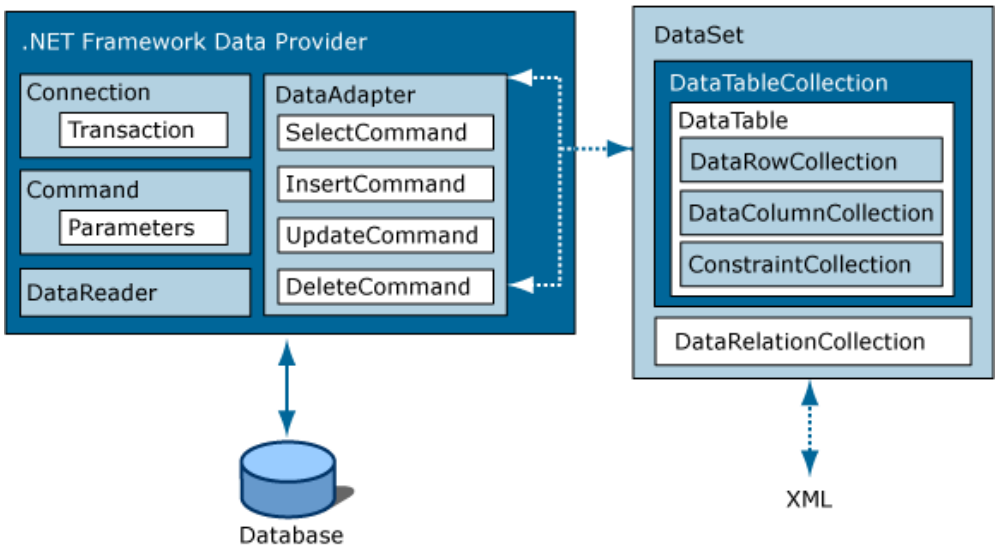


Figura 1: Arquitectura de ADO.NET
Referencia: hampprogramandoando.blogspot.com

Componentes de ADO.NET

Los dos componente principales de ADO.NET para el acceso a datos y su manipulación son los proveedores de datos .NET Fraework y DataSet.

LINQ to SQL

LINQ to SQL admite consultas en un modelo de objetos asignado a las estructuras de datos de una base de datos relacional sin utilizar un modelo conceptual intermedio. Cada tabla se representa mediante una clase distinta, acoplando de manera precisa el modelo de objetos al esquema de

la base de datos relacional. LINQ to SQL traduce Language-integrated queries del modelo de objetos a Transact-SQL y lo envía a la base de datos para su ejecución. Cuando la base de datos devuelve los resultados, LINQ to SQL los vuelve a traducir a objetos.

ADO.NET Entity Framework

ADO.NET Entity Framework está diseñado para permitir que los desarrolladores creen aplicaciones de acceso a los datos programando en un modelo de aplicación conceptual en lugar de programar directamente en un esquema de almacenamiento relacional. El objetivo es reducir la cantidad de código y mantenimiento que se necesita para las aplicaciones orientadas a datos

WCF Data Services

Describe cómo se usa Servicios de datos de WCF para implementar servicios de datos en web o en una intranet. Los datos se estructuran como entidades y relaciones de acuerdo a las especificaciones de Entity Data Model. Los datos implementados en este modelo se pueden direccionar mediante el protocolo HTTP estándar

XML Y ADO.NET

ADO.NET aprovecha la eficacia de XML para proporcionar acceso a datos sin conexión. ADO.NET fue diseñado teniendo en cuenta las clases de XML incluidas en .NET Framework; ambos son componentes de una única arquitectura.

PROVEEDORES DE DATOS ADO.NET

Los proveedores de datos .NET Framework sirven para conectarse a una base de datos, ejecutar comandos y recuperar resultados. Esos resultados se procesan directamente, se colocan en un DataSet con el fin de que el usuario pueda verlos cuando los necesite, se combinan con datos de varios orígenes o se utilizan de forma remota entre niveles. Los proveedores de datos .NET Framework son ligeros, de manera que crean un nivel mínimo entre el origen de datos y el código, con lo que aumenta el rendimiento sin sacrificar funcionalidad.

En la tabla siguiente se muestran los proveedores de datos .NET que se incluyen en el Framework .NET

Proveedor de Datos .NET	Descripción
.NET Framework Proveedor de datos para SQL Server	Proporciona acceso a datos para Microsoft SQL Server. Utiliza la librería System.Data.SqlClient.
Proveedor de datos .NET Framework para OLE DB	Para orígenes de datos que se exponen mediante OLE DB. Utiliza la librería System.Data.OleDb.
Proveedor de datos .NET Framework para ODBC	Para orígenes de datos que se exponen mediante ODBC. Utiliza la librería System.Data.Odbc.
Proveedor de datos .NET Framework para Oracle	Para orígenes de datos de Oracle. El proveedor de datos .NET Framework para Oracle es compatible con la versión 8.1.7 y posteriores del software de cliente de

	Oracle y utiliza la librería System.Data.OracleClient.
Proveedor EntityClient	Proporciona acceso a datos para las aplicaciones de Entity Data Model. Utiliza la librería System.Data.EntityClient.

Los cuatro objetos centrales que constityen un proveedor de datos de .NET Framework son:

Objeto	Descripción
Connection	Establece una conexión a una fuente de datos. La clase base para todos los objetos Connection es **DbConnection**.
Command	Ejecuta un comando en una fuente de datos. Expone Parameters y puede ejecutarse en el ámbito de un objeto Transaction desde Connection. La clase base para todos los objetos Command es **DbCommand**.
DataReader	Lee un flujo de datos de solo avance y solo lectura desde una fuente de datos. La clase base para todos los objetos DataReader es **DbDataReader**.
DataAdapter	Llena un DataSet y realiza las actualizaciones necesarias en una fuente de datos. La clase base para todos los objetos DataAdapter es **DbDataAdapter**.

Junto con las clases principales citadas en la tabla anterior, los proveedores de datos .NET incluyen los siguientes objetos:

Objeto	Descripción
Transaction	Incluye operaciones de actualización en las transacciones que se realizan en el origen de datos. ADO.NET es también compatible con las transacciones que usan clases en el espacio de nombres System.Transactions.
CommandBuilder	Un objeto auxiliar que genera automáticamente las propiedades de comando de un **DataAdapter** o que obtiene de un procedimiento almacenado información acerca de parámetros con la que puede rellenar la colección **Parameters** de un objeto **Command**.
Parameter	Define los parámetros de entrada y salida para los comandos y procedimientos almacenados
ConnectionStringBuilder	Un objeto auxiliar que proporciona un modo sencillo de crear y administrar el contenido de las cadenas de conexión utilizadas por los objetos **Connection**.
Exception	Se devuelve cuando se detecta un error en el origen de dato
ClientPermission	Se proporciona para los atributos de seguridad de acceso del código de los proveedores de datos

DATASET EN ADO.NET

El objeto DataSet es esencial para la compatibilidad con escenarios de datos distribuidos desconectados con ADO.NET.

El objeto DataSet es una representación residente en memoria de datos que proporciona un modelo de programación relacional coherente independientemente del origen de datos. Se puede utilizar con muchos y distintos orígenes de datos, con datos XML o para administrar datos locales de la aplicación.

El DataSet representa un conjunto completo de datos que incluye tablas relacionadas y restricciones, así como relaciones entre las tablas. En la siguiente ilustración se muestra el modelo de objetos DataSet.

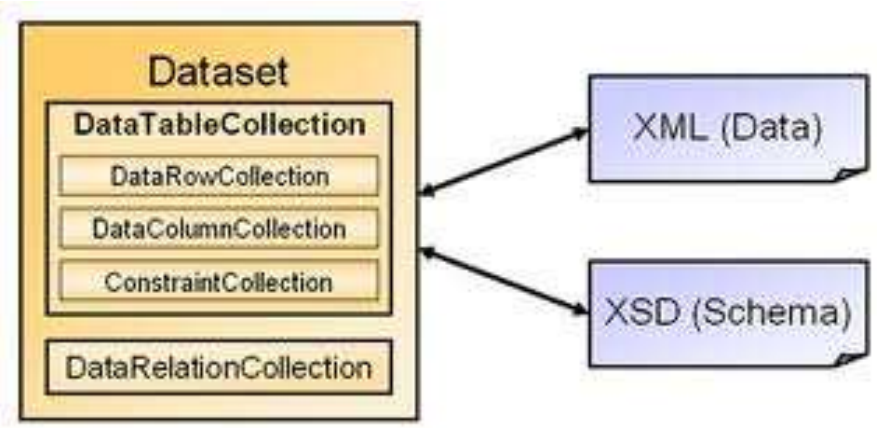


Figura 2: DataSet
Referencia: edn.embarcadero.com

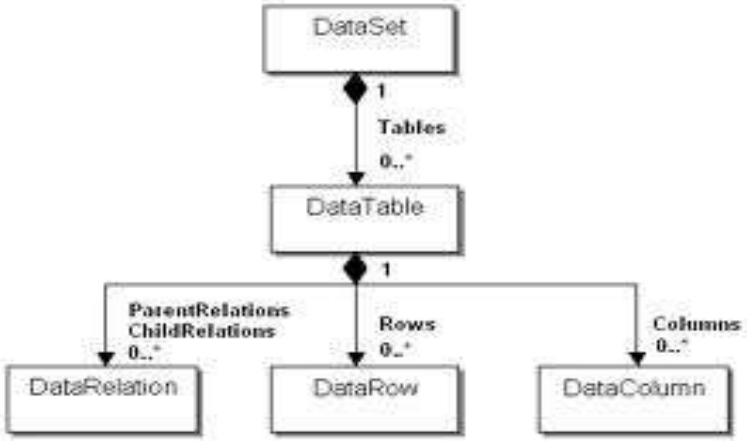


Figura 3: DataSet y Componenets
Referencia: www.c-sharpcorner.com

1.2 MANEJANDO UNA CADENA DE CONEXIÓN A UN ORIGEN DE DATOS

En ADO.NET se utiliza un objeto **Connection** para conectar con un determinado origen de datos mediante una cadena de conexión en la que se proporciona la información de autenticación necesaria. El objeto Connection utilizado depende del tipo de origen de datos.

Cada proveedor de datos .NET Framework incluye un objeto Connection:

- Proveedor de datos para OLE DB incluye un objeto OleDbConnection,
- Proveedor de datos para SQL Server incluye un objeto SqlConnection,
- Proveedor de datos para ODBC incluye un objeto OdbcConnection y
- Proveedor de datos para Oracle incluye un objeto OracleConnection.

Conectar a SQL Server mediante ADO.NET

Para conectarse a Microsoft SQL Server 7.0 o posterior, utilice el objeto SqlConnection del proveedor de datos .NET Framework para SQL Server. El proveedor de datos .NET Framework para SQL Server admite un formato de cadena de conexión que es similar al de OLE DB (ADO).

En el ejemplo siguiente se muestra la forma de crear un abrir una conexión a un origen de datos en SQL Server

```
SqlConnection cn = new SqlConnection(" Data Source=(local); Initial Catalog=NorthWind; user id=sa; pwd=sql");

cn.Open()
```

Donde:

Data Source: Origen de datos
Initial Catalog=Nombre de la base de datos
User id=usuario
Pwd=clave

Cerrar una Conexión

Debe cerrar siempre el objeto Connection cuando deje de usarlo. Esta operación se puede realizar mediante los métodos Close o Dispose del objeto Connection.

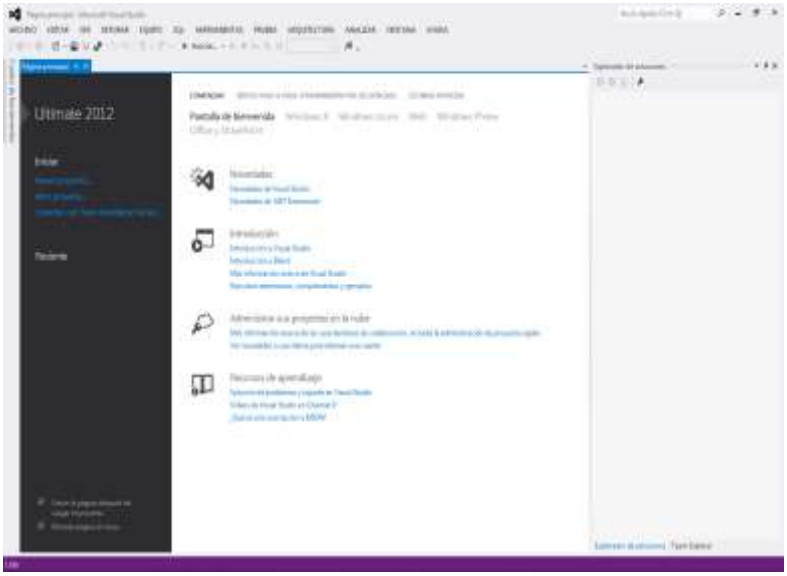
Las conexiones no se liberan automáticamente cuando el objeto Connection queda fuera de ámbito o es reclamado por el garbageCollector.

LABORATORIO 1.1

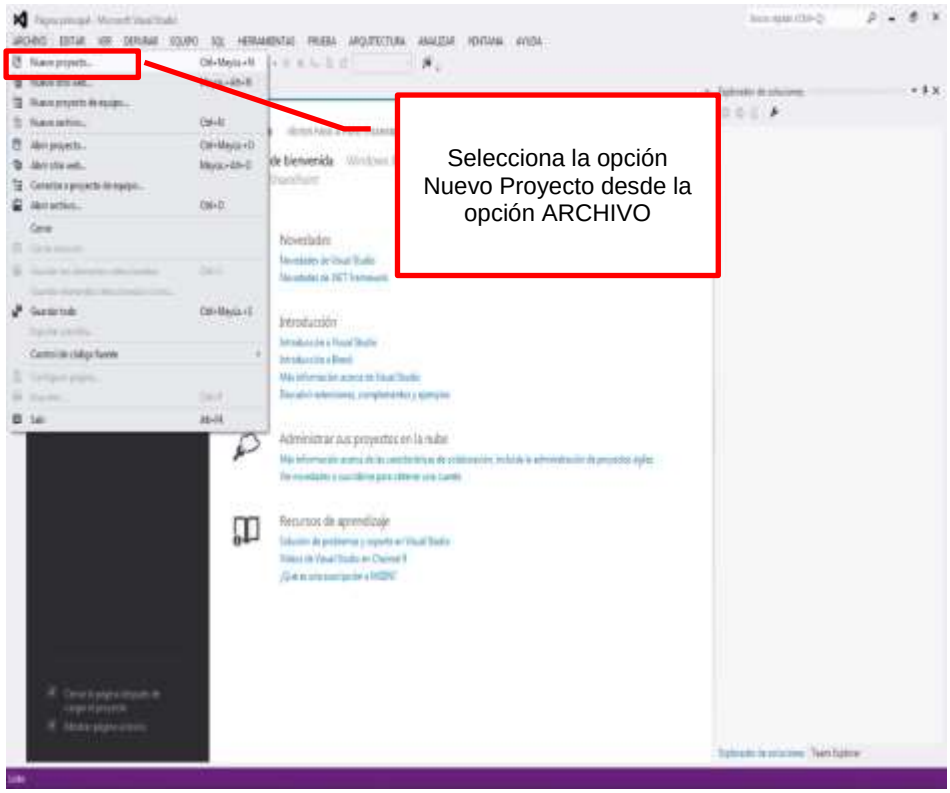
Implemente un Formulario donde Liste todos los Productos almacenados en la base de datos

1. Inicio del Proyecto

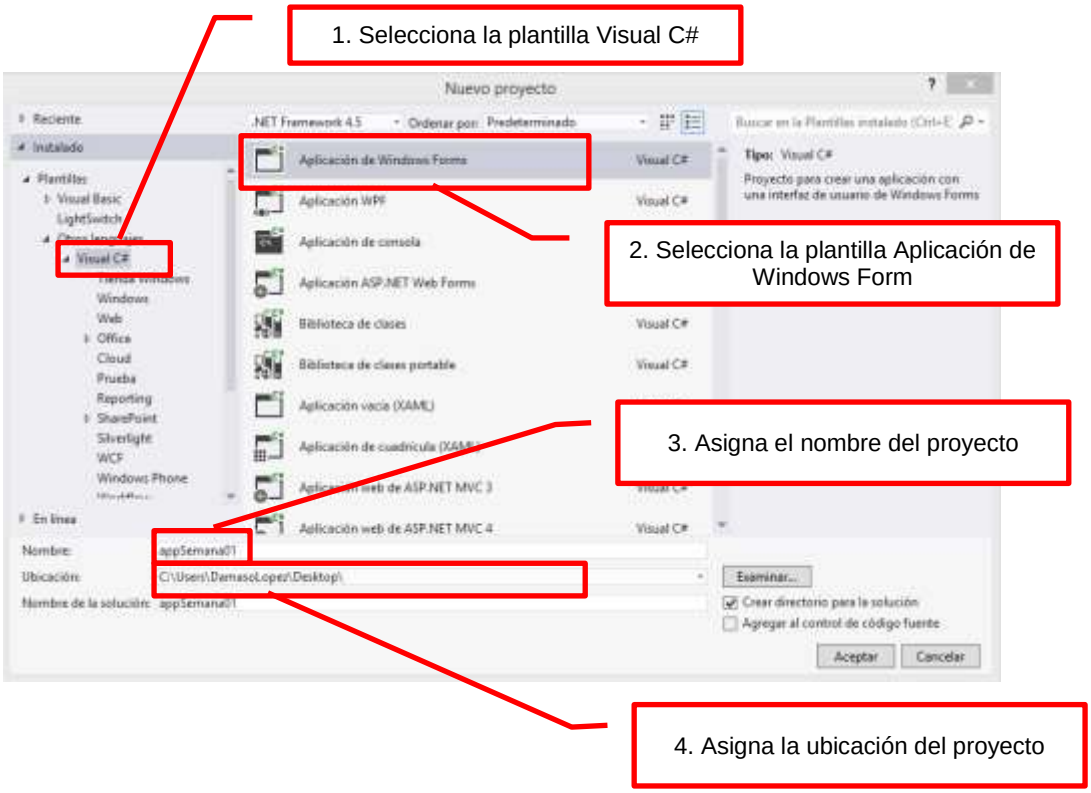
- Cargar la aplicación del el Menú INICIO.
- Seleccione Microsoft Visual Studio 2012 el cual se visualiza el IDE



- Seleccione desde ARCHIVO la opción Nuevo Proyecto desde el IDE



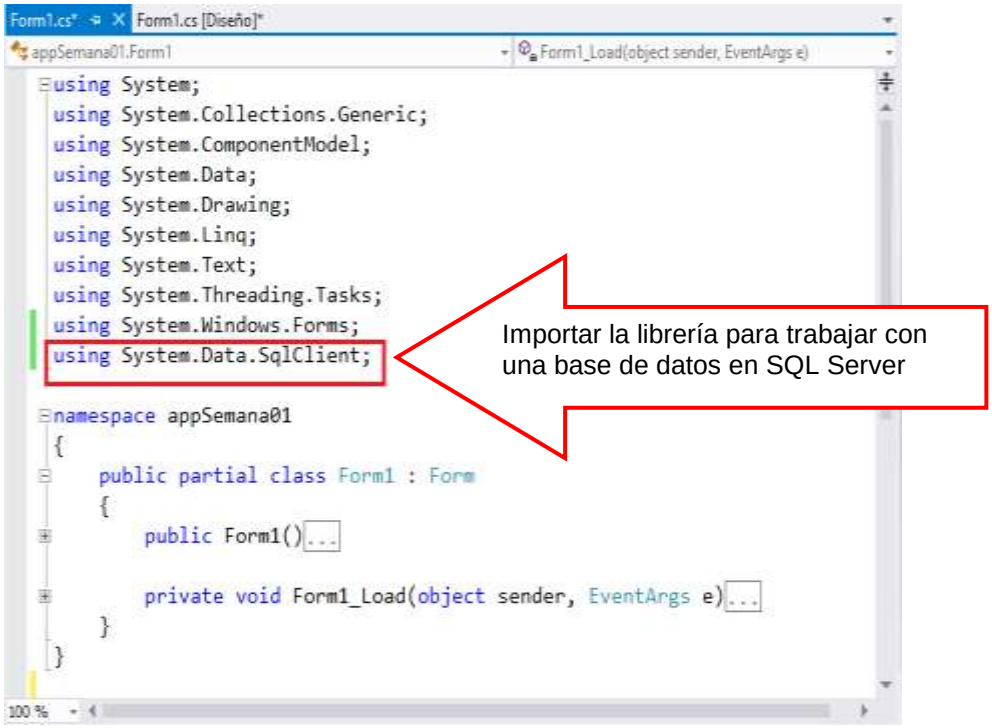
- En la Ventana Nuevo Proyecto , seleccione la plantilla Visual Basic
- Luego, seleccione la plantilla **Aplicación de Windows Forms**
- A continuación, asigne un nombre al proyecto y su ubicación
- Presione el botón ACEPTAR



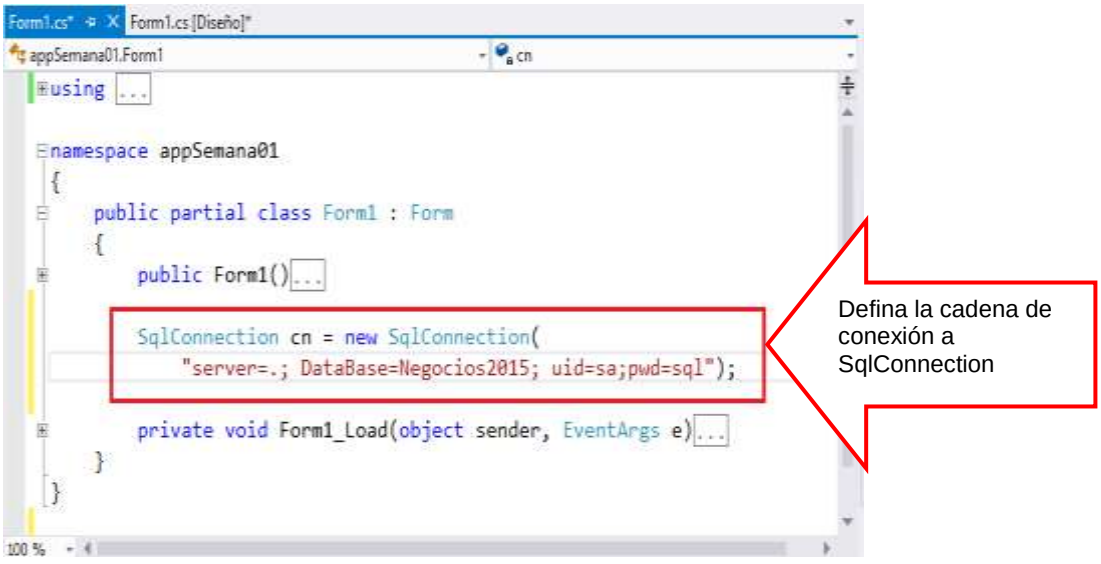
Diseño de Formulario

Programación

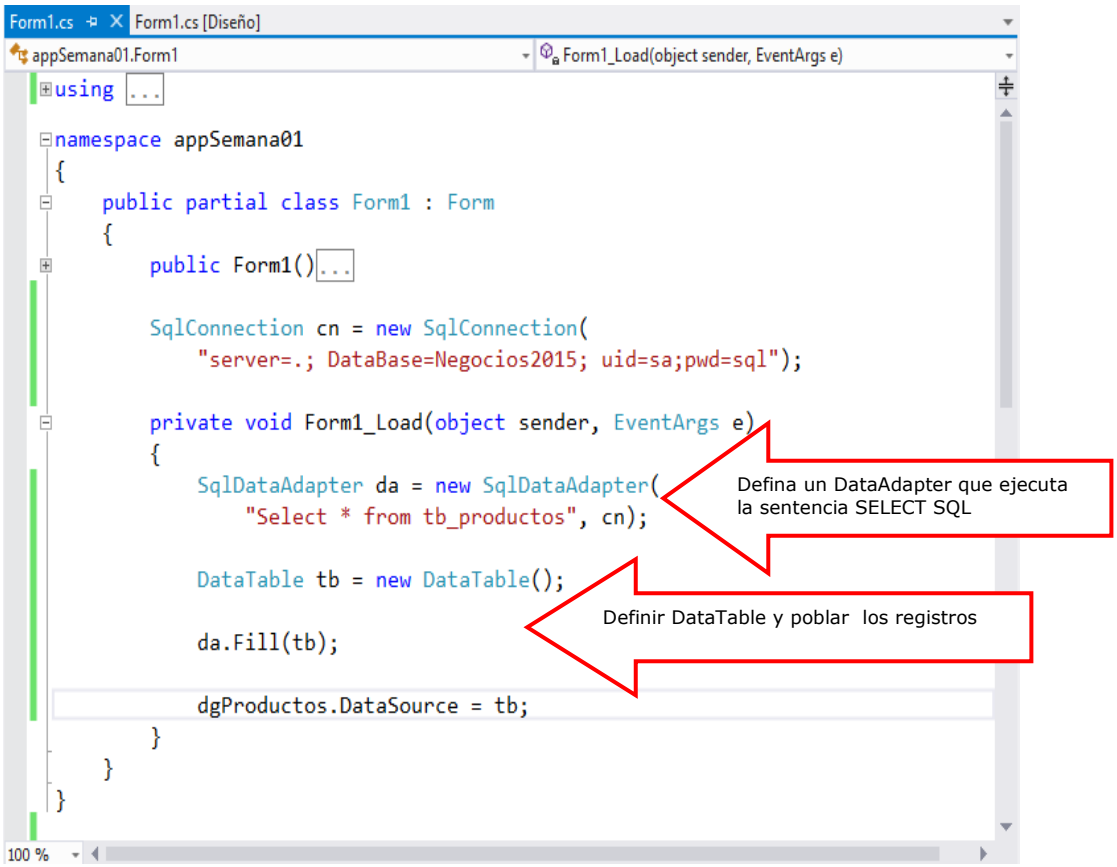
Importar las librerías de trabajo: System.Data.SqlClient, tal como se muestra



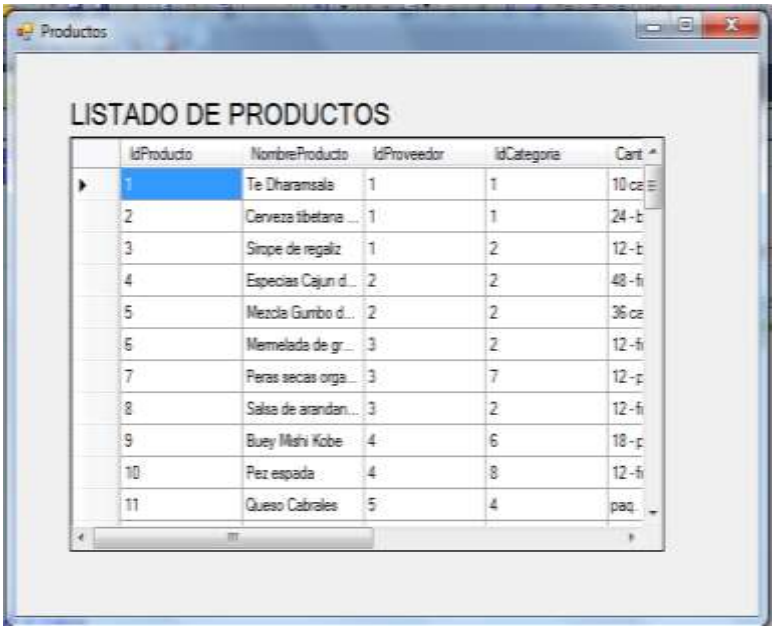
Declare la conexión a la base de datos Negocios2015; el servidor es SUITE101-SER; autenticado a nivel SQL Server: usuario=sa y password=sql



Programa el evento Load del Formulario: ejecuta la sentencia SELECT de tb_productos, los registros resultantes se poblaran en el DataTable tb, mostrando los resultados en el DataGridView



Para ejecutar el formulario presiona la tecla F5, donde se listarán los registros en el control DataGridView

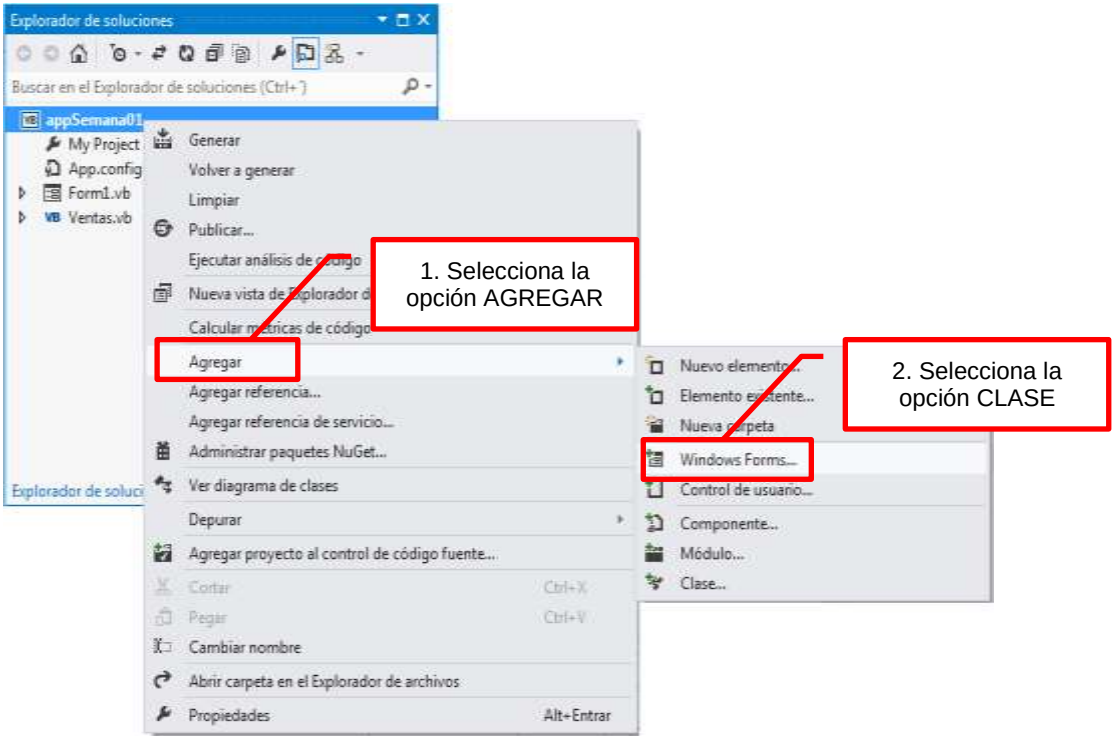


LABORATORIO 1.2

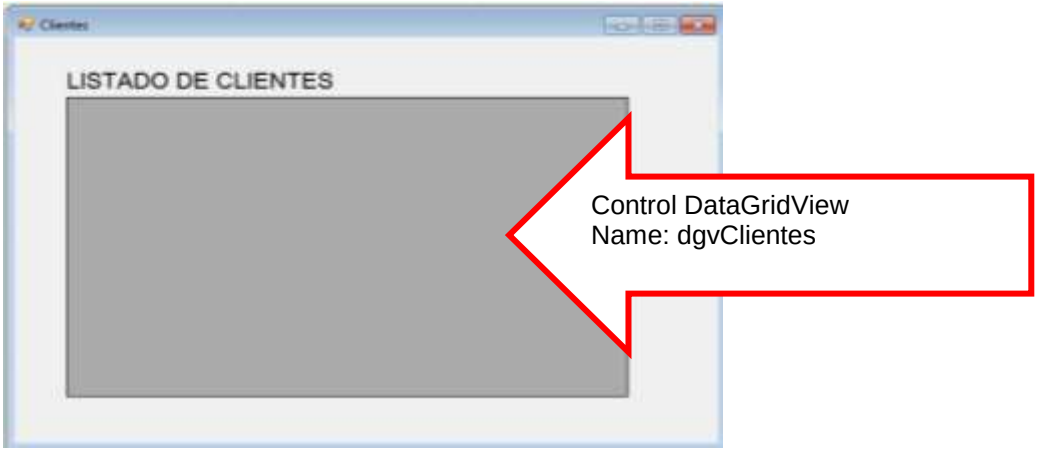
Implemente un Formulario donde Liste todos los Clientes almacenados en la base de datos; considere que en la consulta debe listar el nombre del país correspondiente a cada cliente

1. Para continuar con el mismo proyecto, agregamos un Formulario:

- Hacer clic derecho en el proyecto
- Seleccione la opción Agregar
- Seleccione la opción Windows Forms.
- Al visualizar la ventana Ítem, simplemente presione el botón AGREGAR.

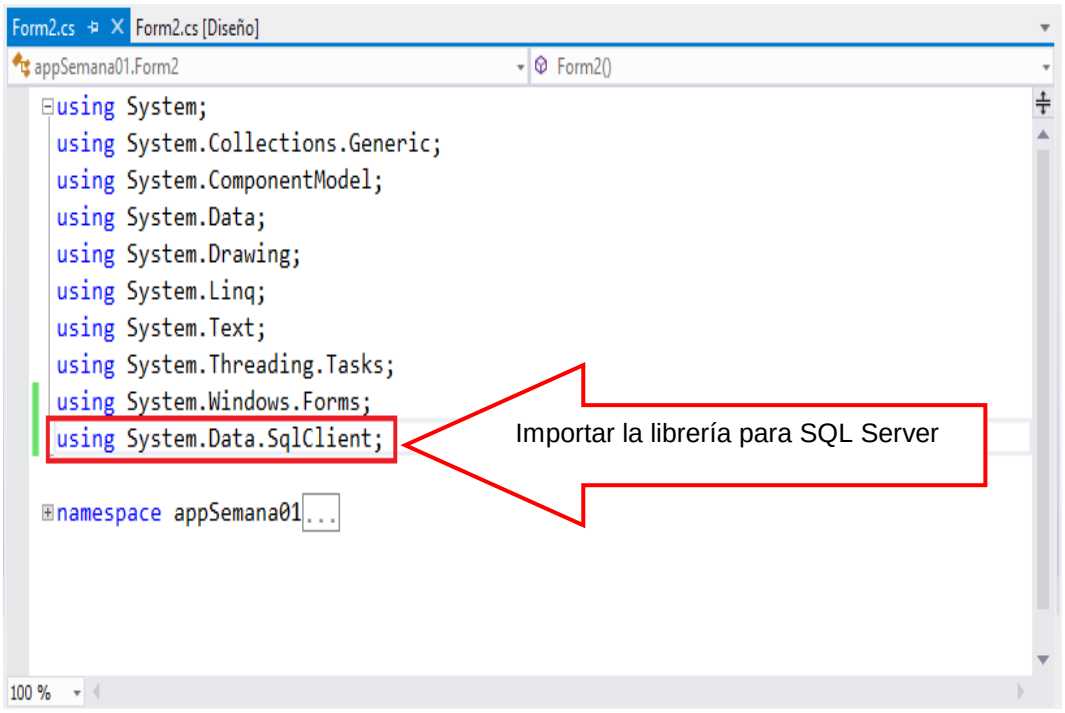


Diseño de Formulario

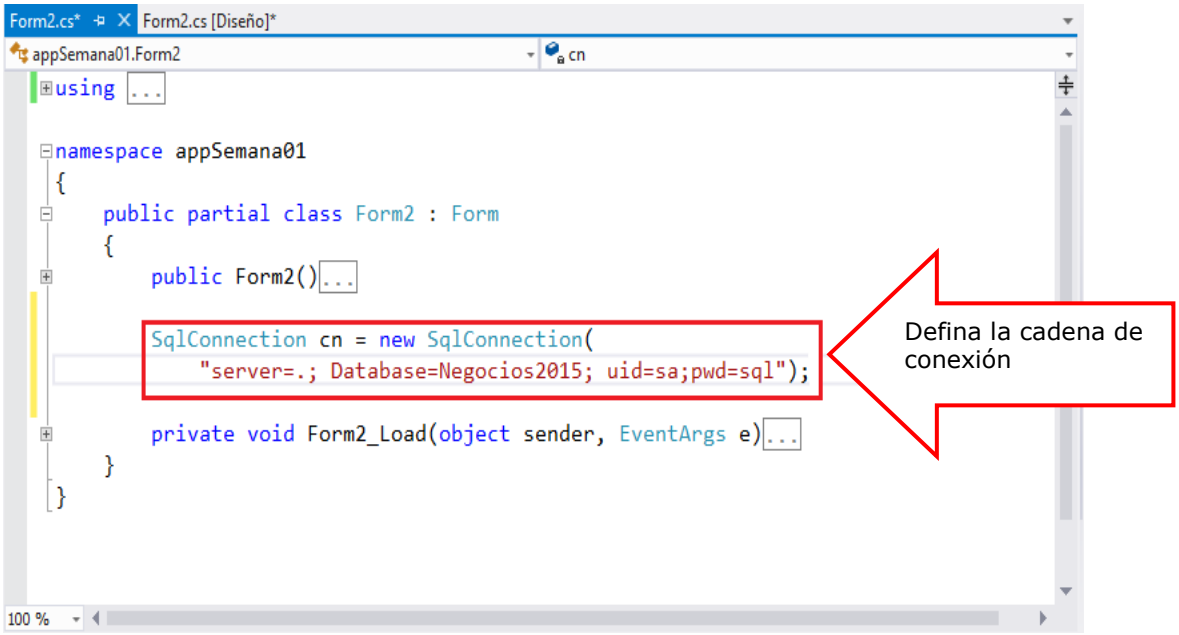


Programación

Importar las librerías de trabajo: Systema.Data.SqlClient, tal como se muestra



A continuación defina la conexión a la base de datos: Negocios2015; autenticando a nivel SQL Server (user id y password)



Programa el evento Load del Formulario, donde al cargar el Formulario: ejecuta la sentencia SELECT SQL de tb_clientes y tb_paises, los registros resultantes se poblaran en el DataTable tb, mostrando los resultados en el DataGridView

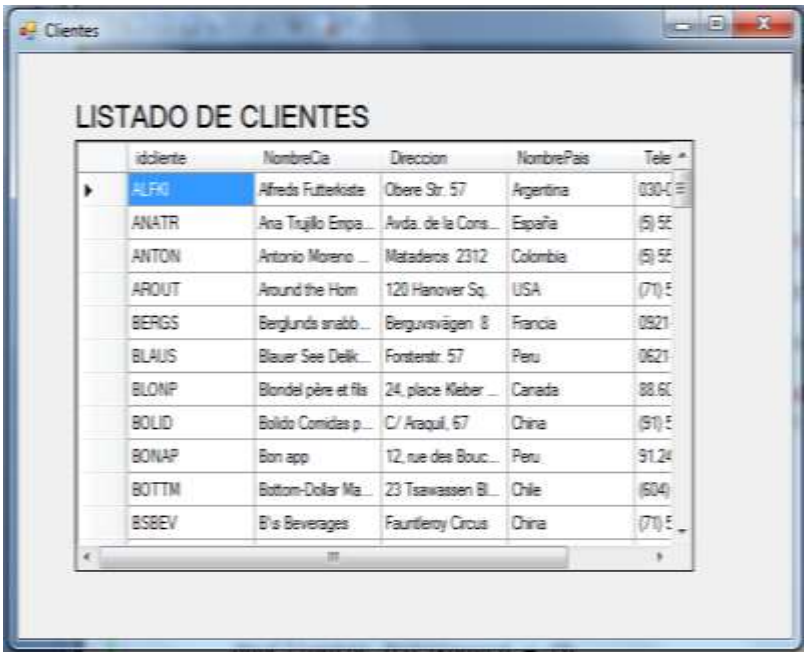
```
using ...
namespace appSemana01
{
    public partial class Form2 : Form
    {
        public Form2()...

        SqlConnection cn = new SqlConnection(
            "server=.; Database=Negocios2015; uid=sa;pwd=sql");

        private void Form2_Load(object sender, EventArgs e)
        {
            SqlDataAdapter da = new SqlDataAdapter(
            "Select idcliente, NombreCia, Direccion, NombrePais, Telefono " +
            "From tb_clientes c inner join tb_paises p " +
            "on p.idpais=c.idpais", cn);

            DataTable tb=new DataTable();

            da.Fill(tb);

            dgClientes.DataSource=tb;
        }
    }
}
```

DataAdapter ejecuta la sentencia SELECT (uso de join)

Definir el DataTable y poblarlo con el método Fill()

Mostrar los datos en el DataGridView

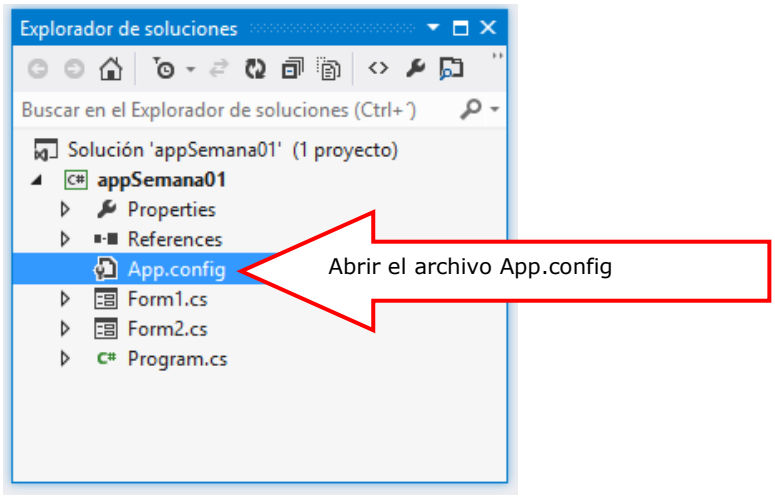
Presiona la tecla F5, donde se ejecuta la consulta de los registros de clientes, visualizando los registros en el control DataGridView



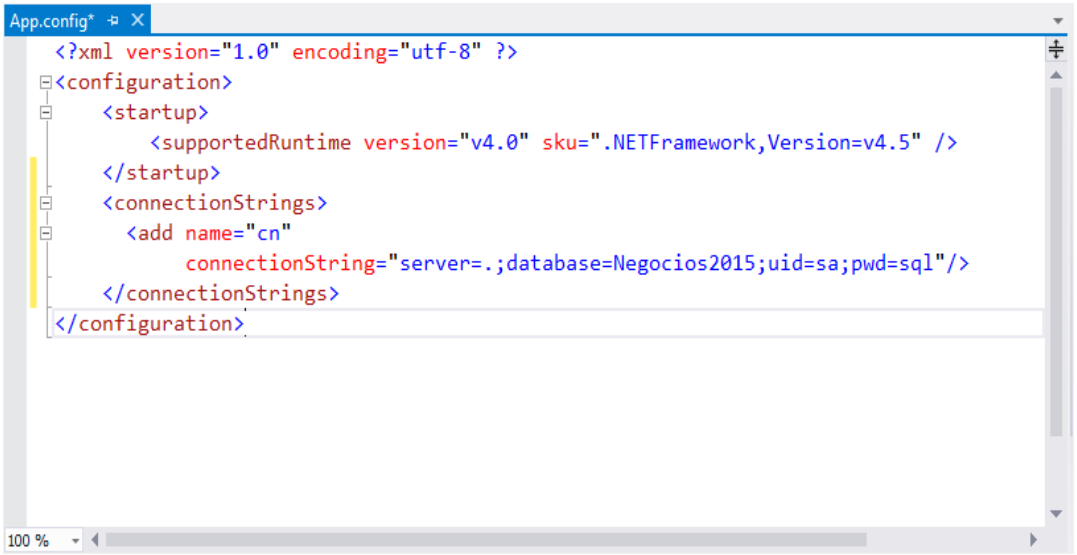
LABORATORIO 1.3

Implemente un Formulario donde Liste todos los Pedidos registrados en la base de datos; considere que la cadena de conexión deberá publicar en el app.config

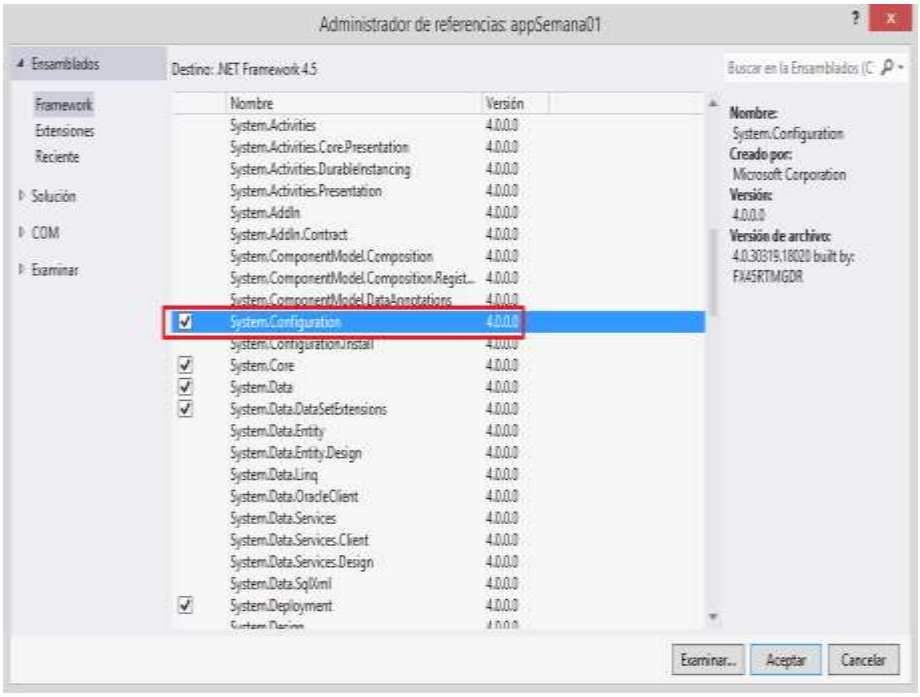
Generar una cadena de conexión pública: Abrir el archivo App.config, del explorador de proyectos, tal como se muestra



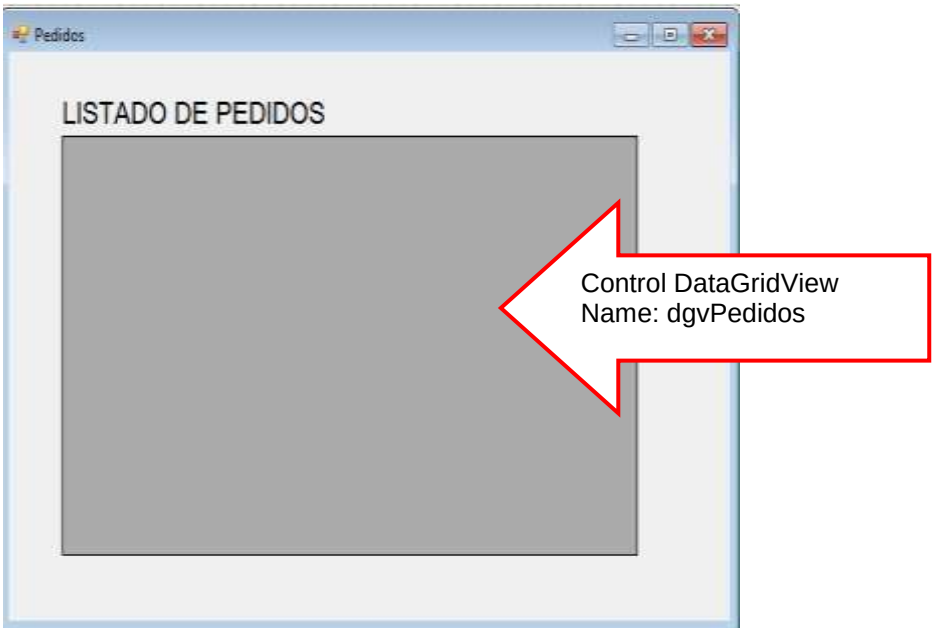
Defina la cadena de conexión dentro de la etiqueta <connectionStrings> en el App.config.



A continuación, agregar la referencia: System.Configuration para trabajar con la conexión publicada en el app.Config. Desde la opción Proyecto, selecciona la opción Agregar Referencia…

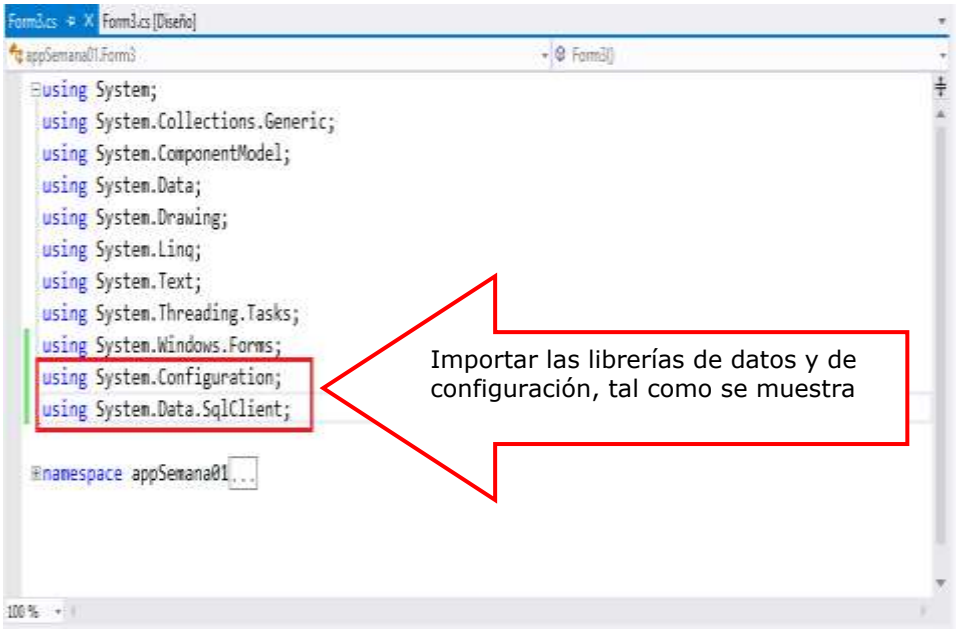


Diseño de Formulario

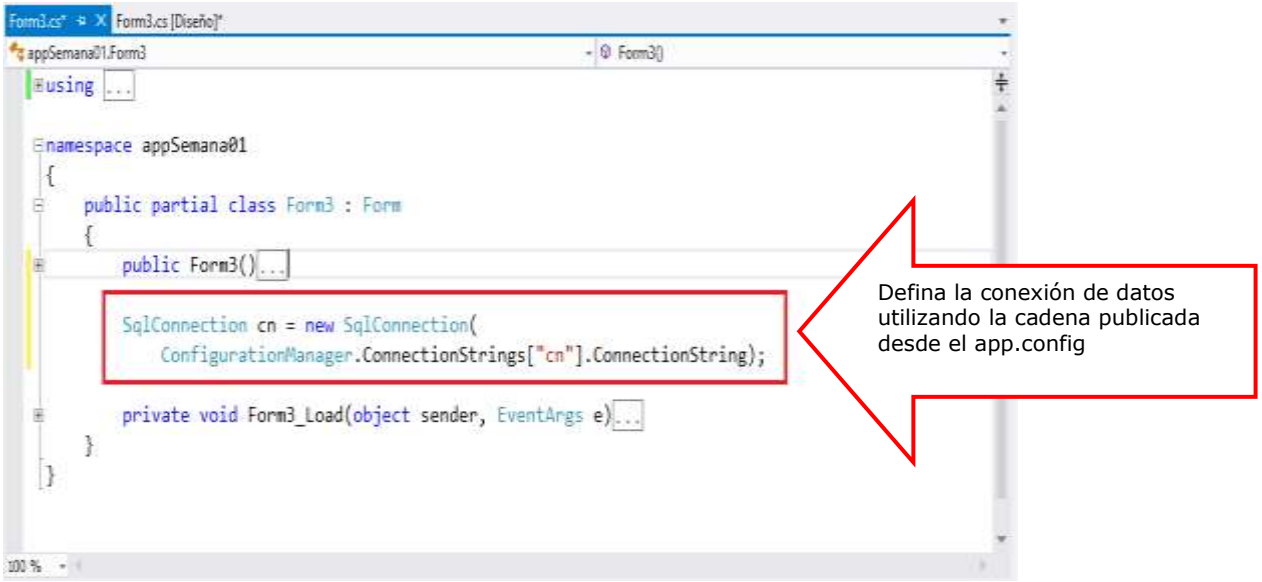


Programación

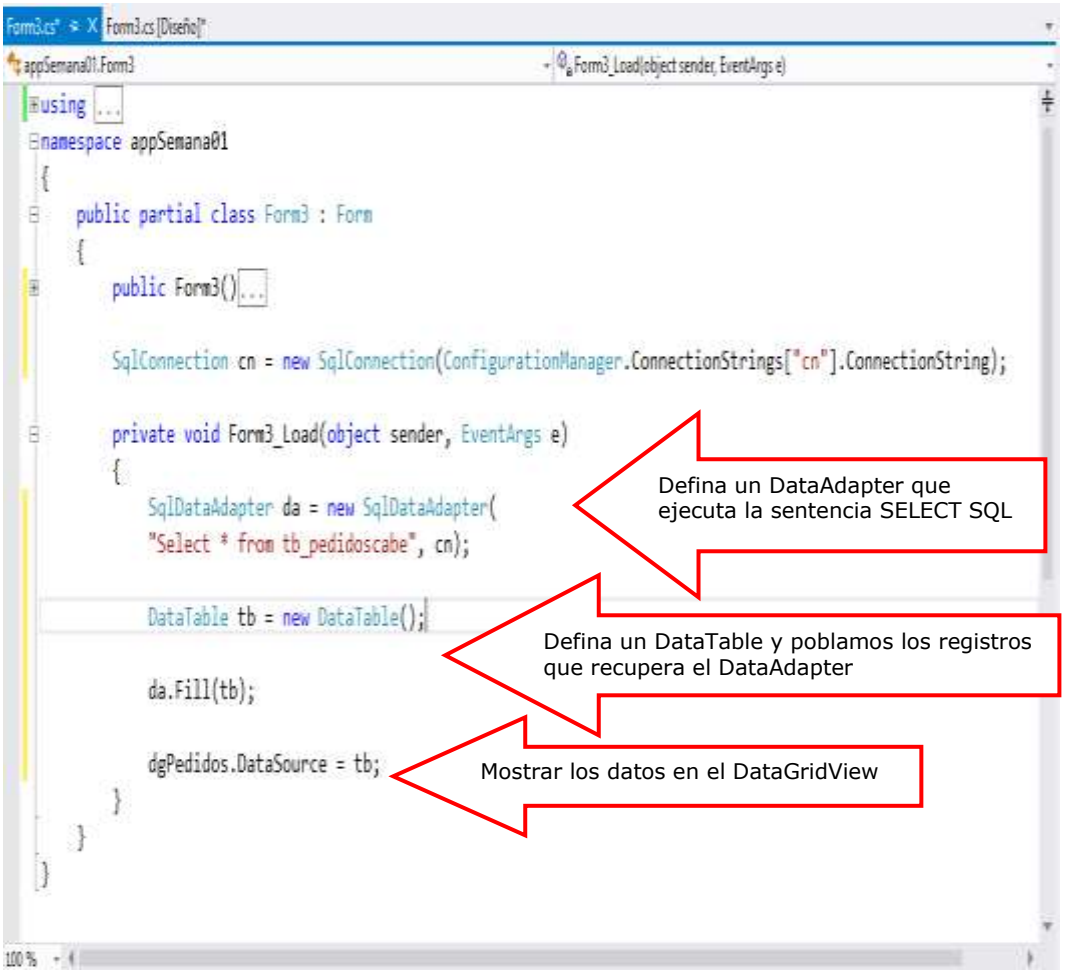
Defina las librerías de trabajo: System.Data.SqlClient (Base de Datos en SQL Server) y el System.Configuration.ConfigurationManager



Defina la conexión a la base de datos Negocios2015 instanciando el SQLConnection



Defina la consulta SQL en el DataAdapter donde retorna los registros de pedidos almacenando los registros en el DataTable, para luego mostrar los registros en el DataGridView1.



Presione F5, para ejecutar el formulario

Resumen

- Tradicionalmente, el procesamiento de datos ha dependido principalmente de un modelo de dos niveles basado en una conexión. A medida que aumenta el uso que hace el procesamiento de datos de arquitecturas de varios niveles, los programadores están pasando a un enfoque sin conexión con el fin de proporcionar una mejor escalabilidad a sus aplicaciones.
- Los dos componente principales de ADO.NET para el acceso a datos y su manipulación son los proveedores de datos .NET Fraework y DataSet.
- Los proveedores de datos .NET Framework sirven para conectarse a una base de datos, ejecutar comandos y recuperar resultados. Los proveedores de datos .NET Framework son ligeros, de manera que crean un nivel mínimo entre el origen de datos y el código, con lo que aumenta el rendimiento sin sacrificar funcionalidad.
- El proveedor de datos .NET Framework para SQL Server utiliza la librería System.Data.SqlClient
- Los principales componentes de un proveedor de datos .NET Framework son:
  - Connection
  - Command
  - DataReader
  - DataAdapter
- El objeto DataSet es esencial para la compatibilidad con escenarios de datos distribuidos desconectados con ADO.NET. Representa un conjunto completo de datos que incluye tablas relacionadas y restricciones; así como relaciones entre las tablas
- En ADO.NET se utiliza un objeto Connection para conectar con un determinado origen de datos mediante una cadena de conexión en la que se proporciona la información de autenticación necesaria. El objeto Connection utilizado depende del tipo de origen de datos.
- Para conectarse a Microsoft SQL Server 7.0 o posterior, utilice el objeto SqlConnection del proveedor de datos .NET Framework para SQL Server. El proveedor de datos .NET Framework para SQL Server admite un formato de cadena de conexión que es similar al de OLE DB (ADO).
- Debe cerrar siempre el objeto Connection cuando deje de usarlo. Esta operación se puede realizar mediante los métodos Close o Dispose del objeto Connection.

- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  http://msdn.microsoft.com/es-es/library/27y4ybxw(v=vs.110).aspx

  http://hampprogramandoando.blogspot.com/2013/05/ado-net-45-parte-iv-consulta-de-datos.html

UNIDAD

1

OPERACIONES CONECTADAS A UN ORIGEN DE DATOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno realiza operaciones de consulta y actualización de datos en el entorno de una aplicación Windows conectado a un origen de datos utilizando la librería ADO.NET

TEMARIO

Tema 2: Recuperación de datos (4 horas)

2.1 Consultando datos sobre un origen de datos
- 2.1.1 Realizando una consulta de datos utlizando la clase DataAdapter
- 2.1.2 Realizando una consulta de datos utilizando parámetros en el proceso
- 2.1.3 Trabajando con el DataReader.

ACTIVIDADES PROPUESTAS

- Los alumnos diseñan formularios que manejan la conexión a una fuente de datos para realizar operaciones de consulta a la fuente de datos.
- Los alumnos desarrollan los laboratorios de esta semana

2. RECUPERACIÓN DE DATOS EN ADO.NET

La función principal de cualquier aplicación que trabaje con una fuente de datos es conectarse a dicha fuente de datos y recuperar los datos que se almacenan.

Los proveedores de .NET Framework para ADO.NET sirven como puente entre una aplicación y un origen de datos, permitiendo ejecutar instrucciones SQL para recuperar datos mediante el DataAdapter o el DataReader.

2.1. REALIZANDO UNA CONSULTA CON DATAADAPTER

La clase DataAdapter se encarga de las operaciones entre la capa de datos y la capa intermedia donde los datos son transferidos. Se puede decir que sirve como puesnte entre un objeto DataSet y un origen de datos asociados para recuperar y guardar datos.

Básicamente, permite rellenar (Fill) el objeto DataSet para que sus datos coincidan con los del origen de datos y permite actualizar (Update) el origen de datos para que sus datos coincidan con los del DataSet.

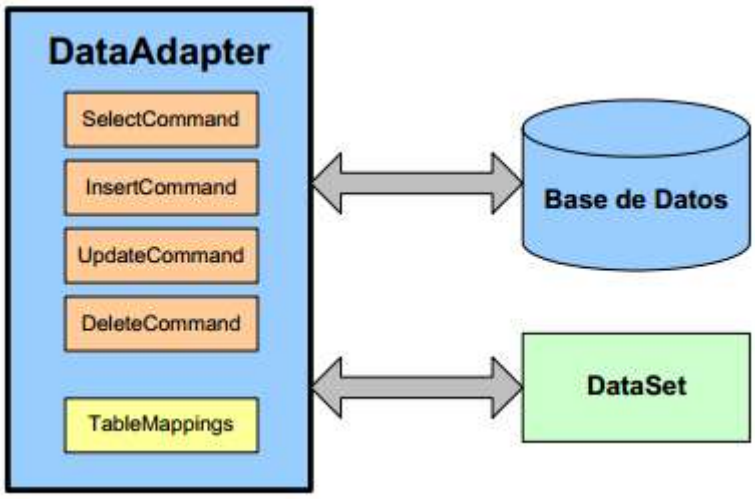


Figura 1: Diagrama del objeto DataAdapter
Referencia: http://ehu.es/mrodriguez/archivos/csharppdf/ADONET/ADONET.pdf

Los constructores de la clase son:

- DataAdapter (Command selectCommand). Utiliza un comando de selección como parámetro.
- DataAdapter (string selectText, string selectConnectionString). Se utiliza una sentencia SQL de selección y una cadena de conexión como parámetros
- DataAdapter (string selectText, Connection selectConnection). Se utiliza los parámetros sentencia SQL de selección y un objeto de tipo conexión.

En la interface IDataAdapter se declaran los siguientes métodos (toda clase que implemente esta interface está obligata a implementarlos)

- Fill (DataSet). Rellena un objeto de la clase DataSet

- FillSchema (Dataset, Tipo de schema). Rellena un esquema con un DataSet indicando el tipo de esquema a rellenar
- Update (DataSet). Actualiza el DataSet correspondiente
- GetFillParameters(). Recoge el conjunto de parámetros cuando se ejecuta una consulta de selección.

Cada proveedor de datos de .NET FrameWork cuenta con un objeto DataAdapter:

Proveedor	Descripcion
OledbDataAdapter	Proveedor de datos para OLEDB
SqlDataAdapter	Proveedor de datos para SQL Server
OdbcDataAdapter	Proveedor de datos para ODBC
OracleDataAdapter	Proveedor de datos para Oracle

2.2 REALIZANDO UNA CONSULTA CON PARÀMETROS

Cuando el DataAdapter ejecuta instrucciones de consulta con parámetros, se deben definen qué parámetros de entrada y de salida se deben crear. Para crear un parámetro en un DataAdapter, se utiliza el método: Parameters.Add() o Parameteres.AddWithValue().

El método **Parameters.Add()** se debe especificar el nombre de columna, tipo de datos y tamaño, asignando el valor del parámetro definido. El método Add de la colección Parameters toma el nombre del parámetro, el tipo de datos, el tamaño (si corresponde al tipo) y el nombre de la propiedad SourceColumn de DataTable

El método **Parameters.AddWithValue()** se debe especificar el nombre del parámetro y su valor.

2.3 TRABAJANDO CON DATAREADER

La recuperación de datos mediante DataReader implica crear una instancia del objeto Command y de un DataReader a continuación, para lo cual se llama a **Command**.**ExecuteReader** a fin de recuperar filas de un origen de datos.

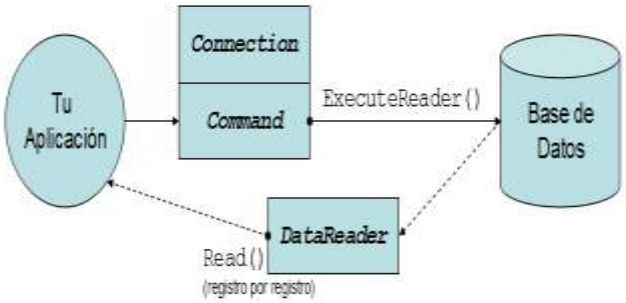


Figura 1: Diagrama del objeto DataReader
Referencia: http://www.dotnetero.com/2006/08/adonet-para-novatos.html

Puede utilizar el método **Read** del objeto DataReader para obtener una fila a partir de los resultados de una consulta.Para tener acceso a cada columna de la fila devuelta, puede pasar a DataReader el nombre o referencia numérica de la columna en cuestión.Sin embargo, el mejor rendimiento se logra con los métodos que ofrece DataReader y que permiten tener acceso a los valores de las columnas en sus tipos de datos nativos (GetDateTime, GetDouble, GetGuid, GetInt32, etc.).

Para obtener una lista de métodos de descriptor de acceso con tipo para **DataReaders** de proveedores de datos:

Proveedor	Descripcion
OledbDataReader	Proveedor de datos para OLEDB
SqlDataReader	Proveedor de datos para SQL Server
OdbcDataReader	Proveedor de datos para ODBC
OracleDataReader	Proveedor de datos para Oracle

Cerrar el DataReader

Siempre debe llamar al método **Close** cuando haya terminado de utilizar el objeto DataReader.

Si Command contiene parámetros de salida o valores devueltos, éstos no estarán disponibles hasta que se cierre el **DataReader**.

Tenga en cuenta que mientras está abierto un **DataReader**, ese **DataReader** utiliza de forma exclusiva el objeto **Connection**. No se podrá ejecutar ningún comando para el objeto **Connection** hasta que se cierre el **DataReader** original, incluida la creación de otro **DataReader**.

Recuperar varios conjuntos de resultados con NextResult

En el caso en que se devuelvan varios resultados, el DataReader proporciona el método **NextResult** para recorrer los conjuntos de resultados en orden.

Obtener información del esquema a partir del DataReader

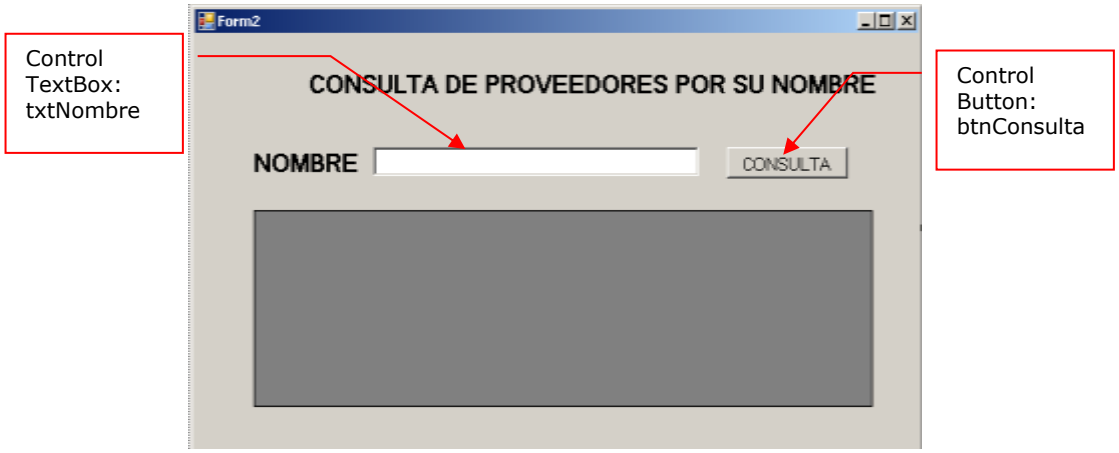
Mientras hay abierto un **DataReader**, puede utilizar el método **GetSchemaTable** para recuperar información del esquema acerca del conjunto actual de resultados. **GetSchemaTable** devuelve un objeto **DataTable** rellenado con filas y columnas que contienen la información del esquema del conjunto actual de resultados.

DataTable contiene una fila por cada una de las columnas del conjunto de resultados. Cada columna de una fila de la tabla de esquema está asociada a una propiedad de la columna que se devuelve en el conjunto de resultados. **ColumnName** es el nombre de la propiedad y el valor de la columna es el de la propiedad. En el ejemplo de código siguiente se muestra la información del esquema de **DataReader**.

LABORATORIO 2.1

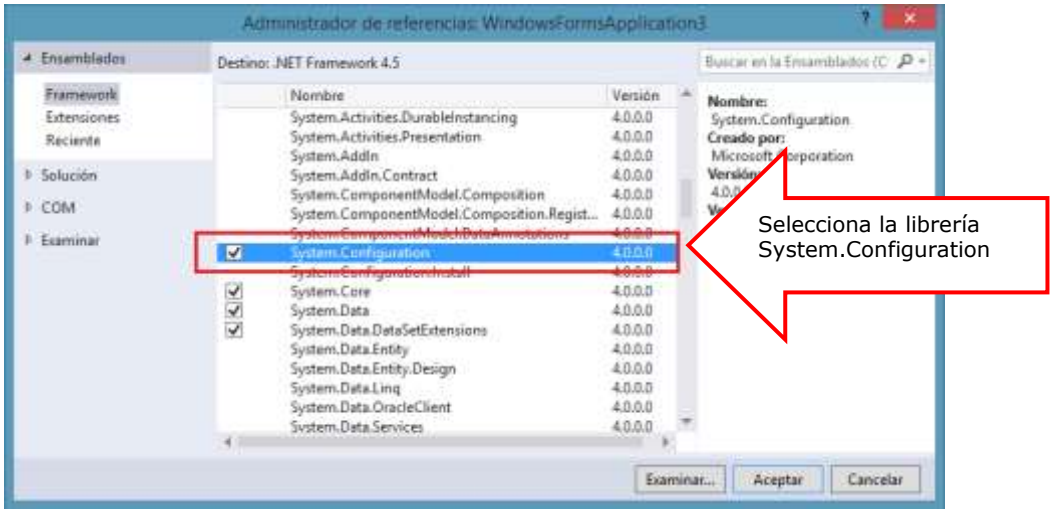
Implemente un Formulario donde visualice los registros de los proveedores que coincida con las iniciales de su nombre ingresado por teclado.

Diseño de Formulario



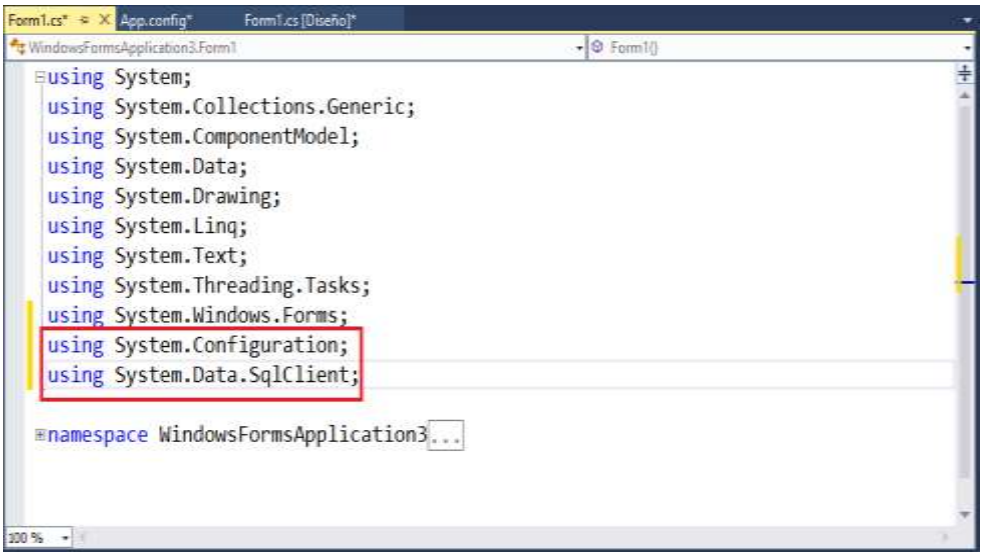
Defina la cadena de conexión en el app.config, en la etiqueta <connectionStrings>

Desde la opción Proyecto, selecciona la opción Agregar Referencia.Selecciona System.Configuration para trabajar con la conexión publicada en el app.Config

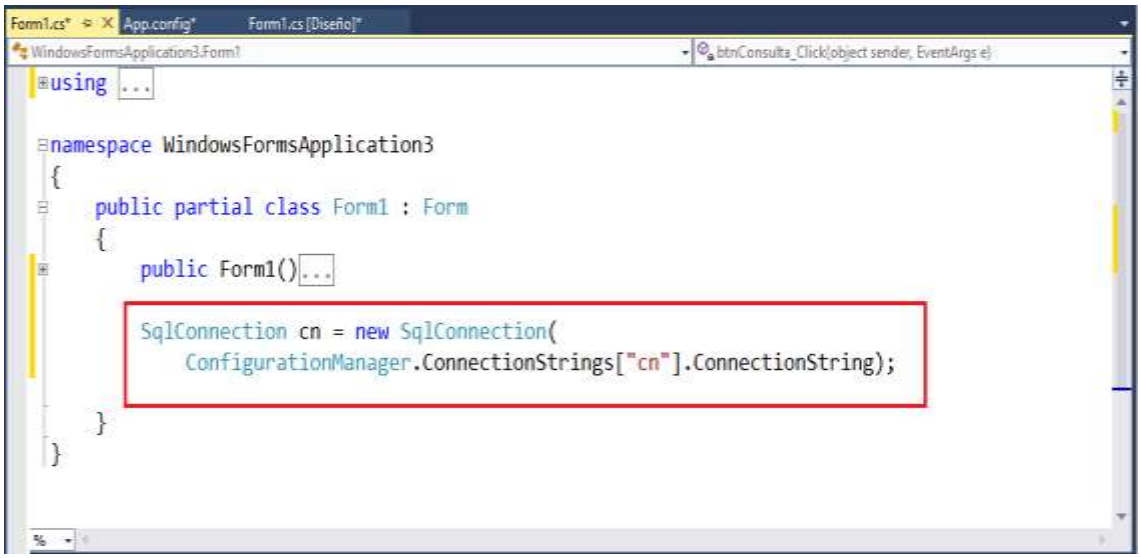


PROGRAMACION

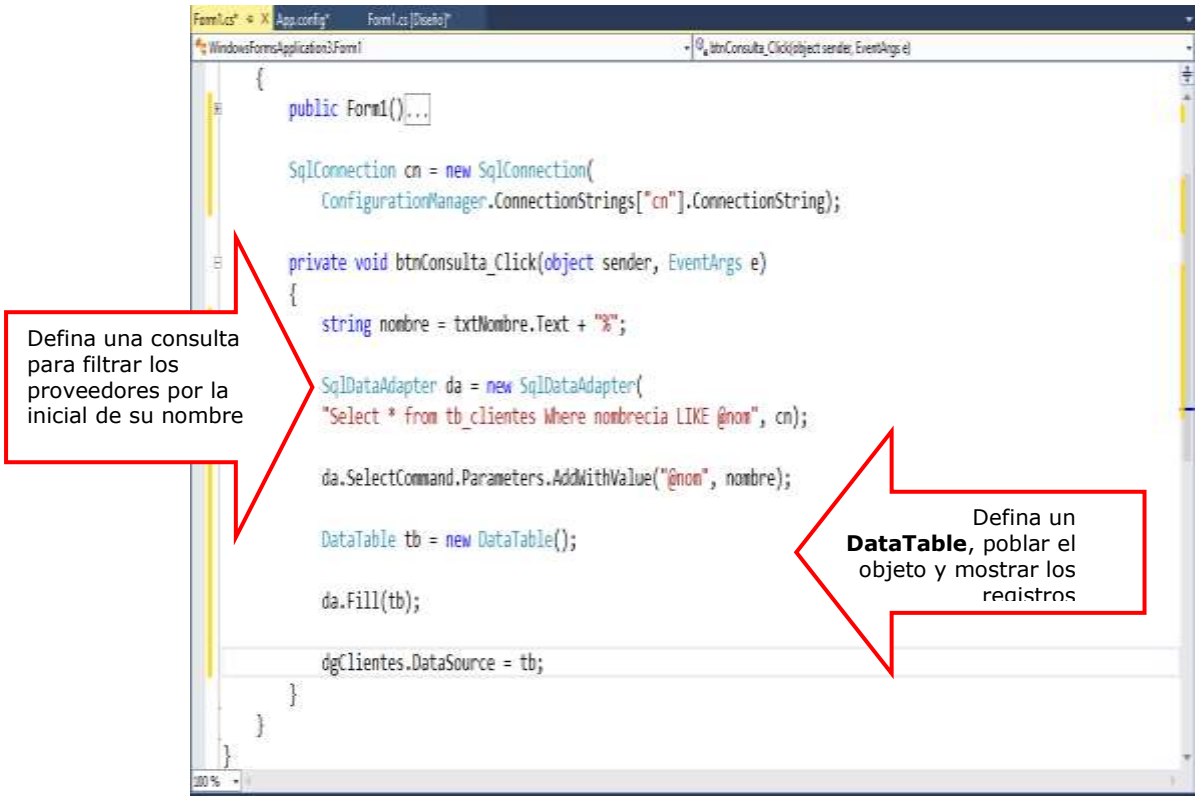
Defina las librerías de trabajo: **System.Data.SqlClient** (Base de Datos en SQL Server) y el **System.Configuration.ConfigurationManager**



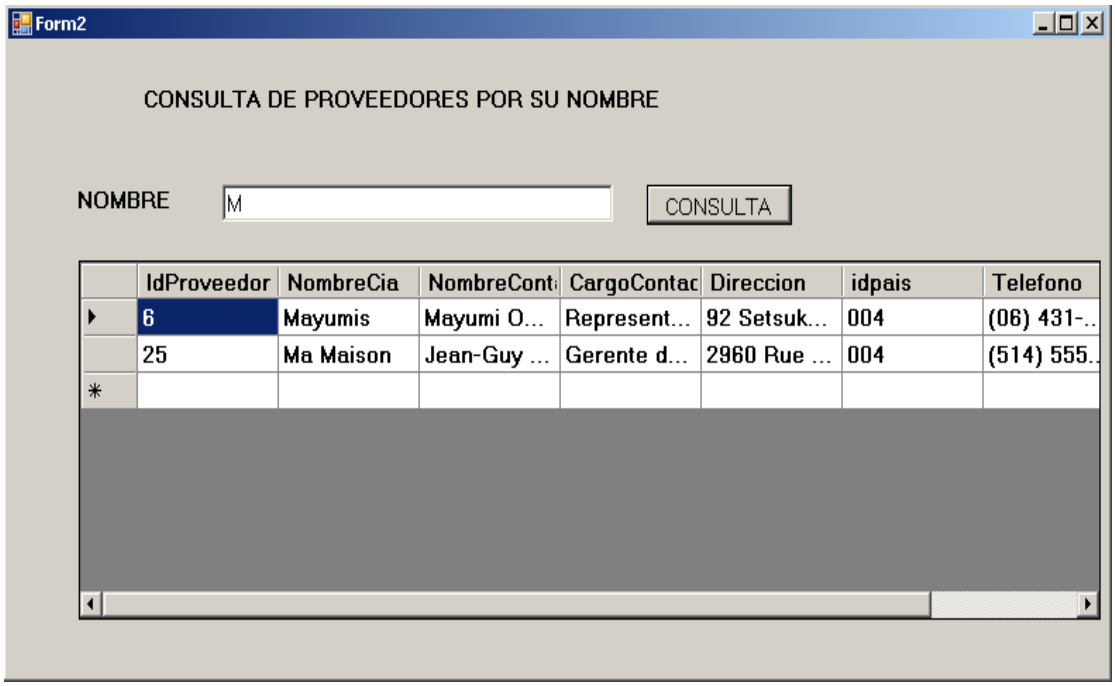
Defina la conexión a la base de datos utilizando la conexión publicada en el app.Config, tal como se muestra



Programa el evento Click del botón Consulta, defina la sentencia SQL que filtre los proveedores por su campo NombreCia.



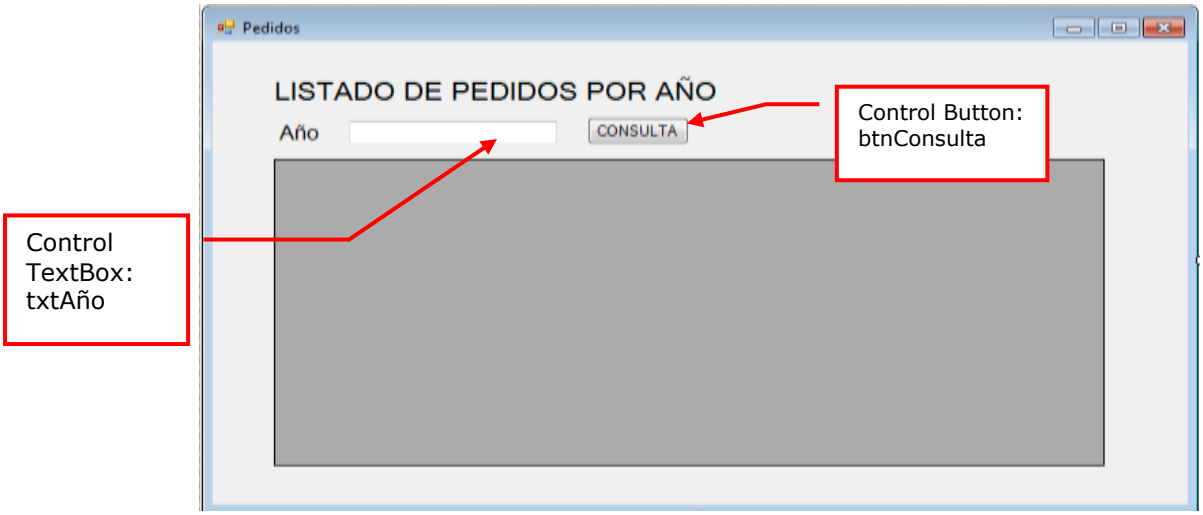
Ejecuta el formulario, presiona la tecla F5; ingresa un valor o inicial del nombre en el control TextBox: txtnombre y presione el botón Consulta.



LABORATORIO 2.2

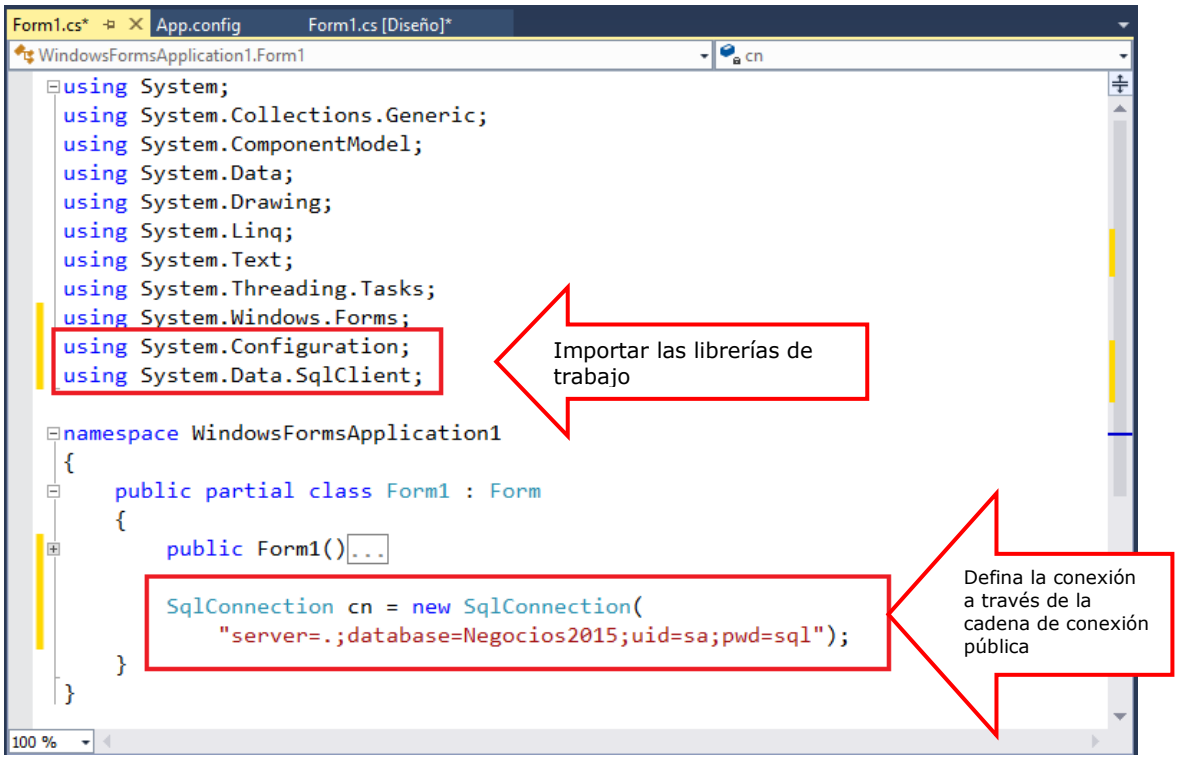
Implemente un Formulario donde visualice los registros de los pedidos por un determinado año ingresado por teclado.

Diseño de Formulario

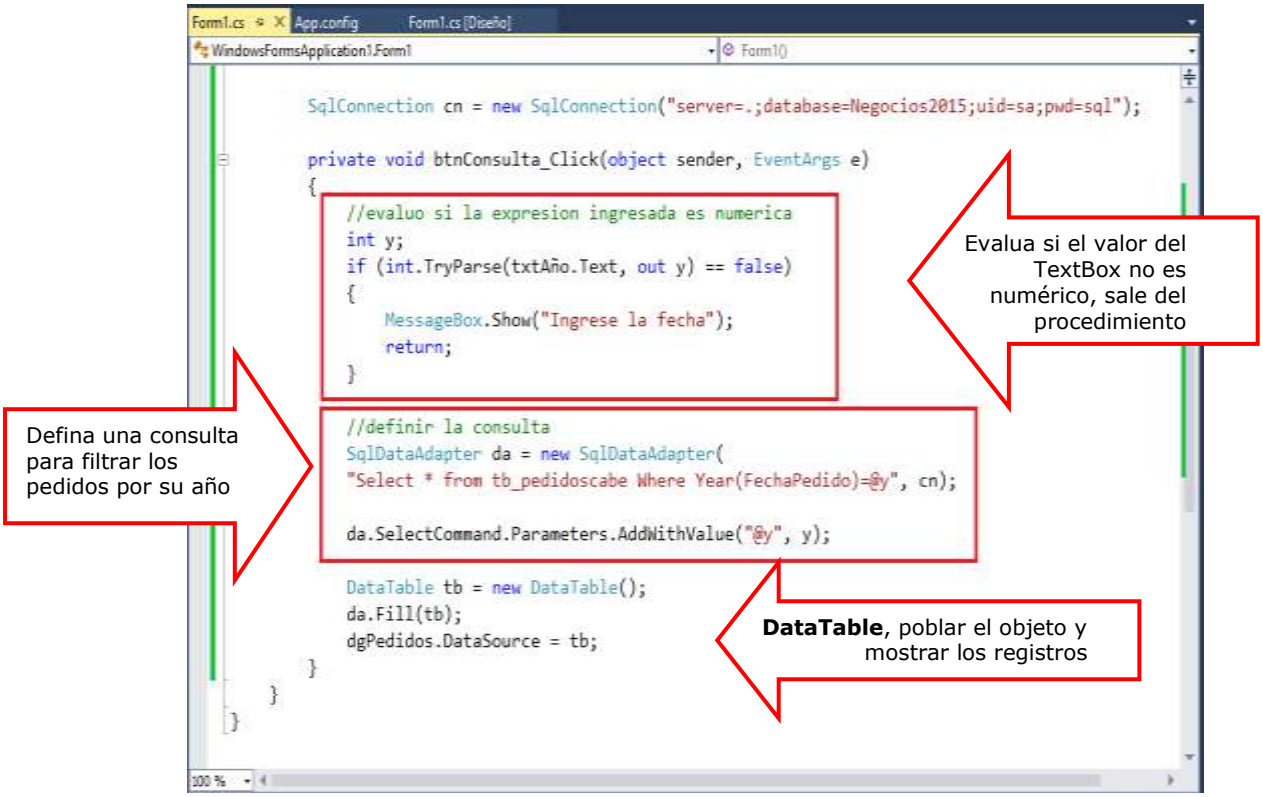


PROGRAMACION

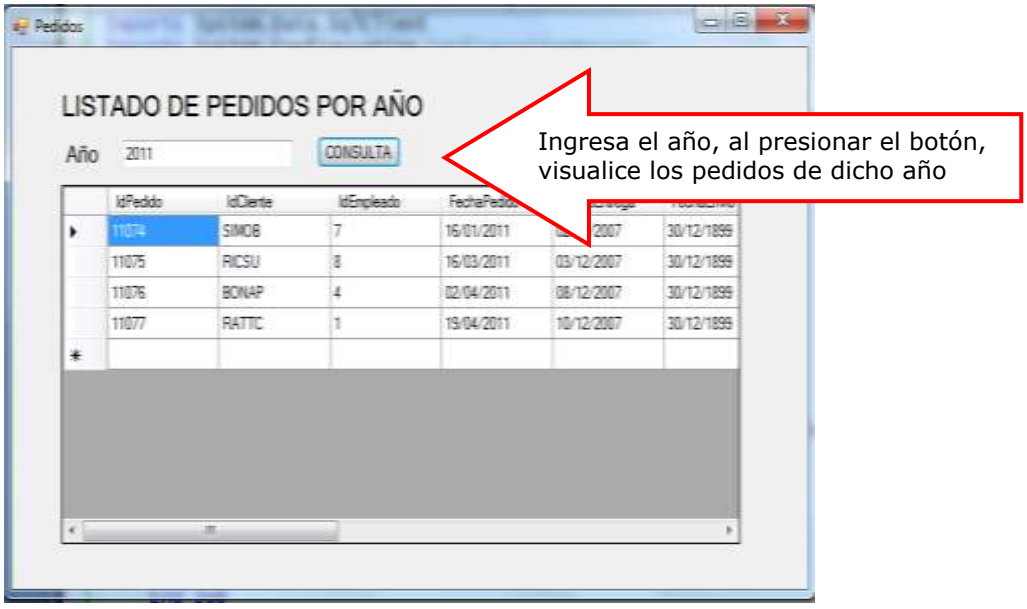
Defina las librerías de trabajo: **System.Data.SqlClient** (Base de Datos en SQL Server) y el **System.Configuration.ConfigurationManager**



Programa el evento Click del botón Consulta para ejecutar la consulta de registros de tb_pedidoscabe por un determinado año de FechaPedido, ingresado desde el control textBox, comparando con la columna Year(FechaPedido)



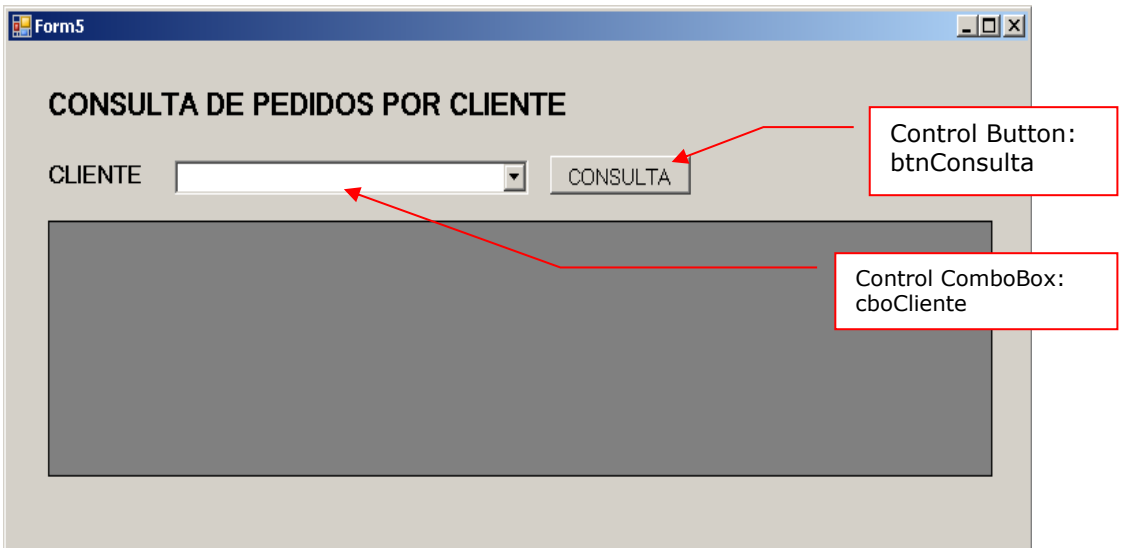
Presiona la tecla F5, ingrese el año desde el TextBox, al presionar el botón Consulta visualiza los pedidos por el año de la fechaPedido ingresado



LABORATORIO 2.3

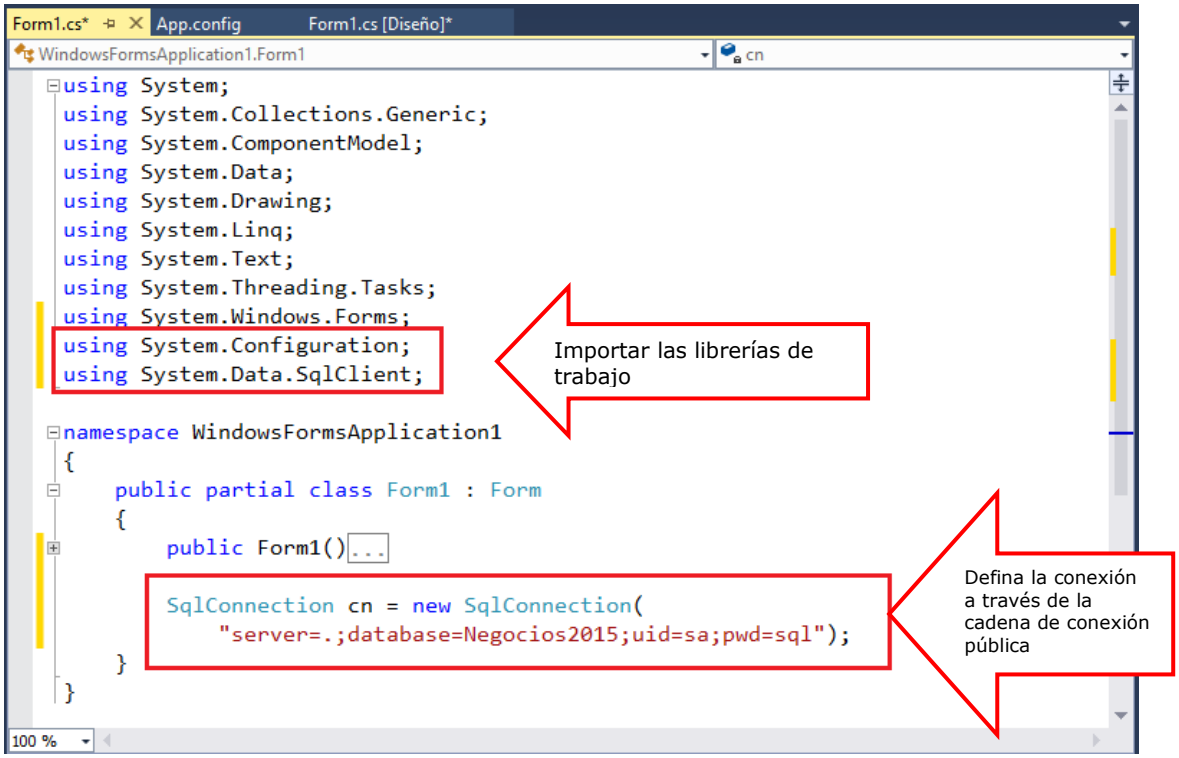
Implemente un Formulario donde liste todos los PedidosCabe de un cliente seleccionado desde un control ComboBox

Diseño de Formulario

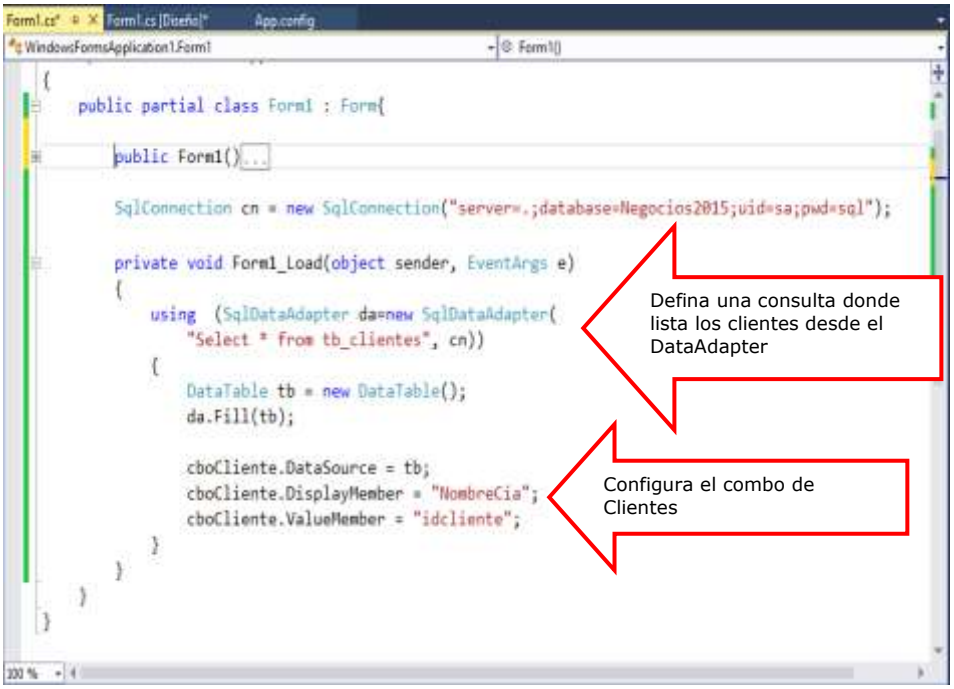


PROGRAMACION

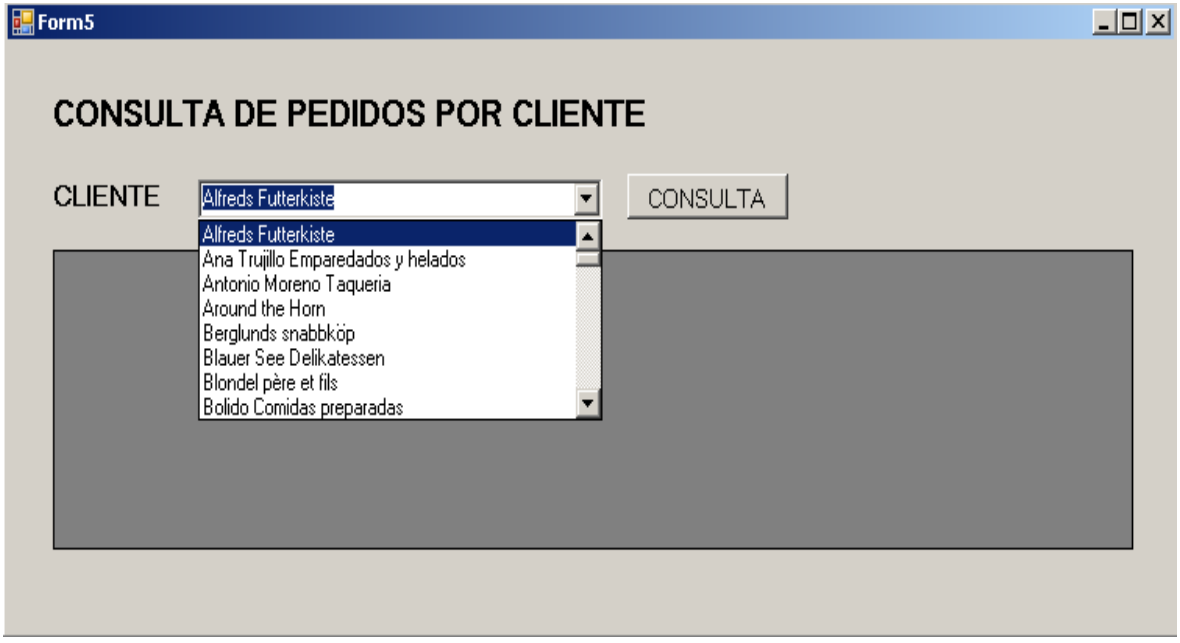
Defina las librerías de trabajo: **System.Data.SqlClient** (Base de Datos en SQL Server) y el **System.Configuration.ConfigurationManager**



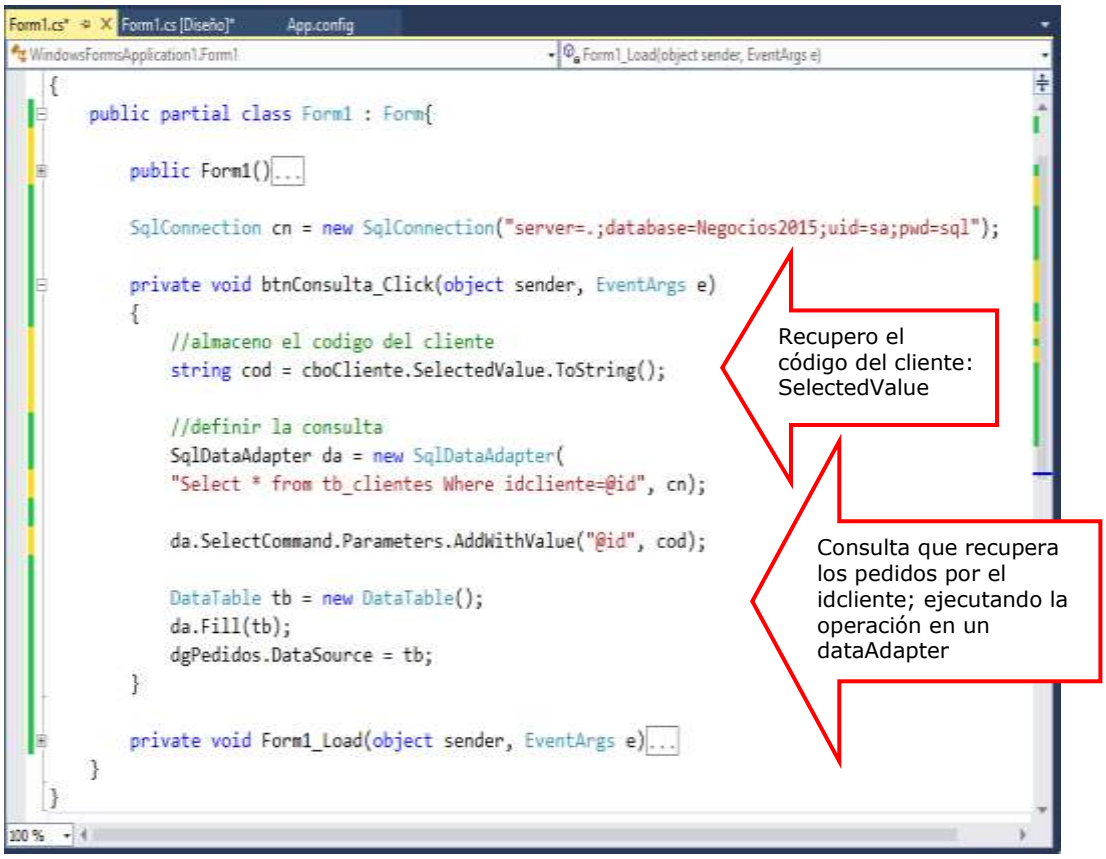
Defina la sentencia SQL para listar los registros de clientes en el DataAdapter, almacenando o poblando los registros en el DataTable llamado “tabla”. A continuación configuramos el ComboBox para cargar los clientes.



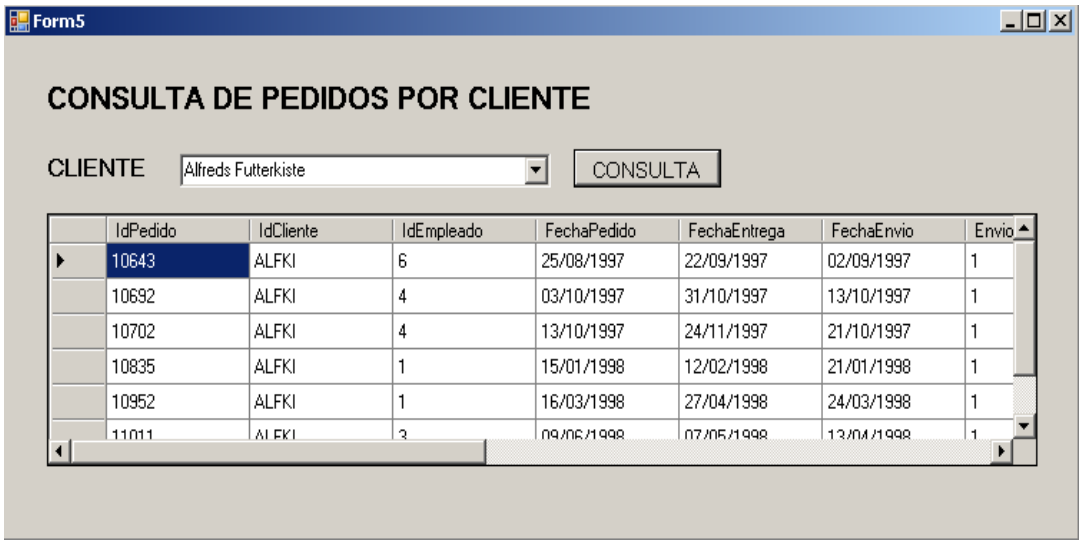
Presiona la tecla F5, para verificar que los datos se visualizarn en el control ComboBox1, tal como se muestra.



Codifique el evento Click del botón Consulta, donde listará los pedidos de un cliente seleccionado desde el ComboBox cboCliente.



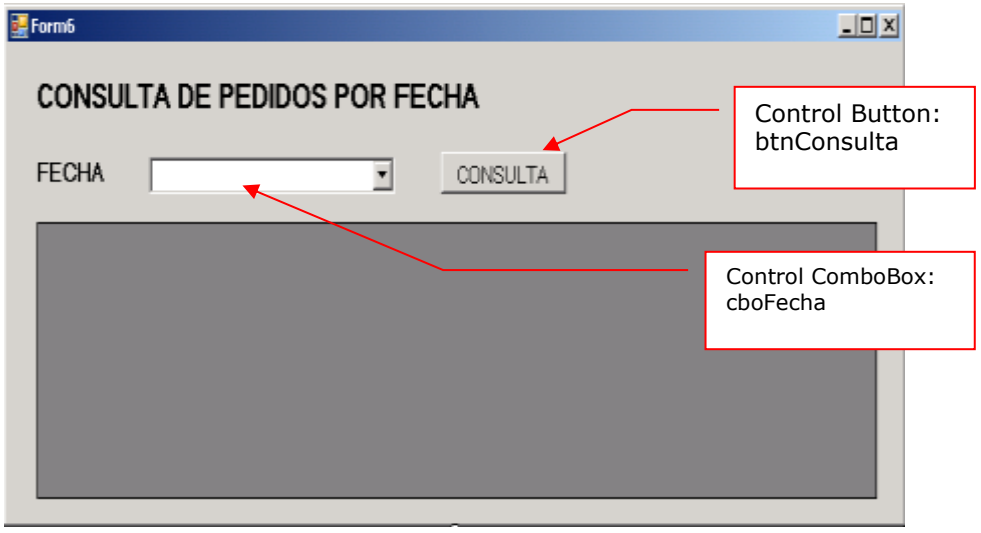
Presione F5; seleccione un cliente y presione el botón Consulta, se visualiza los pedidos del cliente seleccionado



LABORATORIO 2.4

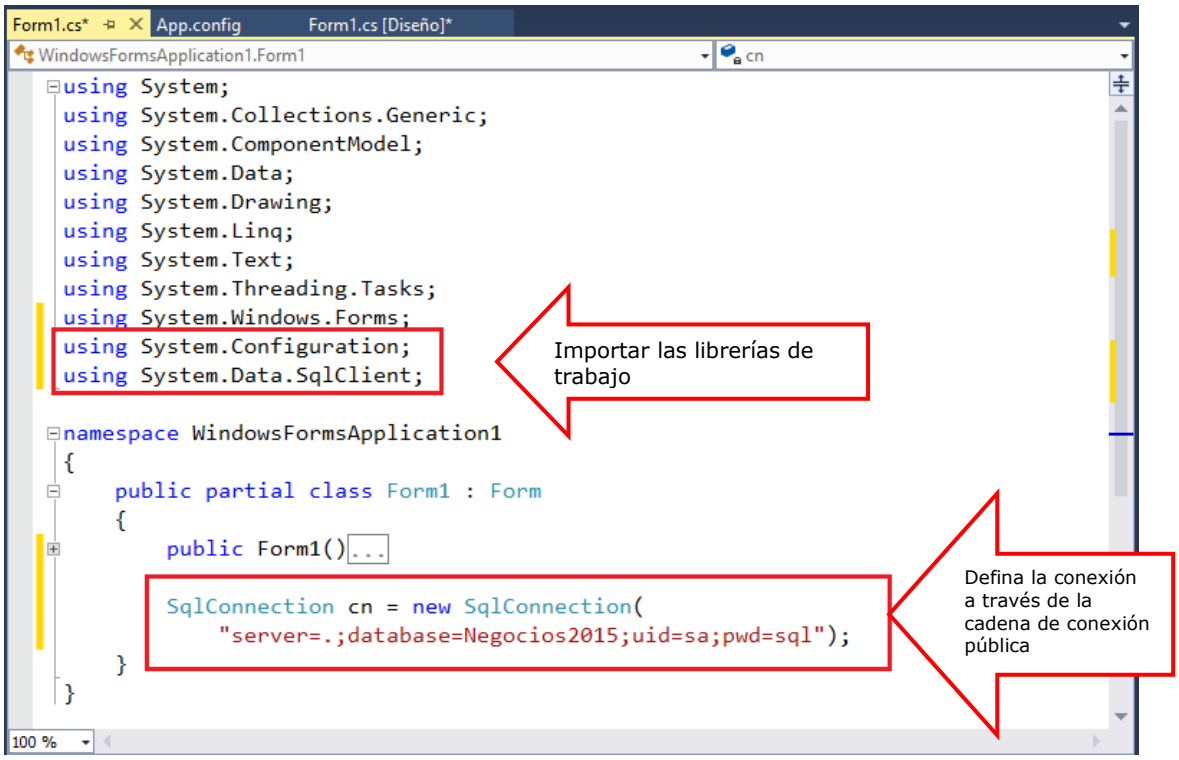
Implemente un Formulario donde liste todos los PedidosCabe de una fecha seleccionada desde un control ComboBox

Diseño de Formulario

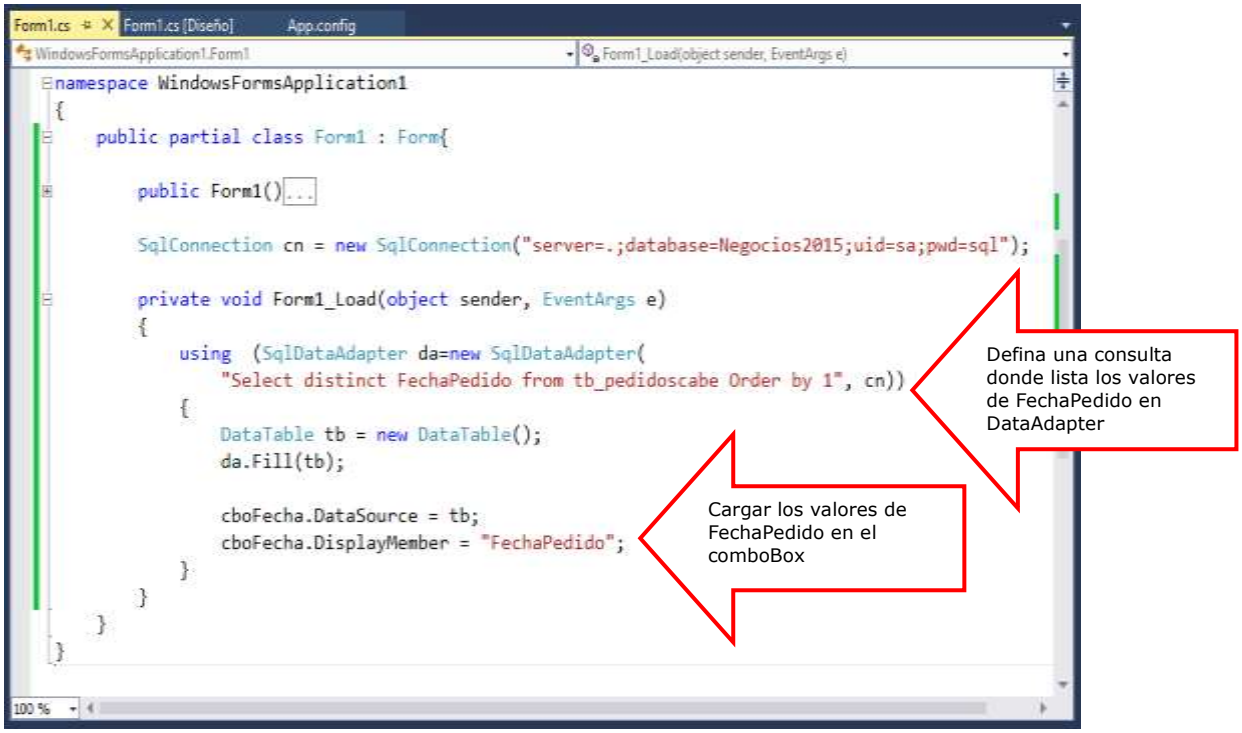


PROGRAMACIÓN

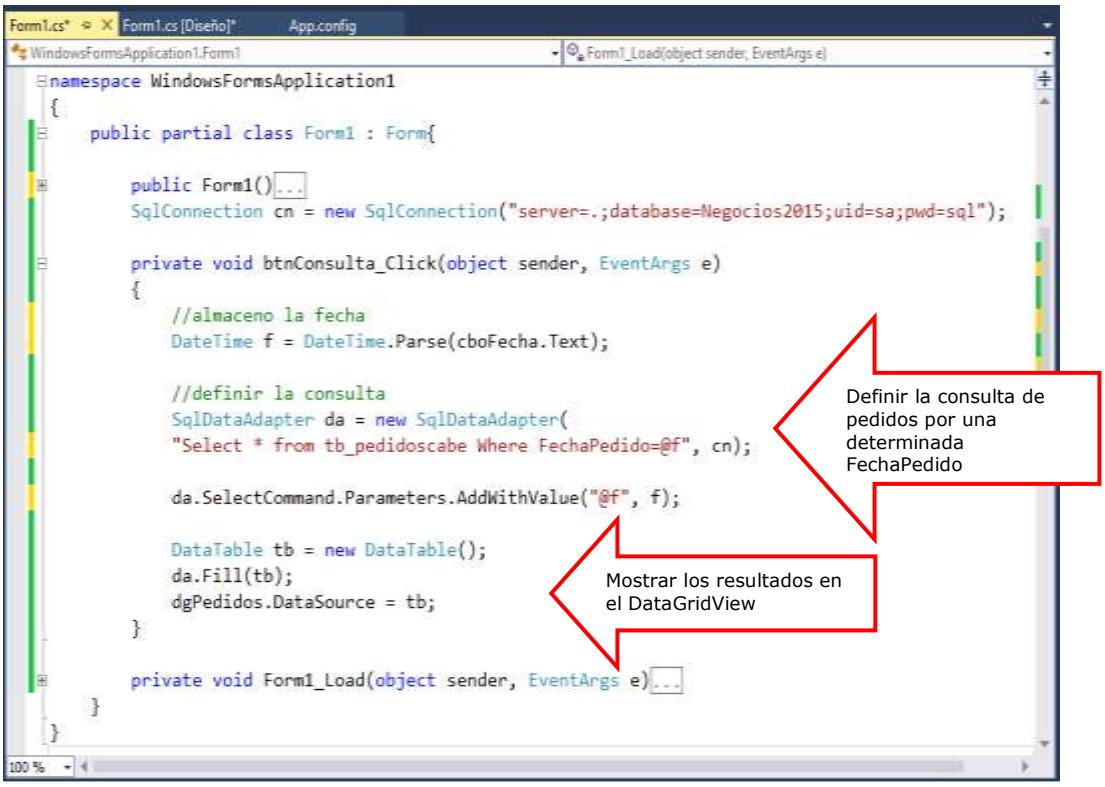
Defina las librerías de trabajo: **System.Data.SqlClient** (Base de Datos en SQL Server) y el **System.Configuration.** Defina la conexión a la base de datos.



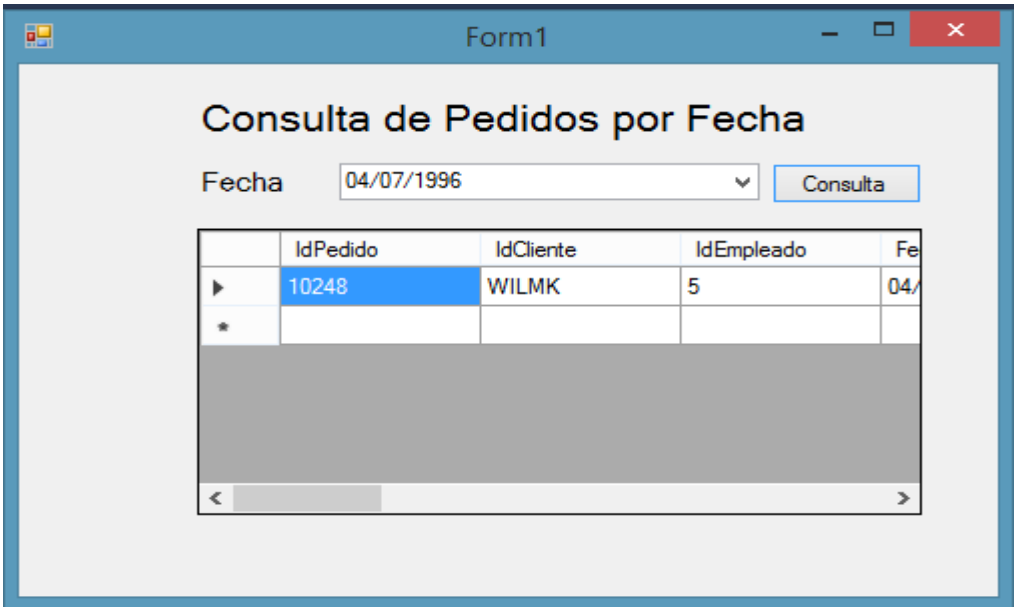
Defina la conexión a la base de datos. Programa el evento Load del Formulario, para cargar los valores del campo FechaPedido en el combo cboFecha



Programe el evento Click del botón Consulta. Recupera la fecha seleccionado del comboBox y lo utilizamos en la consulta, para listar los pedidos por Fecha



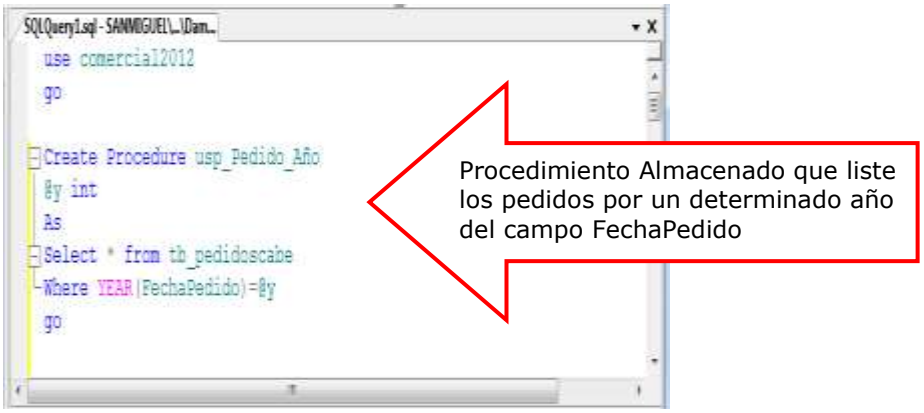
Presione la tecla F5; seleccione una fecha desde el control ComboBox1, al presionar el botón Consulta visualice los pedidos de la fecha seleccionada



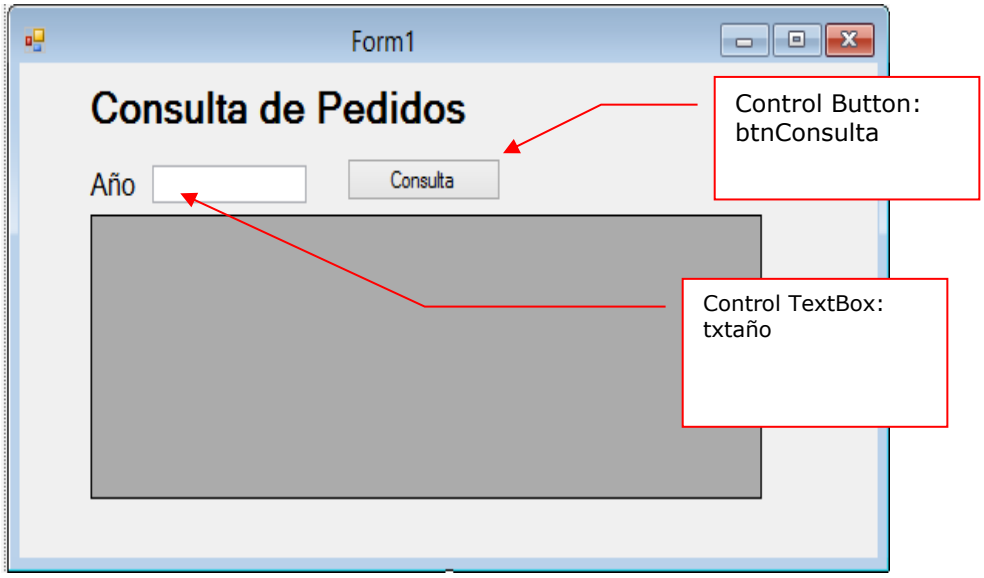
LABORATORIO 2.5

Implementa un Formulario donde Liste todos los Pedidos registrados por un Año específico

Defina el procedimiento Almacenado de la consulta

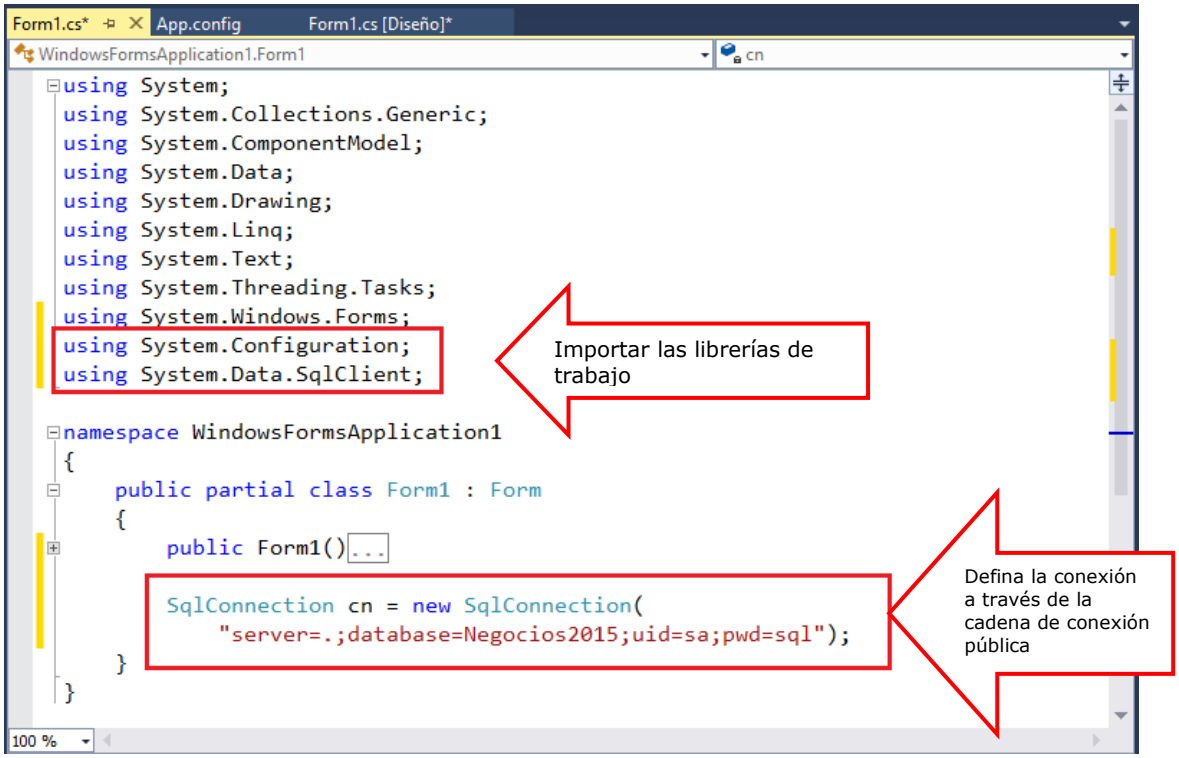


Diseño del Formulario

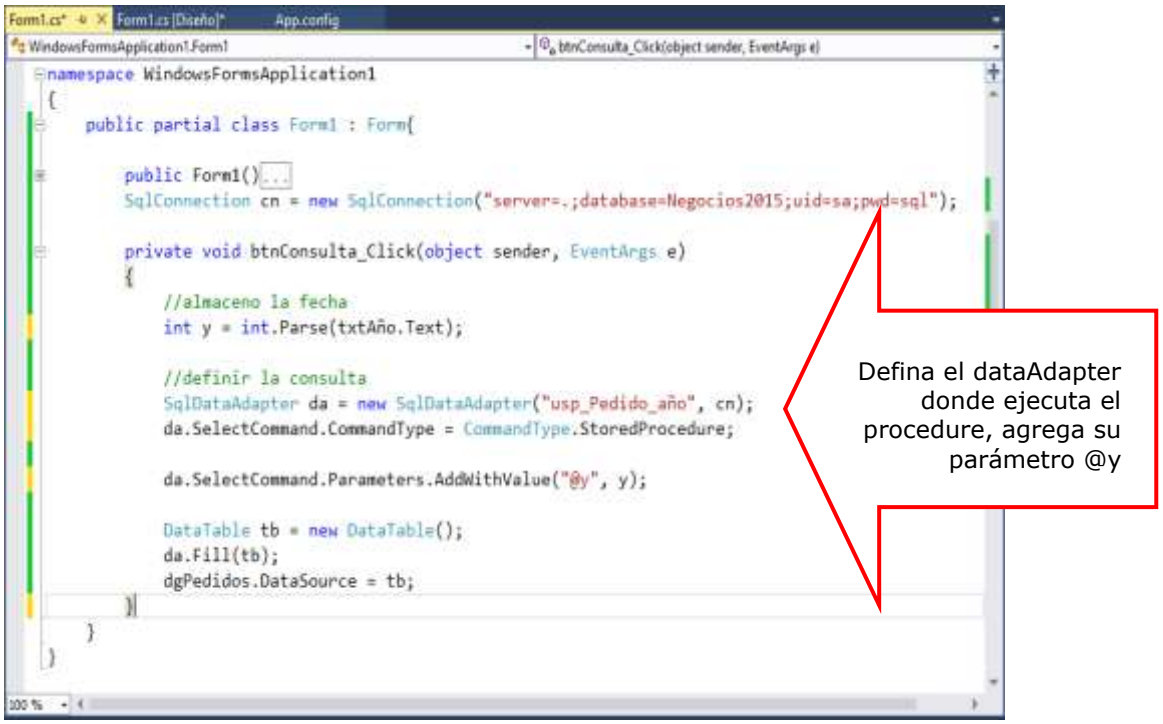


PROGRAMACIÓN

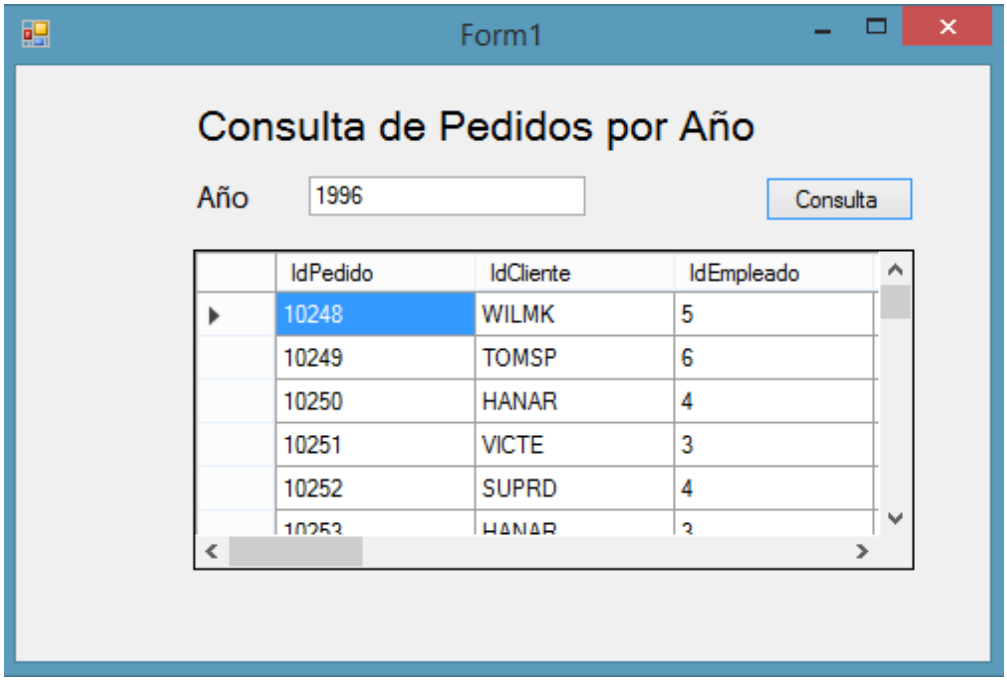
Defina las librerías: Data.SqlClient y Configuration. Instancia la conexión a la base de datos utilizando la conexión publicada en el app.config



Programa el evento Click del control btnConsulta donde ejecute el procedimiento almacenado "usp_Pedido_Año", utilizando un DataAdapter, visualizando los registros en el DataGridView1



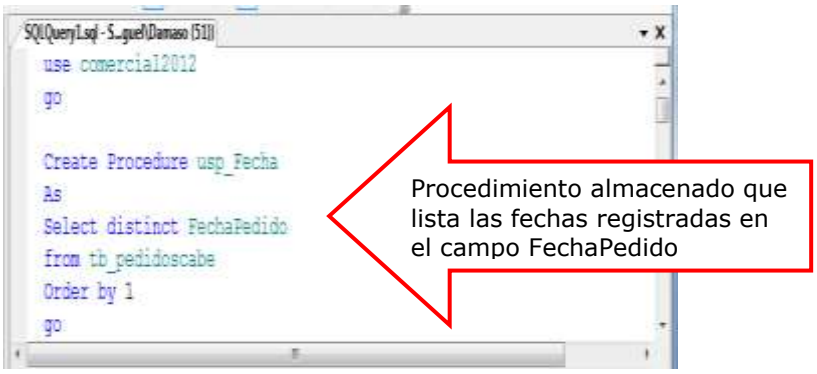
Presione la tecla F5 para ejecutar el Formulario, ingrese el año desde un textBox, al presionar el botón CONSULTA, visualizamos los pedidos del año ingresado

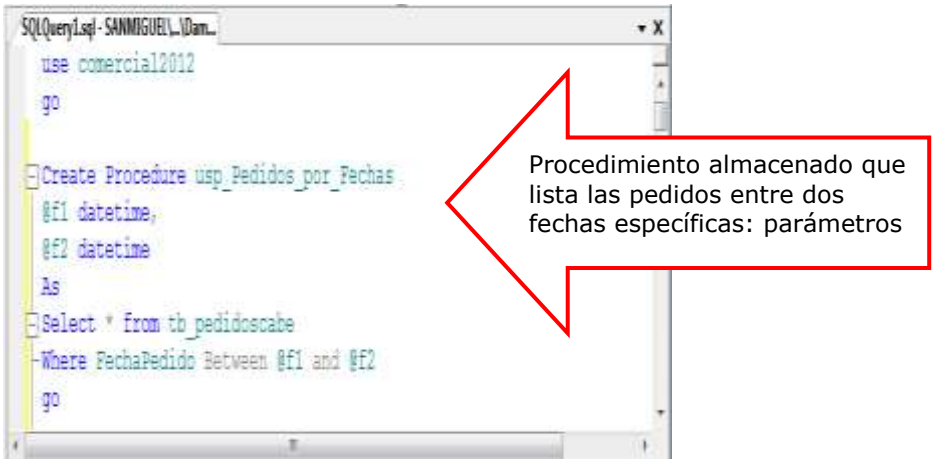


LABORATORIO 2.6

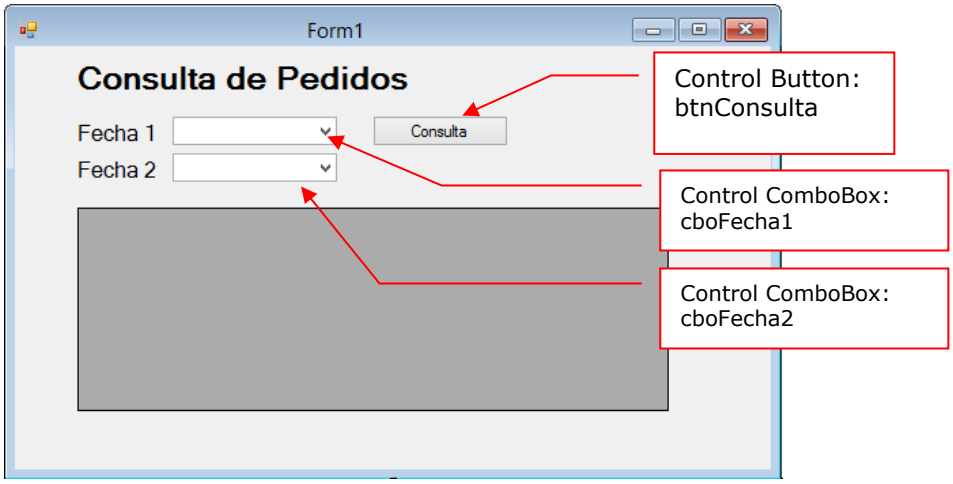
Implementa un Formulario que permita listar los pedidos registrados entre un rango de dos fechas

Defina el procedimiento Almacenado de la consulta



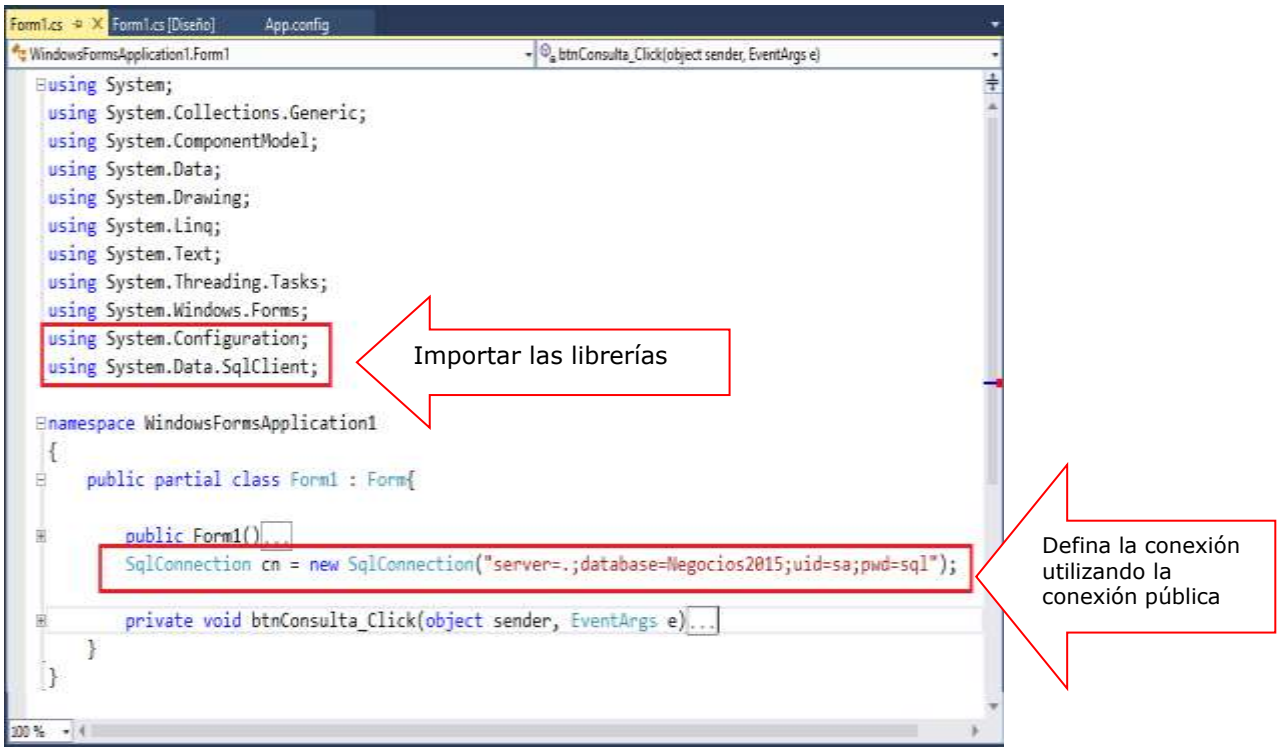


Diseño del Formulario

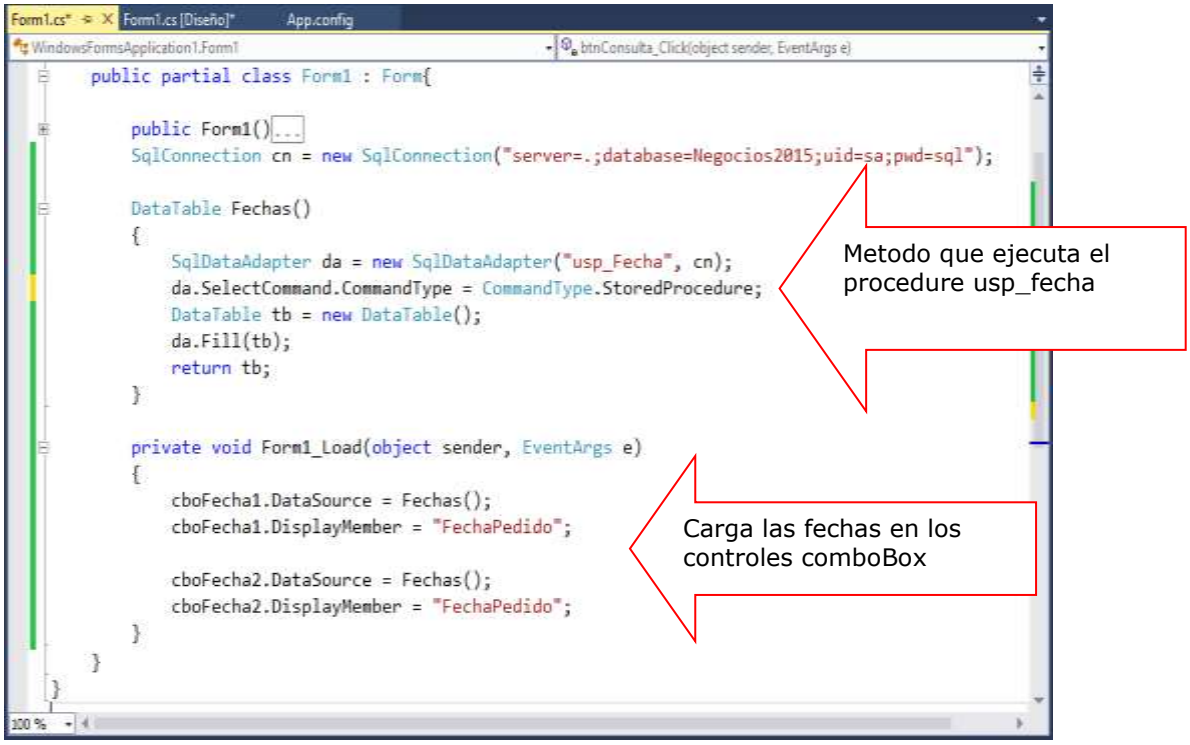


PROGRAMACIÓN

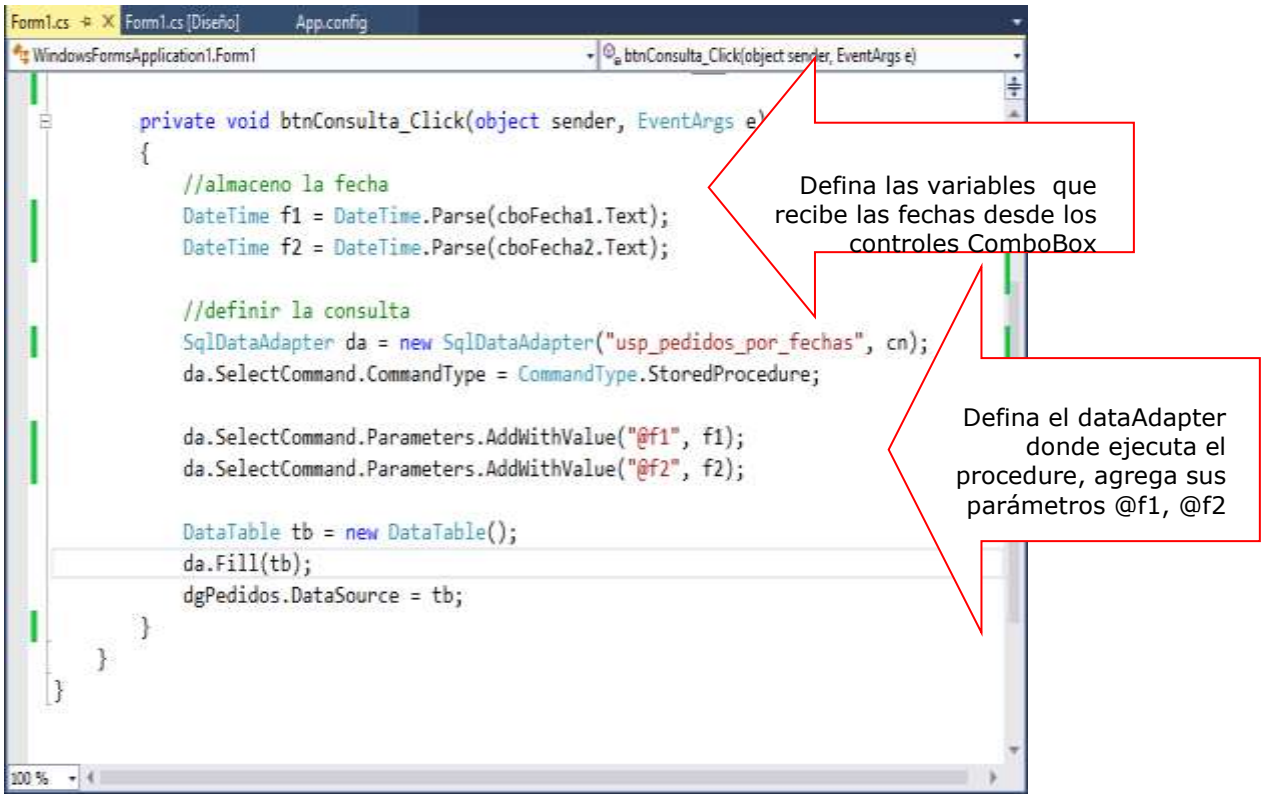
Defina las librerías de trabajo: SqlClient y Configuration. Instancia la conexión a la base de datos utilizando la conexión publicada en el app.config



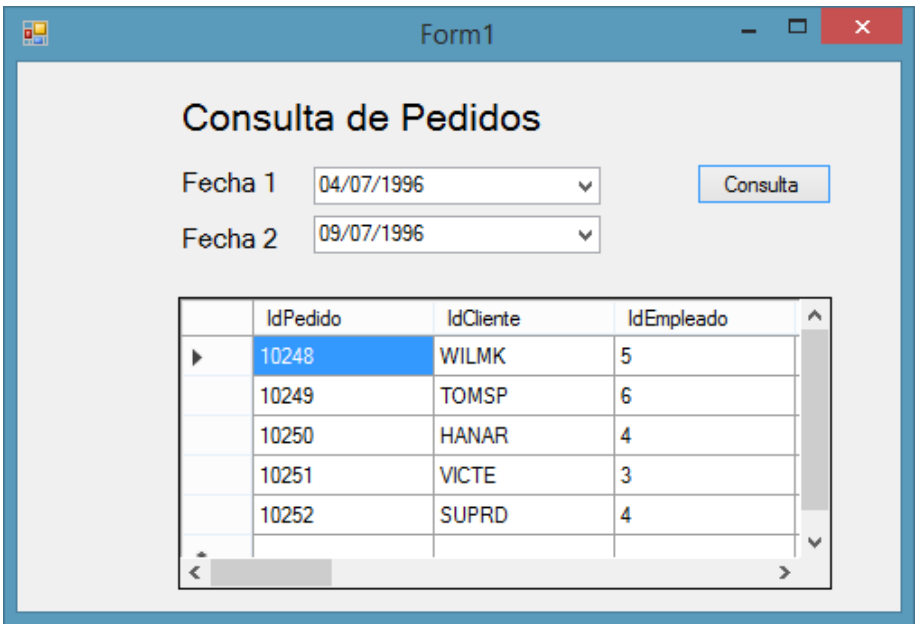
Defina un método llamado Fechas, donde ejecuta el procedimiento almacenado de las fechas. Programa el evento Load para cargar las fechas en los comboBox.



Programa el evento Click del control btnConsulta donde ejecute el procedimiento almacenado “usp_Pedidos_por_Fechas”, visualizando los registros en dgPedidos.



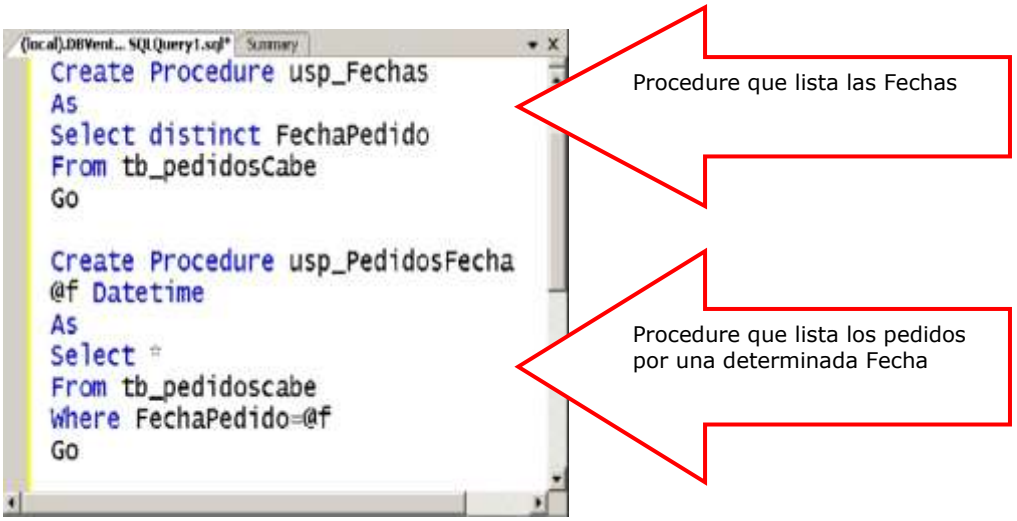
Presione la tecla F5 para ejecutar el Formulario, seleccione las fechas desde los controles ComboBox, al presionar el botón CONSULTA, visualizamos los pedidos entre las fechas seleccionadas



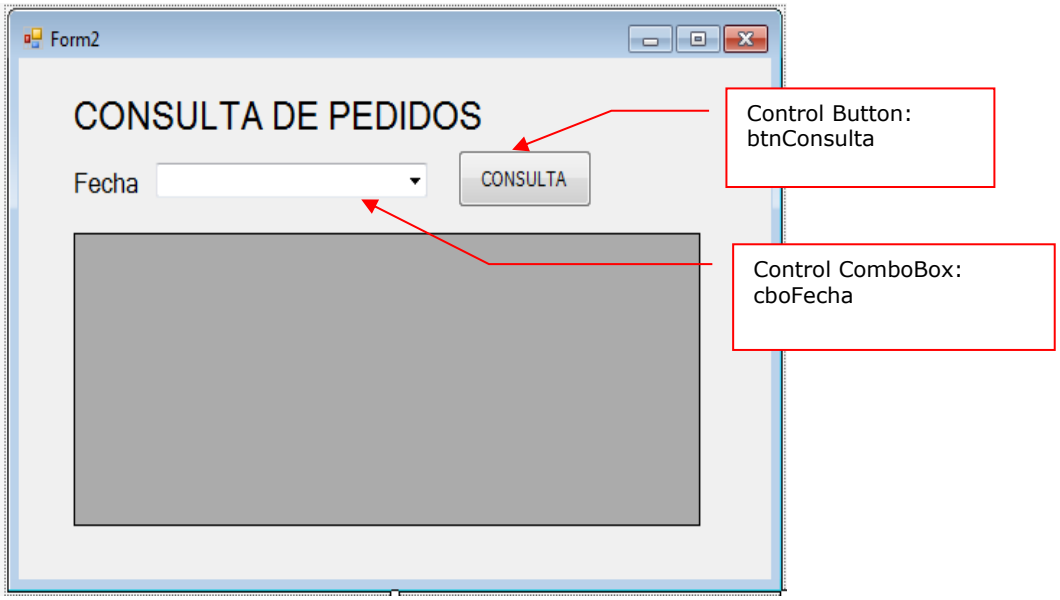
LABORATORIO 2.7

Implementa un Formulario que permita listar los registros de tb_pedidoscabe registrados por una determinada Fecha (FechaPedido). Implemente la solución utilizando **DataReader**

Defina los procedimientos almacenados de la consulta

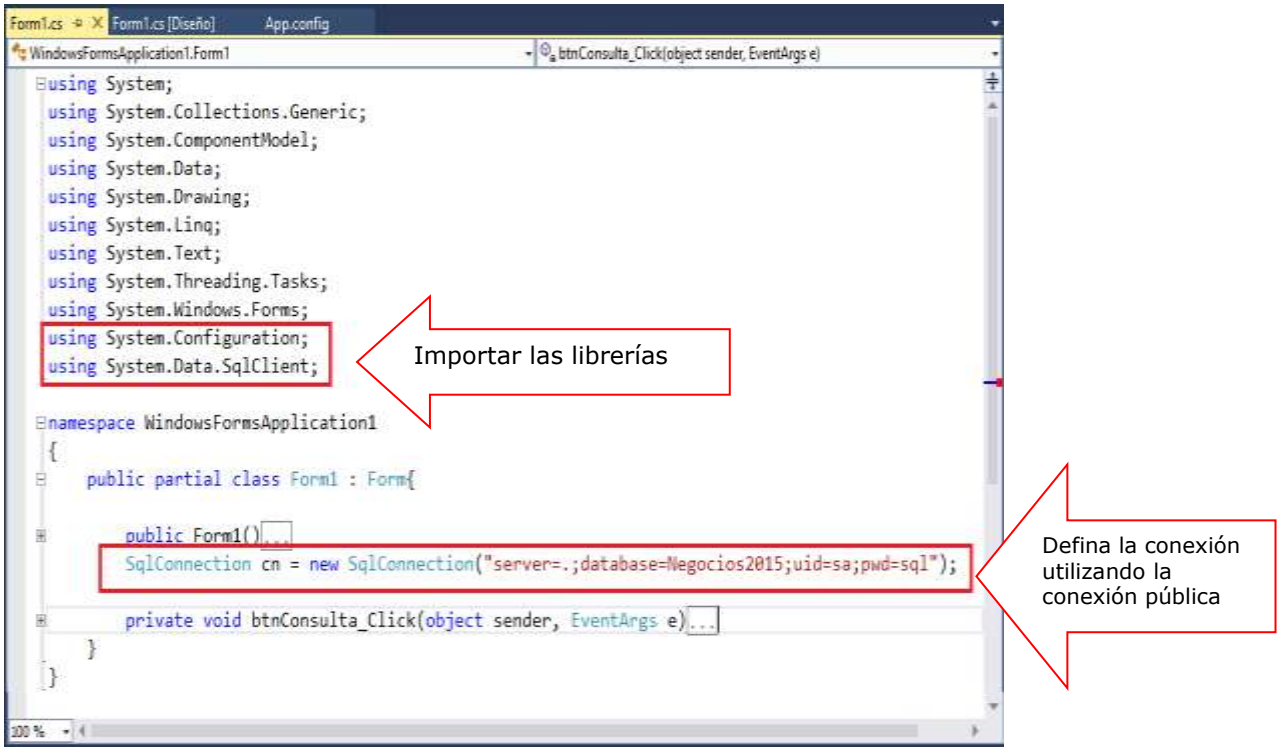


DISEÑA EL FORMULARIO

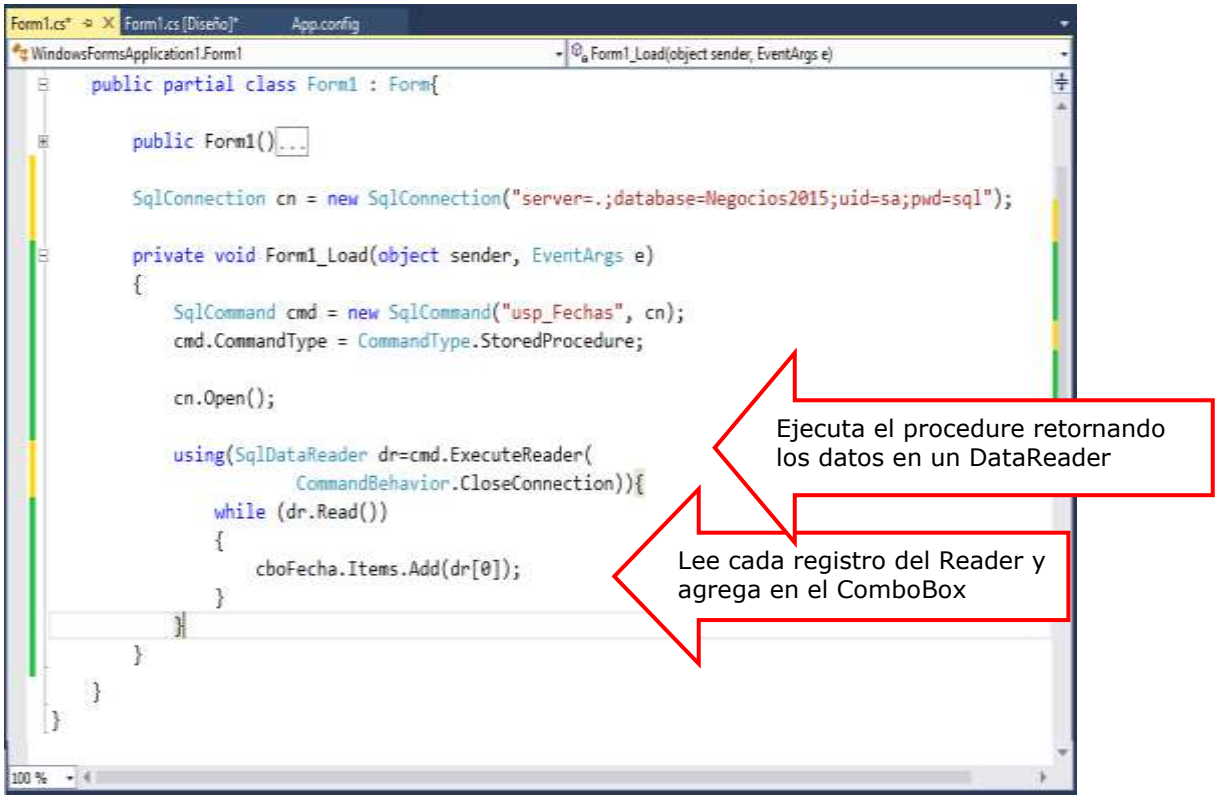


PROGRAMACION

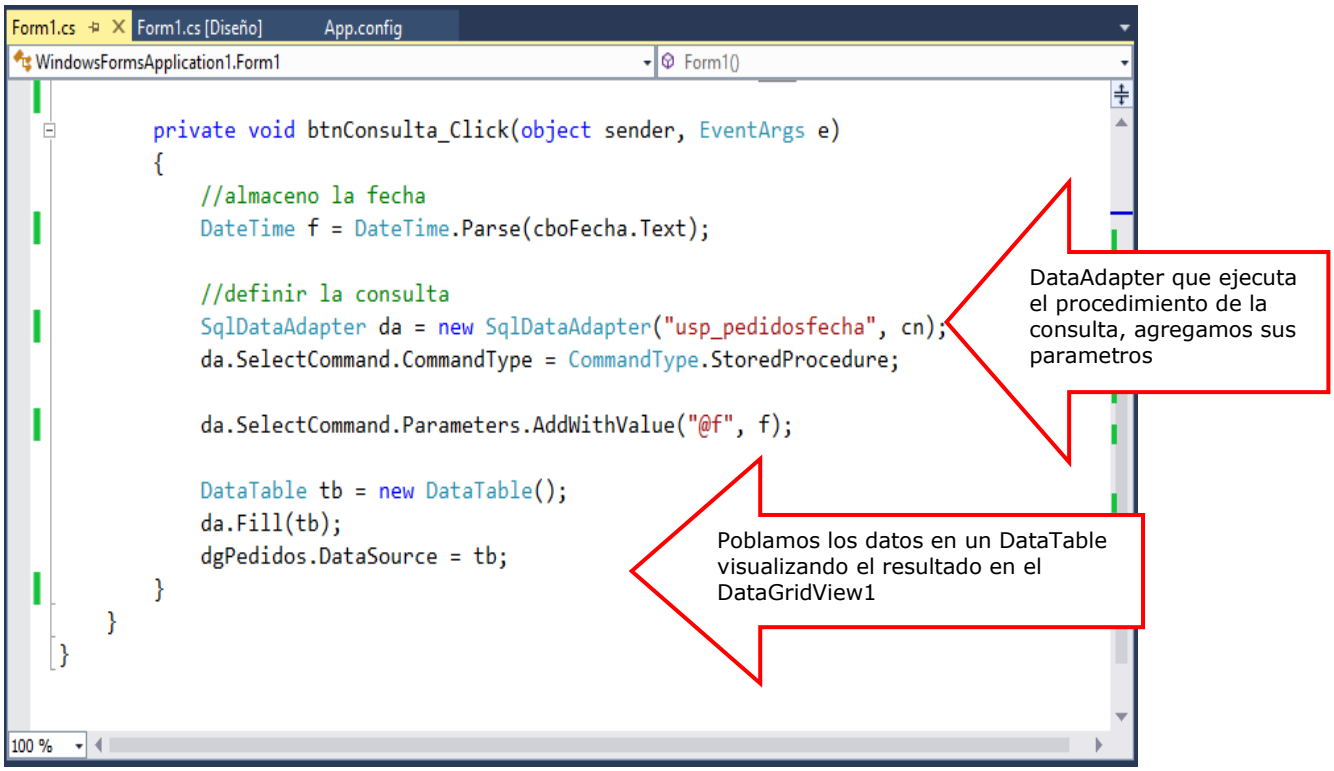
Defina las librerías: Data.SqlClient y Configuration.ConfigurationManager. Instancia la conexión a la base de datos utilizando la conexión publicada en el app.config



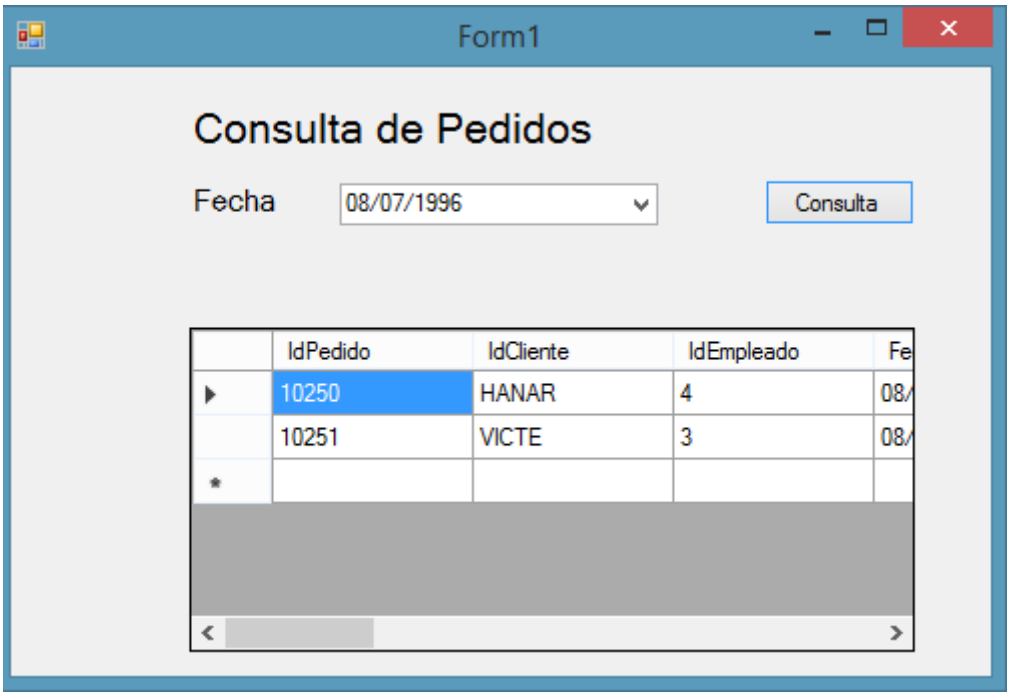
Programa el Load del Formulario; ejecute el procedimiento almacenado usp_fechas en un objeto Command almacenando los registros resultantes en el DataReader. A continuación el DataReader cargará la fecha al ComboBox



Programa el evento Clic del botón Consulta, selecciona una fecha, visualice los Pedidos de la fecha seleccionada, ejecuta el procedimiento almacenado usp_PedidosFecha



Ejecuta el Formulario, selecciona una Fecha, al presionar el botón Consulta se visualizan los registros de la tabla tb_pedidoscabe



LABORATORIO 2.8: PROPUESTO

Implementar un Formulario donde Liste los Pedidos al seleccionar un Mes y Año. Utilice DataReader para dar solucion

Procedimientos almacenados para recuperar los meses y años

```
Create Procedure usp_Meses
As
Select distinct MONTH(FechaPedido) as M,
DATENAME(m,FechaPedido) as Mes
From tb_pedidoscabe
Order by 1

Create Procedure usp_Años
As
Select distinct YEAR(FechaPedido) as Y
From tb_pedidoscabe
Order by 1
```

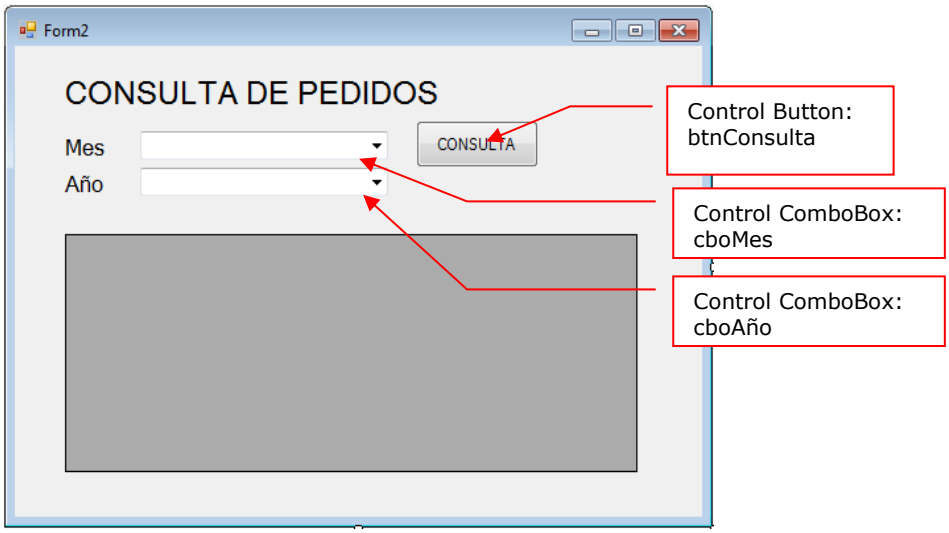
Procedimiento almacenado que liste los Meses de FechaPedido, donde retorna su valor numérico y su expresión

Procedimiento almacenado que liste los Años de FechaPedido

```
Create Procedure usp_PedidosMesAño
@mes int,
@año int
As
Select * from tb_pedidoscabe
where MONTH(FechaPedido)=@mes And
      YEAR(FechaPedido)=@año

go
```

Procedimiento almacenado que liste los registros de pedidos por Mes y Año de FechaPedido, defina los parámetros para realizar la evaluación

DISEÑO DEL FORMULARIO



Resumen

- La función principal de cualquier aplicación que trabaje con una fuente de datos es conectarse a dicha fuente de datos y recuperar los datos que se almacenan.
- La clase DataAdapter se encarga de las operaciones entre la capa de datos y la capa intermedia donde los datos son transferidos. Se puede decir que sirve como puesnte entre un objeto DataSet y un origen de datos asociados para recuperar y guardar datos.
- Básicamente, permite rellenar (Fill) el objeto DataSet para que sus datos coincidan con los del origen de datos y permite actualizar (Update) el origen de datos para que sus datos coincidan con los del DataSet.
- La recuperación de datos mediante DataReader implica crear una instancia del objeto Command y de un DataReader a continuación, para lo cual se llama a **Command**.**ExecuteReader** a fin de recuperar filas de un origen de datos.
- Puede utilizar el método **Read** del objeto DataReader para obtener una fila a partir de los resultados de una consulta.Para tener acceso a cada columna de la fila devuelta, puede pasar a DataReader el nombre o referencia numérica de la columna en cuestión.Sin embargo, el mejor rendimiento se logra con los métodos que ofrece DataReader y que permiten tener acceso a los valores de las columnas en sus tipos de datos nativos (GetDateTime, GetDouble, GetGuid, GetInt32, etc.).
- Siempre debe llamar al método **Close** cuando haya terminado de utilizar el objeto DataReader. Si Command contiene parámetros de salida o valores devueltos, éstos no estarán disponibles hasta que se cierre el **DataReader**.
- En el caso en que se devuelvan varios resultados, el DataReader proporciona el método **NextResult** para recorrer los conjuntos de resultados en orden

- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  - http://msdn.microsoft.com/es-es/library/bh8kx08z(v=vs.110).aspx
  - http://msdn.microsoft.com/es-es/library/ms254931(v=vs.110).aspx

UNIDAD

1

OPERACIONES CONECTADAS A UN ORIGEN DE DATOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno realiza operaciones de consulta y actualización de datos en el entorno de una aplicación Windows conectado a un origen de datos utilizando la librería ADO.NET

TEMARIO

Tema 3: Manipulación de datos (6 horas)

1. Operaciones de modificación de datos sobre un origen de datos
   1.1. Manejo de la clase Command
   1.2. Ejecutando operaciones de actualización de datos utilizando una sentencia SQL o ejecutando un procedimiento almacenado
   1.3. Manejo de parámetros en actualización de datos

ACTIVIDADES PROPUESTAS

- Los alumnos diseñan formularios que manejan la conexión a una fuente de datos para realizar operaciones de actualización de datos a la fuente de datos.
- Los alumnos desarrollan los laboratorios de esta semana

3. OPERACIÓN DE MODIFICACION DE DATOS SOBRE UN ORIGEN DE DATOS

Las instrucciones SQL que modifican datos (por ejemplo INSERT, UPDATE o DELETE) no devuelven ninguna fila. De la misma forma, muchos procedimientos almacenados realizan alguna acción pero no devuelven filas. Para ejecutar comandos que no devuelvan filas, cree un objeto Command con el comando SQL adecuado y una Connection, incluidos los Parameters necesarios. El comando se debe ejecutar con el método ExecuteNonQuery del objeto Command.

El método ExecuteNonQuery devuelve un entero que representa el número de filas que se ven afectadas por la instrucción o por el procedimiento almacenado que se haya ejecutado. Si se ejecutan varias instrucciones, el valor devuelto es la suma de los registros afectados por todas las instrucciones ejecutadas.

3.1 Manejo de la clase COMMAND

Una vez establecida una conexión a un origen de datos, puede ejecutar comandos y devolver resultados desde el mismo mediante un objeto **DbCommand**.

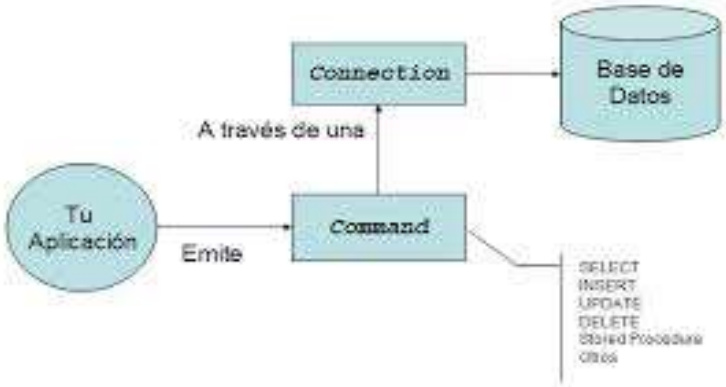


Figura 1: Diagrama del objeto Command
Referencia: http://www.dotnetero.com/2006/08/adonet-para-novatos.html

Para crear un comando, puede utilizar uno de los constructores de comando del proveedor de datos .NET Framework con el que esté trabajando. Los constructores pueden aceptar argumentos opcionales, como una instrucción SQL para ejecutar en el origen de datos, un objeto **DbConnection** o un objeto **DbTransaction**. También puede configurar dichos objetos como propiedades del comando. También puede crear un comando para una determinada conexión mediante el método **CreateCommand** de un objeto **DbConnection**. La instrucción SQL que ejecuta el comando se puede configurar mediante la propiedad CommandText.

Cada proveedor de datos de .NET FrameWork cuenta con un objeto Command:

Proveedor	Descripcion
OledbCommand	Proveedor de datos para OLEDB
SqlCommand	Proveedor de datos para SQL Server
OdbcCommand	Proveedor de datos para ODBC
OracleCommand	Proveedor de datos para Oracle

EJECUCION DE UN OBJETO COMMAND

Cada proveedor de datos .NET Framework incluido en .NET Framework dispone de su propio objeto command que hereda de DbCommand.

El proveedor de datos .NET Framework para OLE DB incluye un objeto **OleDbCommand**, el proveedor de datos .NET Framework para SQL Server incluye un objeto **SqlCommand**, el proveedor de datos .NET Framework para ODBC incluye un objeto **OdbcCommand** y el proveedor de datos .NET Framework para Oracle incluye un objeto **OracleCommand**.

Cada uno de estos objetos expone métodos para ejecutar comandos que se basan en el tipo de comando y el valor devuelto deseado, tal como se describe en la tabla siguiente:

Commando	Valor de retorno
ExecuteReader	Devuelve un objeto **DataReader**.
ExecuteScalar	Devuelve un solo valor escalar.
ExecuteNonQuery	Ejecuta un comando que no devuelve ninguna fila.
ExecuteXMLReader	Devuelve un valor **XmlReader**. Solo está disponible para un objeto SqlCommand.

Cada objeto command fuertemente tipado admite también una enumeración **CommandType** que especifica cómo se interpreta una cadena de comando, tal como se describe en la tabla siguiente

Commando	Valor de retorno
Text	Comando de SQL que define las instrucciones que se van a ejecutar en el origen de dato
StoredProcedure	Nombre del procedimiento almacenado. Puede usar la propiedad **Parameters** de un comando para tener acceso a los parámetros de entrada y de salida y a los valores devueltos, independientemente del método Execute al que se llame. Al usar **ExecuteReader**, no es posible el acceso a los valores devueltos y los parámetros de salida hasta que se cierra **DataReader**
TableDirect	Nombre de una tabla.

EJECUCIÓN DE UNA CONSULTA QUE RETORNE UN CONJUNTO DE RESULTADOS

El objeto Command de ADO.NET tiene un método **ExecuteReader** que permite ejecutar una consulta que retorna uno o más conjunto de resultados.

Al ejecutar el método **ExecuteReader**, podemos pasar un parámetro al método el cual representa la enumeración **CommandBehavior**, que permite controlar al Command como se ejecutado.

A continuación mostramos la descripción de los enumerables del CommandBehavior:

Valor	Descripción
CommandBehavior.CloseConnection	Se utiliza para cerrar la conexión en forma automática, tan pronto como el dataReader es cerrado.
CommandBehavior.SingleResult	Retorna un único conjunto de resultados
CommandBehavior.SingleRow	Retorna una fila

3.2 EJECUTANDO OPERACIONES DE ACTUALIZACION DE DATOS UTILIZANDO SQL O PROCEDIMIENTO ALMACENADO

Los procedimientos almacenados ofrecen numerosas ventajas en el caso de aplicaciones que procesan datos. Mediante el uso de procedimientos almacenados, las operaciones de bases de datos se pueden encapsular en un solo comando, optimizar para lograr el mejor rendimiento, y mejorar con seguridad adicional.

Aunque es cierto que para llamar a un procedimiento almacenado basta con pasar en forma de instrucción SQL su nombre seguido de los argumentos de parámetros, el uso de la colección **Parameters** del objeto **DbCommand** de ADO.NET permite definir más explícitamente los parámetros del procedimiento almacenado, y tener acceso a los parámetros de salida y a los valores devueltos.

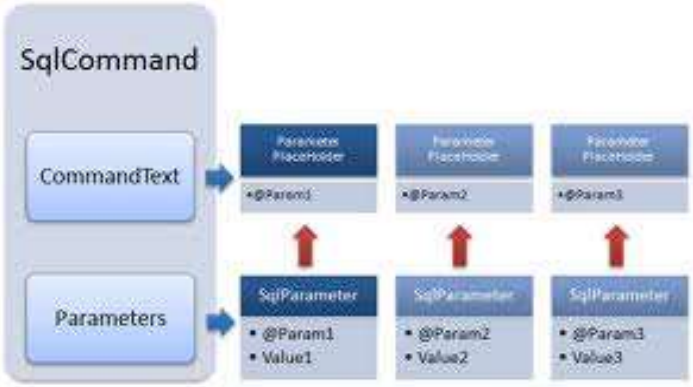


Figura 2: Diagrama del objeto SqlCommand
Referencia: http://yinyangit.wordpress.com/2011/08/05/ado-net-tutorial-lesson-06-adding-parameters-to-sqlcommands/

3.3 MANEJO DE PARAMETROS EN ACTUALIZACION DE DATOS

Cuando se usan parámetros con **SqlCommand** para ejecutar un procedimiento almacenado de SQL Server, los nombres de los parámetros agregados a la colección **Parameters** deben coincidir con los nombres de los marcadores de parámetro del procedimiento almacenado.

El proveedor de datos de .NET Framework para SQL Server no admite el uso del marcador de posición de signo de interrogación de cierre (?) para pasar parámetros a una instrucción SQL o a un procedimiento almacenado. Este proveedor trata los parámetros del procedimiento almacenado como parámetros con nombre y busca marcadores de parámetros coincidentes.

Para crear un objeto **DbParameter**, se puede usar su constructor o bien se puede agregar a **DbParameterCollection** mediante una llamada al método Add de la colección **DbParameterCollection**.

El método Add acepta como entrada argumentos del constructor o cualquier objeto de parámetro ya existente, en función del proveedor de datos.

En el caso de los parámetros que no sean de entrada (INPUT), debe de asignarse la propiedad ParameterDirection y especifique cual es el tipo de dirección del parámetro: **InputOutput**, **Output** o **ReturnValue**

El tipo de datos de un parámetro es específico del proveedor de datos de .NET Framework. Al especificar el tipo, el valor de Parameter se convierte en el tipo del proveedor de datos de .NET Framework antes de pasar el valor al origen de datos. Si lo desea, puede especificar el tipo de un objeto Parameter de forma genérica estableciendo la propiedad **DbType** del objeto **Parameter** en un **DbType** determinado.

Las instrucciones SQL que modifican datos (por ejemplo INSERT, UPDATE o DELETE) no devuelven ninguna fila. De la misma forma, muchos procedimientos almacenados realizan alguna acción pero no devuelven filas. Para ejecutar comandos que no devuelvan filas, cree un objeto Command con el comando SQL adecuado y una Connection, incluidos los **Parameters** necesarios. El comando se debe ejecutar con el método **ExecuteNonQuery** del objeto **Command**.

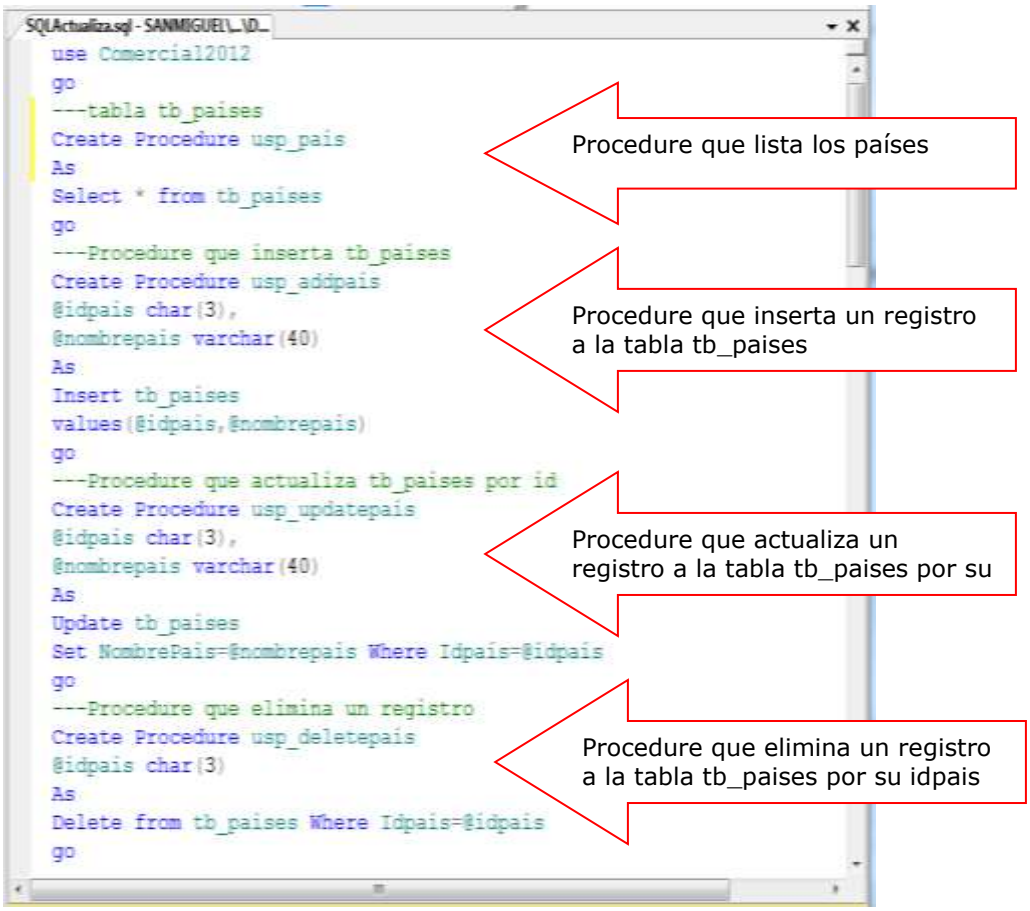
El método **ExecuteNonQuery** devuelve un entero que representa el número de filas que se ven afectadas por la instrucción o por el procedimiento almacenado que se haya ejecutado. Si se ejecutan varias instrucciones, el valor devuelto es la suma de los registros afectados por todas las instrucciones ejecutadas.

LABORATORIO 3.1

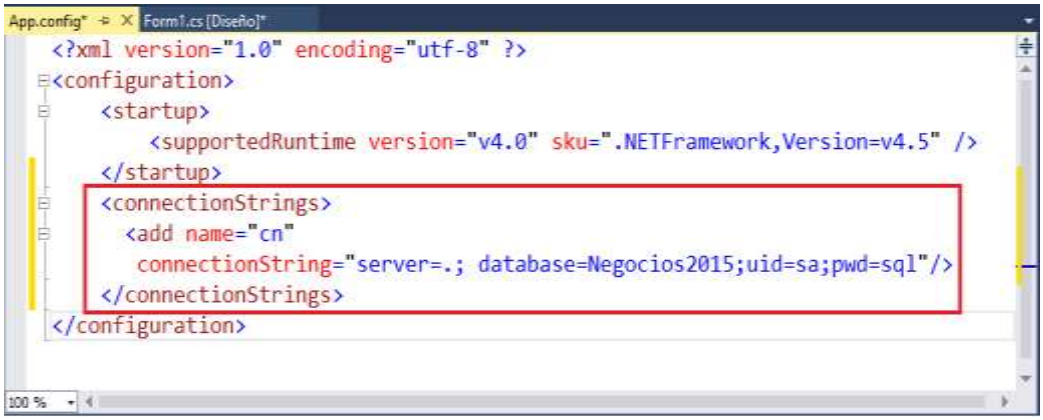
Mantenimiento de datos

Se desea implementar un Formulario donde realice el mantenimiento a la tabla tb_paises, utilice procedimientos almacenados en el mantenimiento

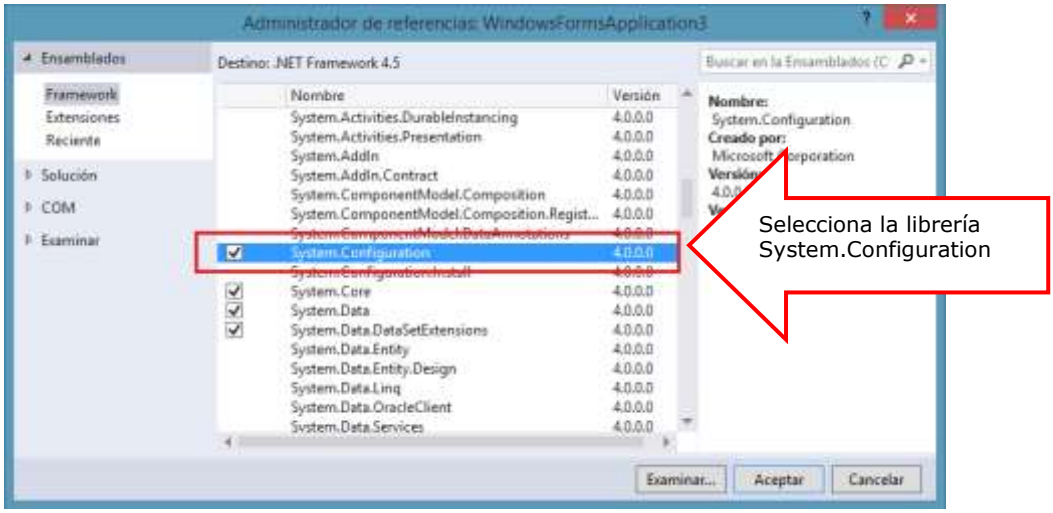
DISEÑO DE PROCEDIMIENTOS ALMACENADOS



En el proyecto defina la cadena de conexión <connectionStrings> en el App.config.

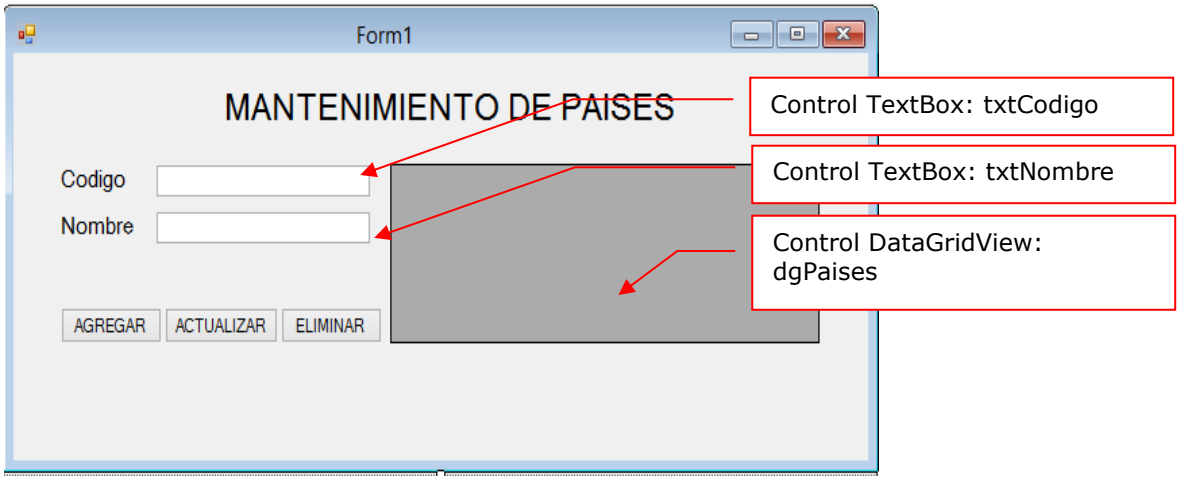


A continuación, agregar la referencia: System.Configuration para trabajar con la conexión publicada en el app.Config. Desde la opción Proyecto, selecciona la opción Agregar Referencia…



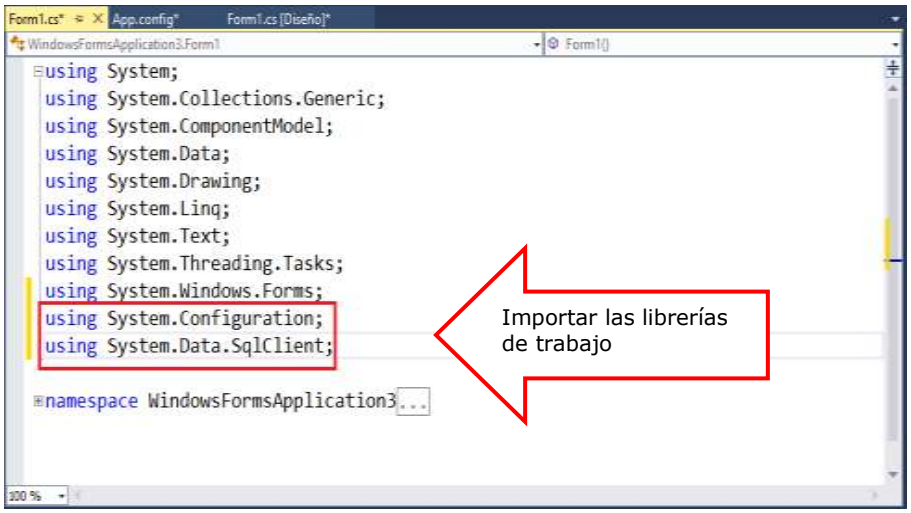
Diseño del Formulario

A continuación diseñe el formulario, tal como se muestra

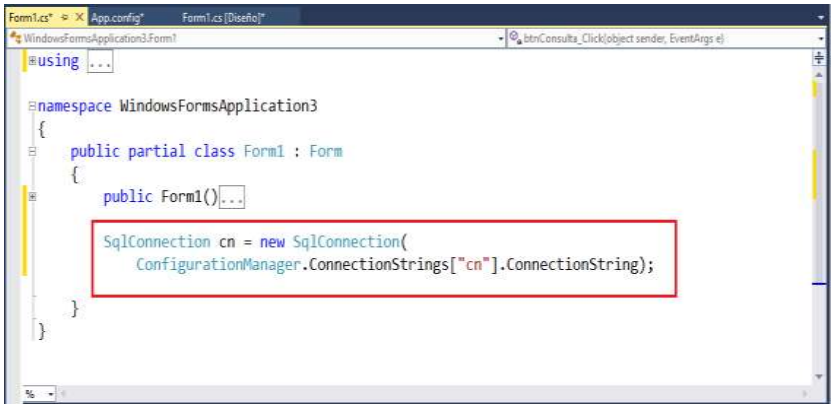


PROGRAMACIÓN

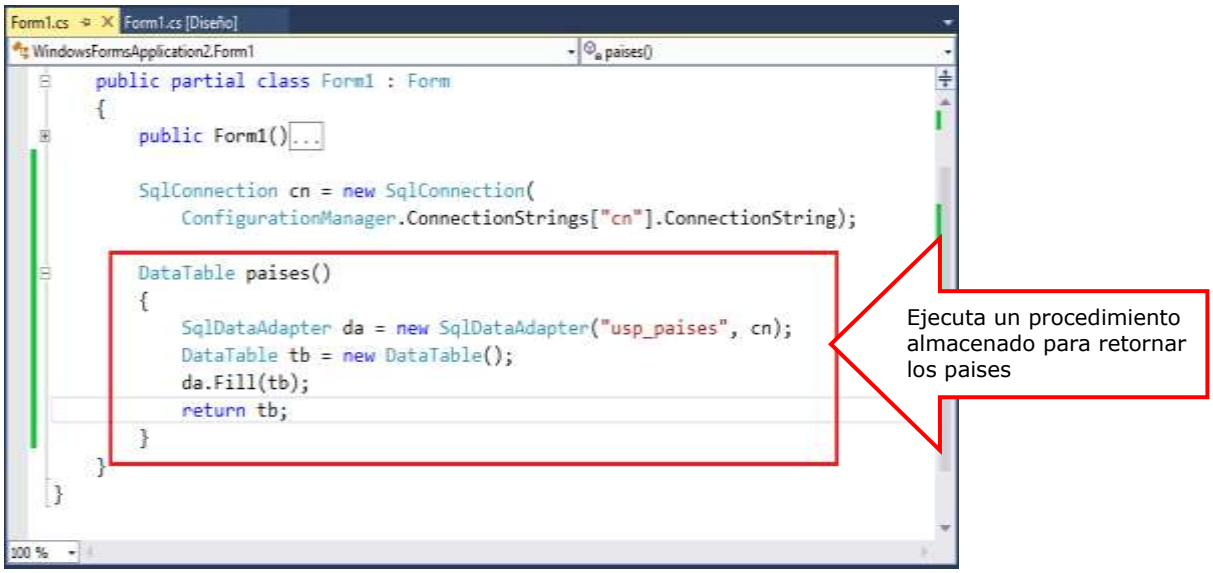
Defina las librerías de trabajo: System.Data.SqlClient y System.Configuration



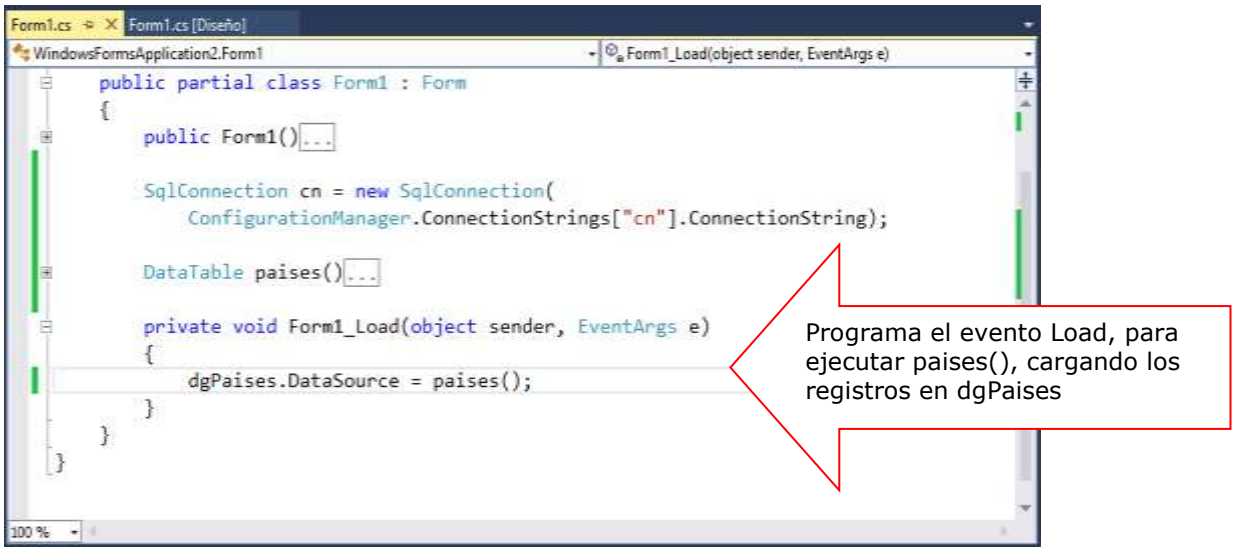
Defina la conexión de la base de datos en el formulario



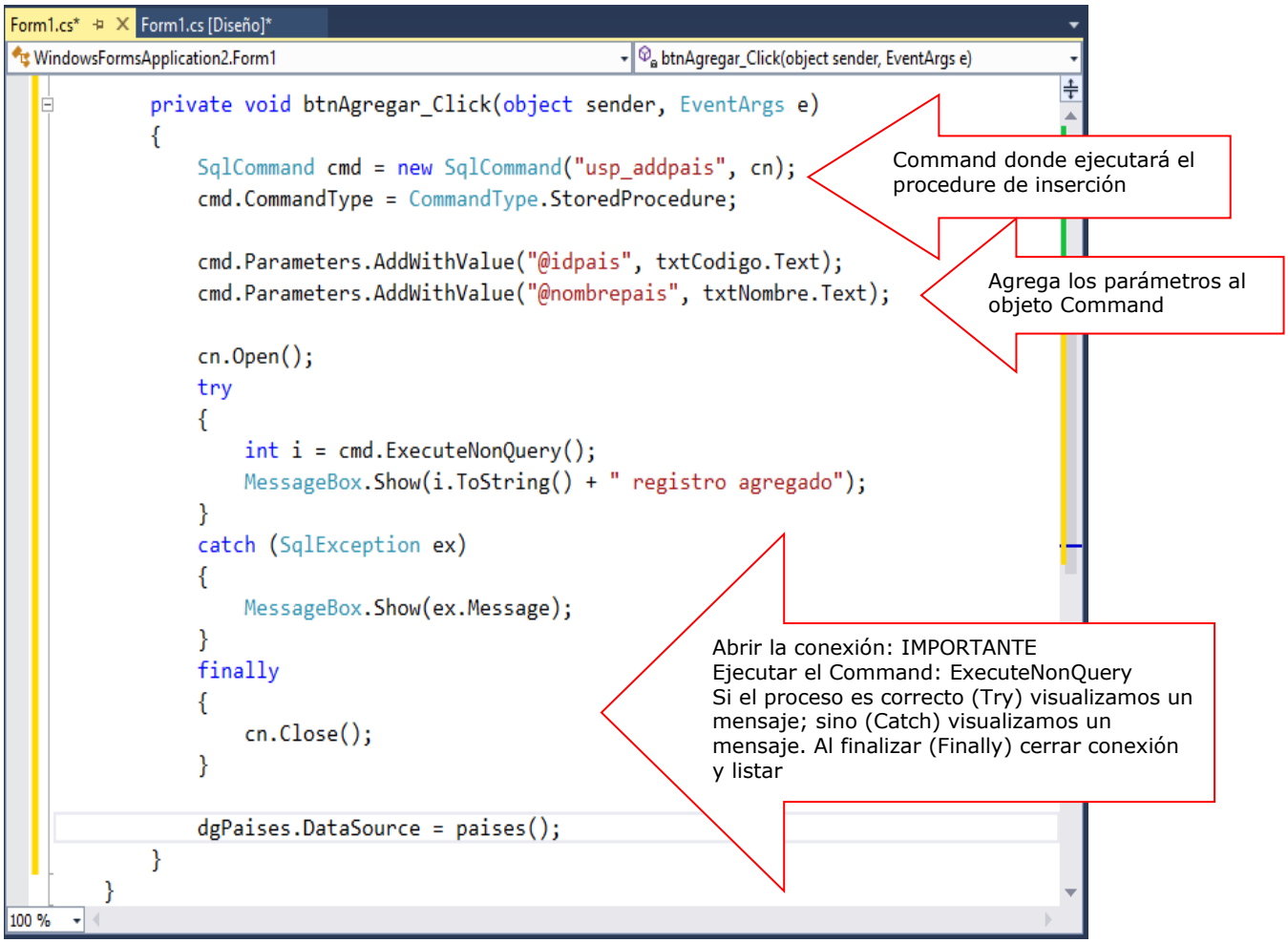
Defina un método de tipo DataTable donde retorna los registros de la tabla tb_paises.



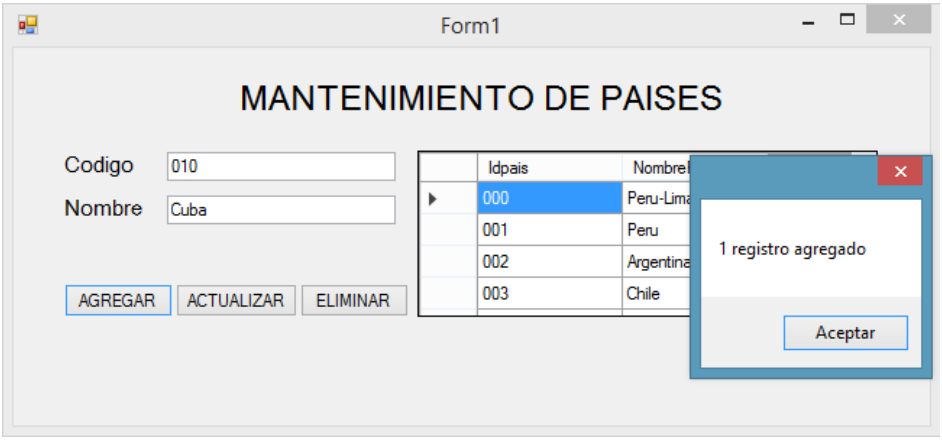
Programa el evento Load del Formulario, donde al cargar el Formulario ejecutamos la función paises(), visualizando los resultado en el control DataGridView: dgPaises



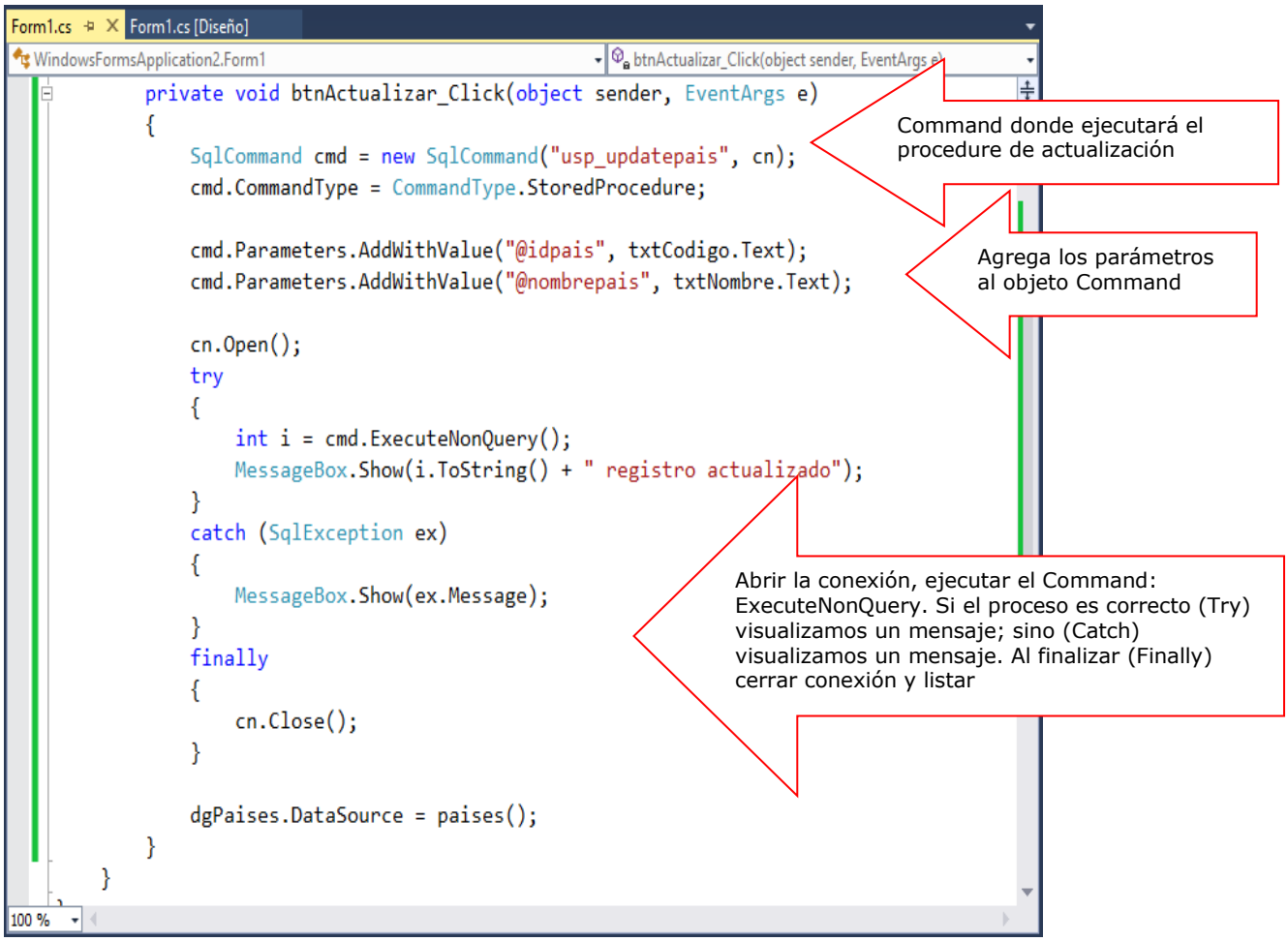
Programa el evento Click del botón Agregar. En este proceso el objeto Command para ejecutar un procedimiento almacenado de inserción (usp_addpais)



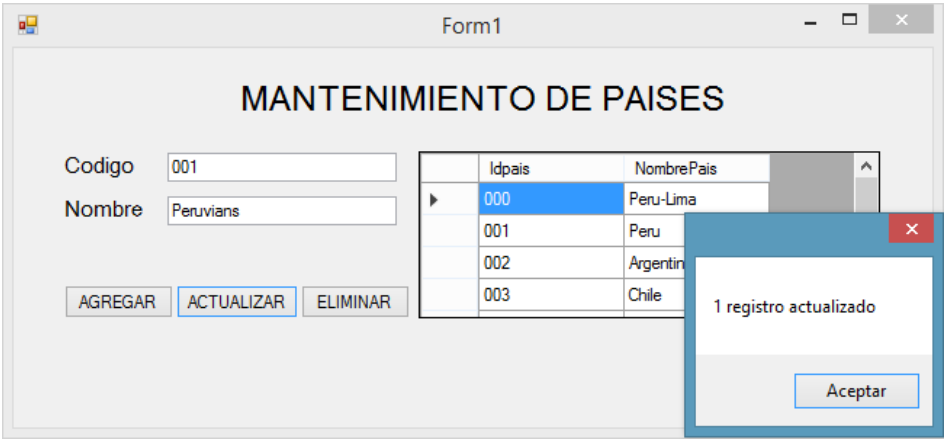
Ejecutamos la página para comprobar el proceso: ingresa el código y nombre del país. Al presionar el botón AGREGAR, se ejecuta el proceso visualizando un mensaje, tal como se muestra.



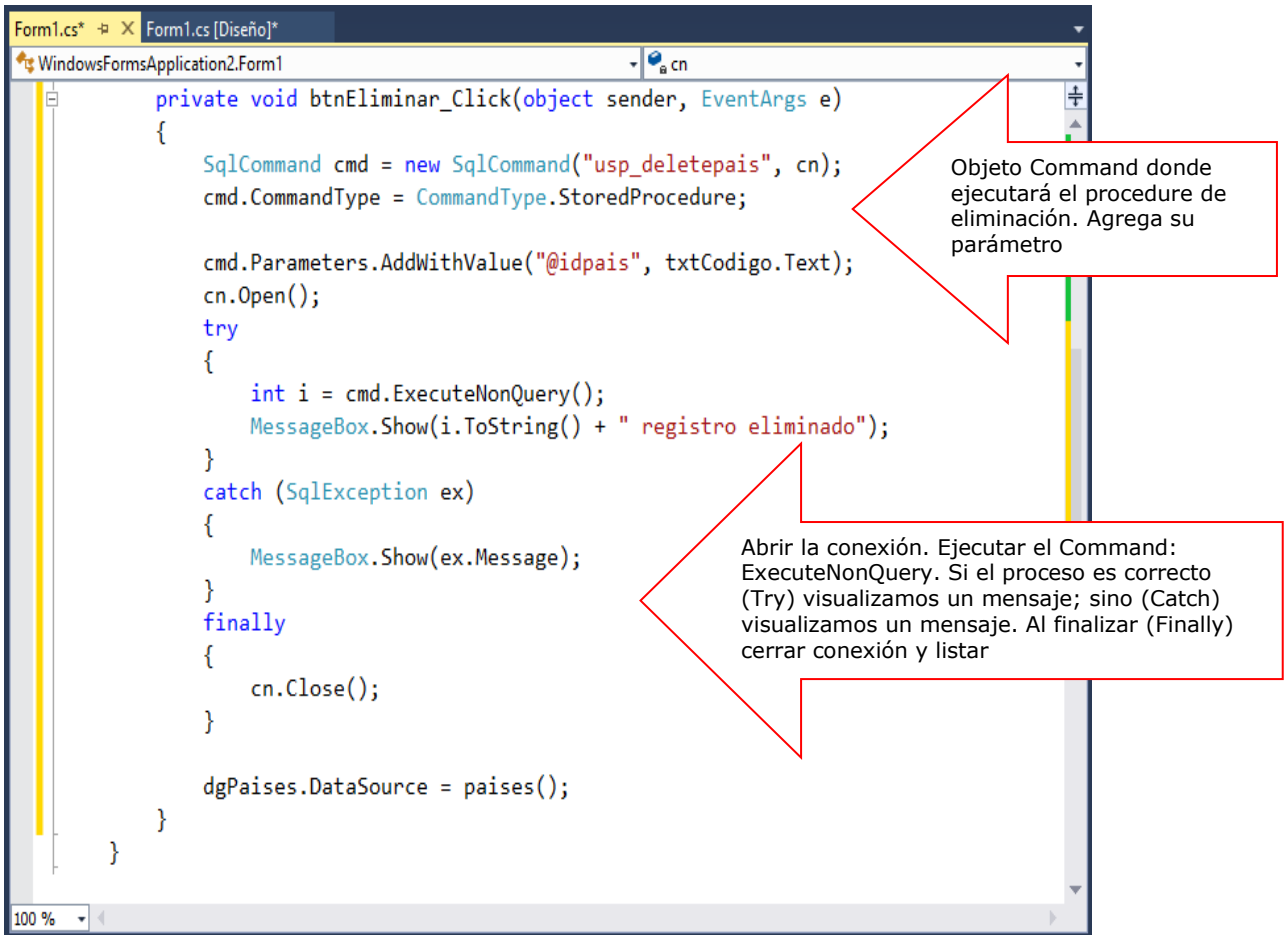
Programa el evento Click del botón Actualizar, definimos el objeto Command para ejecutar el procedure usp_updatepais, tal como se muestra



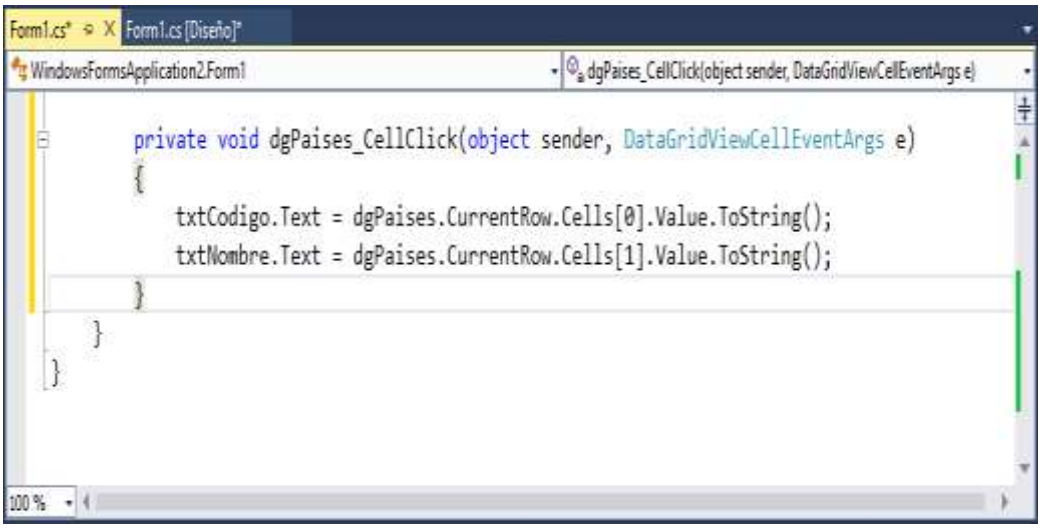
Ejecutamos la página para comprobar el proceso: ingresa el código y nombre del país. Al presionar el botón ACTUALIZAR, se ejecuta el proceso visualizando un mensaje, tal como se muestra



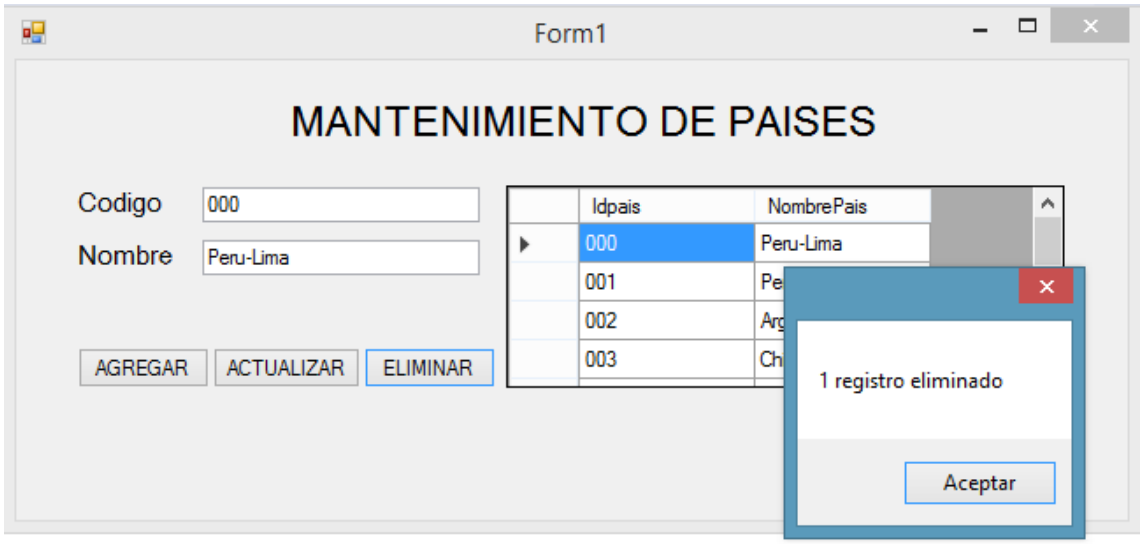
Programa el evento Click del botón Eliminar, donde ejecutamos el proceso para eliminar un registro a la tabla tb_paises por su campo idpais. En este proceso utilizamos el objeto Command para ejecutar el procedure usp_deletepais



Programa el evento CellClick del control dgPais, donde al seleccionar una fila (CurrentRow), visualizamos los datos en los controles



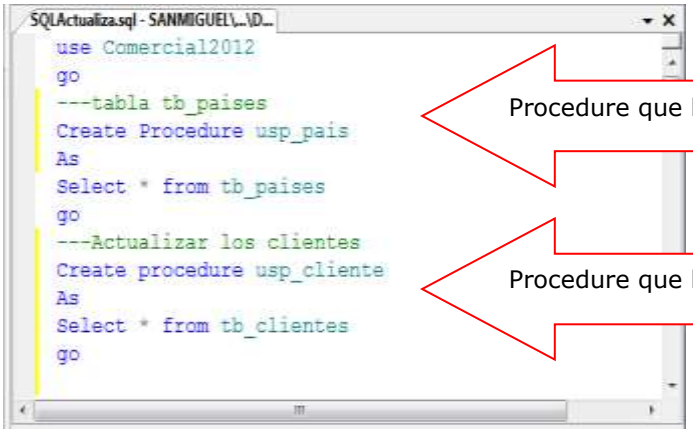
Presiona la tecla F5 para ejecutar los procesos del Formulario

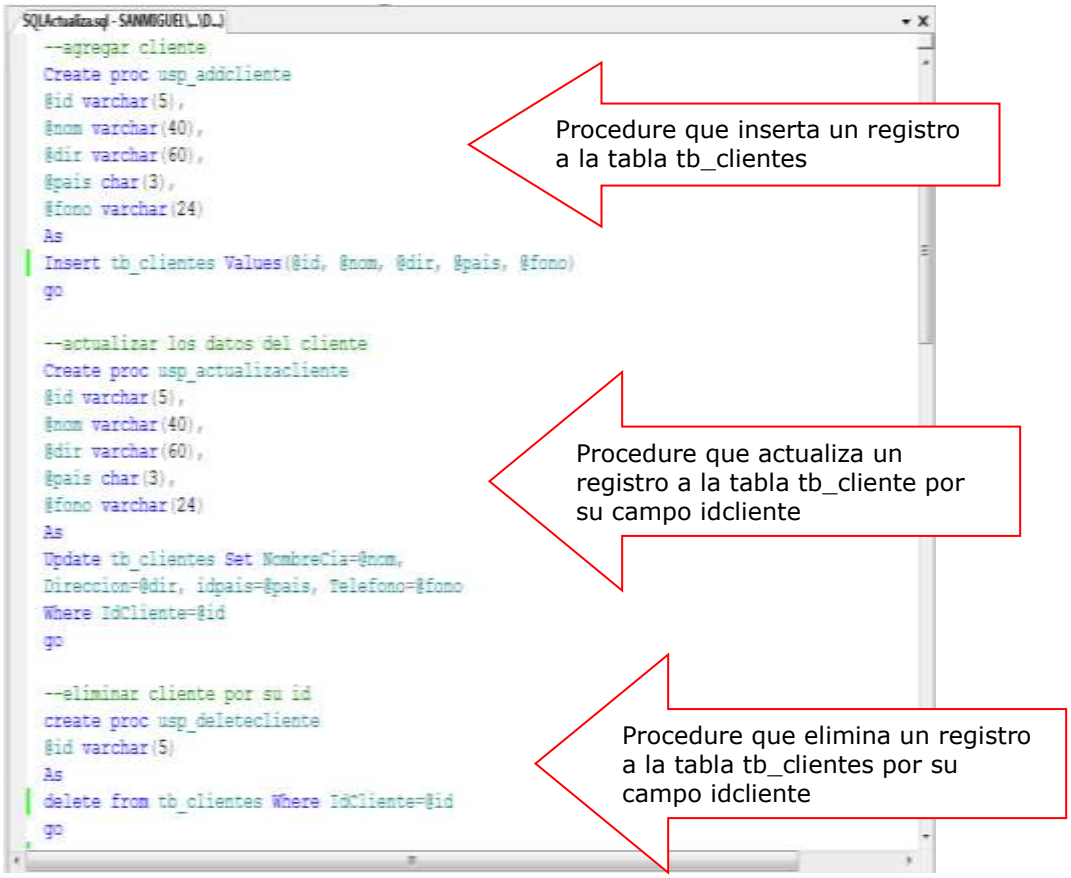


LABORATORIO 3.2

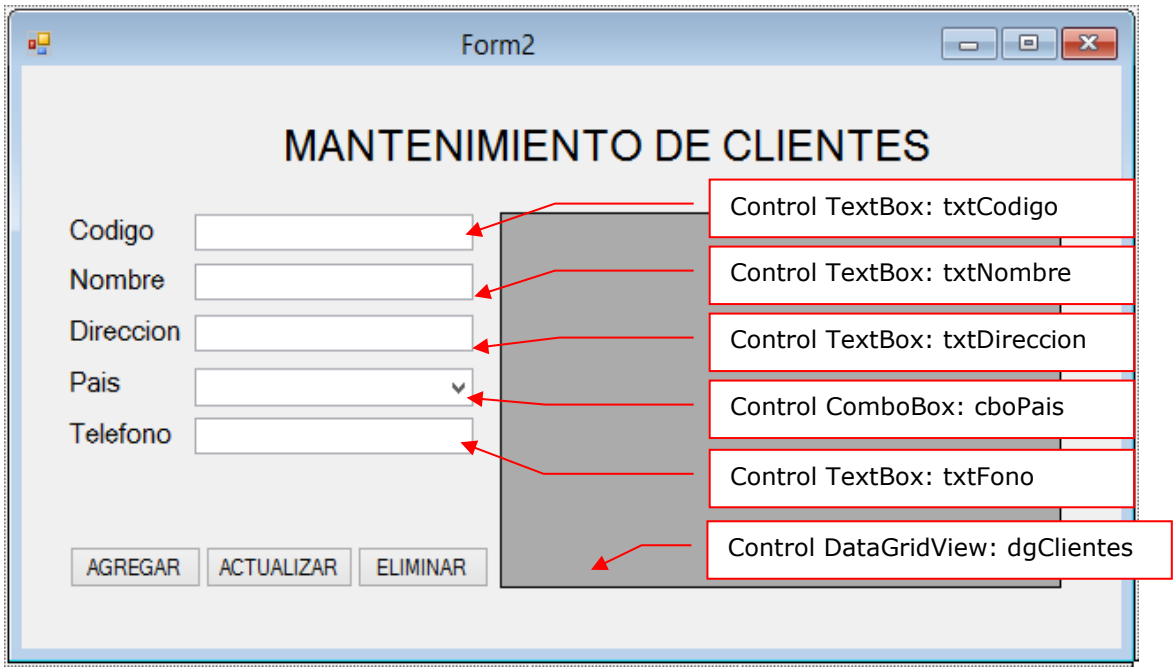
Se desea implementar un Formulario donde realice el mantenimiento a la tabla tb_clientes, utilice procedimientos almacenados en el mantenimiento.

Defina los procedimientos almacenados del proceso



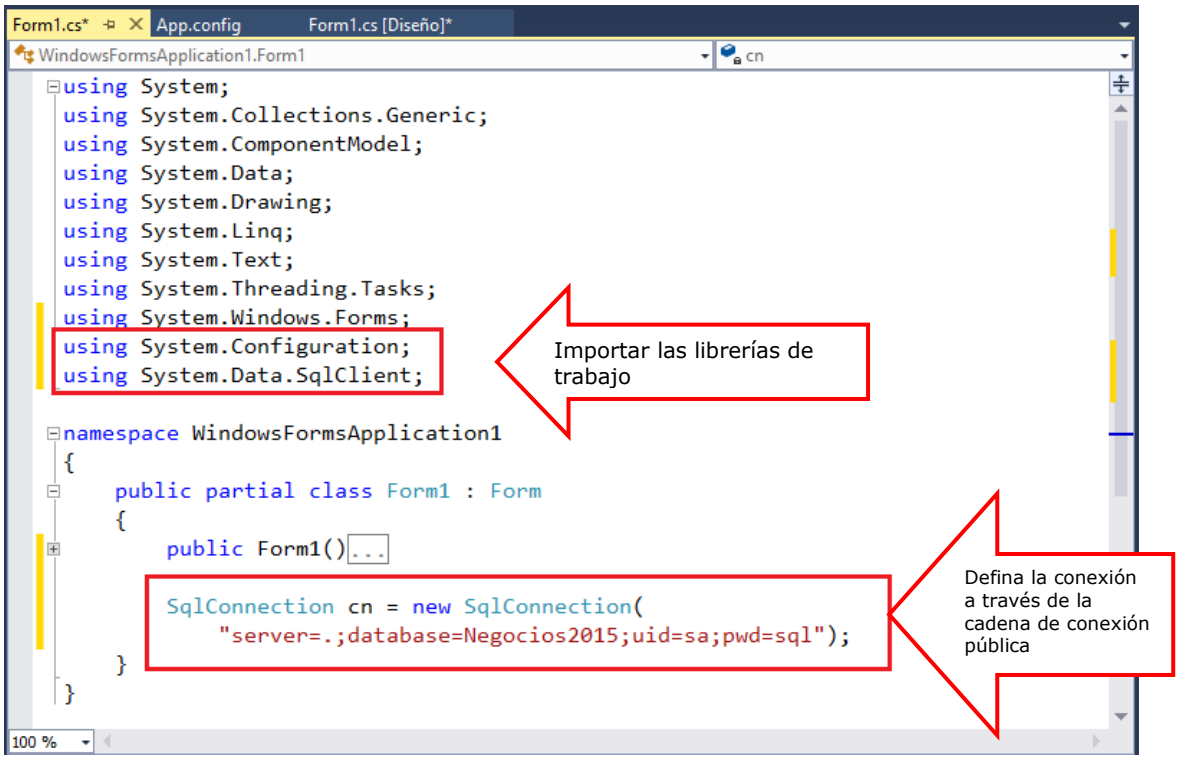


DISEÑO DEL FORMULARIO

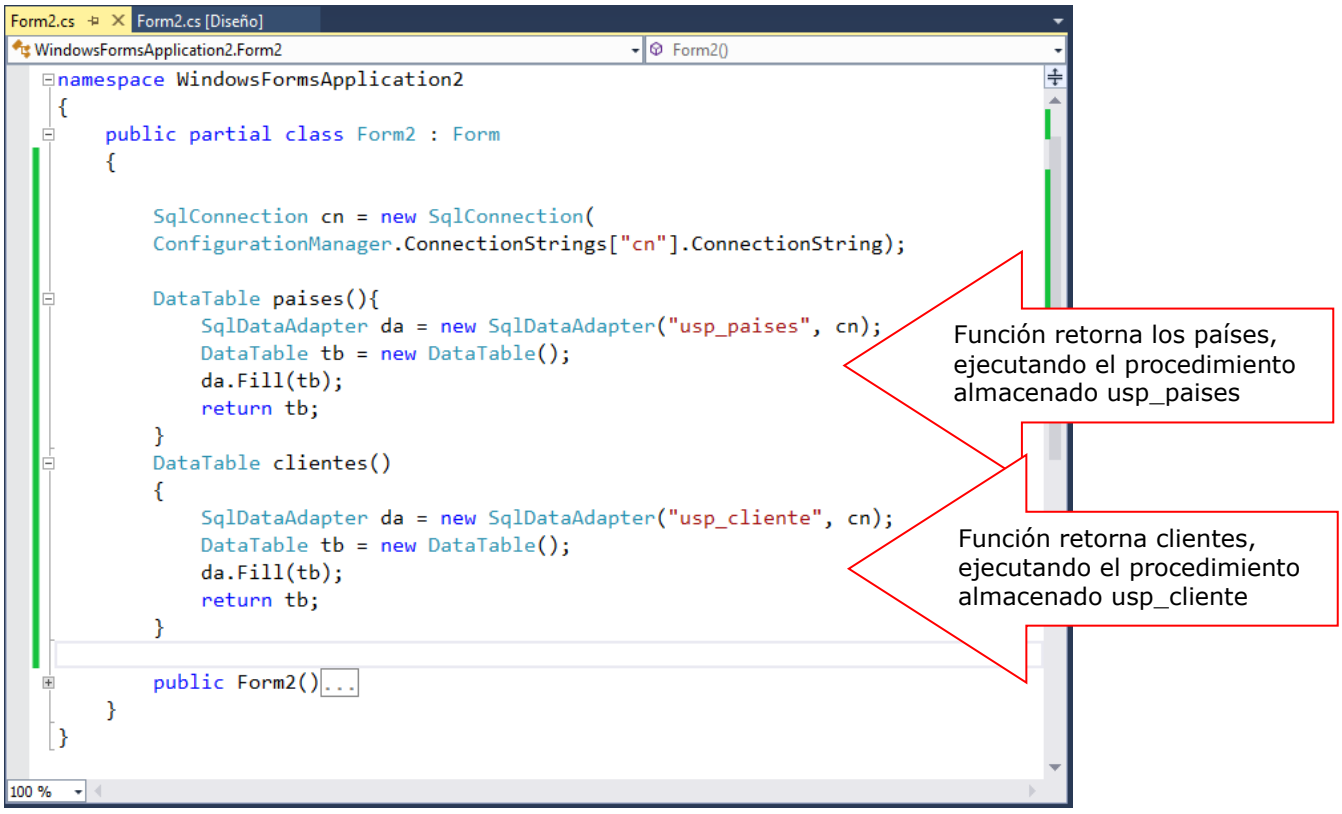


PROGRAMACIÓN

Defina las librerías de trabajo: System.Data.SqlClient y System.Configuration. Defina la conexión a la base de datos Negocios2015 instanciando el SQLConnection

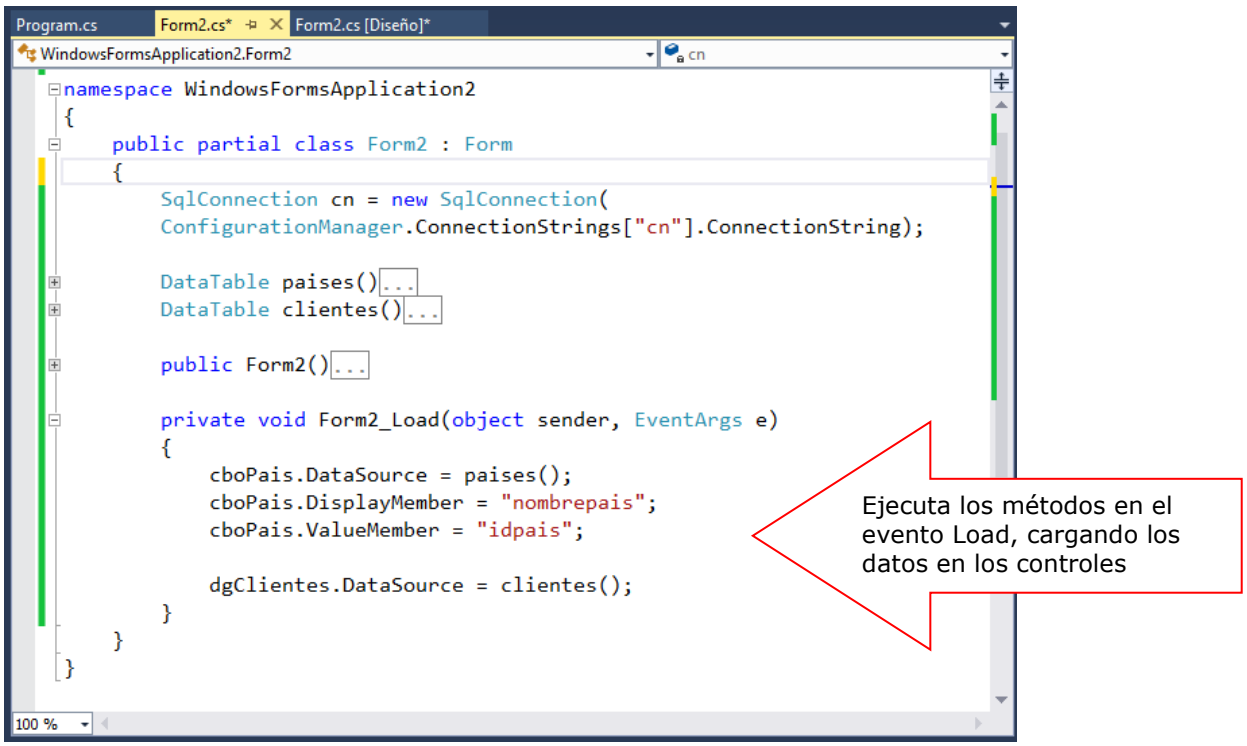


Defina una función de tipo DataTable donde retorna los registros de la tabla tb_paises; y otra función DataTable donde retorna los registros de la tabla tb_clientes



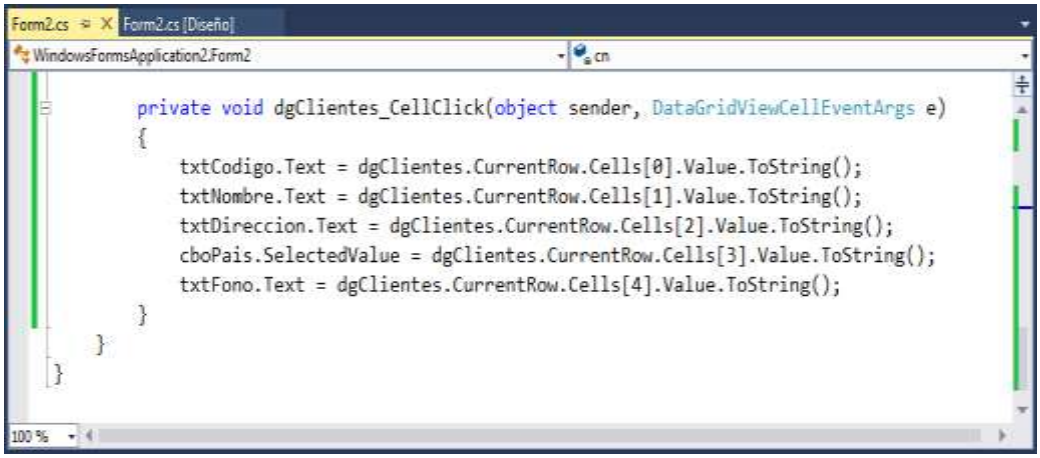

```
namespace WindowsFormsApplication2
{
    public partial class Form2 : Form
    {

        SqlConnection cn = new SqlConnection(
        ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

        DataTable paises(){
            SqlDataAdapter da = new SqlDataAdapter("usp_paises", cn);
            DataTable tb = new DataTable();
            da.Fill(tb);
            return tb;
        }
        DataTable clientes()
        {
            SqlDataAdapter da = new SqlDataAdapter("usp_cliente", cn);
            DataTable tb = new DataTable();
            da.Fill(tb);
            return tb;
        }

        public Form2()...
    }
}
```

Programa el evento Load del Formulario, cargamos los datos en los respectivos controles: cboPais y dgClientes

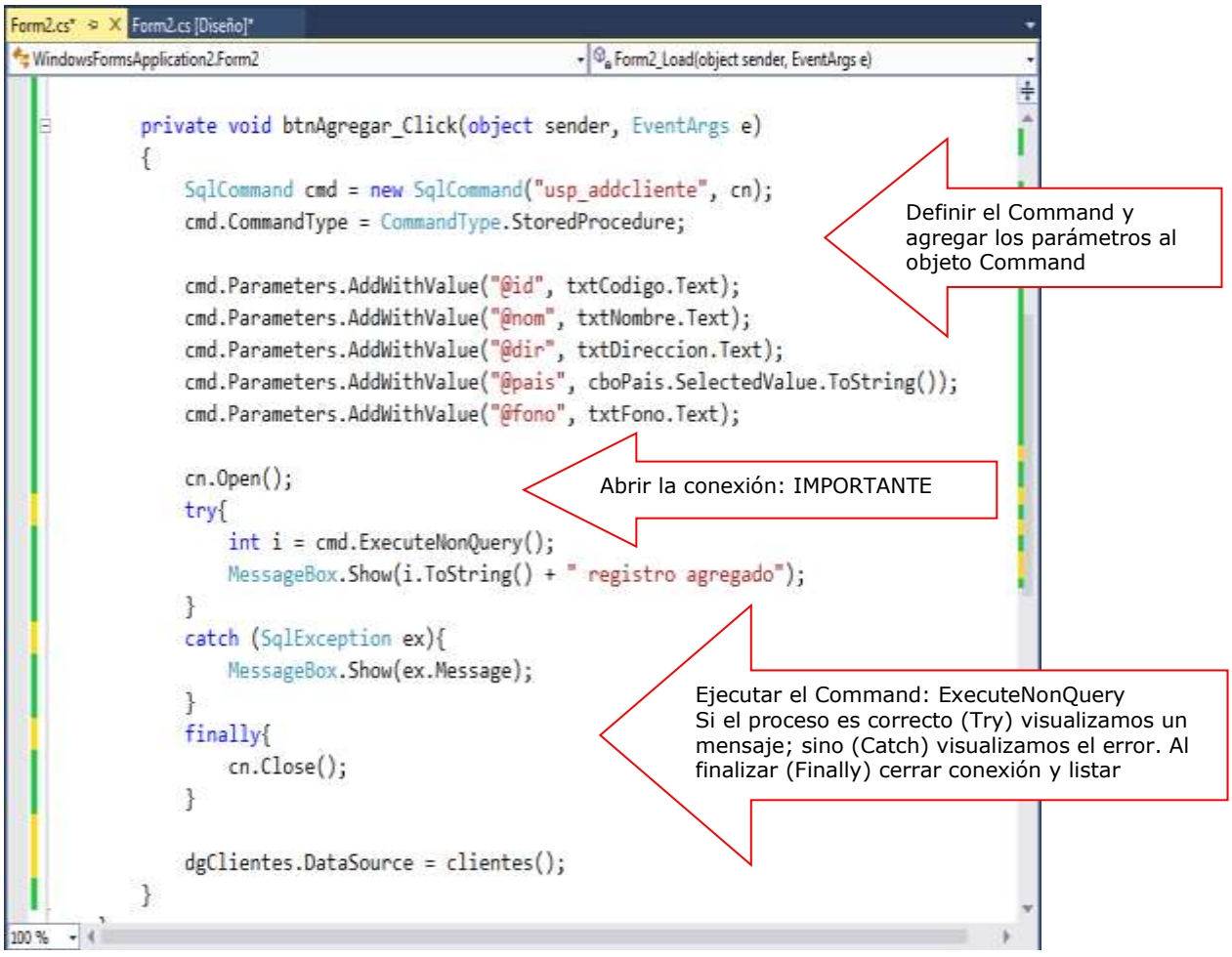



```
namespace WindowsFormsApplication2
{
    public partial class Form2 : Form
    {
        SqlConnection cn = new SqlConnection(
        ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

        DataTable paises()...
        DataTable clientes()...

        public Form2()...

        private void Form2_Load(object sender, EventArgs e)
        {
            cboPais.DataSource = paises();
            cboPais.DisplayMember = "nombrepais";
            cboPais.ValueMember = "idpais";

            dgClientes.DataSource = clientes();
        }
    }
}
```

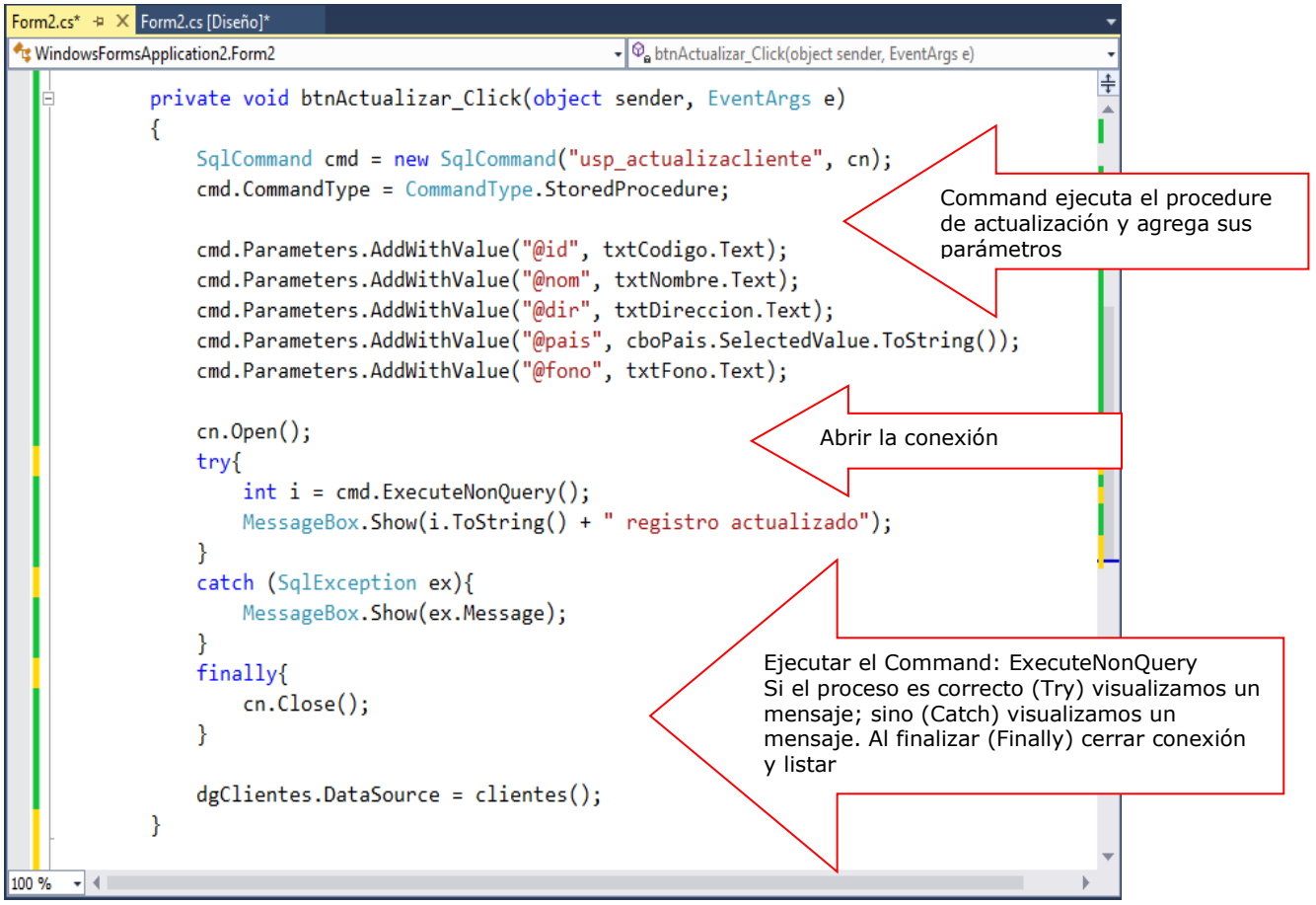
Programa el evento CellClick del control dgCliente, donde al seleccionar una fila (CurrentRow), visualizamos los datos en los controles



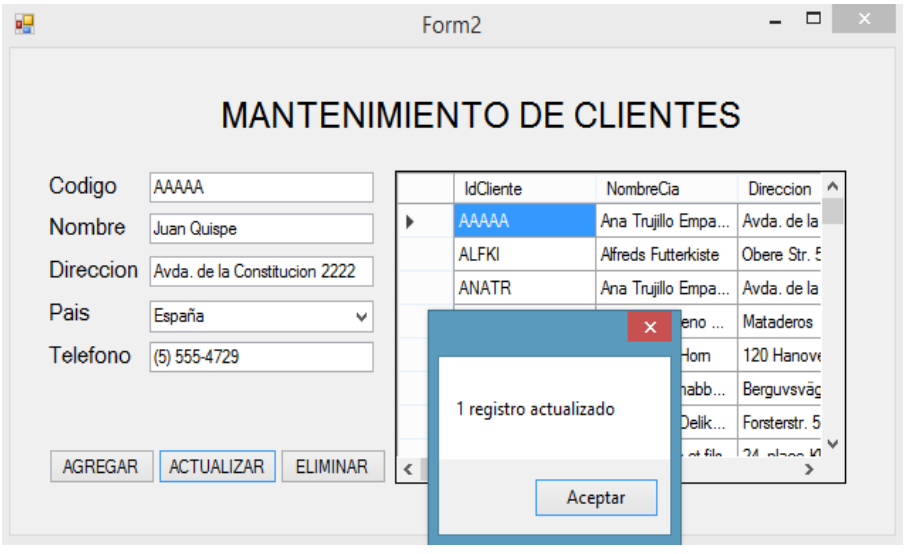
Programa el evento Click del botón Agregar, ejecuta el proceso para agregar un registro a la tabla tb_clientes. En este proceso utilizamos el objeto Command para ejecutar un procedimiento almacenado de inserción (usp_addcliente)



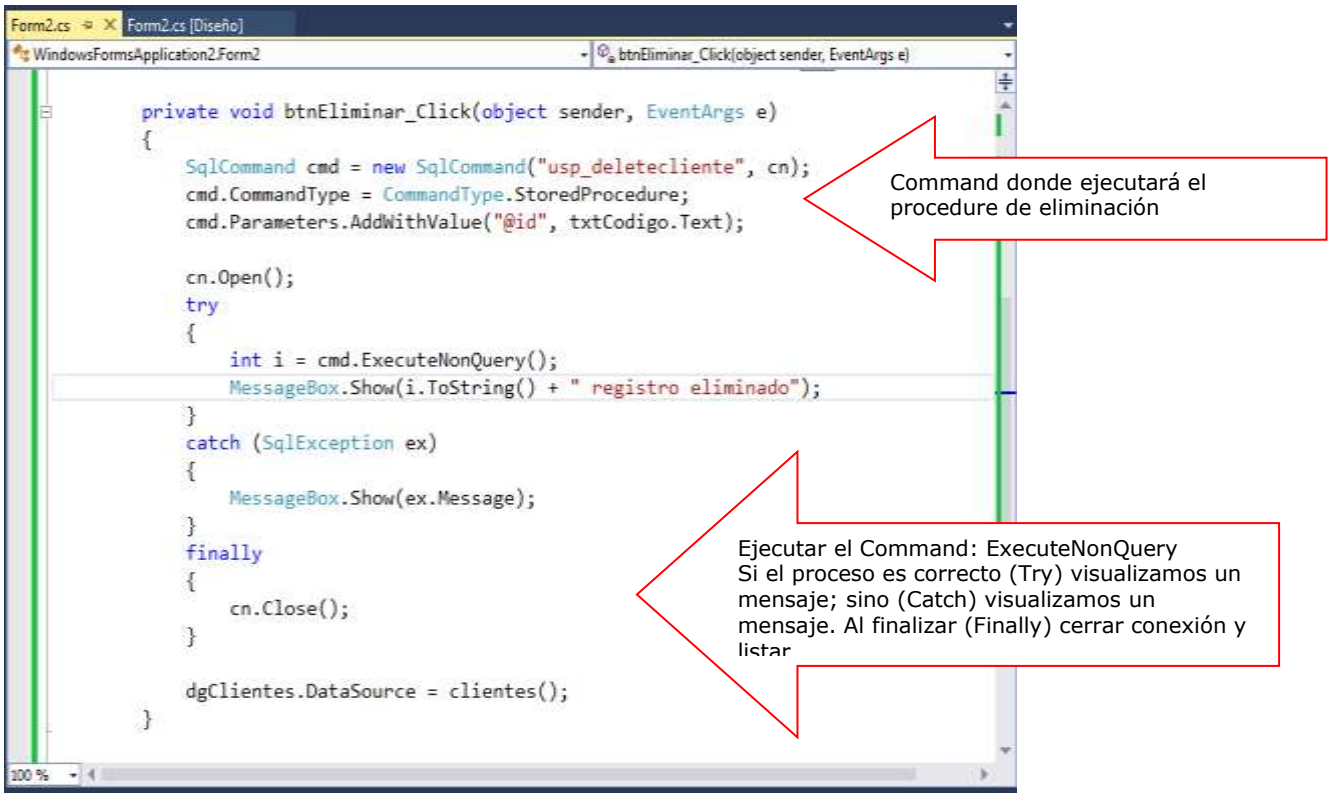
Programa el evento Click del botón Actualizar, defina el objeto Command para ejecutar el procedure usp_updatecliente.



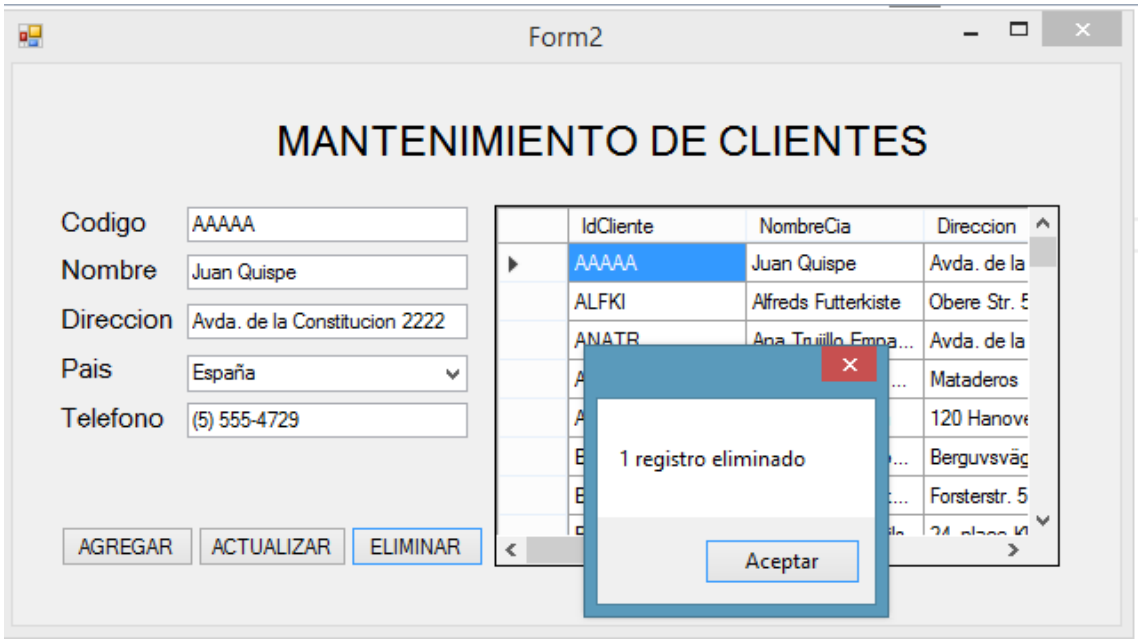
Ejecuta el Formulario para verificar los procesos definidos



Programa el evento Click del botón Eliminar, donde ejecutamos el proceso para eliminar un registro a la tabla tb_clientes por su campo idcliente. En este proceso utilizamos el objeto Command para ejecutar el procedure usp_deletecliente



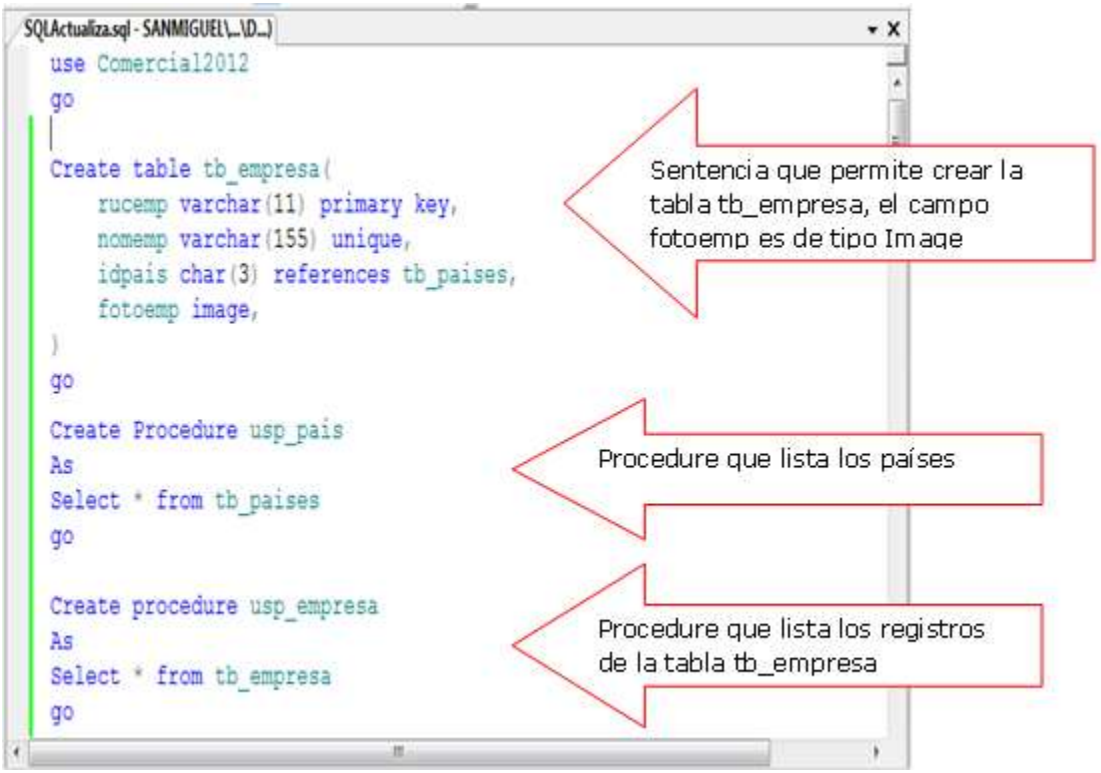
Para ejecutar presione F5

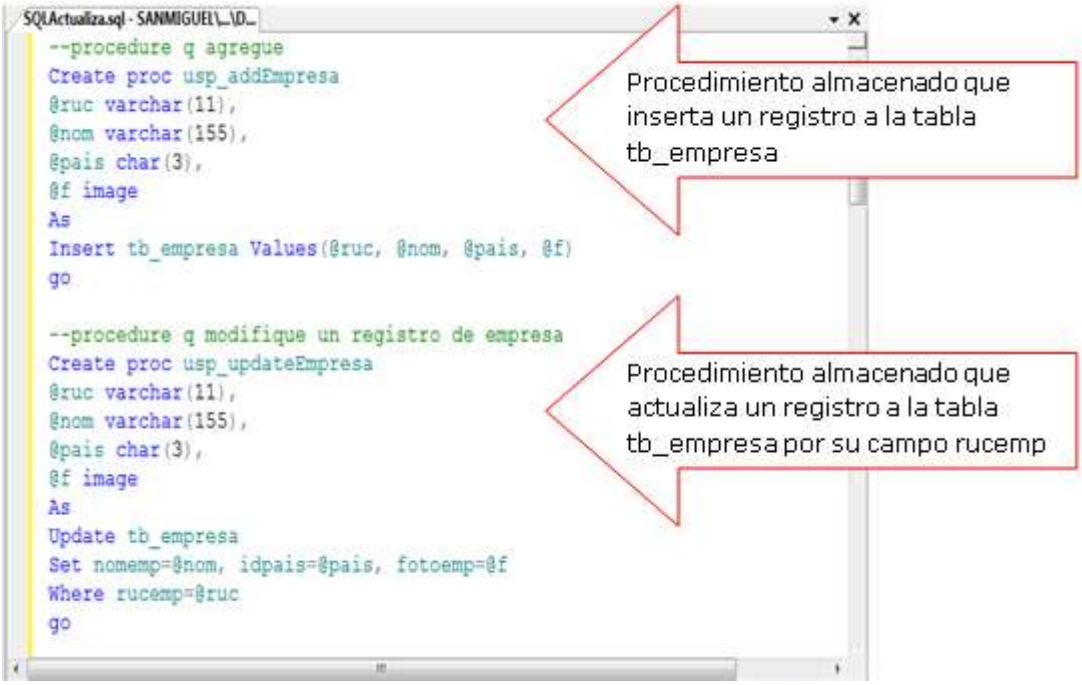


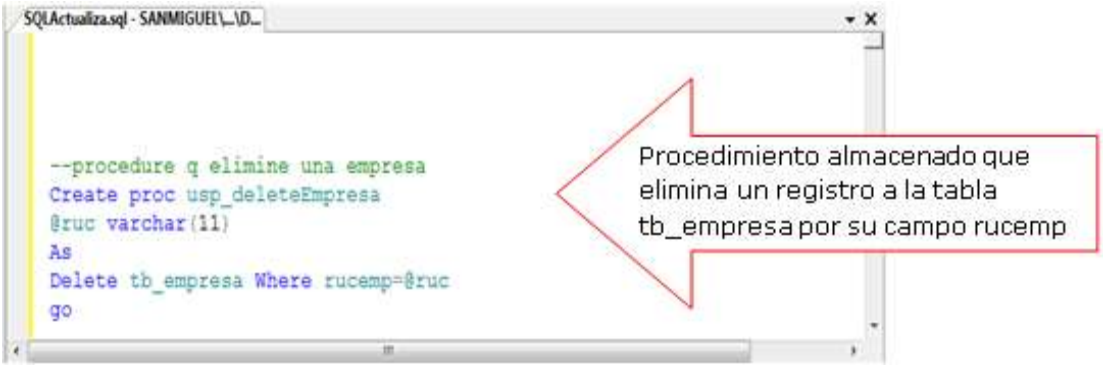
LABORATORIO 3.3

Se desea implementar un Formulario donde realice el mantenimiento a la tabla tb_empresa donde se incluya el logo de la empresa (imágenes o foto), utilice procedimientos almacenados en el mantenimiento.

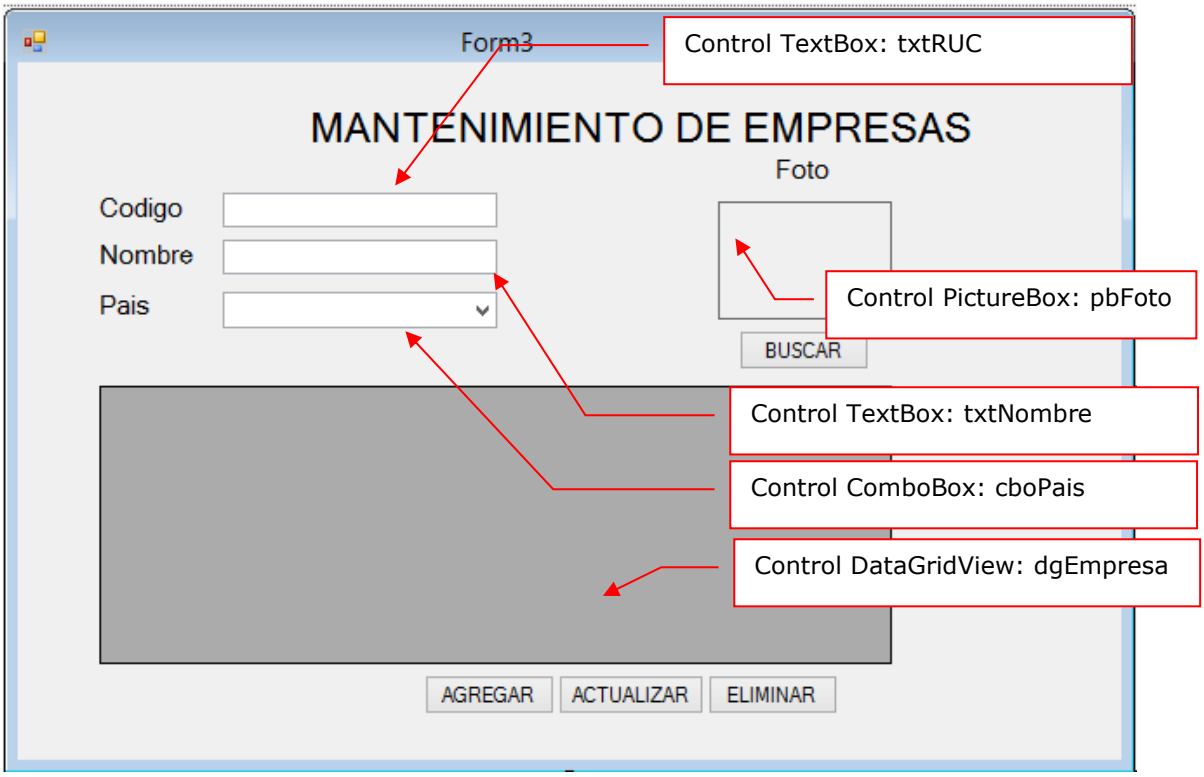
Defina los procedimientos almacenados del proceso





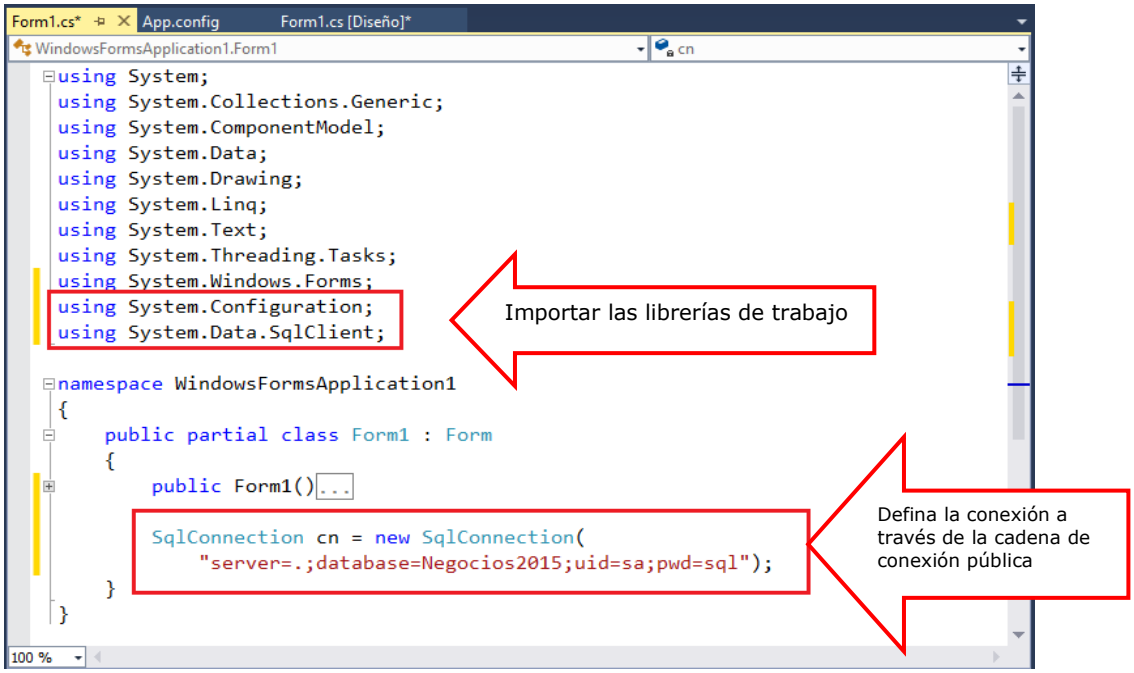


DISEÑO DEL FORMULARIO

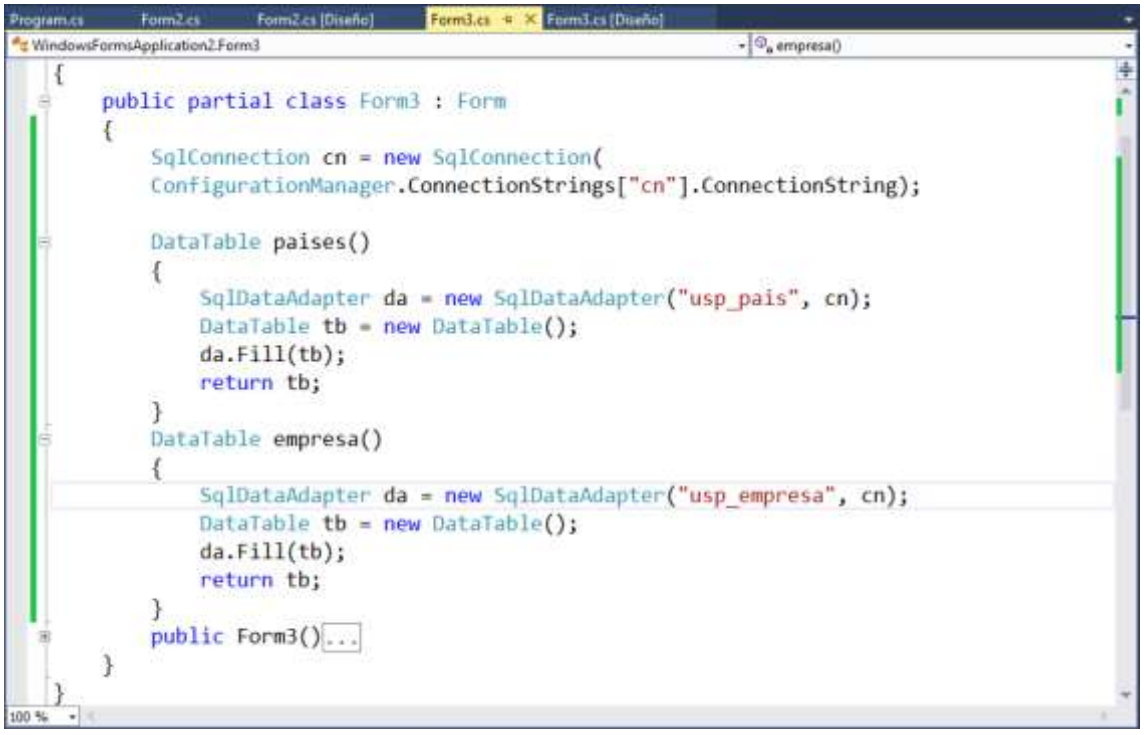


PROGRAMACIÓN

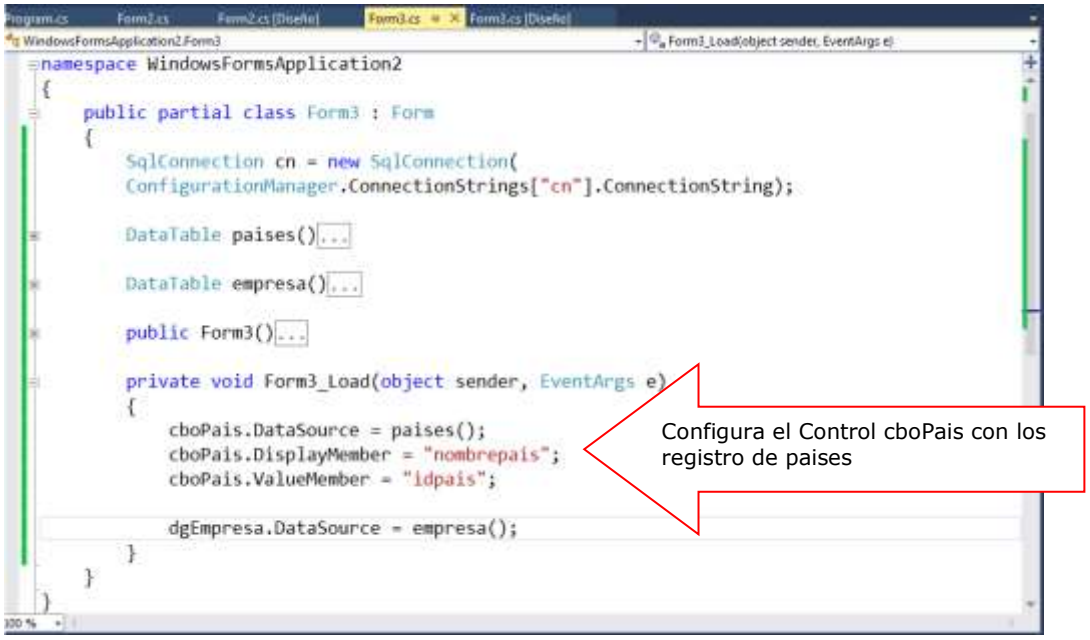
Defina las librerías de trabajo: System.Data.SqlClient (Base de Datos en SQL Server) y el System.Configuration (uso de las etiquetas app.config)



Defina la función DataTable países() donde retorna los registros de tb_paises; y la función empresa() donde retorna los registros de tb_empresa.

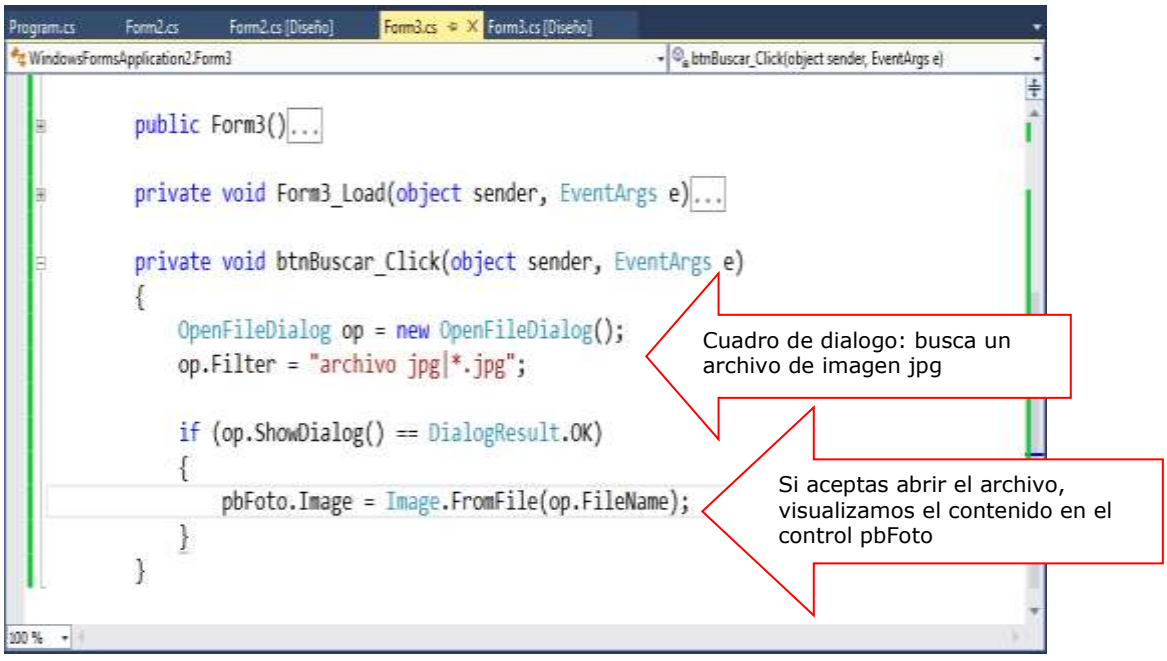


Programa el evento Load del Formulario, donde al cargar el Formulario configura el control ComboBox cboPais con los registros de países, y listamos los registros de empresa en el control DataGridView: dgEmpresa



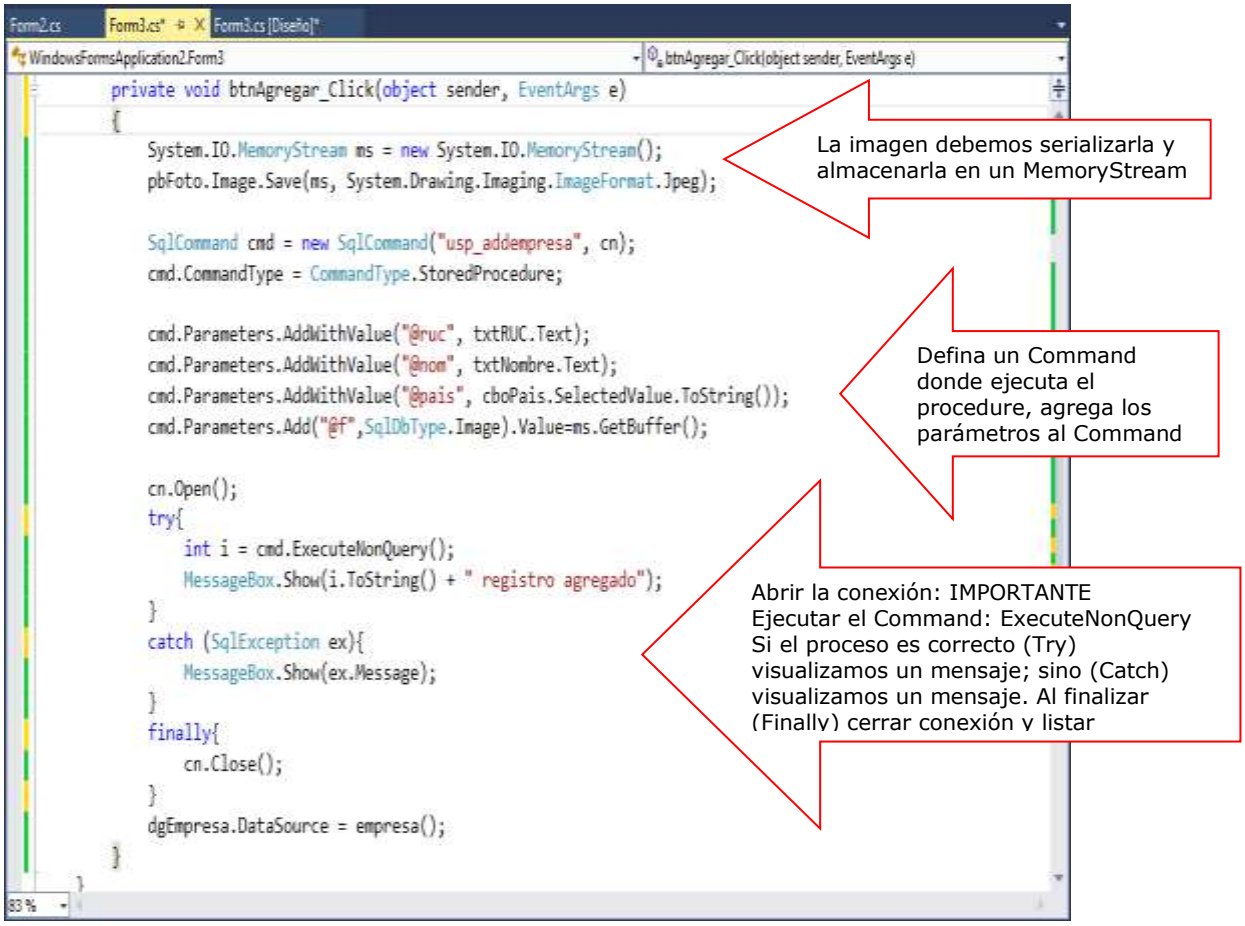

```
namespace WindowsFormsApplication2
{
    public partial class Form3 : Form
    {
        SqlConnection cn = new SqlConnection(
        ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

        DataTable paises()...

        DataTable empresa()...

        public Form3()...

        private void Form3_Load(object sender, EventArgs e)
        {
            cboPais.DataSource = paises();
            cboPais.DisplayMember = "nombrepais";
            cboPais.ValueMember = "idpais";

            dgEmpresa.DataSource = empresa();
        }
    }
}
```

Programa el evento Click del botón Buscar, donde permita buscar un archivo de imagen (jpg) visualizando el archivo de imagen en el control pbFoto

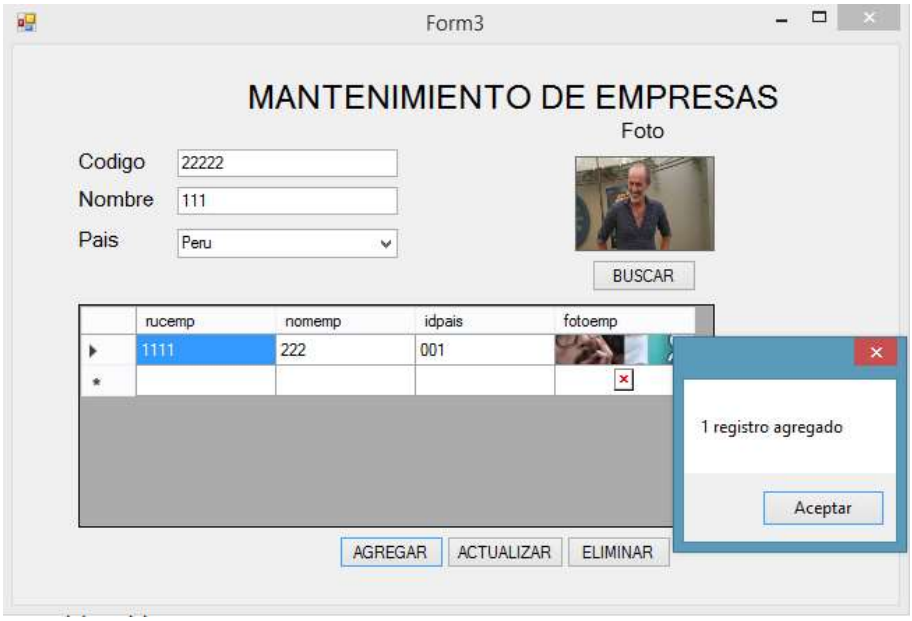



```
public Form3()...

private void Form3_Load(object sender, EventArgs e)...

private void btnBuscar_Click(object sender, EventArgs e)
{
    OpenFileDialog op = new OpenFileDialog();
    op.Filter = "archivo jpg|*.jpg";

    if (op.ShowDialog() == DialogResult.OK)
    {
        pbFoto.Image = Image.FromFile(op.FileName);
    }
}
```

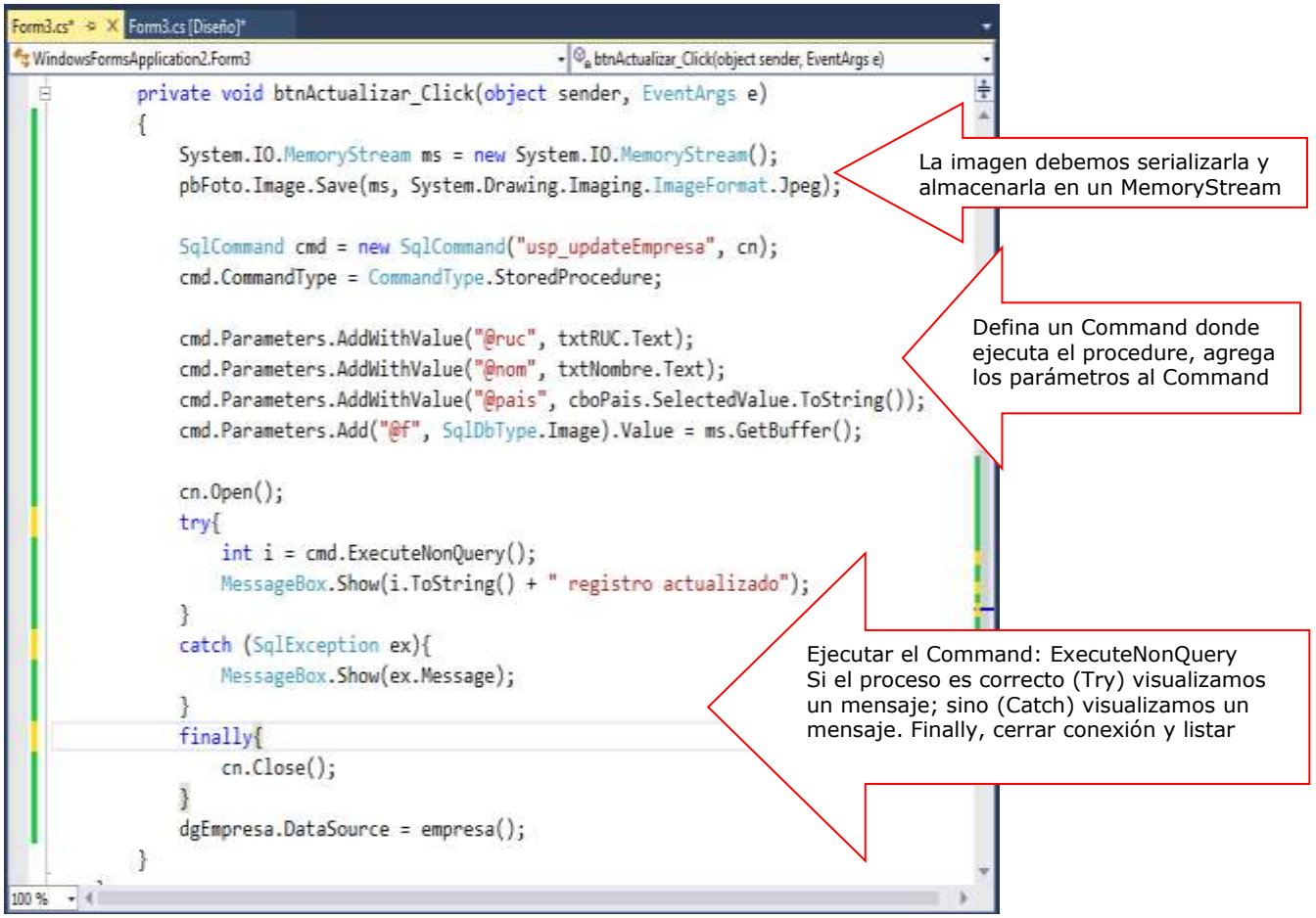
Programa el evento Click del botón Agregar, utilizamos el objeto Command para ejecutar un procedimiento almacenado usp_addempresa



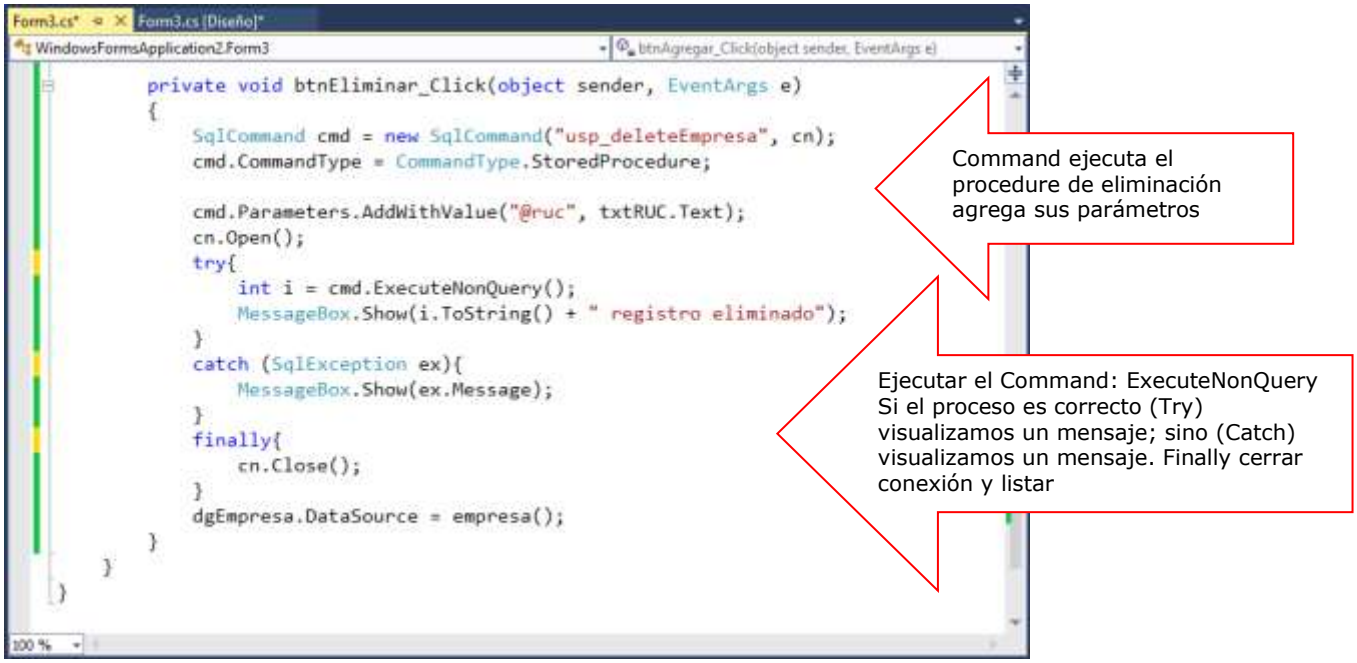
Ejecutamos el formulario, ingresamos datos, al presionar el botón AGREGAR insertamos el registro



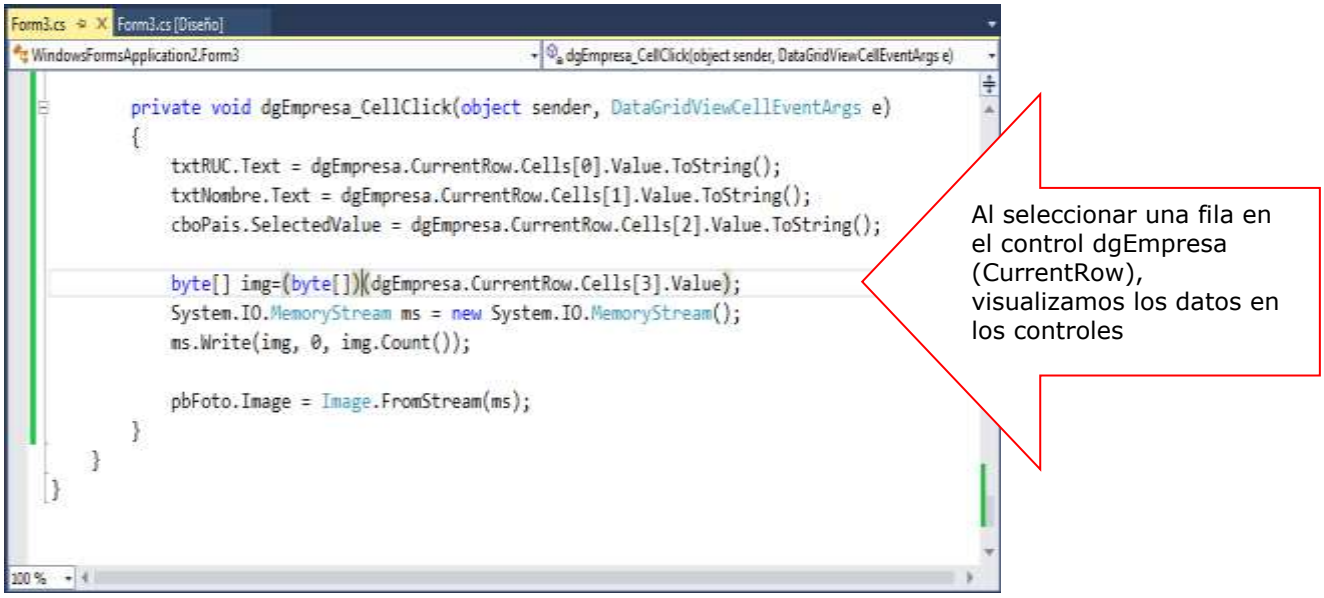
Programa el evento Click del botón Actualizar; en este proceso un Command ejecuta el procedure usp_updateEmpresa donde actualiza los datos de una empresa



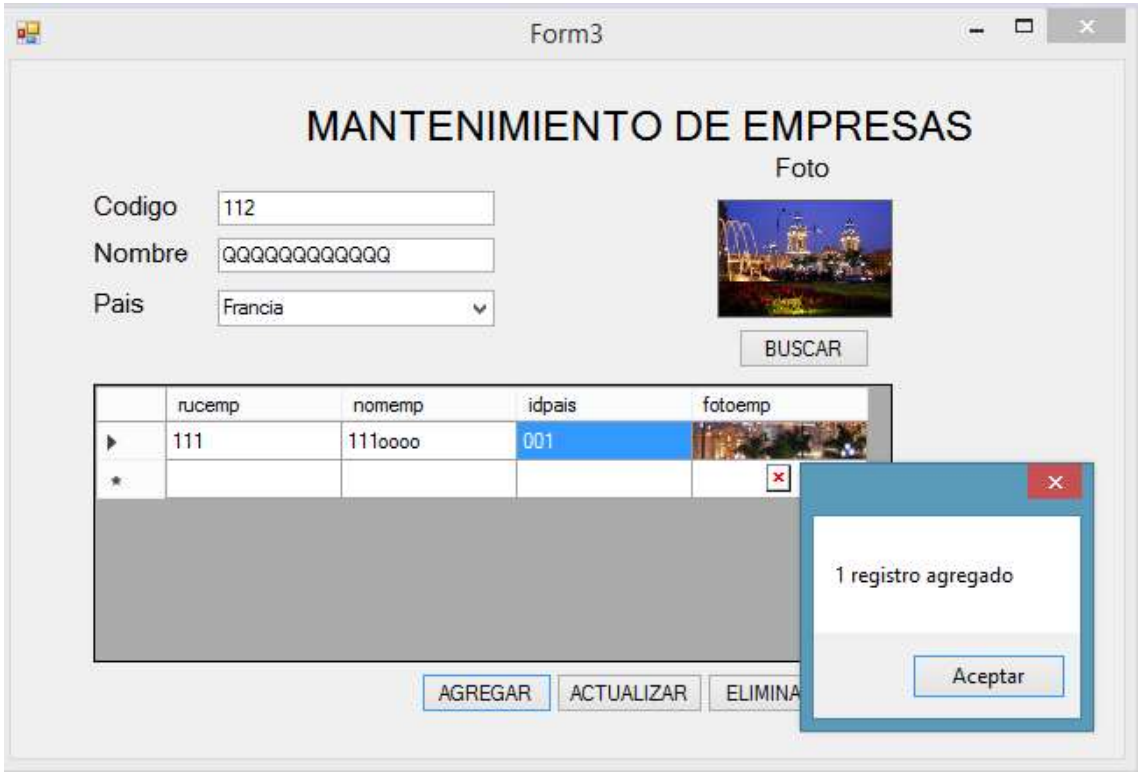
Programa el evento Click del botón Eliminar. En este proceso utilizamos el objeto Command para ejecutar el procedure usp_deleteEmpresa



Programa el evento CellClick del control dgEmpresa, donde al seleccionar una fila, visualizamos los datos en los controles



Presiona la tecla F5 para ejecutar los procesos del Formulario



Resumen

- Para crear un comando, puede utilizar uno de los constructores de comando del proveedor de datos .NET Framework con el que esté trabajando. Los constructores pueden aceptar argumentos opcionales, como una instrucción SQL para ejecutar en el origen de datos, un objeto **DbConnection** o un objeto **DbTransaction**.
- El proveedor de datos .NET Framework para OLE DB incluye un objeto **OleDbCommand**, el proveedor de datos .NET Framework para SQL Server incluye un objeto **SqlCommand**, el proveedor de datos .NET Framework para ODBC incluye un objeto **OdbcCommand** y el proveedor de datos .NET Framework para Oracle incluye un objeto **OracleCommand**.
- El objeto Command de ADO.NET tiene un método **ExecuteReader** que permite ejecutar una consulta que retorna uno o más conjunto de resultados. Al ejecutar el método **ExecuteReader**, podemos pasar un parámetro al método el cual representa la enumeración **CommandBehavior**, que permite controlar al Command como se ejecutado.
- Los procedimientos almacenados ofrecen numerosas ventajas en el caso de aplicaciones que procesan datos. Mediante el uso de procedimientos almacenados, las operaciones de bases de datos se pueden encapsular en un solo comando, optimizar para lograr el mejor rendimiento, y mejorar con seguridad adicional.
- El proveedor de datos de .NET Framework para SQL Server no admite el uso del marcador de posición de signo de interrogación de cierre (?) para pasar parámetros a una instrucción SQL o a un procedimiento almacenado. Este proveedor trata los parámetros del procedimiento almacenado como parámetros con nombre y busca marcadores de parámetros coincidentes.
- Para crear un objeto **DbParameter**, se puede usar su constructor o bien se puede agregar a **DbParameterCollection** mediante una llamada al método Add de la colección **DbParameterCollection**.
- Las instrucciones SQL que modifican datos (por ejemplo INSERT, UPDATE o DELETE) no devuelven ninguna fila. De la misma forma, muchos procedimientos almacenados realizan alguna acción pero no devuelven filas. Para ejecutar comandos que no devuelvan filas, cree un objeto Command con el comando SQL adecuado y una Connection, incluidos los **Parameters** necesarios. El comando se debe ejecutar con el método **ExecuteNonQuery** del objeto **Command**.

- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  - http://msdn.microsoft.com/es-es/library/ms254953(v=vs.110).aspx
  - http://msdn.microsoft.com/es-es/library/yy6y35y8(v=vs.110).aspx
  - http://www.ehu.es/mrodriguez/archivos/csharppdf/ADONET/ADONET.pdf

UNIDAD

1

OPERACIONES CONECTADAS A UN ORIGEN DE DATOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno realiza operaciones de consulta y actualización de datos en el entorno de una aplicación Windows conectado a un origen de datos utilizando la librería ADO.NET

TEMARIO

Tema 4: Manejo de transacciones de datos (3 horas)

1. Transacciones en .NET
   1.1. Implementando una transacción Explicita e implícita
   1.2. Manejo de la clase Committable Transaction en procesos de actualización
   1.3. Manejo de transacciones en la capa de datos

ACTIVIDADES PROPUESTAS

- Los alumnos diseñan formularios que manejan la conexión a una fuente de datos para realizar operaciones de actualización de datos a la fuente de datos.
- Los alumnos desarrollan los laboratorios de esta semana

4. MANEJO DE TRANSACCIONES EN .NET

Cuando se compra un libro de una librería en línea, se intercambia dinero (en forma de crédito) por el libro. Si tiene disponibilidad de crédito, una serie de operaciones relacionadas garantiza que se obtenga el libro y la librería obtiene el dinero. Sin embargo, si la operación sufre de un error durante el intercambio comercial, el erro afecta a la totalidad del proceso. No se obtiene el libro y la librería no obtiene el dinero.

Una transacción consiste en un comando único o en un grupo de comandos que se ejecutan como un paquete. Las transacciones permiten combinar varias operaciones en una sola unidad de trabajo. Si en un punto de la transacción se produjera un error, todas las actualizaciones podrían revertirse y devolverse al estado que tenían antes de la transacción.

Una transacción debe ajustarse a las propiedades: atomicidad, coherencia, aislamiento y durabilidad para poder garantizar la coherencia de datos. La mayoría de los sistemas de bases de datos relacionales, como Microsoft SQL Server, admiten transacciones, al proporcionar funciones de bloqueo, registro y administración de transacciones cada vez que una aplicación cliente realiza una operación de actualización, inserción o eliminación.

4.1 IMPLEMENTANDO UNA TRANSACCION IMPLICITA Y EXPLICITA

Una transacción explícita es aquella en que se define explícitamente el inicio y el final de la transacción. Las aplicaciones utilizan las instrucciones BEGIN TRANSACTION, COMMIT TRANSACTION, COMMIT WORK, ROLLBACK TRANSACTION o ROLLBACK WORK de Transact-SQL para definir transacciones explícitas.

BEGIN TRANSACTION

Marca el punto de inicio de una transacción explícita para una conexión.

COMMIT TRANSACTION o COMMIT WORK

Se utiliza para finalizar una transacción correctamente si no hubo errores. Todas las modificaciones de datos realizadas en la transacción se convierten en parte permanente de la base de datos. Se liberan los recursos ocupados por la transacción.

ROLLBACK TRANSACTION o ROLLBACK WORK

Se utiliza para eliminar una transacción en la que se encontraron errores. Todos los datos modificados por la transacción vuelven al estado en el que estaban al inicio de la transacción. Se liberan los recursos ocupados por la transacción.

En ADO, utilice el método BeginTrans en un objeto Connection para iniciar una transacción explícita. Para finalizar la transacción, llame a los métodos CommitTrans o RollbackTrans del objeto Connection.

En el proveedor administrado de SqlCliente de ADO.NET, utilice el método BeginTransaction en un objeto SqlConnection para iniciar una transacción explícita. Para finalizar la transacción, llame a los métodos Commit() o Rollback() del objeto SqlTransaction.

El modo de transacciones explícitas se mantiene solamente durante la transacción. Cuando la transacción termina, la conexión vuelve al modo de transacción en que estaba antes de iniciar la transacción explícita, es decir, el modo implícito o el modo de confirmación automática.

Cada proveedor de datos de .NET Framework tiene su propio objeto Transaction para realizar transacciones locales. Si necesita que se realice una transacción en una base de datos de SQL Server, seleccione una transacción de **System**.**Data**.**SqlClient**. En transacciones de Oracle, utilice el proveedor **System**.**Data**.**OracleClient**. Además, existe una nueva clase **DbTransaction** disponible para la escritura de código independiente del proveedor que requiere transacciones.

En ADO.NET, las transacciones se controlan con el objeto Connection. Puede iniciar una transacción local con el método **BeginTransaction**. Una vez iniciada una transacción, puede inscribir un comando en esa transacción con la propiedad Transaction de un objeto **Command**. Luego, puede confirmar o revertir las modificaciones realizadas en el origen de datos según el resultado correcto o incorrecto de los componentes de la transacción. Las operaciones para confirmar una transacción es **Commit** y la operación para revertir o deshacer una transacción es **RollBack**.

Una transacción implícita inicia una nueva transacción en una conexión a SQL Server Database Engine (Motor de base de datos de SQL Server) después de confirmar o revertir la transacción actual. No tiene que realizar ninguna acción para delinear el inicio de una transacción, sólo tiene que confirmar o revertir cada transacción. El modo de transacciones implícitas genera una cadena continua de transacciones.

La transacción sigue activa hasta que emita una instrucción COMMIT o ROLLBACK. Una vez que la primera transacción se ha confirmado o revertido, la instancia Motor de base de datos inicia automáticamente una nueva transacción la siguiente vez que la conexión ejecuta una de estas instrucciones. La instancia continúa generando una cadena de transacciones implícitas hasta que se desactiva el modo de transacciones implícitas.

El modo de transacciones implícitas se establece mediante la instrucción SET de Transact-SQL o a través de funciones y métodos de la API de bases de datos.

4.2 MANEJO DE LA CLASE COMMITTABLETRANSACCION EN PROCESOS DE ACTUALIZACION

La clase CommittableTransaction proporciona a las aplicaciones una manera explícita de utilizar una transacción, a diferencia de utilizar implícitamente la clase TransactionScope. A diferencia de la clase **TransactionScope**, el sistema de escritura de la aplicación ha de llamar específicamente a los métodos Commit y Rollback para confirmar o anular la transacción. Sin embargo, sólo el creador de una transacción puede confirmar la transacción. Por consiguiente, las copias de

transacciones que se pueden confirmar, obtenidas a través del método Clone no se pueden confirmar.

USO DEL TRANSACTIONSCOPE

La clase **TransactionScope** crea un bloque de código transaccional al inscribir implícitamente las conexiones en una transacción distribuida. Debe llamar al método **Complete** al final del bloque **TransactionScope** antes de abandonarlo. Al salir del bloque se invoca el método **Dispose**. Si se ha producido una excepción que ocasiona que el código salga del ámbito, la transacción se considera anulada.

Se recomienda el uso de un bloque using para asegurarse de que se llama a **Dispose** en el objeto **TransactionScope** cuando se sale de dicho bloque. Si no se confirman ni revierten las transacciones pendientes, el rendimiento puede verse seriamente afectado ya que el tiempo de espera predeterminado de **TransactionScope** es un minuto. Si no utiliza una instrucción using, todo el trabajo deberá realizarlo en un bloque Try y llamar explícitamente al método **Dispose** en el bloque Finally.

Si se produce una excepción en **TransactionScope**, la transacción se marca como incoherente y se abandona. Se revertirá cuando se elimine el **TransactionScope**. Si no se produce ninguna excepción, las transacciones participantes se confirman.

La clase **TransactionScope** crea una transacción con un **IsolationLevel** predeterminado de **Serializable**. Dependiendo de la aplicación, podría estudiar la posibilidad de reducir el nivel de aislamiento para evitar una elevada contención en la aplicación.

4.3 MANEJO DE TRANSACCIONES EN LAS CAPAS

Para implementar las transacciones utilizamos el método BeginTransaction(), el cual admite varios tipos de parámetros, dependiendo del modo de aislamiento que queramos usar en nuestra base de datos.

Después de que una transacción se confirma o revierte, el nivel de aislamiento de la transacción conserva para todos los comandos posteriores que están en el modo de confirmación automática (el valor predeterminado de SQL Server). Esto puede producir resultados imprevistos, por ejemplo, que se conserve un nivel de aislamiento REPEATABLE READ y se bloquee a los demás usuarios de una fila. Para restaurar el nivel de aislamiento en el valor predeterminado (LECTURA CONFIRMADA), ejecutar Transact-SQL SET la instrucción de la LECTURA CONFIRMADA de NIVEL OF AISLAMIENTO OF TRANSACTION o, la llamada SqlConnection.BeginTransaction seguido por SqlTransaction.Commit.

Respecto a la instruccion using(){}, con ella conseguimos asegurarnos de que siempre se cierre la conexión.

En el caso de querer realizar operaciones en bloque con tablas relacionadas por ejemplo, podremos usar SqlTransaction, de forma que abstraemos el manejo de transacciones a las demás capas de la arquitectura (si trabajamos en una arquitectura de n-capas) aunque puede ser una solución costosa en diseño si no se planifica bien o también podemos usar TransactionScope si queremos manejar las operaciones desde la capa de negocio.

LABORATORIO 4.1

Se desea implementar un Formulario donde realice el mantenimiento a la tabla tb_paises, utilice procedimientos almacenados en el mantenimiento. Implemente transacciones en el proceso.

Defina los procedimientos almacenados del proceso

```
use Comercial2012
go
---tabla tb_paises
Create Procedure usp_pais
As
Select * from tb_paises
go
---Procedure que inserta tb_paises
Create Procedure usp_addpais
@idpais char(3),
@nombrepais varchar(40)
As
Insert tb_paises
values(@idpais,@nombrepais)
go
---Procedure que actualiza tb_paises por id
Create Procedure usp_updatepais
@idpais char(3),
@nombrepais varchar(40)
As
Update tb_paises
Set NombrePais=@nombrepais Where Idpais=@idpais
go
---Procedure que elimina un registro
Create Procedure usp_deletepais
@idpais char(3)
As
Delete from tb_paises Where Idpais=@idpais
go
```

Procedure que lista los países

Procedure que inserta un registro a la tabla tb_paises

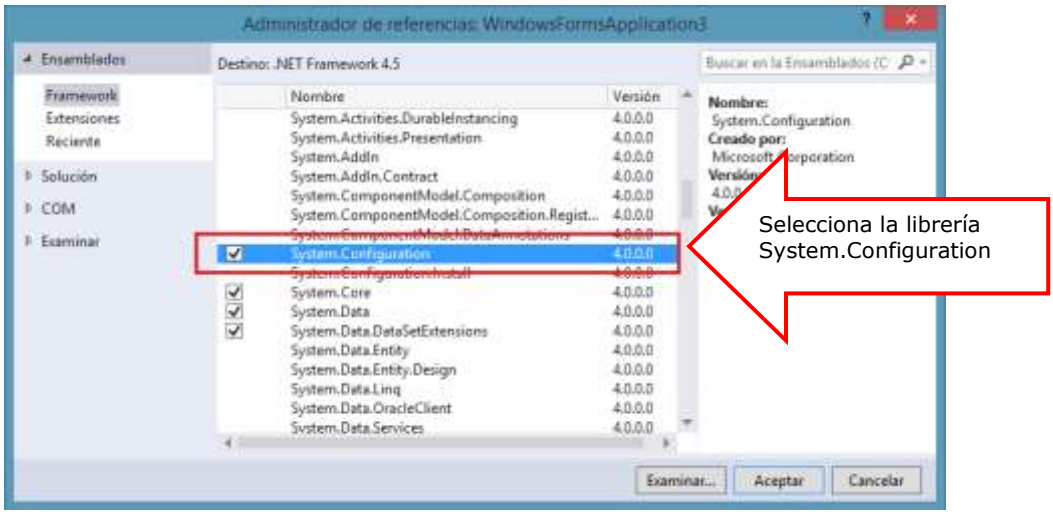
Procedure que actualiza un registro a la tabla tb_paises por su

Procedure que elimina un registro a la tabla tb_paises por su idpais

En el proyecto defina la cadena de conexión <connectionStrings> en el App.config.

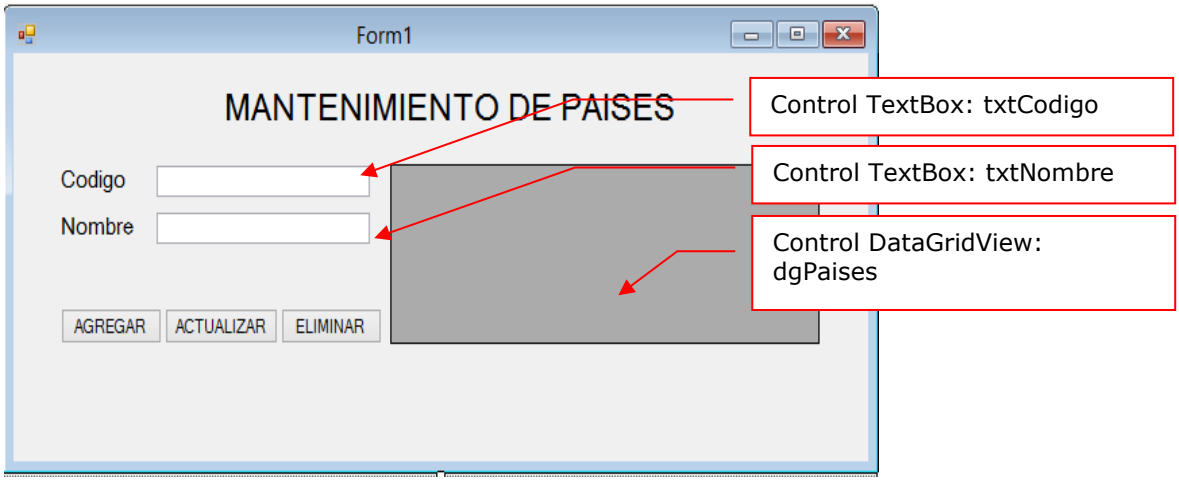
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
    <connectionStrings>
      <add name="cn"
       connectionString="server=.; database=Negocios2015;uid=sa;pwd=sql"/>
    </connectionStrings>
</configuration>
```

A continuación, agregar la referencia: System.Configuration para trabajar con la conexión publicada en el app.Config.



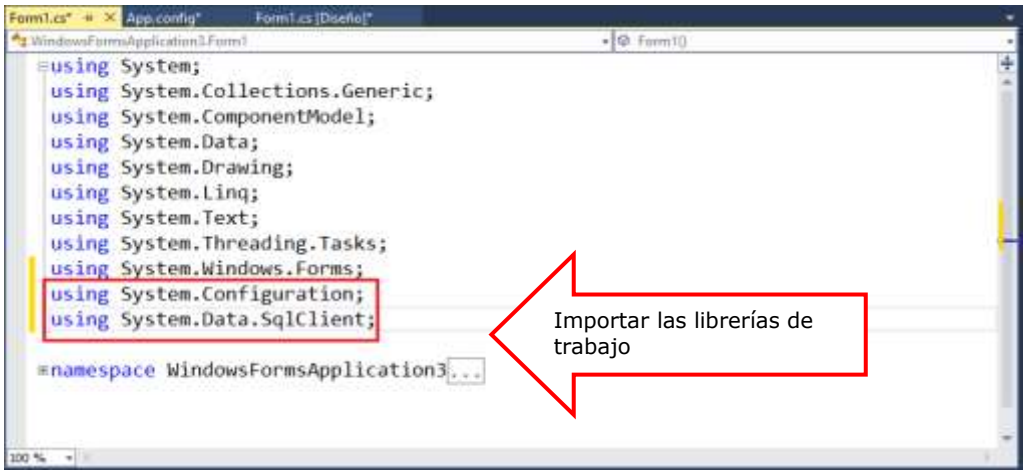
Diseño del Formulario

A continuación diseñe el formulario, tal como se muestra

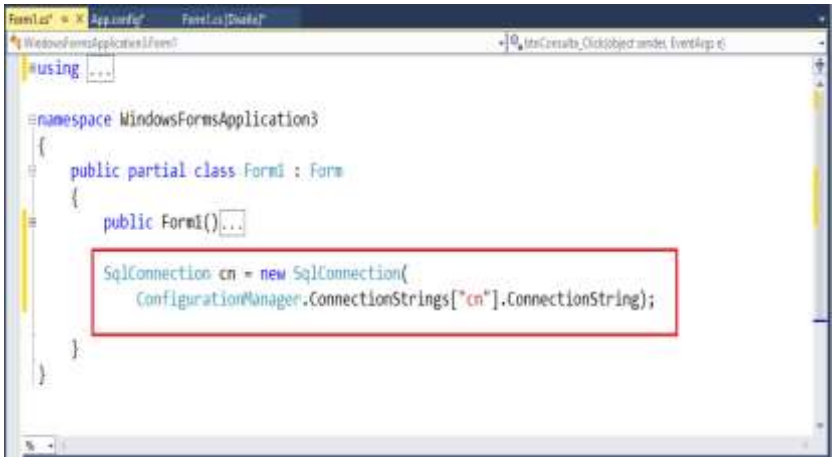


PROGRAMACIÓN

Defina las librerías de trabajo: System.Data.SqlClient y System.Configuration

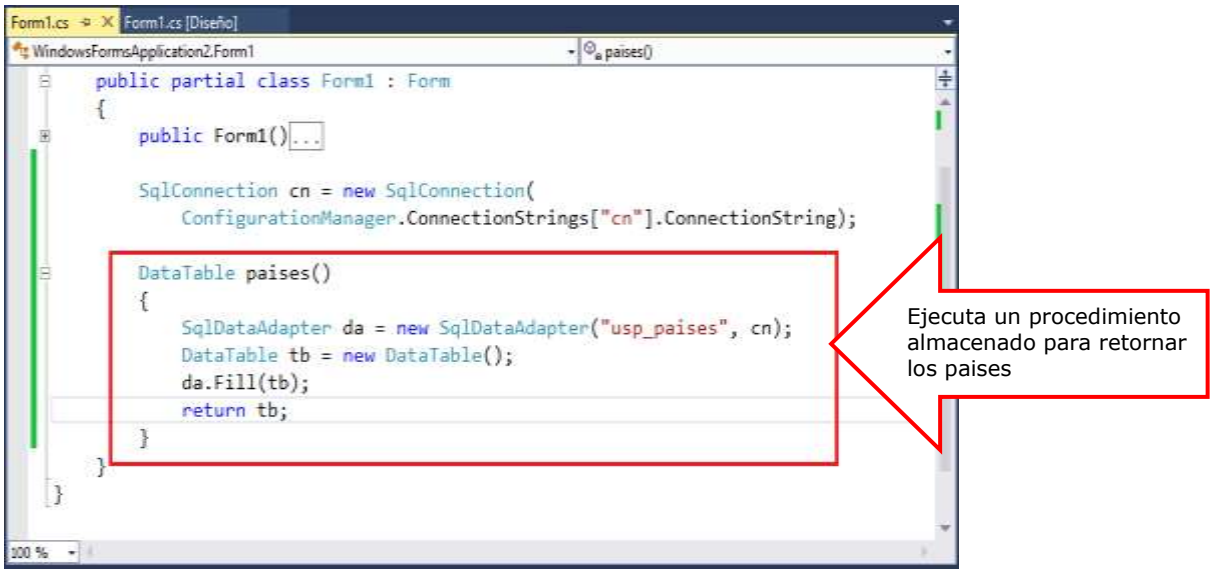


Defina la conexión a la base de datos Negocios2015 instanciando el SQLConnection



```
using ...

namespace WindowsFormsApplication3
{
    public partial class Form1 : Form
    {
        public Form1()...

        SqlConnection cn = new SqlConnection(
            ConfigurationManager.ConnectionStrings["cn"].ConnectionString);
    }
}
```

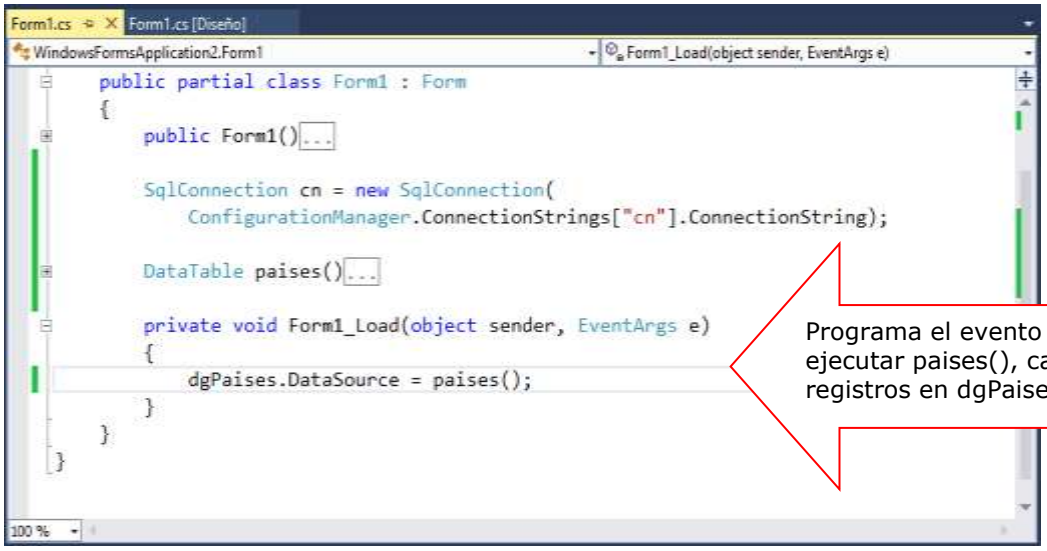
Defina una función de tipo DataTable donde retorna los registros de la tabla tb_paises; en este proceso ejecutamos el procedimiento almacenado usp_pais



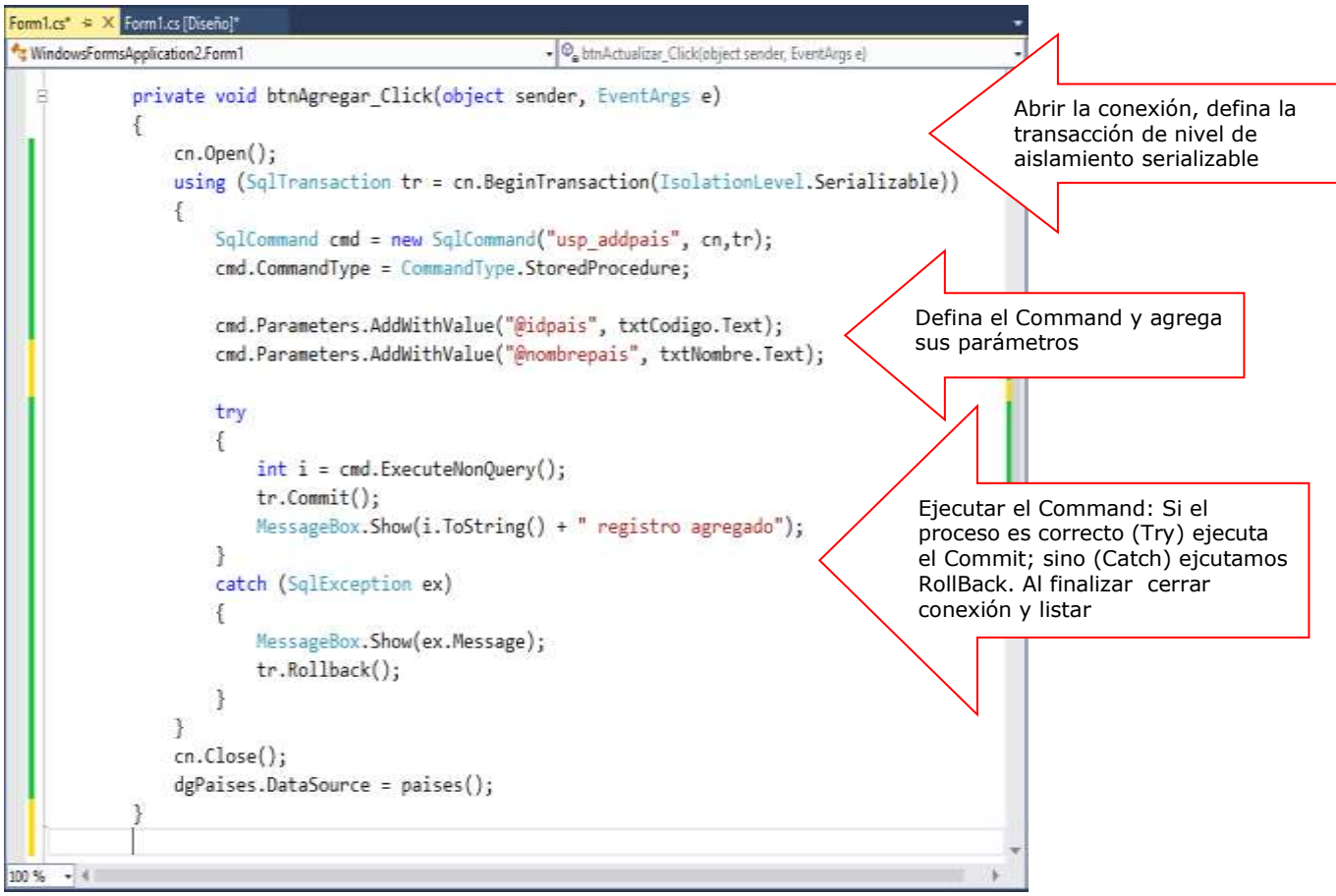
```
Form1.cs | Form1.cs [Diseño]
WindowsFormsApplication2.Form1 | paises()

public partial class Form1 : Form
{
    public Form1()...

    SqlConnection cn = new SqlConnection(
        ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

    DataTable paises()
    {
        SqlDataAdapter da = new SqlDataAdapter("usp_paises", cn);
        DataTable tb = new DataTable();
        da.Fill(tb);
        return tb;
    }
}
```

Ejecuta un procedimiento almacenado para retornar los paises

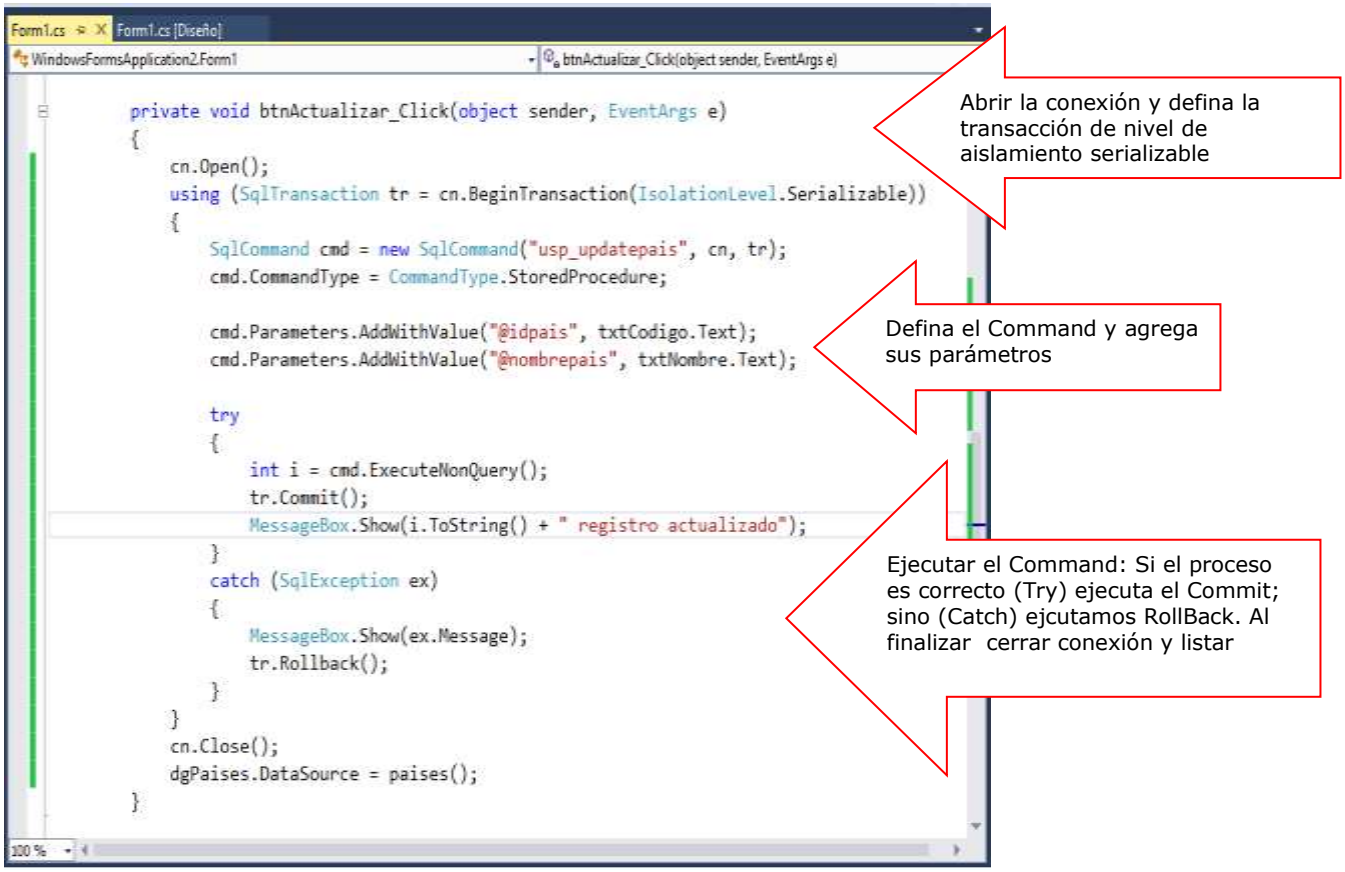
Programa el evento Load del Formulario, donde al cargar el Formulario ejecutamos la función paises(), visualizando los resultado en el control DataGridView: dgPaises



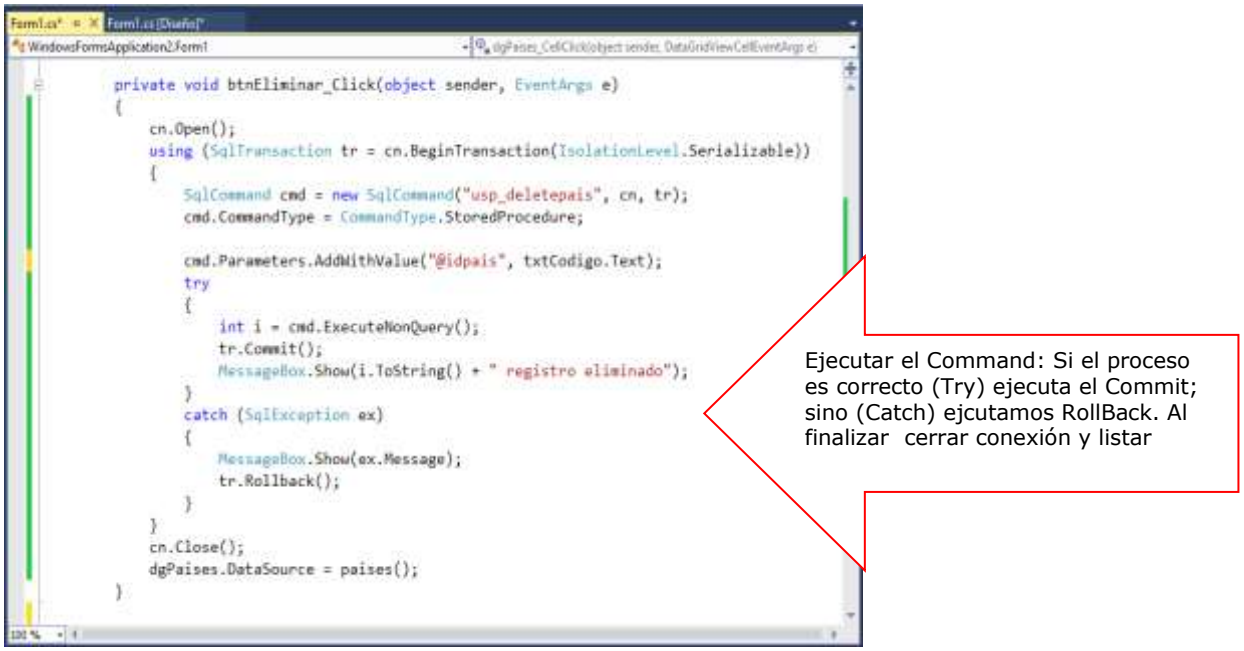
Programa el evento Click del botón Agregar, donde ejecutamos el proceso para agregar un registro a la tabla tb_paises. En este proceso utilizamos el objeto Command para ejecutar un procedimiento almacenado de inserción (usp_addpais)



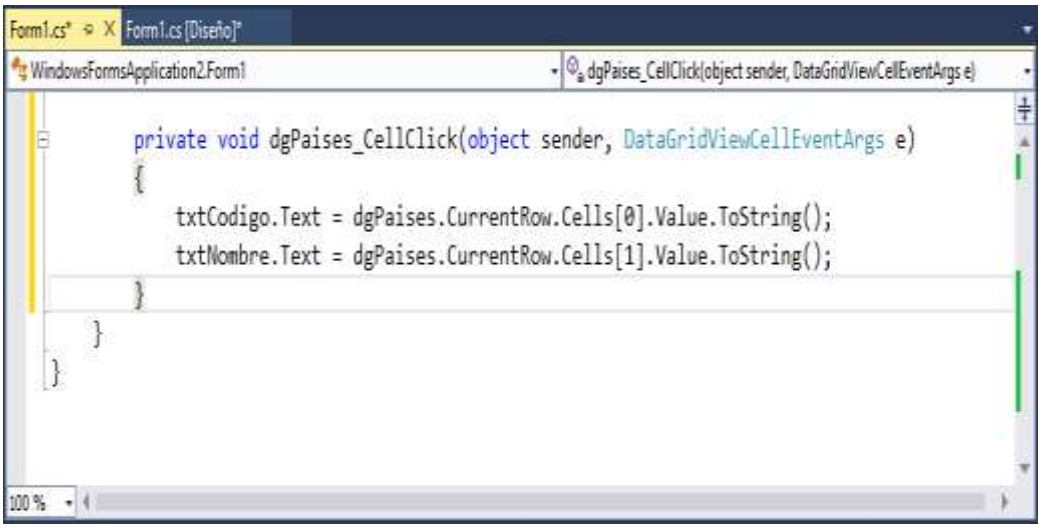
Programa el evento Click del botón Modificar, donde ejecutamos el proceso para modificar un registro a la tabla tb_paises por su campo idpais. En este proceso utilizamos el objeto Command para ejecutar el procedure usp_updatepais



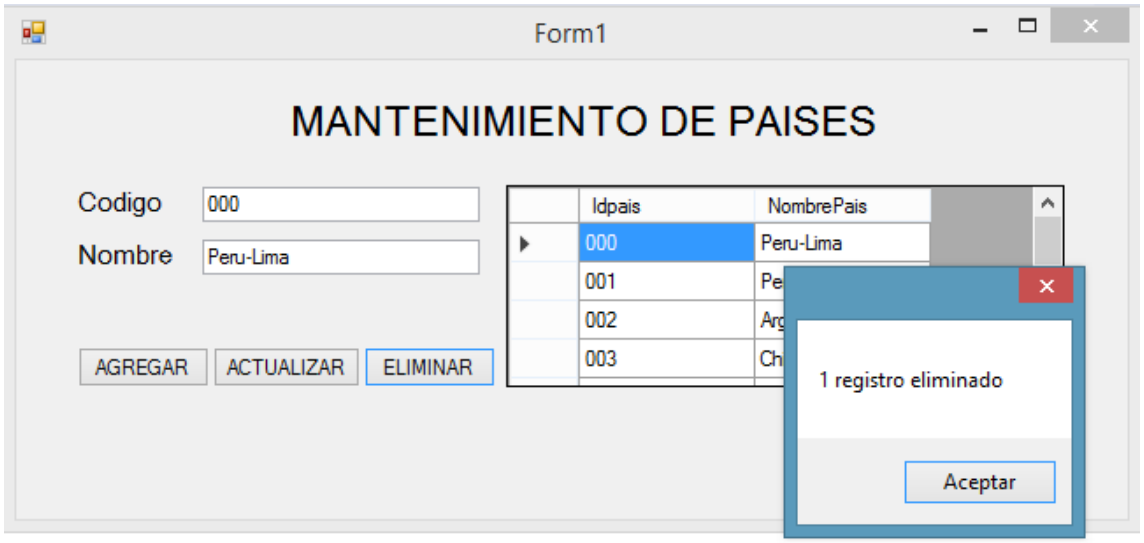
Programa el evento Click del botón Eliminar, donde ejecutamos el proceso para eliminar un registro a la tabla tb_paises por su campo idpais. En este proceso utilizamos el objeto Command para ejecutar el procedure usp_deletepais



Programa el evento CellClick del control dgPais, donde al seleccionar una fila (CurrentRow), visualizamos los datos en los controles



Presiona la tecla F5 para ejecutar los procesos del Formulario



LABORATORIO 4.2

Se desea implementar un Formulario donde realice el mantenimiento a la tabla tb_empresa donde se incluya el logo de la empresa (imágenes o foto), utilice procedimientos almacenados en el mantenimiento.

Defina los procedimientos almacenados del proceso

```
use Comercial2012
go
Create Procedure usp_pais
As
Select * from tb_paises
go

Create procedure usp_empresa
As
Select * from tb_empresa
go
```

Procedure que lista los países

Procedure que lista los registros de la tabla tb_empresa

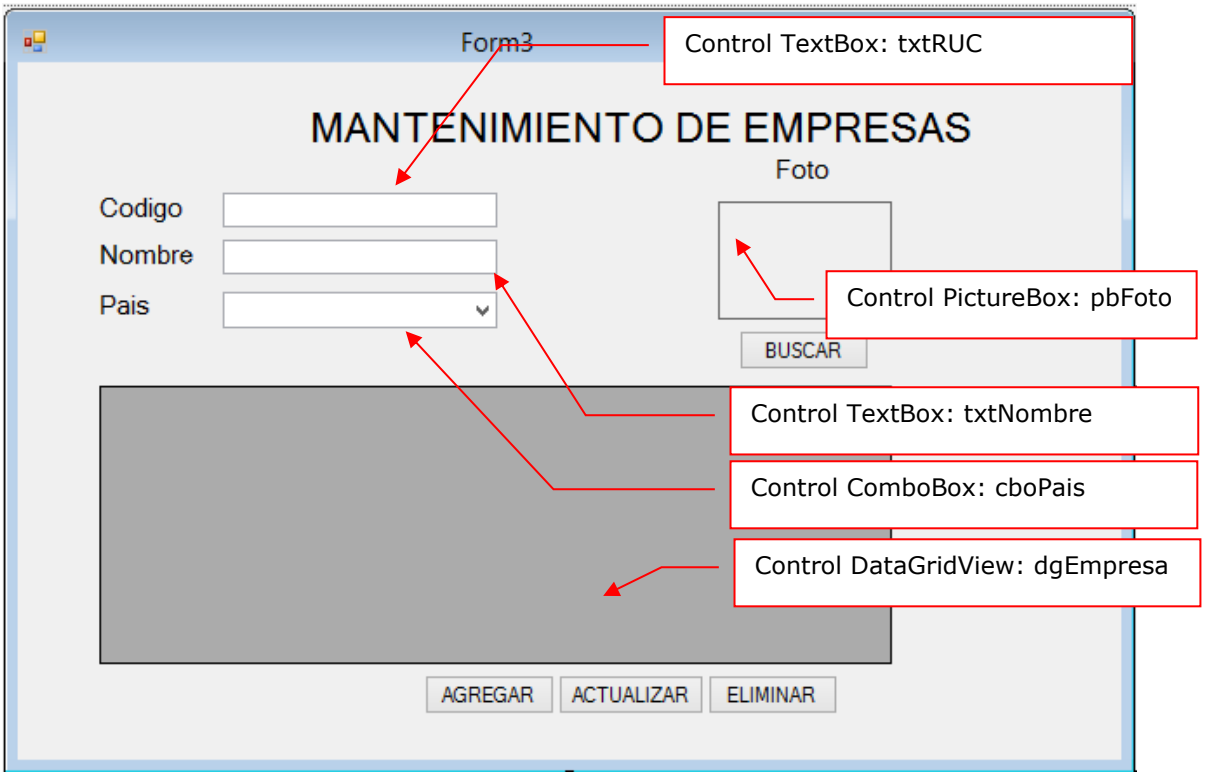
```
--procedure q agregue
Create proc usp_addEmpresa
@ruc varchar(11),
@nom varchar(155),
@pais char(3),
@f image
As
Insert tb_empresa Values(@ruc, @nom, @pais, @f)
go

--procedure q modifique un registro de empresa
Create proc usp_updateEmpresa
@ruc varchar(11),
@nom varchar(155),
@pais char(3),
@f image
As
Update tb_empresa
Set nomemp=@nom, idpais=@pais, fotoemp=@f
Where rucemp=@ruc
go

--procedure q elimine una empresa
Create proc usp_deleteEmpresa
@ruc varchar(11)
As
Delete tb_empresa Where rucemp=@ruc
go
```

Procedimiento almacenado que inserta un registro a la tabla tb_empresa

Procedimiento almacenado que actualiza un registro a la tabla tb_empresa por su campo rucemp

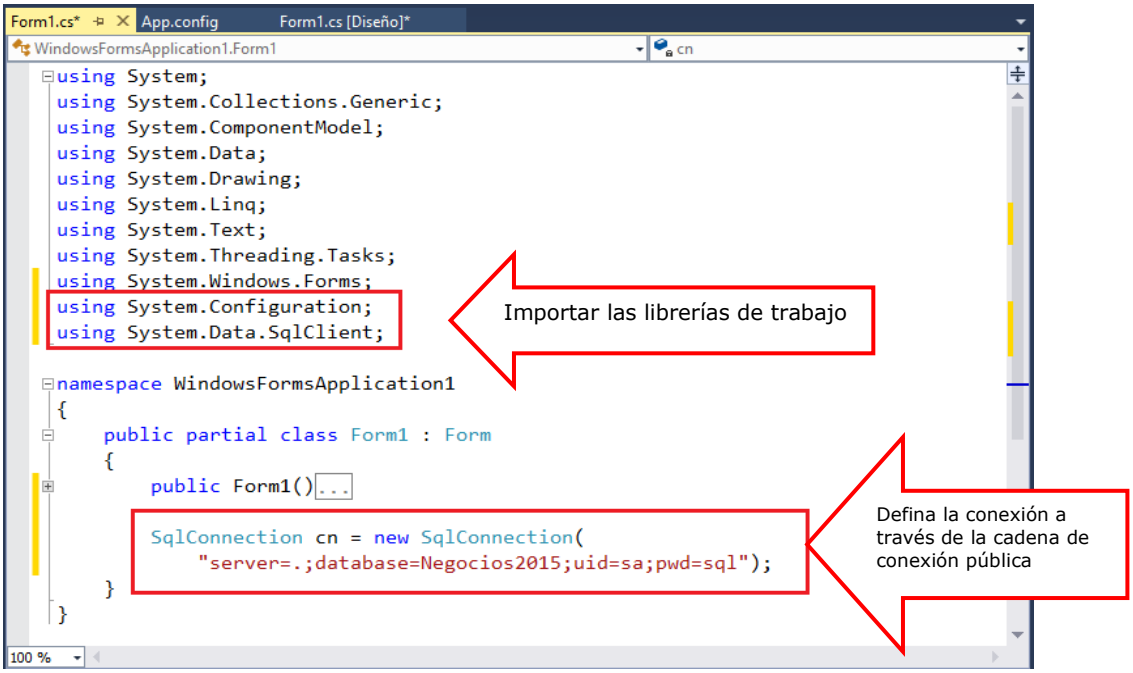
Procedimiento almacenado que elimina un registro a la tabla tb_empresa por su campo rucemp

DISEÑO DEL FORMULARIO.
A continuación diseñe el formulario, tal como se muestra

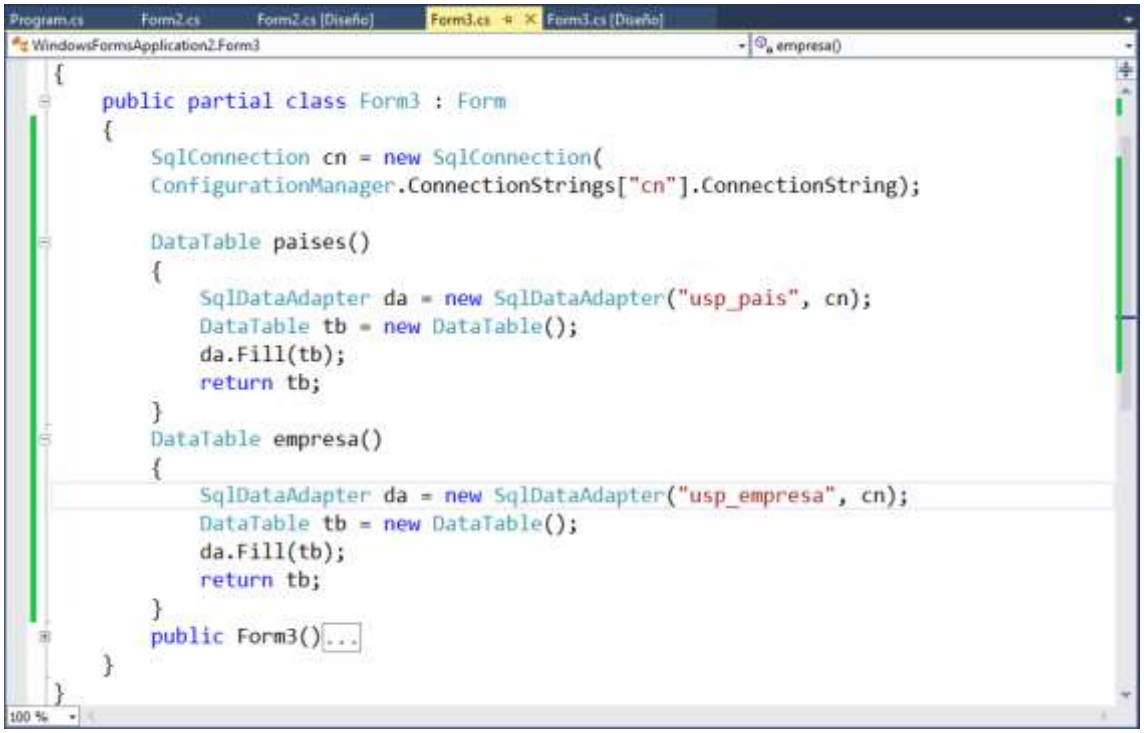


PROGRAMACIÓN

Defina las librerías de trabajo: System.Data.SqlClient (Base de Datos en SQL Server) y el System.Configuration (uso de las etiquetas app.config)



Defina la función DataTable países() donde retorna los registros de tb_paises; y la función empresa() donde retorna los registros de tb_empresa.

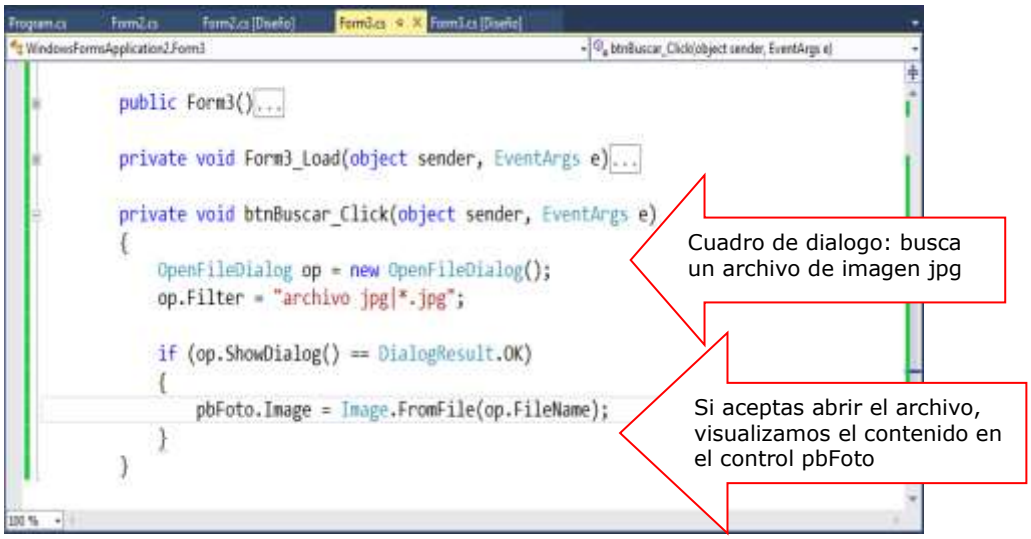


```
{
    public partial class Form3 : Form
    {
        SqlConnection cn = new SqlConnection(
        ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

        DataTable paises()
        {
            SqlDataAdapter da = new SqlDataAdapter("usp_pais", cn);
            DataTable tb = new DataTable();
            da.Fill(tb);
            return tb;
        }
        DataTable empresa()
        {
            SqlDataAdapter da = new SqlDataAdapter("usp_empresa", cn);
            DataTable tb = new DataTable();
            da.Fill(tb);
            return tb;
        }
        public Form3()...
    }
}
```

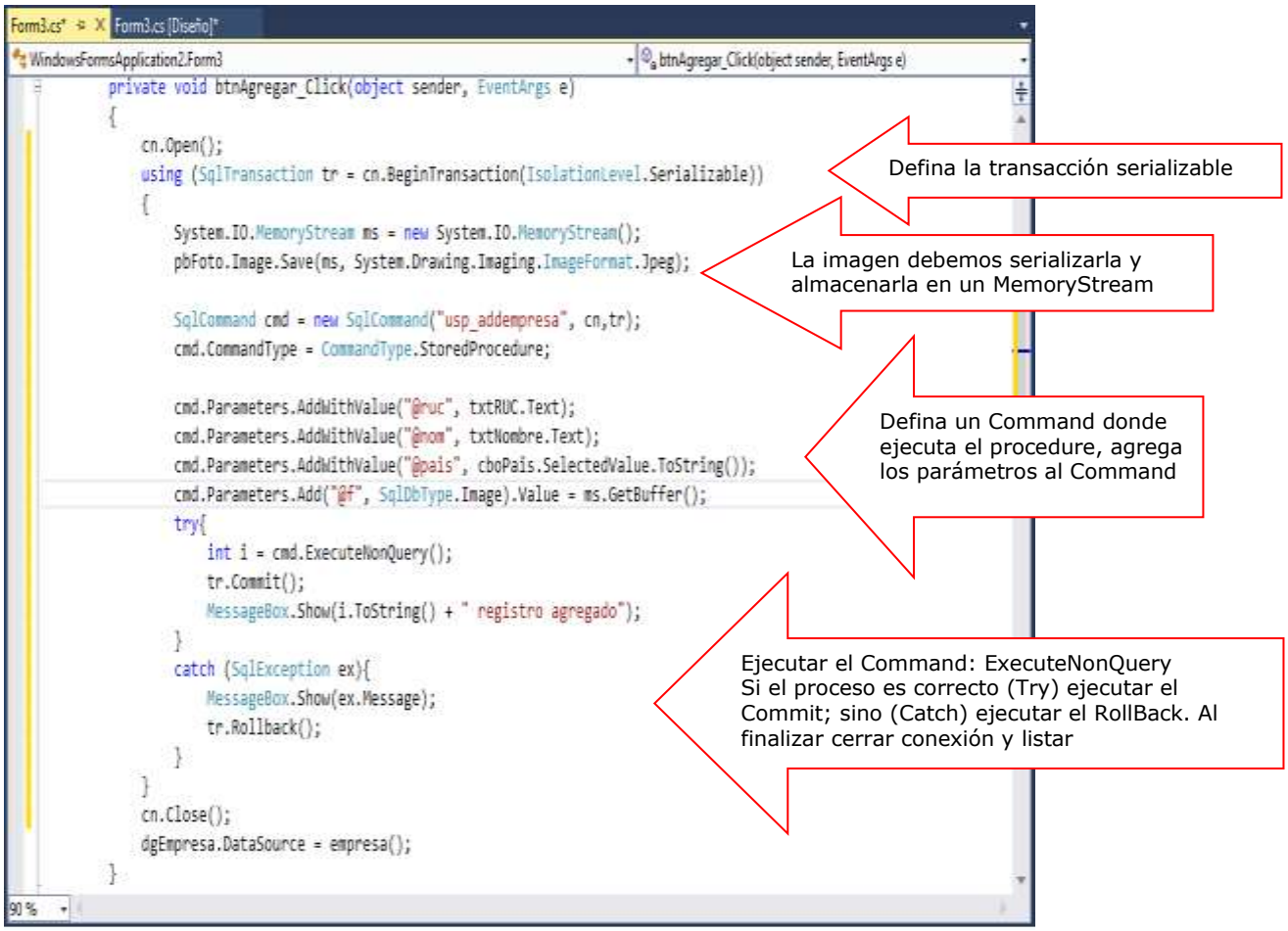
Programa el evento Load del Formulario, donde al cargar el Formulario configura el control ComboBox cboPais con los registros de países, y listamos los registros de empresa en el control DataGridView: dgEmpresa

```
namespace WindowsFormsApplication2
{
    public partial class Form3 : Form
    {
        SqlConnection cn = new SqlConnection(
        ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

        DataTable paises()...

        DataTable empresa()...

        public Form3()...

        private void Form3_Load(object sender, EventArgs e)
        {
            cboPais.DataSource = paises();
            cboPais.DisplayMember = "nombrepais";
            cboPais.ValueMember = "idpais";

            dgEmpresa.DataSource = empresa();
        }
    }
}
```

Programa el evento Click del botón Buscar, donde permita buscar un archivo de imagen (.jpg) visualizando el archivo de imagen en el control pbFoto



Programa el evento Click del botón Agregar, donde ejecutamos el proceso para agregar un registro a la tabla tb_empresa. En este proceso utilizamos el objeto Command para ejecutar un procedimiento almacenado usp_addempresa



Programa el evento Click del botón Actualizar, donde ejecutamos el proceso para modificar un registro a la tabla tb_Empresa por su campo RUC. En este proceso utilizamos el objeto Command para ejecutar el procedure usp_updateEmpresa

```
private void btnActualizar_Click(object sender, EventArgs e)
{
    cn.Open();
    using (SqlTransaction tr = cn.BeginTransaction(IsolationLevel.Serializable))
    {
        System.IO.MemoryStream ms = new System.IO.MemoryStream();
        pbFoto.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);

        SqlCommand cmd = new SqlCommand("usp_updateempresa", cn, tr);
        cmd.CommandType = CommandType.StoredProcedure;

        cmd.Parameters.AddWithValue("@ruc", txtRUC.Text);
        cmd.Parameters.AddWithValue("@nom", txtNombre.Text);
        cmd.Parameters.AddWithValue("@pais", cboPais.SelectedValue.ToString());
        cmd.Parameters.Add("@f", SqlDbType.Image).Value = ms.GetBuffer();
        try{
            int i = cmd.ExecuteNonQuery();
            tr.Commit();
            MessageBox.Show(i.ToString() + " registro actualizado");
        }
        catch (SqlException ex){
            MessageBox.Show(ex.Message);
            tr.Rollback();
        }
    }
    cn.Close();
    dgEmpresa.DataSource = empresa();
}
```

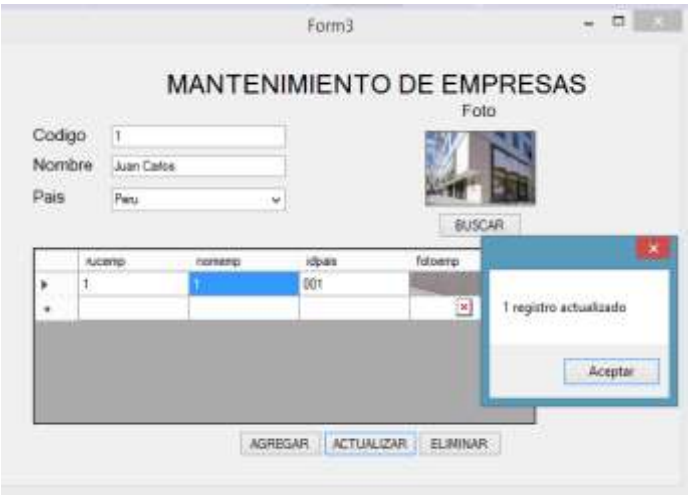
Defina la transacción serializable

La imagen debemos serializarla y almacenarla en un MemoryStream

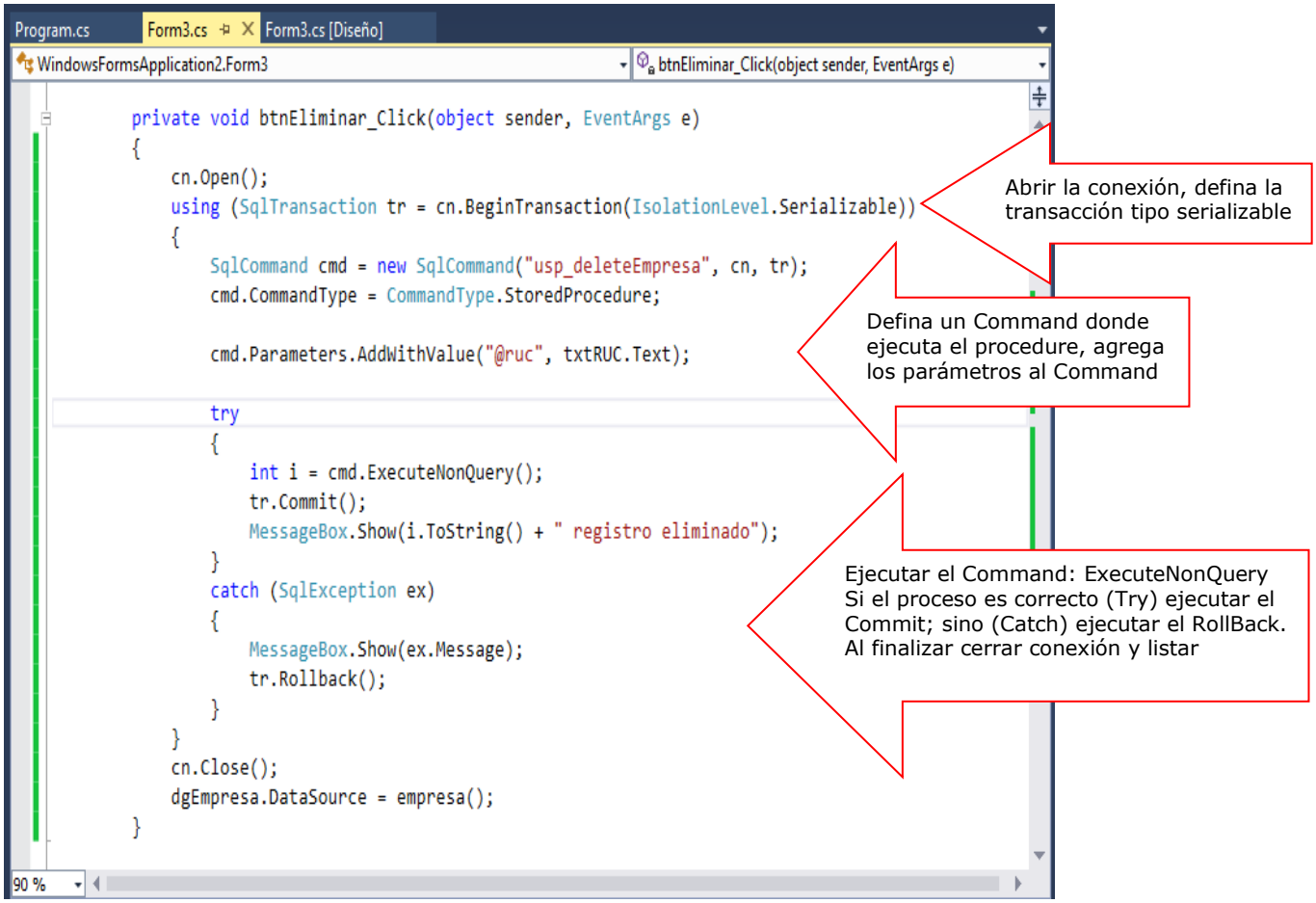
Defina un Command donde ejecuta el procedure, agrega los parámetros al Command

Ejecutar el Command: ExecuteNonQuery
Si el proceso es correcto (Try) ejecutar el Commit; sino (Catch) ejecutar el RollBack. Al finalizar cerrar conexión y listar

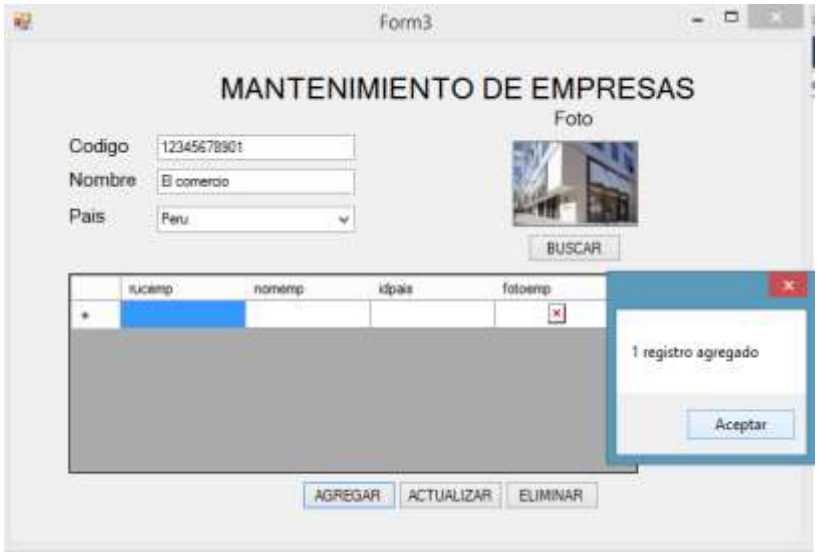
Ejecutamos el Formulario para comprobar los procesos



Programa el evento Click del botón Eliminar, donde ejecutamos el proceso para eliminar un registro a la tabla tb_Empresa por su campo Ruc. En este proceso utilizamos el objeto Command para ejecutar el procedure usp_deleteEmpresa



Presiona la tecla F5 para ejecutar los procesos del Formulario



Resumen

- Una transacción consiste en un comando único o en un grupo de comandos que se ejecutan como un paquete. Las transacciones permiten combinar varias operaciones en una sola unidad de trabajo. Si en un punto de la transacción se produjera un error, todas las actualizaciones podrían revertirse y devolverse al estado que tenían antes de la transacción.
- Una transacción debe ajustarse a las propiedades: atomicidad, coherencia, aislamiento y durabilidad para poder garantizar la coherencia de datos. La mayoría de los sistemas de bases de datos relacionales, como Microsoft SQL Server, admiten transacciones, al proporcionar funciones de bloqueo, registro y administración de transacciones cada vez que una aplicación cliente realiza una operación de actualización, inserción o eliminación
- Una transacción se considera local cuando consta de una única fase y es controlada directamente por la base de datos. Cada proveedor de datos de .NET Framework tiene su propio objeto Transaction para realizar transacciones locales. Si necesita que se realice una transacción en una base de datos de SQL Server, seleccione una transacción de **System**.**Data**.**SqlClient**. En transacciones de Oracle, utilice el proveedor **System**.**Data**.**OracleClient**. Además, existe una nueva clase **DbTransaction** disponible para la escritura de código independiente del proveedor que requiere transacciones.
- En ADO.NET, las transacciones se controlan con el objeto Connection. Puede iniciar una transacción local con el método **BeginTransaction**. Una vez iniciada una transacción, puede inscribir un comando en esa transacción con la propiedad Transaction de un objeto **Command**. Luego, puede confirmar o revertir las modificaciones realizadas en el origen de datos según el resultado correcto o incorrecto de los componentes de la transacción. Las operaciones para confirmar una transacción es **Commit** y la operación para revertir o deshacer una transacción es **RollBack**.
- La clase **TransactionScope** crea un bloque de código transaccional al inscribir implícitamente las conexiones en una transacción distribuida. Debe llamar al método **Complete** al final del bloque **TransactionScope** antes de abandonarlo. Al salir del bloque se invoca el método **Dispose**. Si se ha producido una excepción que ocasiona que el código salga del ámbito, la transacción se considera anulada
- La clase **TransactionScope** crea una transacción con un **IsolationLevel** predeterminado de **Serializable**. Dependiendo de la aplicación, podría estudiar la posibilidad de reducir el nivel de aislamiento para evitar una elevada contención en la aplicación.
- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  - http://msdn.microsoft.com/es-es/library/777e5ebh(v=vs.110).aspx
  - http://msdn.microsoft.com/es-es/library/2k2hy99x(v=vs.110).aspx
  - http://msdn.microsoft.com/es-es/library/ms254973(v=vs.110).aspx

UNIDAD

2

OPERACIONES DESCONECTADAS A UN ORIGEN DE DATOS

Al término de la unidad, el alumno realiza operaciones de consulta y actualización de datos en el entorno de una aplicación Windows desconectado de un origen de datos utilizando la librería ADO.NET

Temario

Tema 5: Manipulación de datos desconectados (4 horas)

1. Manejo de datos desconectados en un DataSet
   1.1. Operaciones de actualización con datos desconectados
   1.2. Actualizacion del origen de datos: ejecutando el CommandBuilder
   1.3. Filtrando los datos desconectados: uso del DataView

ACTIVIDADES PROPUESTAS

- Los alumnos reconocen el modelo desconectado de ADO.NET.
- Los alumnos manejan los objetos desconectados de datos
- Los alumnos realizan operaciones de actualización en un DataSet

5. MANIPULACION DE DATOS DESCONECTADOS

El **DataSet** de ADO.NET es una representación de datos residente en memoria que proporciona un modelo de programación relacional coherente independientemente del origen de datos que contiene. Un **DataSet** representa un conjunto completo de datos, incluyendo las tablas que contienen, ordenan y restringen los datos, así como las relaciones entre las tablas.

En la siguiente ilustración se muestra el modelo de objeto DataSet.

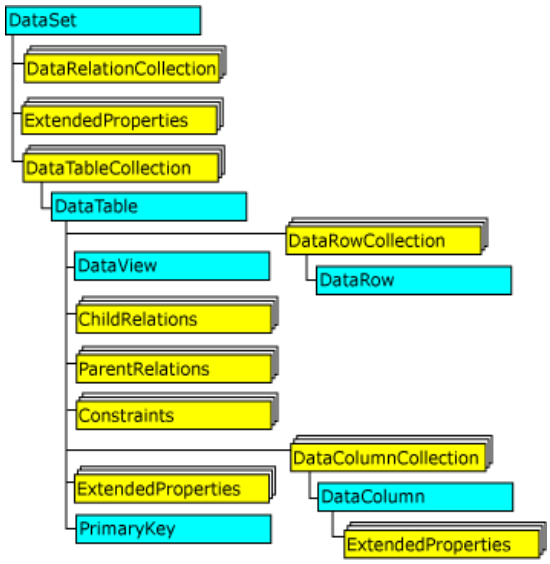


Figura 1: Modelo del objetos DataSet
Referencia: http://msdn.microsoft.com/es-es/library/vstudio/zb0sdh0b(v=vs.100).aspx

Creación de un DataSet

Puede crear una instancia de **DataSet** llamando al constructor **DataSet**. Si lo desea, especifique un nombre de argumento. Si no especifica ningún nombre para el **DataSet**, se establecerá el nombre "**NewDataSet**".

```
DataSet ds = new DataSet("dsFactura");
```

También es posible crear un nuevo **DataSet** basado en un DataSet existente. El nuevo **DataSet** puede ser una copia exacta del **DataSet** existente; un clon del **DataSet** que copia la estructura relacional o el esquema, pero que no contiene ningún dato del **DataSet** existente; o un subconjunto del **DataSet**, que contiene solamente las filas modificadas del **DataSet** existente mediante el método **GetChanges**.

```
DataSet dsclone = ds.Clone();
```

Añadir una Tabla al DataSet

ADO.NET permite crear objetos **DataTable** y agregarlos a un **DataSet** existente. Es posible establecer información de restricciones para una **DataTable** mediante las propiedades **PrimaryKey** y **Unique**.

Por ultimo, el código establece una columna como columna de clave principal.

```
DataSet ds = new DataSet("dsFactura");
DataTable dt = new DataTable();
ds.Tables.Add(dt);
```

Añadir columnas al DataTable

El esquema de un **DataTable** esta representado por los objetos **DataColumns** y **ForeignKeyConstraints** y **UniqueConstraints**. Para crear un objeto **DataColumn** se instancia el objeto y ejecutando el metodo **Add**. El metodo Add acepta:

- ColumnName, nombre de la columna,
- DataType, tipo de dato de la columna,
- Expression, argumento que crea una columna

El ejemplo siguiente, creamos una tabla y agregamos 4 columnas

```
DataTable tb= new DataTable("Recibo");
tb.Columns.Add("numero", Type.GetType("System.String"));
tb.Columns.Add("fecha", Type.GetType("System.DateTime"));
tb.Columns.Add("cliente", Type.GetType("System.String"));
tb.Columns.Add("monto", Type.GetType("System.Decimal"));
```

Creación de expresiones

Puede definirse una expresión para una columna, habilitándolo para contener un valor calculado desde otra columna en la misma fila o desde valores de varias columnas de las filas en la tabla. El **DataType** desde ser el apropido para el valor de la expresión.

En el siguiente ejemplo, agregar la columna renta que representa el 10% del monto del recibo

```
tb.Columns.Add("renta",Type.GetType("System.Decimal"),
"monto*0.10");
```

Creación de clave primaria en un DataTable

Una tabla, por lo general, tienen una columna o grupo de columnas que identifica en formaunica a cada fila dentro de la tabla. La identificación es llamada **Primay Key**.

La propiedad **PrimaryKey** recibe los valores como un array de objetos **DataColumns**. El siguiente ejemplo permite definir un primary key de una columna.

```
tb.PrimaryKey = new DataColumn[] { tb.Columns[0] };
```

El siguiente ejemplo permite definer un Primary Key formado por dos objetos DataColumns

```
tb.PrimaryKey = new DataColumn[] {
      tb.Columns[0], tb.Columns[1] };
```

Añadir Relaciones

En un DataSet que contiene varios objetos **DataTable**, es posible utilizar objetos DataRelation para relacionar una tabla con otra, navegar por las tablas y devolver filas secundarias o primarias de una tabla relacionada.

Los argumentos necesarios para crear una **DataRelation** son un nombre para la **DataRelation** que se va a crear y una matriz de una o más referencias **DataColumn** a las columnas que actúan como columnas primaria y secundaria en la relación. Una vez creado un objeto **DataRelation**, es posible utilizarlo para navegar por las tablas y recuperar valores.

Al agregar una **DataRelation** a una **DataSet**, se agrega de forma predeterminada una **UniqueConstraint** a la tabla primaria y una **ForeignKeyConstraint** a la tabla secundaria.

```
DataRelation rel1= new DataRelation( "r1",
         ds.Tables["cliente"].Columns["idcli"],
         ds.Tables["factura"].Columns["idcli"]);

ds.Relations.Add(rel1);
```

5.1 OPERACIONES DE ACTUALIZACION CON DATOS DESCONECTADOS

La Clase DataAdapter permite devolver únicamente una página de datos mediante sobrecargas del método **Fill**. Sin embargo, quizás no sea la mejor opción para paginar a través de resultados de consultas grandes ya que, aunque el **DataAdapter** rellena la **DataTable** o el **DataSet** de destino solo con los registros solicitados, se siguen utilizando los recursos para devolver toda la consulta.

Cada proveedor de datos de .NET FrameWork cuenta con un objeto DataAdapter:

Proveedor	Descripcion
OledbDataAdapter	Proveedor de datos para OLEDB
SqlDataAdapter	Proveedor de datos para SQL Server
OdbcDataAdapter	Proveedor de datos para ODBC
OracleDataAdapter	Proveedor de datos para Oracle

Configuración de un DataAdapter

El DataAdapter tiene 4 propiedades que son utilizadas para recibir y actualizar los datos al origen de datos

Propiedades	Descripción
insertCommand	Comando para insertar filas en un almacen de datos

UpdateCommand	Comando para modificar los datos de las filas en el almacen de datos
DeleteCommand	Comando para eliminar filas en el almacen de datos
SelectCommand	Comando para recuperar filas en el almacen de datos

El método **Update** de **DataAdapter** se llama para reflejar en el origen de datos todos los cambios efectuados en DataSet. El método **Update**, al igual que el método **Fill**, acepta como argumentos una instancia de **DataSet** y, de forma opcional, un objeto **DataTable** o un nombre de **DataTable**. La instancia de **DataSet** es el **DataSet** que contiene los cambios efectuados, y **DataTable** identifica la tabla desde la que se pueden recuperar esos cambios. Si no se especifica **DataTable**, se utiliza el primer **DataTable** de **DataSet**.

Al llamar al método **Update**, **DataAdapter** analiza los cambios efectuados y ejecuta el comando apropiado (INSERT, UPDATE o DELETE). Cuando **DataAdapter** encuentra un cambio en DataRow, utiliza los comandos **InsertCommand**, **UpdateCommand** o **DeleteCommand** para reflejarlo. De esta forma, se obtiene el máximo rendimiento de la aplicación de ADO.NET al especificar la sintaxis del comando en la fase de diseño y utilizar, siempre que es posible, procedimientos almacenados.

Antes de llamar a **Update** deben establecerse de forma explícita los comandos. Si se llama a **Update** y el comando correspondiente a una actualización determinada no existe (por ejemplo, no hay un comando **DeleteCommand** para las filas eliminadas), se inicia una excepción.

Llamar a **AcceptChanges** en **DataSet**, **DataTable** o **DataRow** hará que todos los valores **Original** de **DataRow** se sobrescriban con los valores **Current** de **DataRow**. Si se han modificado los valores de campo que identifican de forma única a una fila, los valores **Original** dejarán de coincidir con los valores del origen de datos después de llamar a **AcceptChanges**.

Se llama automáticamente a **AcceptChanges** para cada fila durante una llamada al método **Update** de **DataAdapter**. Puede conservar los valores originales durante una llamada al método **Update** estableciendo primero la propiedad **AcceptChangesDuringUpdate** de **DataAdapter** en false o creando un controlador de eventos para el evento **RowUpdated** y estableciendo **Status** en **SkipCurrentRow**

5.2 ACTUALIZACION DE UN ORIGEN DE DATOS: EJECUTANDO UN COMMANDBUILDER

SqlDataAdapter no genera automáticamente las instrucciones de Transact-SQL requeridas para realizar los cambios realizados en un DataSet con la instancia asociada de SQL Server. El objeto **SqlCommandBuilder** ejecuta automáticamente las instrucciones de Transact-SQL para las actualizaciones de una sola tabla si se establece el valor de la propiedad SelectCommand de SqlDataAdapter.

A continuación, cualquier instrucción de Transact-SQL adicional que no se establezca se genera mediante **SqlCommandBuilder**.

El objeto **SqlCommandBuilder** se registrará a sí mismo como agente de escucha de eventos de RowUpdating siempre que se establezca la propiedad DataAdapter. Sólo se pueden asociar un objeto SqlDataAdapter y un objeto **SqlCommandBuilder** simultáneamente.

Para generar instrucciones INSERT, UPDATE o DELETE, **SqlCommandBuilder** utiliza la propiedad SelectCommand con el fin de recuperar automáticamente un conjunto de metadatos requerido. Si se cambia SelectCommand una vez que se han recuperado los metadatos, como tras la primera actualización, se debe llamar al método RefreshSchema para actualizar los metadatos. SelectCommand también debe devolver como mínimo una clave principal o una columna única. Si no hay ninguna, se genera una excepción InvalidOperation, y no se genera ningún comando.

SqlCommandBuilder también utiliza las propiedades Connection, CommandTimeout y Transaction a las que hace referencia SelectCommand. El usuario debe llamar a RefreshSchema cuando se modifica una o varias de estas propiedades, o se reemplaza SelectCommand. En caso contrario, las propiedades InsertCommand, UpdateCommand y DeleteCommand conservan sus valores anteriores. Si se llama a Dispose, se elimina la asociación de SqlCommandBuilder con SqlDataAdapter y los comandos generados se dejan de utilizar.

5.3 FILTRANDO LOS DATOS DESCONECTADOS: USO DEL DATAVIEW

La clase DataView representa una vista personalizada que puede enlazar datos de un **DataTable** para realizar operaciones de ordenamiento, filtro, búsqueda, edición y navegación. El **DataView** no almacena datos, sino que representa una vista conectada al **DataTable** correspondiente. Los cambios en los datos de **DataView** afectarán a **DataTable**. Los cambios en los datos de **DataTable** afectarán a toda los **DataView** asociados a él.

Creando un DataView

Hay dos formas de crear una **DataView**. Puede utilizar el constructor **DataView** o puede crear una referencia a la propiedad **DefaultView** de la **DataTable**. El constructor **DataView** puede estar vacío o puede aceptar también **DataTable** como único argumento o **DataTable** junto con el criterio de filtro o de ordenación, y un filtro de estado de fila.

Como el índice de una **DataView** se crea al mismo tiempo que **DataView** y cuando se modifica alguna de las propiedades **Sort**, **RowFilter** o **RowStateFilter**, conseguirá un rendimiento óptimo si suministra cualquier criterio inicial de ordenación o filtro como argumentos del constructor al crear la **DataView**. Al crear una **DataView** sin especificar criterios de ordenación o de filtro y establecer posteriormente las propiedades Sort, **RowFilter** o **RowStateFilter** hace que el índice se construya dos veces como mínimo: una vez al crear la **DataView** y la otra cuando se modifica cualquiera de las propiedades de ordenación y filtrado.

Ordenando y Filtrado de Datos

La **DataView** proporciona varias formas de ordenación y filtrado de datos en una **DataTable**:

- Mediante la propiedad **Sort** puede especificar criterios simples o múltiples de ordenación de columnas e incluir parámetros ASC (ascendente) y DESC (descendente).

- Mediante la propiedad **ApplyDefaultSort** puede crear automáticamente un criterio de ordenación, en orden ascendente, basado en la columna o columnas de clave principal de la tabla. **ApplyDefaultSort** solo se aplica cuando la propiedad **Sort** es una referencia nula o una cadena vacía y cuando la tabla tiene definida una clave principal.

- Mediante la propiedad **RowFilter** puede especificar subconjuntos de filas basándose en sus valores de columna.

- Si desea devolver los resultados de una consulta determinada en los datos, en lugar de proporcionar una vista dinámica de un subconjunto de los datos, para conseguir el máximo rendimiento utilice los métodos Find o FindRows de la DataView en lugar de establecer la propiedad RowFilter. El establecimiento de la propiedad RowFilter hace que se recompile el índice de los datos, lo que agrega sobrecarga a la aplicación y reduce el rendimiento. La propiedad RowFilter es más idónea en una aplicación enlazada a datos donde un control enlazado muestra resultados filtrados. Los métodos Find y FindRows aprovechan el índice actual, sin necesidad de volver a crearlo.
- Mediante la propiedad **RowStateFilter** puede especificar las versiones de fila que desea ver. La **DataView** administra implícitamente qué versión de fila exponer, dependiendo del **RowState** de la fila subyacente.

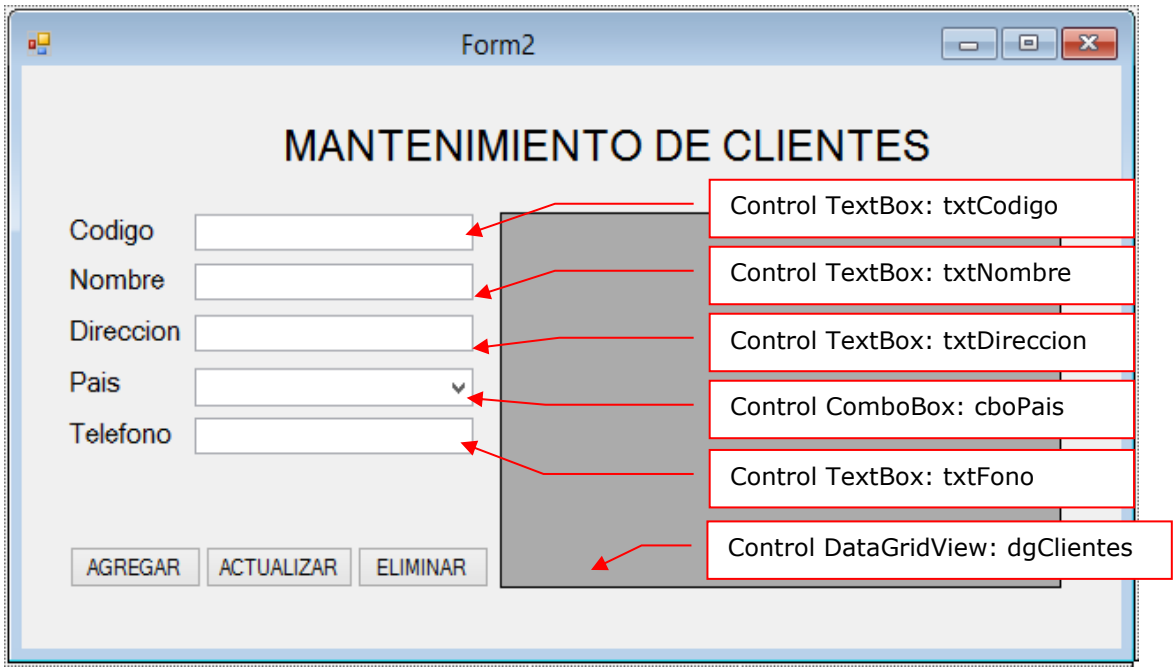
Por ejemplo, si el RowStateFilter está establecido como DataViewRowState.Deleted, la DataView expone la versión de fila Original de todas las filas Deleted porque no hay ninguna versión de fila Current.
En la siguiente tabla se muestran las opciones de DataViewRowState.

Opciones de DataViewRowState	Descripción
CurrentRows	La versión de fila Current de todas las filas Unchanged, Added y Modified. Éste es el valor predeterminado.
Added	La versión de fila Current de todas las filas Added.
Deleted	La versión de fila Original de todas las filas Deleted.
ModifiedCurrent	La versión de fila Current de todas las filas Modified.
ModifiedOriginal	La versión de fila Original de todas las filas Modified.
Ninguna	Ninguna fila.
OriginalRows	La versión de fila Original de todas las filas Unchanged, Modified y Deleted.
Unchanged	La versión de fila Current de todas las filas Unchanged.

LABORATORIO 5.1

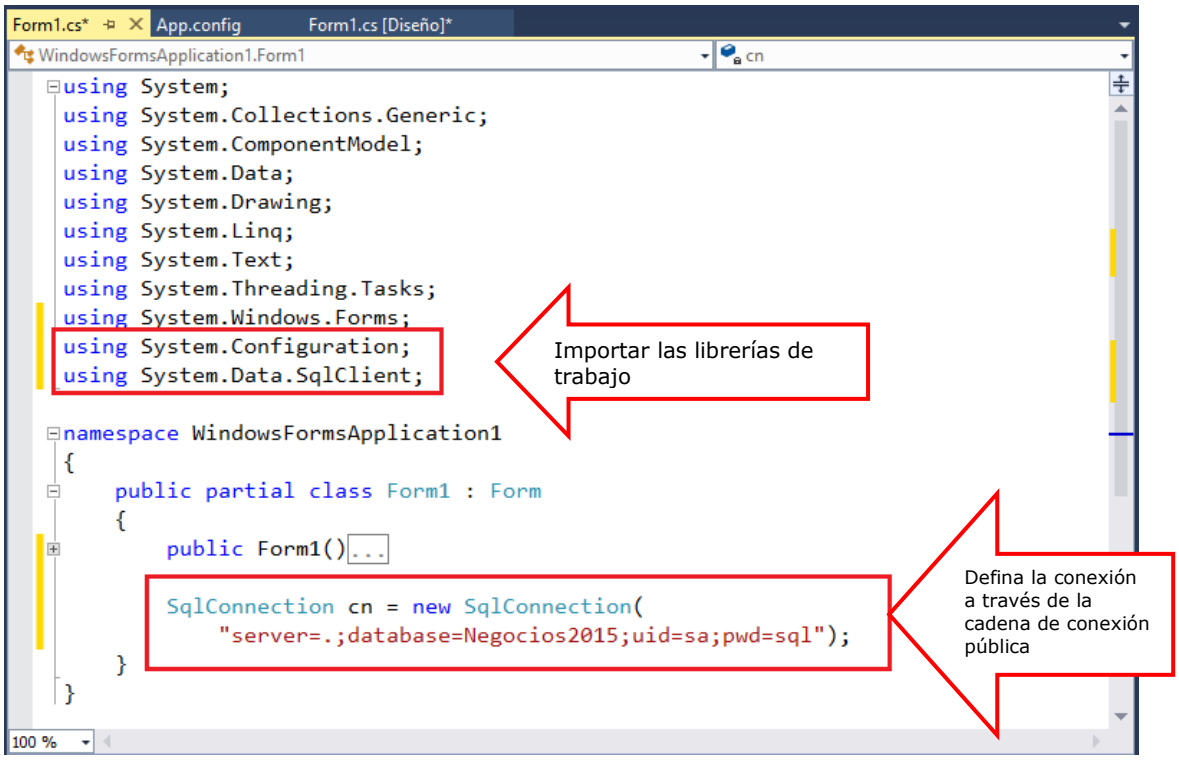
Implementa un Formulario donde permita actualizar los registros de la tabla tb_clientes en forma desconectada

DISEÑO DEL FORMULARIO

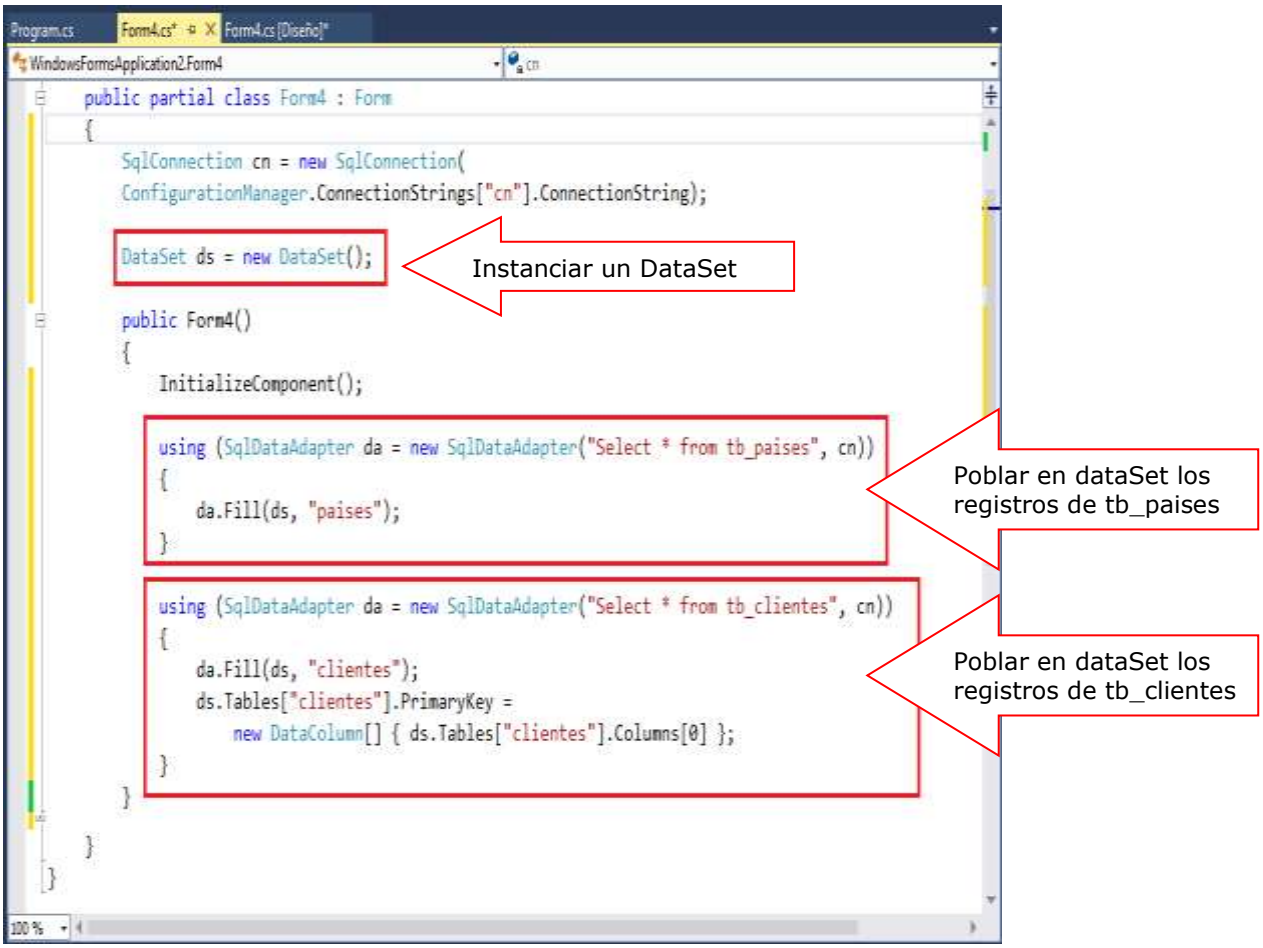


PROGRAMACIÓN

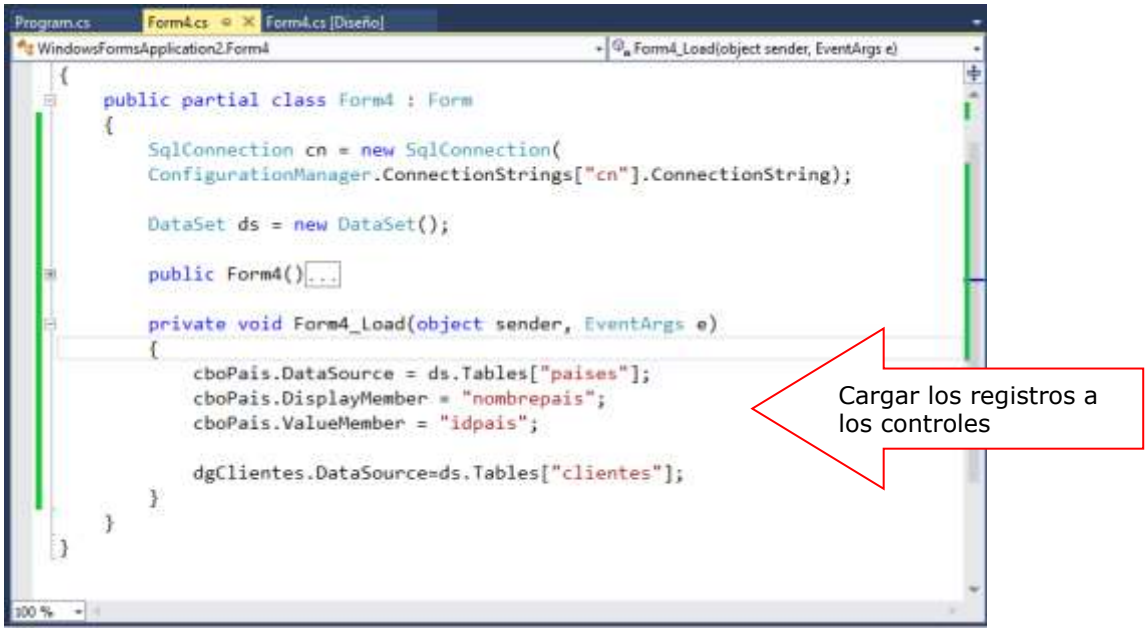
Defina las librerías de trabajo: System.Data.SqlClient y System.Configuration. Defina la conexión a la base de datos Negocios2015 instanciando el SQLConnection



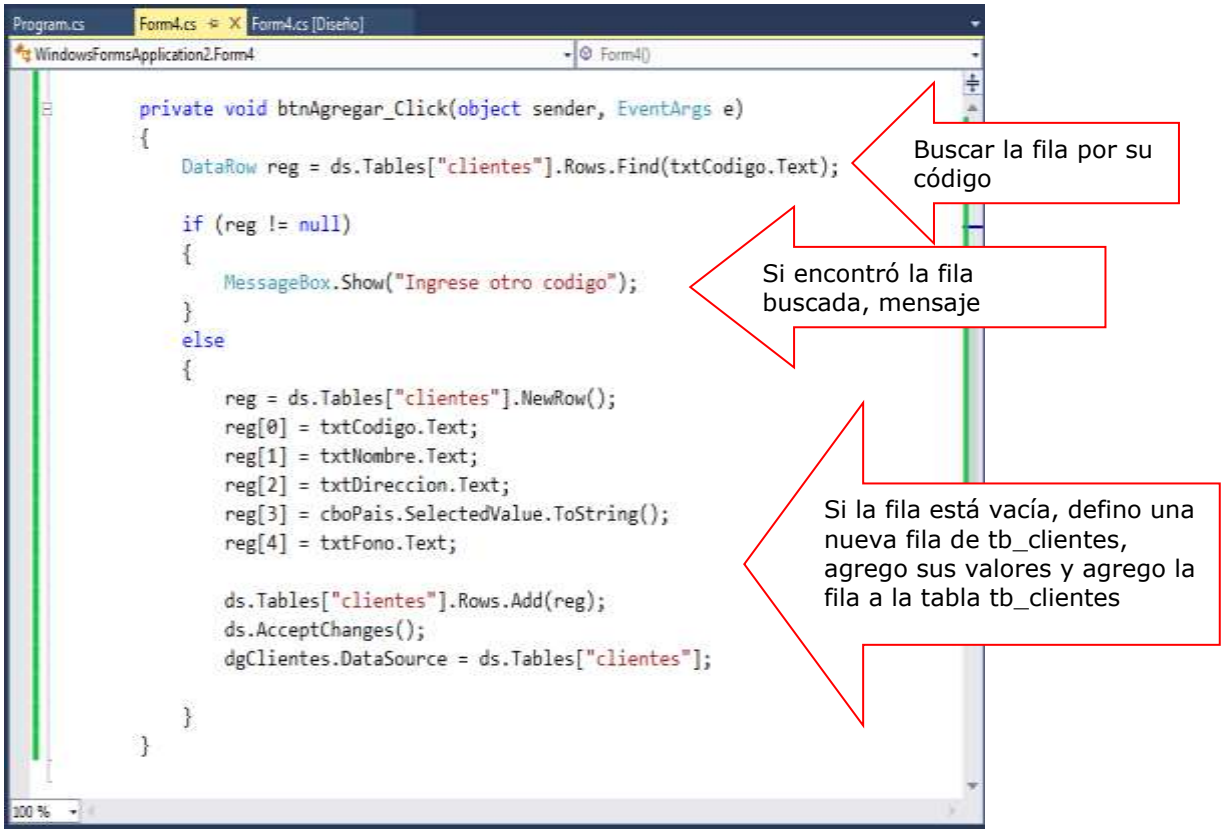
Defina un DataSet, en el constructor poblamos los registros de tb_paises y tb_clientes en el Dataset y lo almacenamos con el nombre de cada tabla.



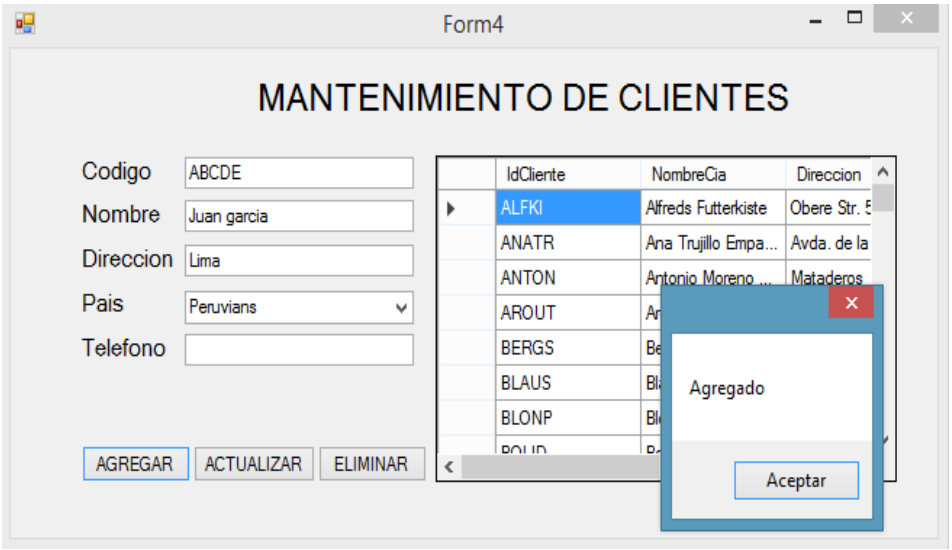
En el evento Load del Formulario, cargamos los datos a los controles



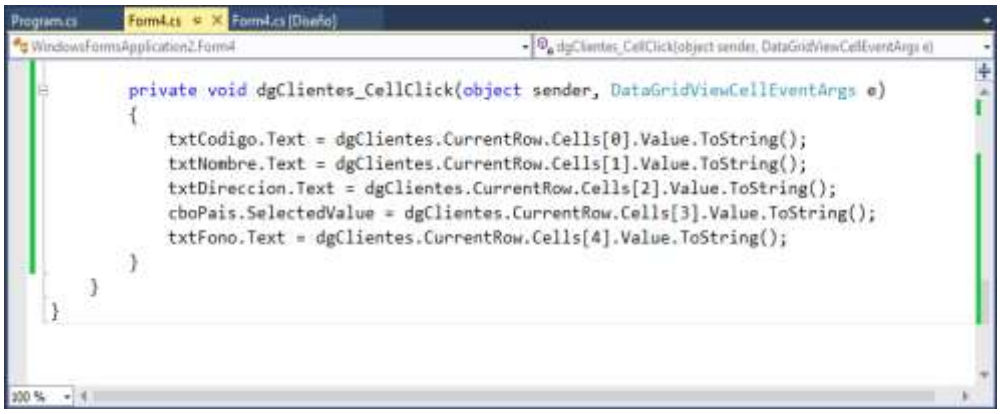
Programa el evento Click del botón Agregar, realizamos la búsqueda del código de clientes y el resultado de la búsqueda es una fila o DataRow que se almacena el reg. Si reg esta vació agregamos una nueva fila, sino mostraremos un mensaje indicando que el código ya existe.



Para verificar, presione F5, ingrese los datos del cliente; al presionar el botón Agregar se inserta el registro en la tabla

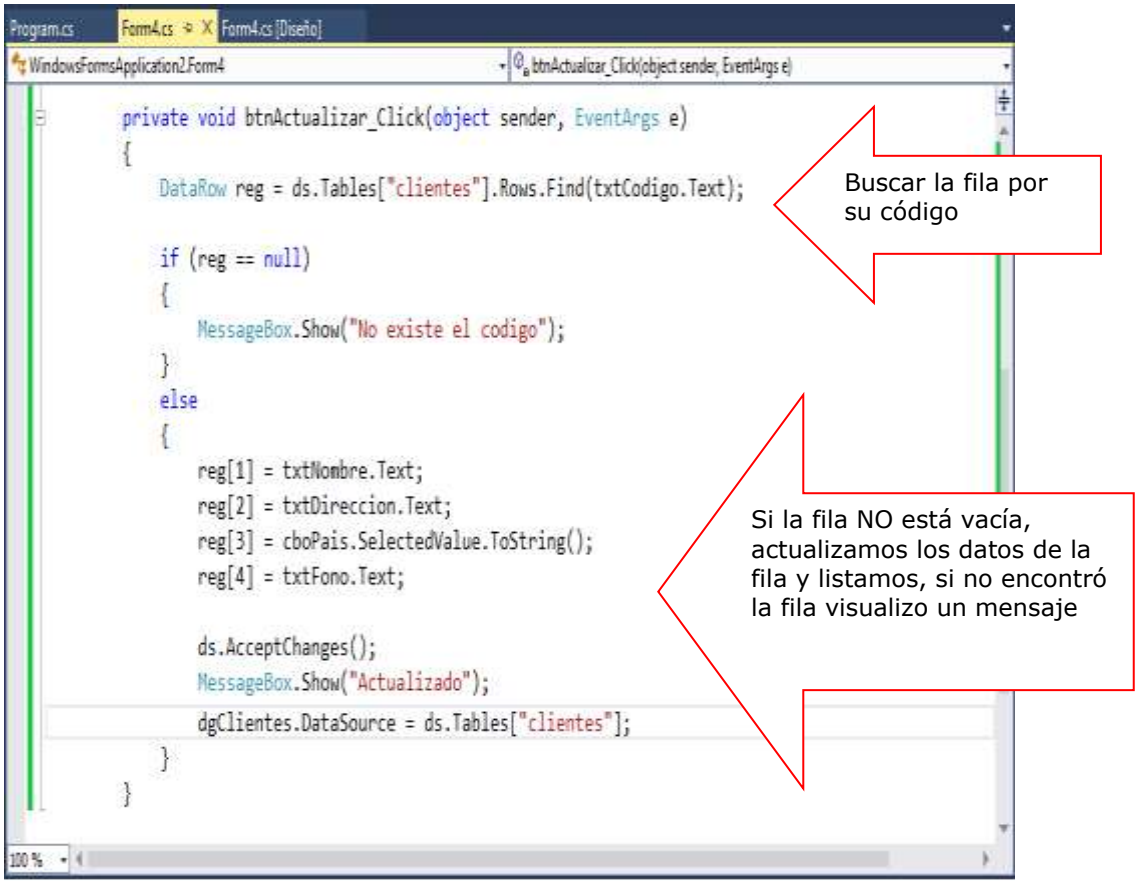


Programación del evento CellClick del control dgClientes, donde al seleccionar una fila, visualizamos los datos del clientes en los controles.



```
private void dgClientes_CellClick(object sender, DataGridViewCellEventArgs e)
{
    txtCodigo.Text = dgClientes.CurrentRow.Cells[0].Value.ToString();
    txtNombre.Text = dgClientes.CurrentRow.Cells[1].Value.ToString();
    txtDireccion.Text = dgClientes.CurrentRow.Cells[2].Value.ToString();
    cboPais.SelectedValue = dgClientes.CurrentRow.Cells[3].Value.ToString();
    txtFono.Text = dgClientes.CurrentRow.Cells[4].Value.ToString();
}
```

Programación del botón Actualizar, en este proceso modificamos un registro de tb_clientes, primero realizamos la búsqueda del código de cliente; y el resultado de la búsqueda es un DataRow que se almacena en re. Si reg esta vacío mostramos un mensaje de error, caso contrario actualizamos los datos del cliente.

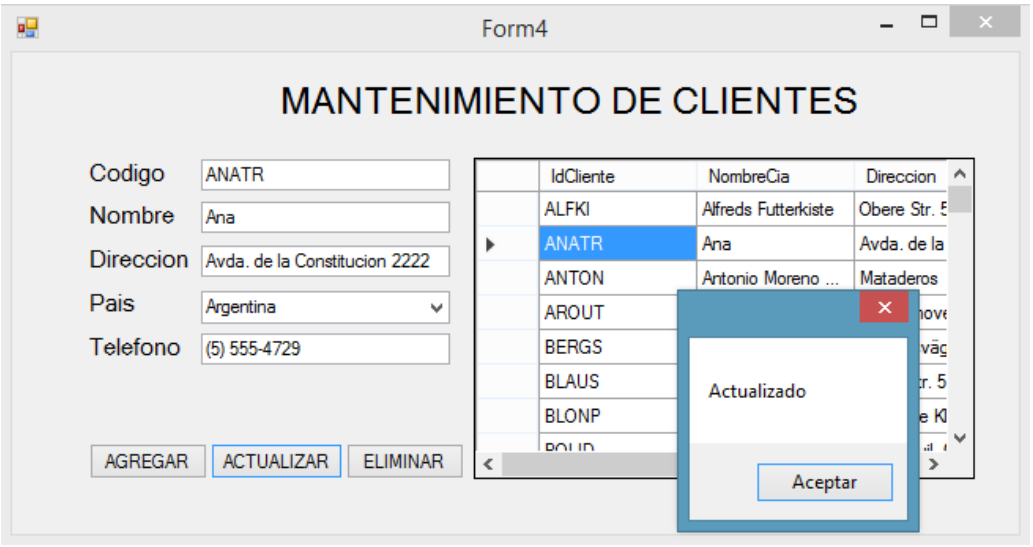


```
private void btnActualizar_Click(object sender, EventArgs e)
{
    DataRow reg = ds.Tables["clientes"].Rows.Find(txtCodigo.Text);

    if (reg == null)
    {
        MessageBox.Show("No existe el codigo");
    }
    else
    {
        reg[1] = txtNombre.Text;
        reg[2] = txtDireccion.Text;
        reg[3] = cboPais.SelectedValue.ToString();
        reg[4] = txtFono.Text;

        ds.AcceptChanges();
        MessageBox.Show("Actualizado");
        dgClientes.DataSource = ds.Tables["clientes"];
    }
}
```

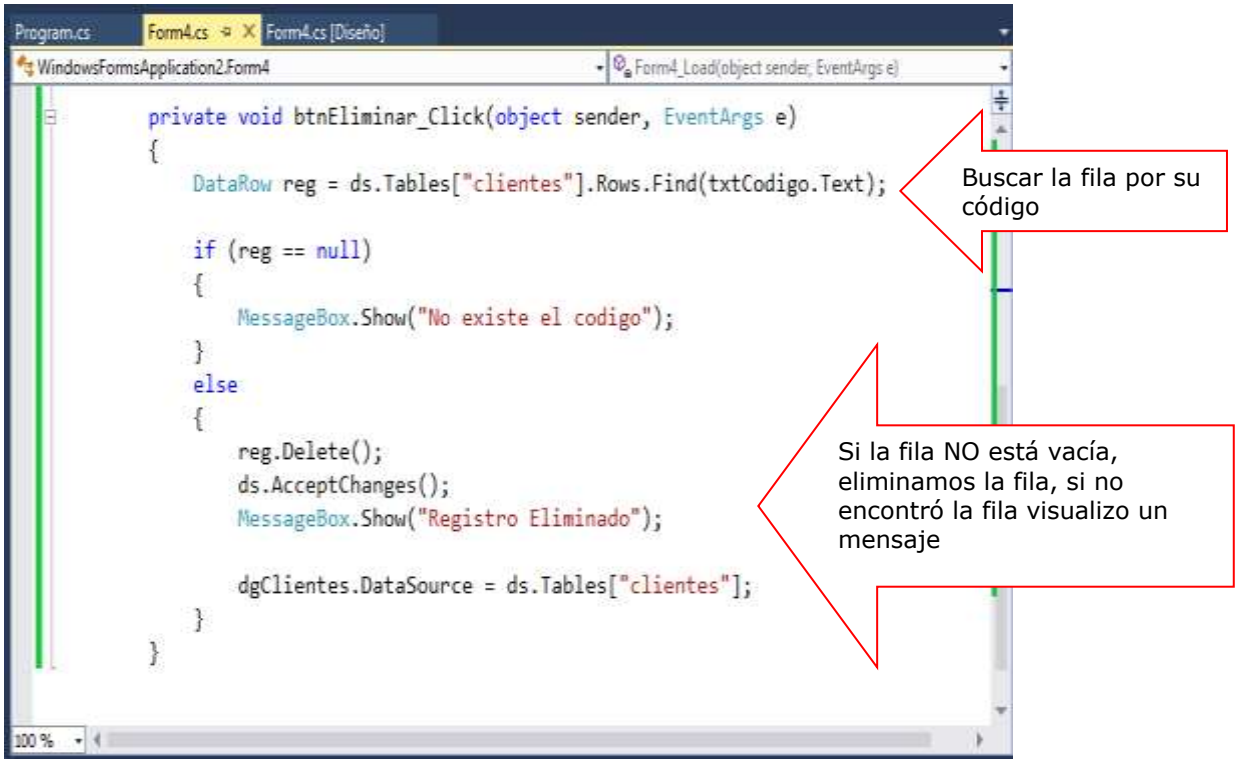
Buscar la fila por su código

Si la fila NO está vacía, actualizamos los datos de la fila y listamos, si no encontró la fila visualizo un mensaje

Para verificar, presione F5, realice la búsqueda del registro por su código, al visualizar los datos modificamos en los TextBox, para confirmar presione el botón Actualizar.



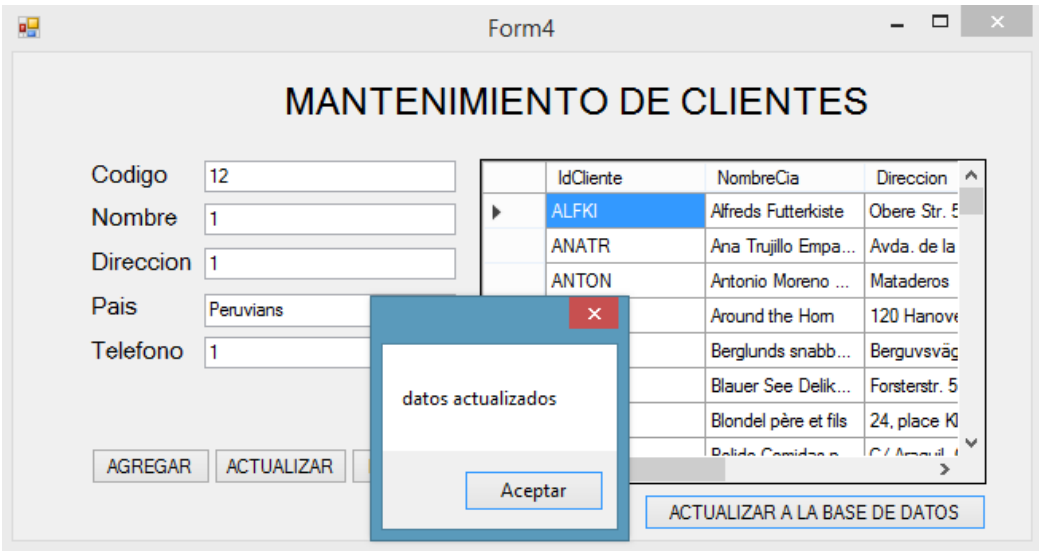
Programación del botón Eliminar, en este proceso eliminamos un cliente por su código, primero realizamos la búsqueda del código; y el resultado de la búsqueda es una fila o DataRow que se almacena el reg; si reg NO está vació, eliminamos el registro



Programación del botón Actualizar a la Base de Datos, en este proceso los datos actualizados en el DataSet deberán actualizarse en la Base de datos, para lo cual utilizamos al objeto DataAdapter, tal como se muestra.

```
private void btnActualizarBD_Click(object sender, EventArgs e)
{
    using(SqlDataAdapter da=new SqlDataAdapter("Select * from tb_clientes",cn)){

        SqlCommandBuilder cmd = new SqlCommandBuilder(da);

        try
        {
            da.Update(ds,"clientes");
            MessageBox.Show("datos actualizados");
            ds.AcceptChanges();

        }
        catch (SqlException ex)
        {
            MessageBox.Show(ex.Message);
            ds.RejectChanges();
        }
    }
}
```

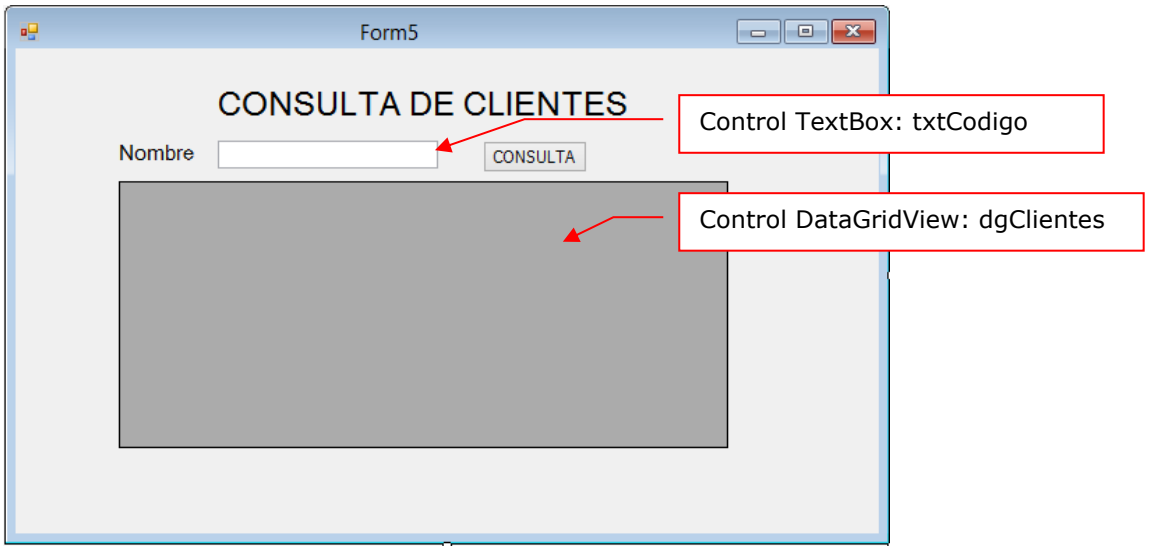
A continuación ejecute el formulario, ingrese, modifique o elimine registro de categorías. Al presionar el botón Actualizar BD, los registros serán actualizados a la Base de datos.



LABORATORIO 5.2

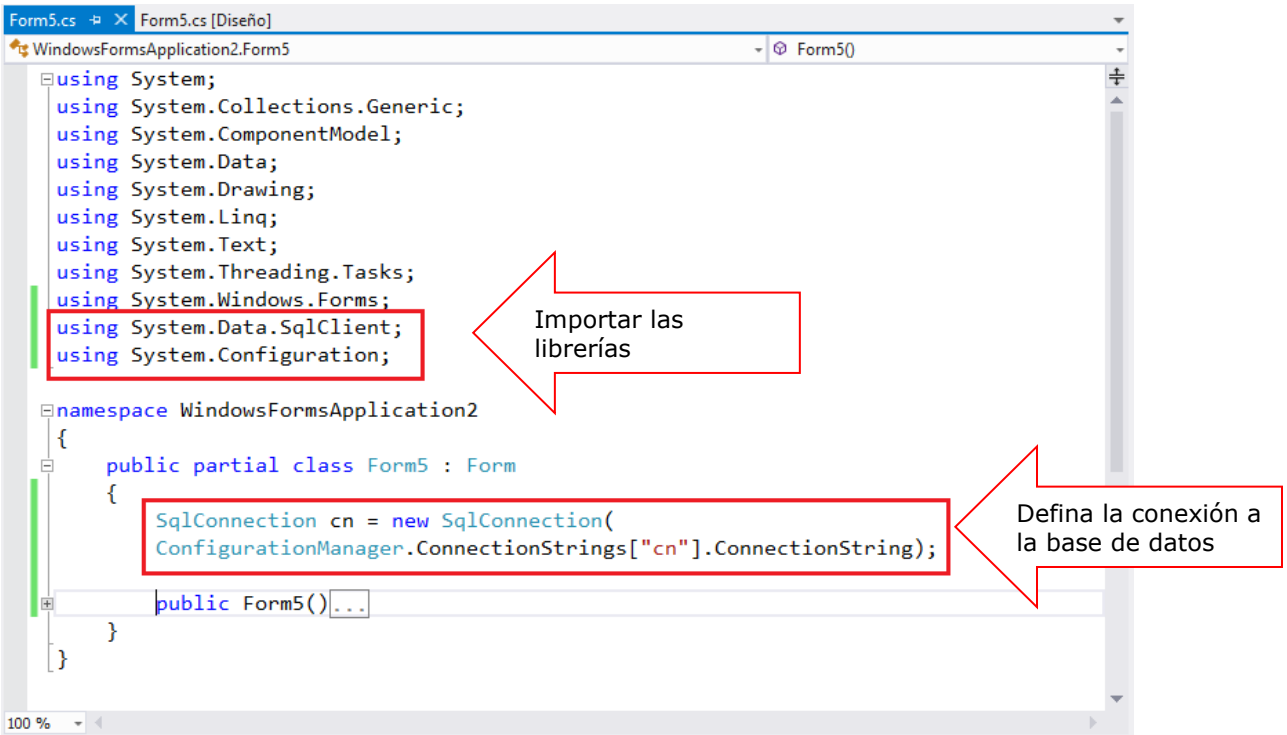
Se desea implementar un Formulario donde permita consultar los registros de la tabla tb_clientes filtrando por las iniciales del nombre. Filtrar los registros utilizando DataView.

DISEÑO DEL FORMULARIO

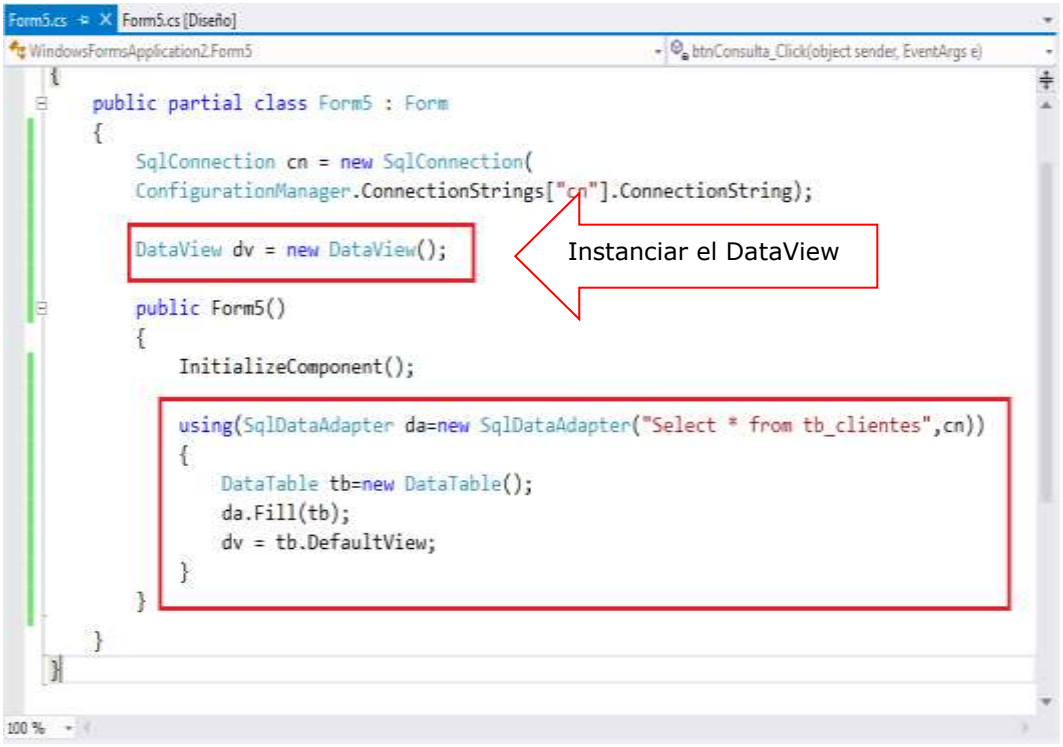


PROGRAMACIÓN

Definir la conexión a la base de datos y el DataSet donde almacene la tabla tb_clientes

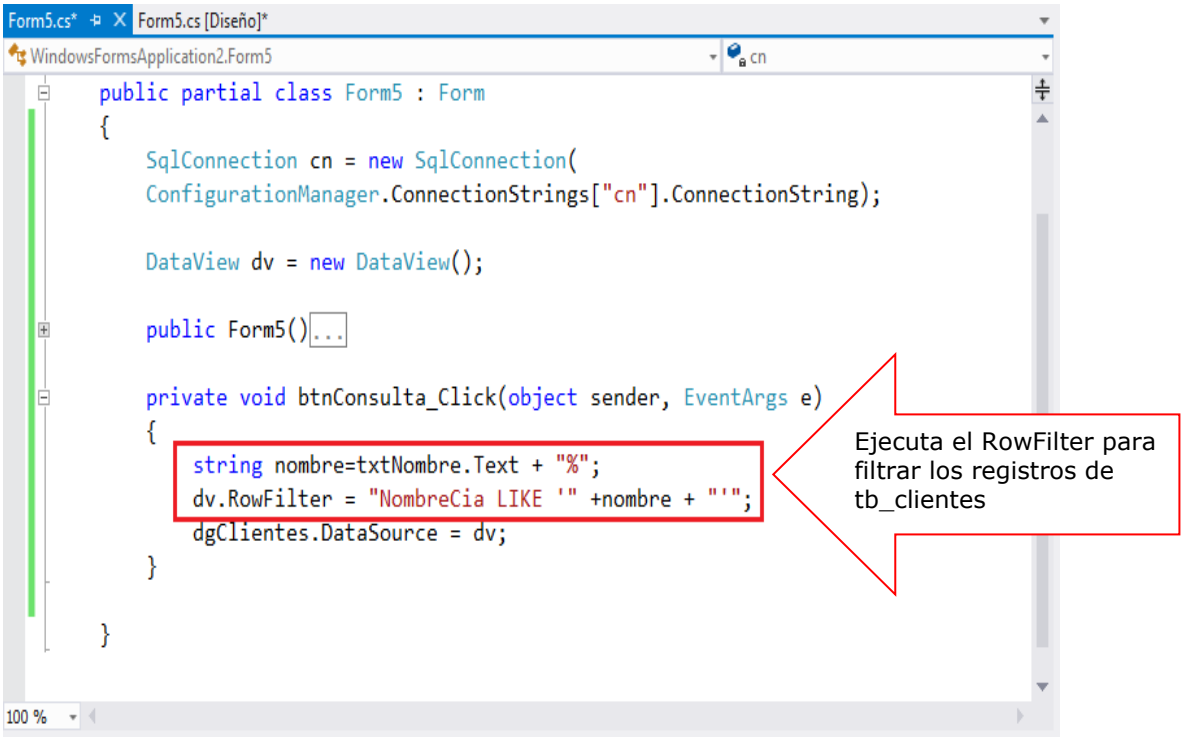


Instancia la clase DataView llamada dv, cargar los registros al DataView en el constructor del formulario.



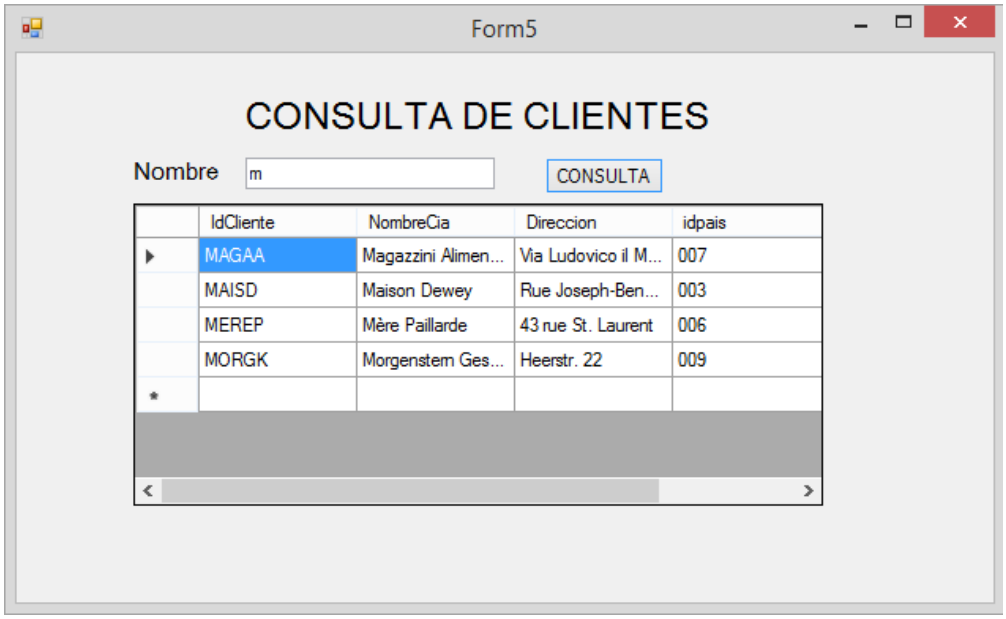

```
public partial class Form5 : Form
{
    SqlConnection cn = new SqlConnection(
    ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

    DataView dv = new DataView();

    public Form5()
    {
        InitializeComponent();

        using(SqlDataAdapter da=new SqlDataAdapter("Select * from tb_clientes",cn))
        {
            DataTable tb=new DataTable();
            da.Fill(tb);
            dv = tb.DefaultView;
        }
    }
}
```

Programa el botón Consulta, donde filtra los registros de tb_clientes por las iniciales del campo NombreCia, ejecutando el método RowFilter del DataView




```
public partial class Form5 : Form
{
    SqlConnection cn = new SqlConnection(
    ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

    DataView dv = new DataView();

    public Form5()...

    private void btnConsulta_Click(object sender, EventArgs e)
    {
        string nombre=txtNombre.Text + "%";
        dv.RowFilter = "NombreCia LIKE '" +nombre + "'";
        dgClientes.DataSource = dv;
    }

}
```

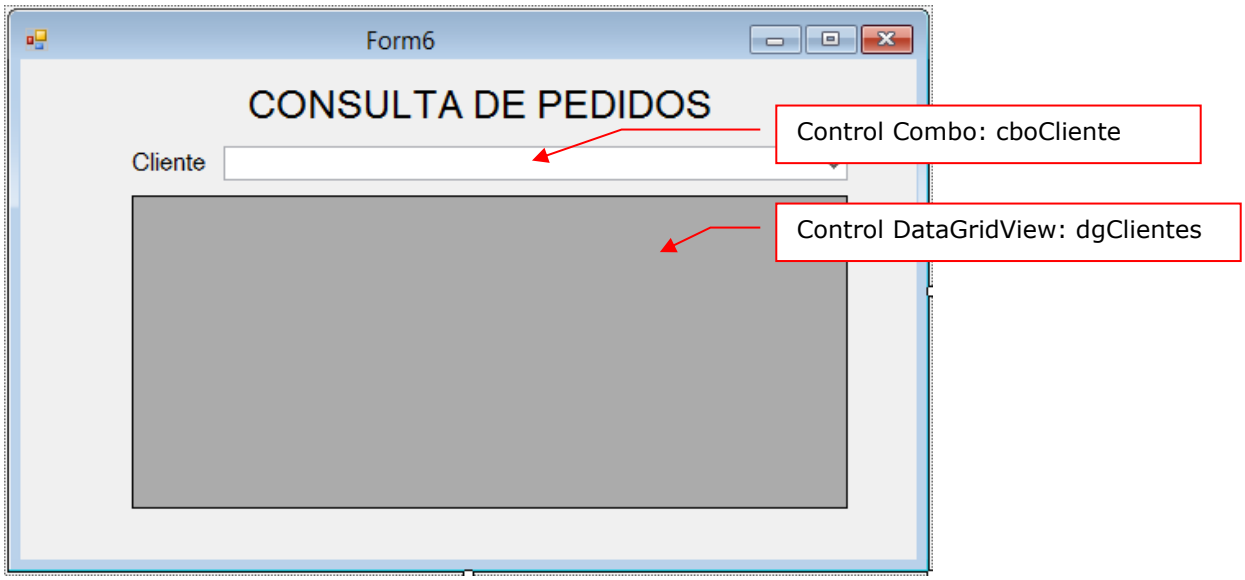
Presiona la tecla F5 para ejecutar la aplicación. Ingrese las iniciales del nombre en el cuadro de Texto, al presionar el botón Consulta, filtrar los clientes por sus iniciales.



LABORATORIO 5.3

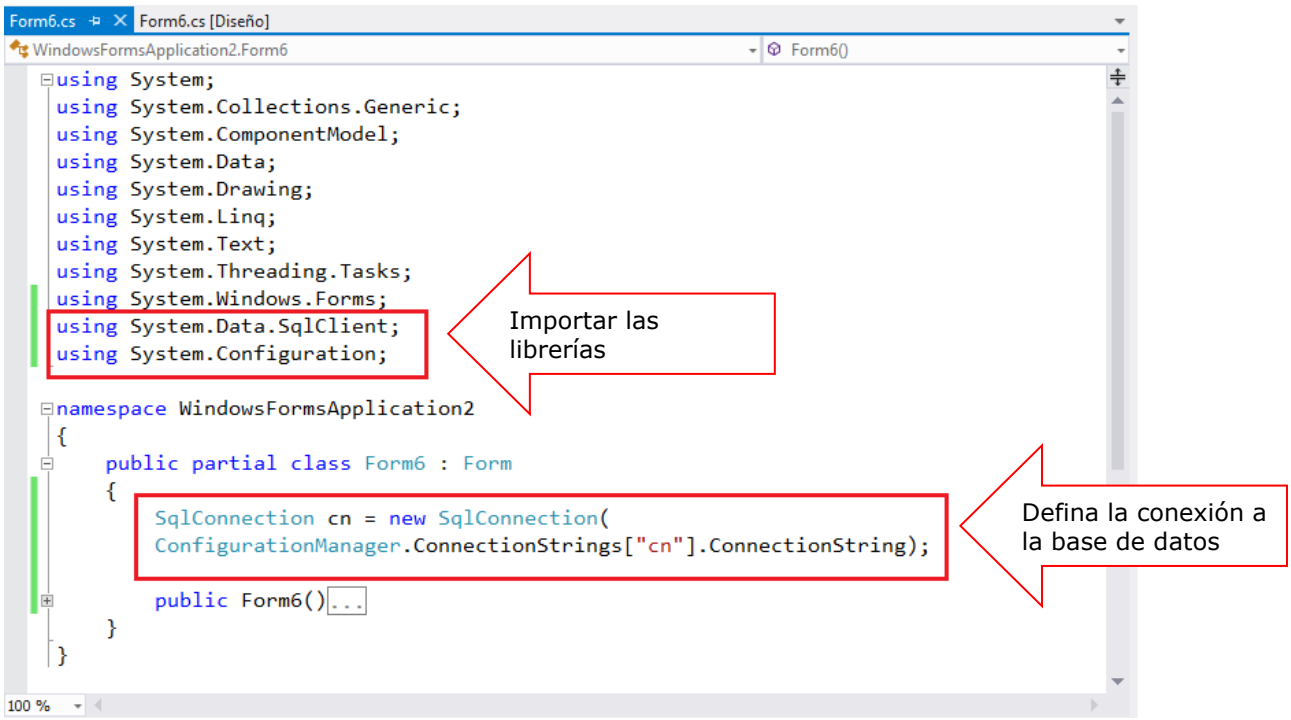
Se desea implementar un Formulario donde permita consultar los registros de la tabla tb_pedidoscabe por un determinado cliente. Filtrar los registros utilizando DataView

DISEÑO DEL FORMULARIO

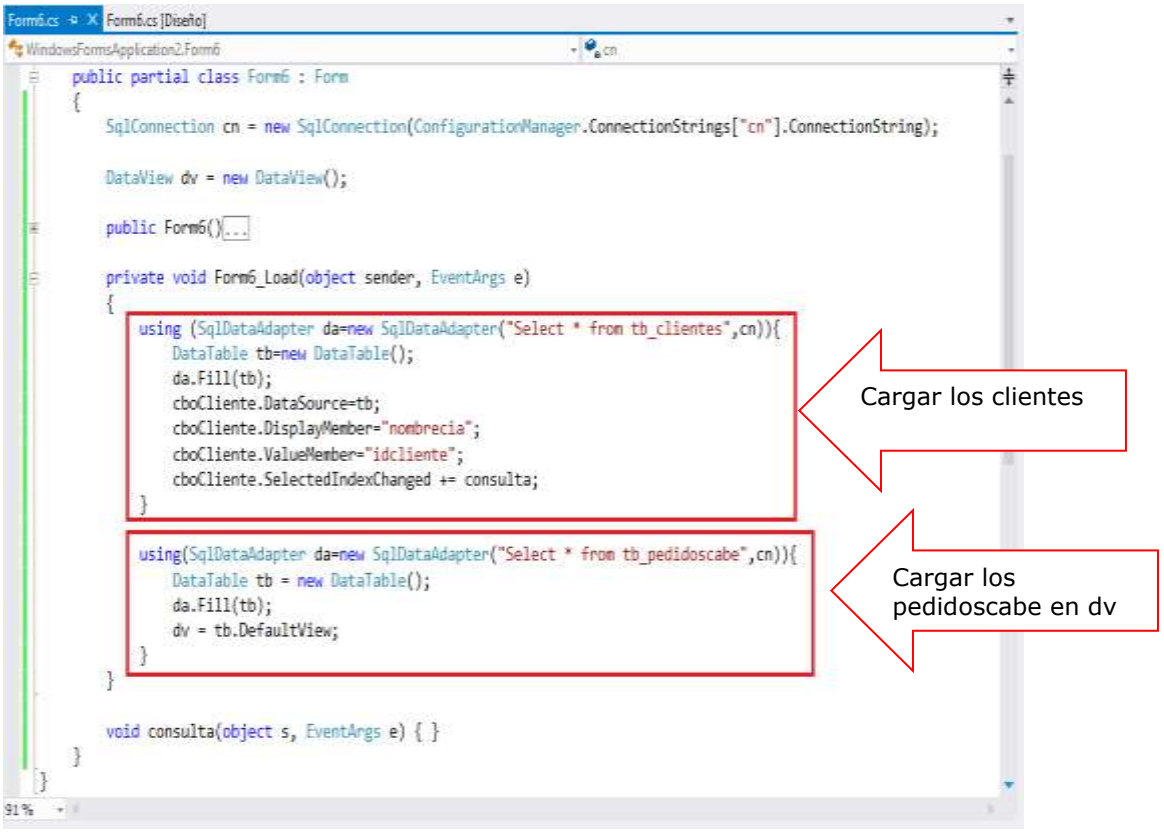


PROGRAMACIÓN

Importar las librerías y definir la conexión a la base de datos.



Programa el evento del Load del formulario que permite cargar los datos de los clientes en el comboBox y cargar los datos de pedidoscabe en el objeto DataView

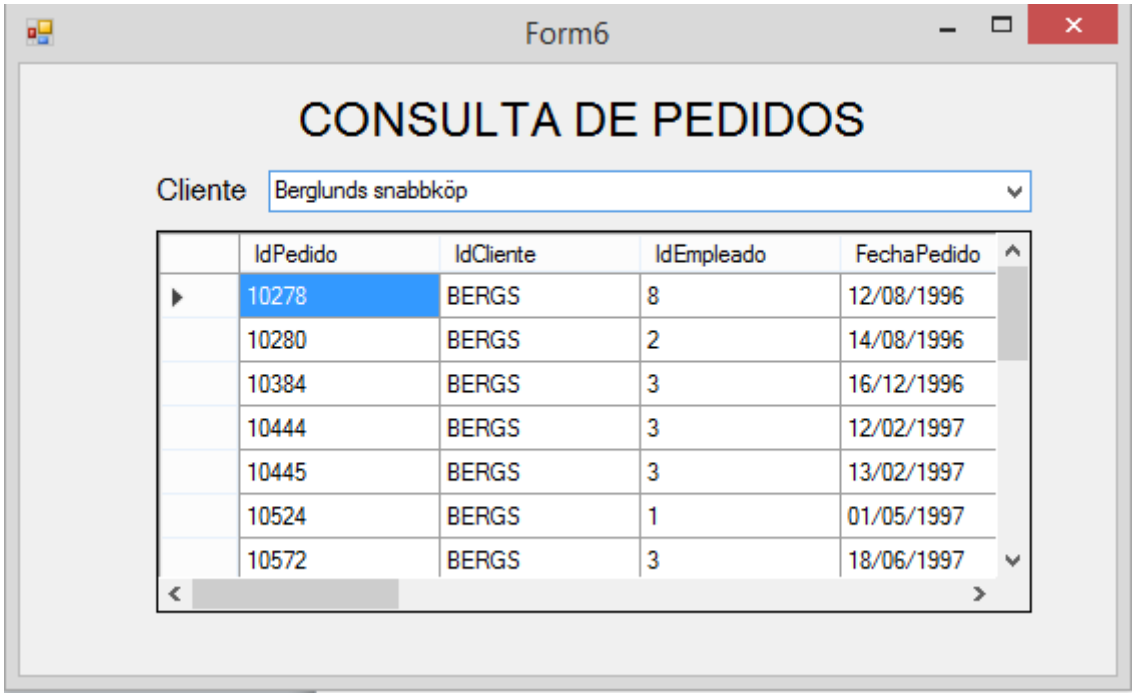



```
public partial class Form6 : Form
{
    SqlConnection cn = new SqlConnection(ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

    DataView dv = new DataView();

    public Form6()...

    private void Form6_Load(object sender, EventArgs e)
    {
        using (SqlDataAdapter da=new SqlDataAdapter("Select * from tb_clientes",cn)){
            DataTable tb=new DataTable();
            da.Fill(tb);
            cboCliente.DataSource=tb;
            cboCliente.DisplayMember="nombrecia";
            cboCliente.ValueMember="idcliente";
            cboCliente.SelectedIndexChanged += consulta;
        }

        using(SqlDataAdapter da=new SqlDataAdapter("Select * from tb_pedidoscabe",cn)){
            DataTable tb = new DataTable();
            da.Fill(tb);
            dv = tb.DefaultView;
        }
    }

    void consulta(object s, EventArgs e) { }
}
```

Programa el metodo consulta, donde filtra los registros de tb_pedidoscabe por su campo idcliente.


```
public partial class Form6 : Form
{
    SqlConnection cn = new SqlConnection(ConfigurationManager.ConnectionStrings["cn"].ConnectionString);

    DataView dv = new DataView();

    public Form6()...

    private void Form6_Load(object sender, EventArgs e)...

    void consulta(object s, EventArgs e)
    {
        string cod = cboCliente.SelectedValue.ToString();

        dv.RowFilter = "idcliente = '" + cod + "'";

        dgPedidos.DataSource = dv;
    }
}
```

Ejecuta el formulario, selecciona un cliente y se visualiza los registros de tb_pedidoscabe por el cliente seleccionado



Resumen

- El **DataSet** de ADO.NET es una representación de datos residente en memoria que proporciona un modelo de programación relacional coherente independientemente del origen de datos que contiene. Un **DataSet** representa un conjunto completo de datos, incluyendo las tablas que contienen, ordenan y restringen los datos, así como las relaciones entre las tablas
- Puede crear una instancia de **DataSet** llamando al constructor **DataSet**. Si lo desea, especifique un nombre de argumento. Si no especifica ningún nombre para el **DataSet**, se establecerá el nombre "**NewDataSet**".
- En un DataSet que contiene varios objetos **DataTable**, es posible utilizar objetos DataRelation para relacionar una tabla con otra, navegar por las tablas y devolver filas secundarias o primarias de una tabla relacionada.
- **DataAdapter** permite devolver únicamente una página de datos mediante sobrecargas del método **Fill**. Sin embargo, quizás no sea la mejor opción para paginar a través de resultados de consultas grandes ya que, aunque el **DataAdapter** rellena la **DataTable** o el **DataSet** de destino solo con los registros solicitados, se siguen utilizando los recursos para devolver toda la consulta.
- El método **Update** de **DataAdapter** se llama para reflejar en el origen de datos todos los cambios efectuados en DataSet. El método **Update**, al igual que el método **Fill**, acepta como argumentos una instancia de **DataSet** y, de forma opcional, un objeto **DataTable** o un nombre de **DataTable.**
- Representa una vista personalizada que puede enlazar datos de un **DataTable** para ordenación, filtrado, búsqueda, edición y navegación. El **DataView** no almacena datos, sino que representa una vista conectada al **DataTable** correspondiente. Los cambios en los datos de **DataView** afectarán a **DataTable**.
- La DataView proporciona varias formas de ordenación y filtrado de datos en una DataTable, la propiedad Sort permite el ordenamiento simple o de varias columnas; la propiedad RowFilter permite filtrar los registros a través de una condición.

- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  - http://msdn.microsoft.com/es-es/library/vstudio/zb0sdh0b(v=vs.100).aspx
  - http://msdn.microsoft.com/es-es/library/ss7fbaez(v=vs.110).aspx
  - http://msdn.microsoft.com/es-es/library/33y2221y(v=vs.110).aspx
  - http://msdn.microsoft.com/es-es/library/13wb36xf(v=vs.110).aspx
  - http://msdn.microsoft.com/es-es/library/system.data.dataview(v=vs.110).aspx

UNIDAD

2

OPERACIONES DESCONECTADAS A UN ORIGEN DE DATOS

Al término de la unidad, el alumno realiza operaciones de consulta y actualización de datos en el entorno de una aplicación Windows desconectado de un origen de datos utilizando la librería ADO.NET

Temario

Tema 6: Implementación de la arquitectura de capas para actualización de datos

1. Arquitectura de capas en el proceso de datos
   1.1. Introducción a la arquitectura de capas
   1.2. Definición de la capa de entidades, de datos y de negocios
   1.3. Implementación de las capas en el proceso del negocio utilizando la capa de presentación: Formularios
2. Manejo de “N” capas en una aplicación Maestro-Detalle

ACTIVIDADES PROPUESTAS

- Los alumnos reconocen el modelo de capas.
- Los alumnos manejan las capas de datos y de procesos
- Los alumnos realizan operaciones de actualización de datos utilizando el modelo de capas.

6 .ARQUITECTURA DE CAPAS EN EL PROCESO DE DATOS

6.1 INTRODUCCION A LA ARQUITECTURA DE CAPAS

La programación por capas es una arquitectura cliente-servidor en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación sería el modelo de interconexión de sistemas abiertos.

Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

En el diseño de sistemas informáticos actual se suelen usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

El diseño más utilizado actualmente es el diseño en tres niveles (o en tres capas).

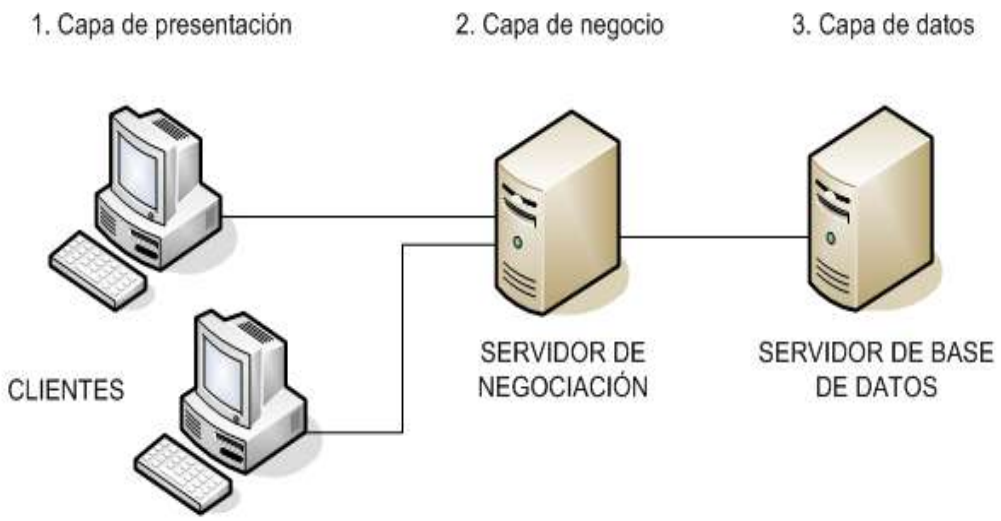


Figura 1: Modelo de "N" Capas
Referencia: http://iutll-abdd.blogspot.com/2012/05/arquitectura-de-n-capas.html

6.2 DEFINICIÓN DE LAS CAPAS

Una aplicación que pretenda utilizar la "potencia" o características particulares de un motor de base de datos, incorporará en la Capa de Lógica de Negocios algo de código que no es compatible con otros motores de base de datos. En consecuencia, cambiar la capa de datos significa corregir la capa de lógica de negocios.

Las buenas prácticas de diseño y desarrollo indican que se debe trabajar sobre el siguiente diagrama:

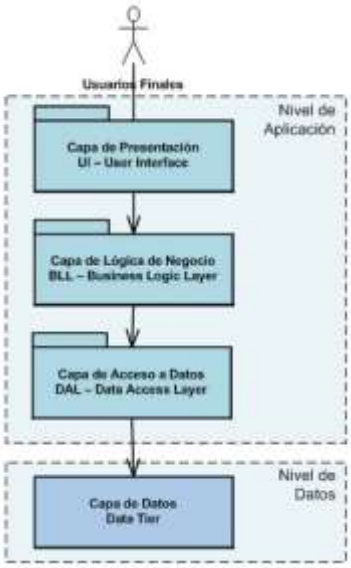


Figura 2: Arquitectura a 3 Capas
Referencia: http://jtentor.com.ar/post/Arquitectura-de-N-Capas-y-N-Niveles.aspx

La nueva capa, se denomina Capa de Acceso a Datos (o Capa de Persistencia) que no es lo mismo que Capa de Datos.

La capa de acceso a datos es una porción de código que justamente realiza el acceso a los datos. De esta manera cuando es necesario cambiar el motor de base de datos, solamente tendremos que corregir esa capa.

En la arquitectura en 3 niveles, existe un nivel intermediario. Esto significa que la arquitectura generalmente está compartida por: Un cliente, es decir, el equipo que solicita los recursos, equipado con una interfaz de usuario (generalmente un navegador Web) para la presentación.

El servidor de aplicaciones (también denominado software intermedio), cuya tarea es proporcionar los recursos solicitados, pero que requiere de otro servidor para hacerlo. El servidor de datos, que proporciona al servidor de aplicaciones los datos que requiere. Comparación entre ambos tipos de arquitecturas La arquitectura en 2 niveles es, por lo tanto, una arquitectura cliente/servidor en la que el servidor es polivalente, es decir, puede responder directamente a todas las solicitudes de recursos del cliente.

Sin embargo, en la arquitectura en 3 niveles, las aplicaciones al nivel del servidor son descentralizadas de uno a otro, es decir, cada servidor se especializa en una determinada tarea, (por ejemplo: servidor web/servidor de bases de datos). La arquitectura en 3 niveles permite: Un mayor grado de flexibilidad Mayor seguridad, ya que la seguridad se puede definir independientemente para cada servicio y en cada nivel Mejor rendimiento, ya que las tareas se comparten entre servidores.

1. Capa de presentación: es la que ve el usuario (también se la denomina "capa de usuario"), es la representación del sistema al usuario. Esta capa permite mostrar la información; y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato).

También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

2. Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

3. Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

6.3 IMPLEMENTACION DE LAS CAPAS EN EL PROCESO DEL NEGOCIO UTILIZANDO LA CAPA DE PRESENTACION: FORMULARIOS

Una aplicación que pretenda utilizar la "potencia" o características particulares de un motor de base de datos, incorporará en la Capa de Lógica de Negocios algo de código que no es compatible con otros motores de base de datos. En consecuencia, cambiar la capa de datos significa corregir la capa de lógica de negocios.

Las buenas prácticas de diseño y desarrollo indican que se debe trabajar sobre el siguiente diagrama:

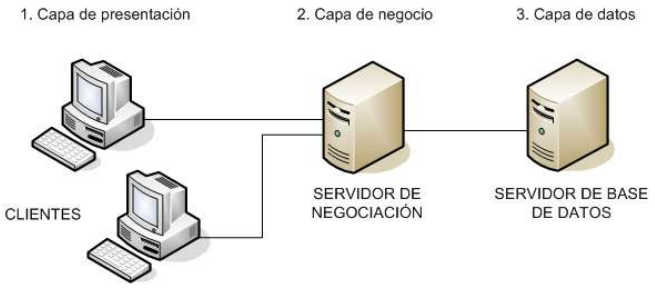


Figura 3: Arquitectura a 3 Capas
Ref: http://es.sandramarramirez.wikia.com/wiki/Programaci%C3%B3n_por_Capas

La capa de Presentación provee su aplicación con una interfase de usuario (IU). Aquí es donde su aplicación presenta información a los usuarios y acepta entradas o respuestas del usuario para usar su programa. Idealmente, la IU no desarrolla ningún procesamiento de negocios o reglas de validación de negocios.

Por el contrario, la IU debería relegar sobre la capa de negocios para manipular estos asuntos. Esto es importante, especialmente hoy en día, debido a que es muy común para una aplicación tener múltiples IU, o para sus clientes o usuarios, que le solicitan que elimine una IU y la remplace con otra. Por ejemplo, usted puede desarrollar una aplicación en C# y remplazarla con una página HTLM., quizás usando tecnología ASP-NET (Active Server Pages creación de páginas dinámicas del lado del servidor).

Comprende las responsabilidades de lógica de presentación:
- Navegabilidad del sistema
- Validación de datos de entrada
- Formateo de los datos de salida
- Internacionalización
- Reenderezado de presentación

Funciones de la capa de Presentación:
- Recoger la información del Usuario.
- Enviar esta información a la Capa de Negocios.
- Recoger resultados de la Capa de Negocios
- Presentar los resultados al usuario.

6.4 MANEJO DE "N" EN UNA APLICACIÓN MAESTRO-DETALLE

En el modelo de acceso a datos, una capa es un nivel lógico en el cual residen componentes o aplicaciones lógicas. Las capas pueden residir en uno a más equipos o servidores, el número de capas hace referencia al número de niveles y no al número de equipos en los cuales los servicios son divididos. Las capas que generalmente se incluyen en aplicaciones son:

Capa de Cliente: conocida como capa de Presentación es la que contiene las interfaces en las que el usuario interactua con el sistema.
Capa de la Lógica de Negocios: el cual contiene la lógica que interactua con el origen de datos. Esta capa intermedia contiene la parte de la aplicación que interactua con los datos, por ejemplo: la creación de una cadena de conexión al origen de datos.
Capa de acceso a Datos: la cual se relaciona directamente con el origen de datos.

Beneficios del trabajo con Capas en maestro-detalle
- Escalabilidad en las aplicaciones
- Distribución mas efectiva
- Cambios en la aplicaciones mas sencillos de manejar e implementar
- Separación de funciones
- Permite aplicaciones en diferentes sistemas operativos
- Clientes menos pesados (thin Client)

LABORATORIO 6.1

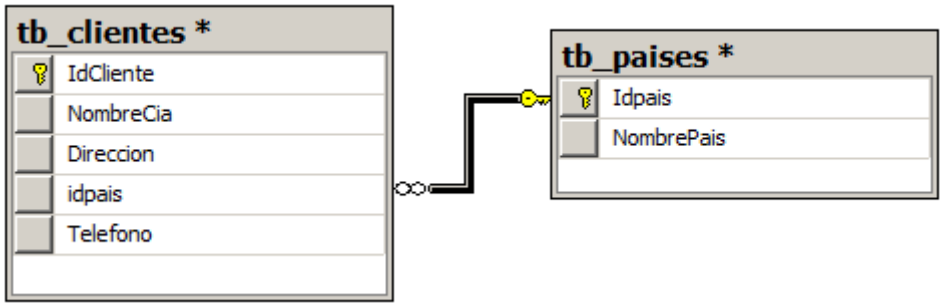
Se desea implementar una solución que permita realizar el mantenimiento a la tabla de Clientes utilizando la programación de capas

Se pide:

1. Diseño e implementación de cada una de las capas
2. Diseño de la capa de presentación, formulario para ejecutar los procesos del negocio.

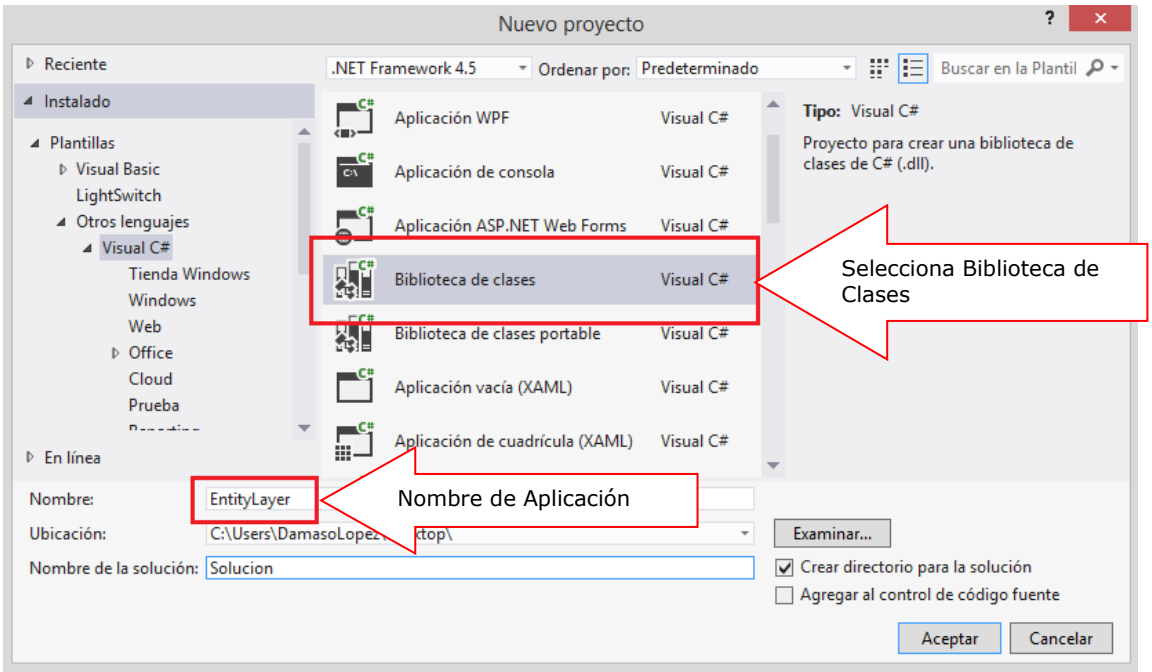
MODELO DE DATOS.

En este proceso trabajaremos con la base de datos Comercial2012, donde realizaremos el mantenimiento a la tabla tb_clientes la cual está relacionada a la tabla tb_paises, tal como se muestra en la gráfica.

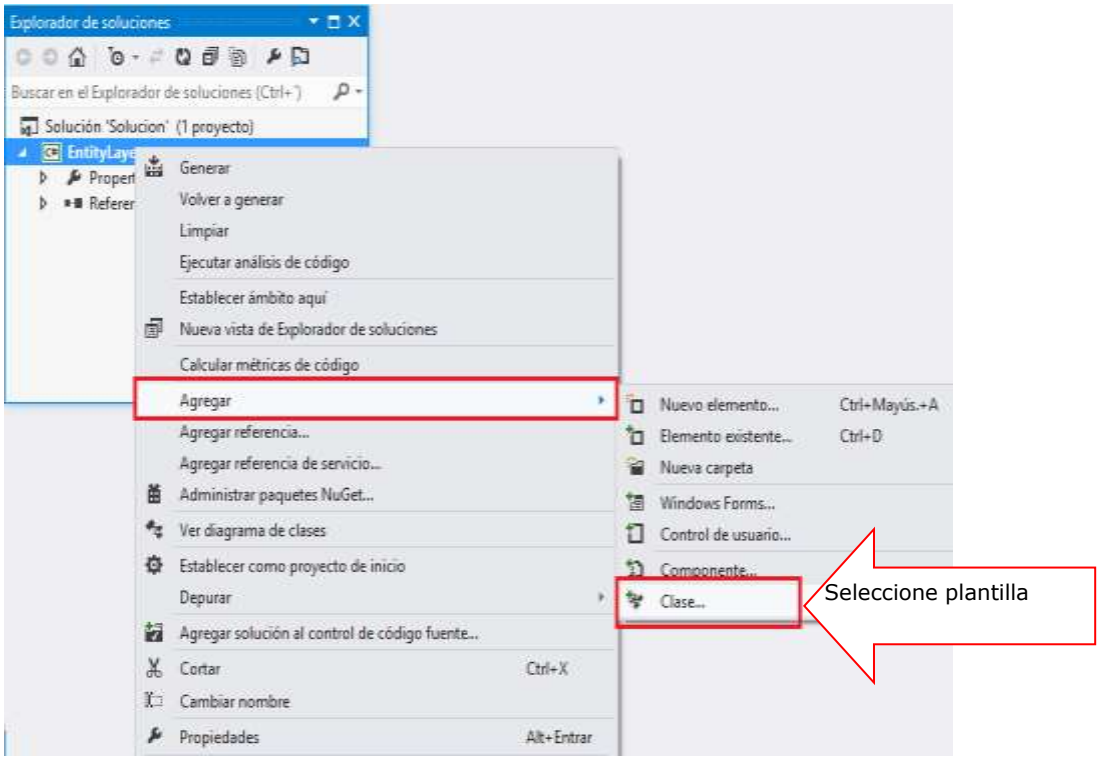


CAPA DE DATOS: BUSINESS ENTITY

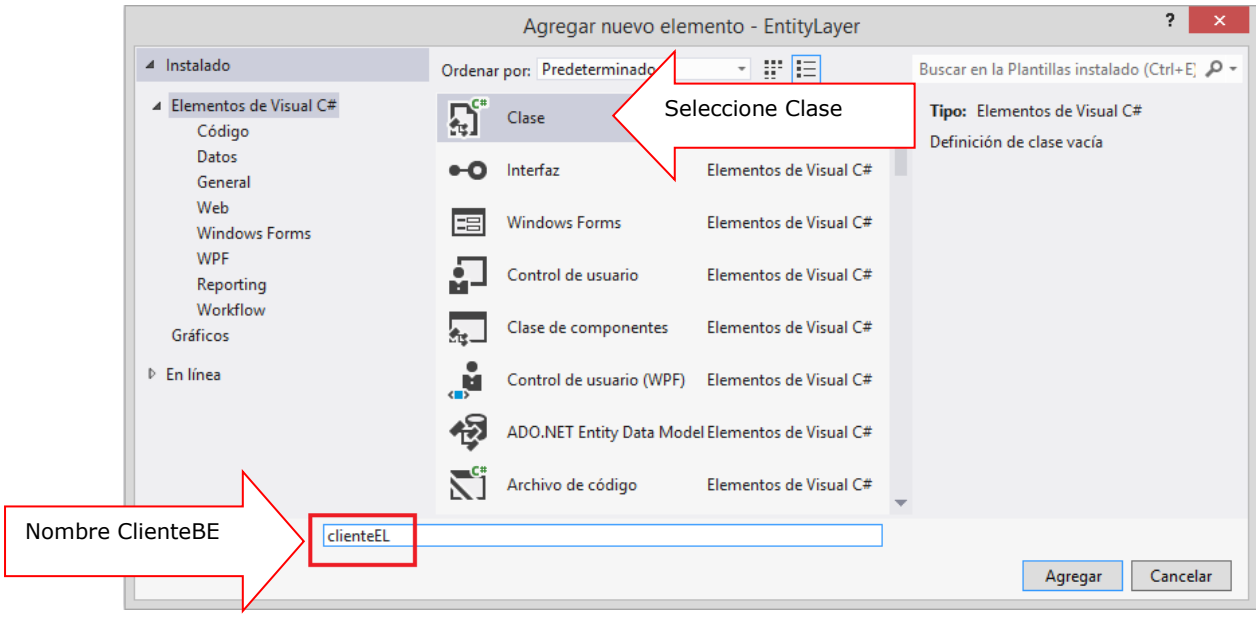
Defina un proyecto de tipo Biblioteca de Clases y la llamaremos EntityLayer; el nombre de la solución será Solucion, tal como se muestra. Terminado de configurar el proyecto presionar el botón ACEPTAR



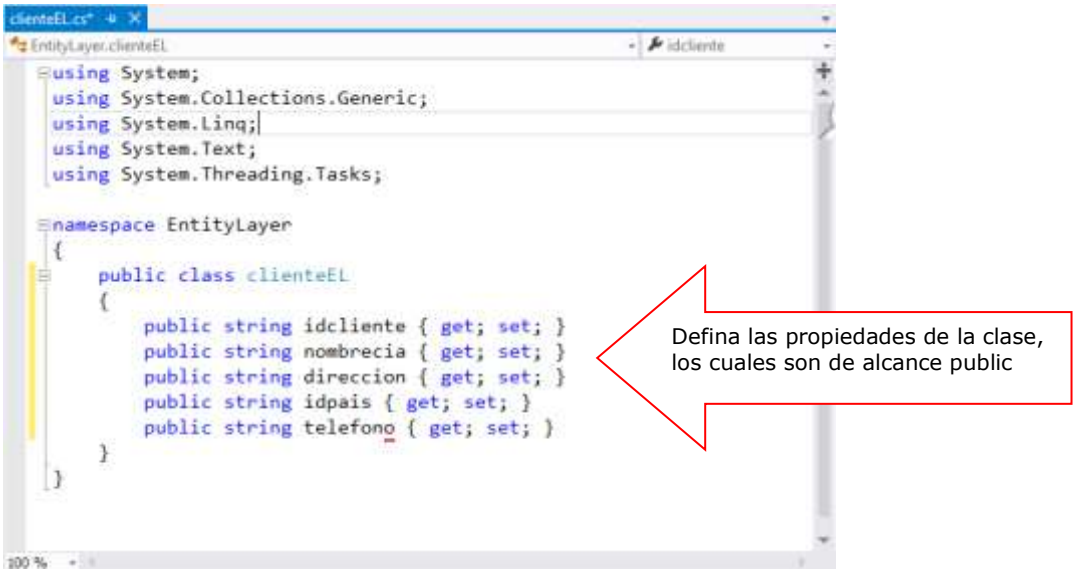
A continuación, agregar una clase al proyecto: Selecciona del proyecto EntityLayer la opción Agregar; desplegada la lista, seleccione la opción Clase… tal como se muestra.



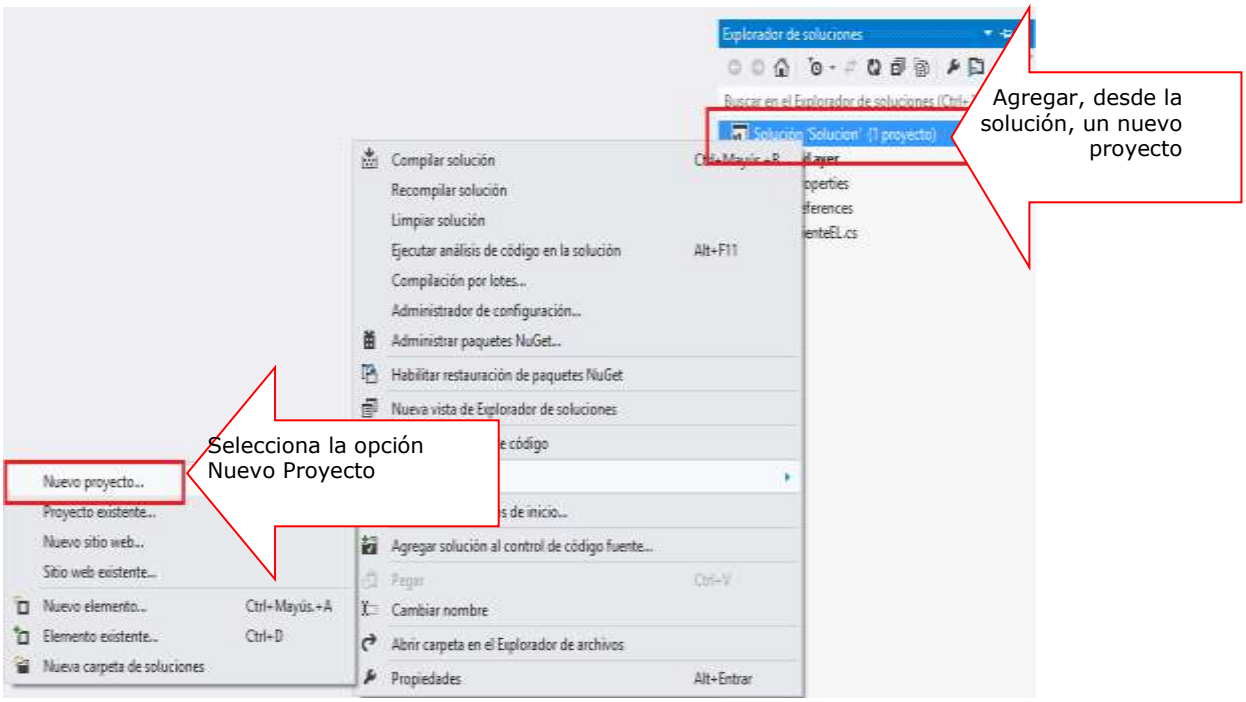
En esta ventana, el elemento Clase se encuentra seleccionado; asigne el nombre a la clase: ClienteBE. Luego presione el botón AGREGAR



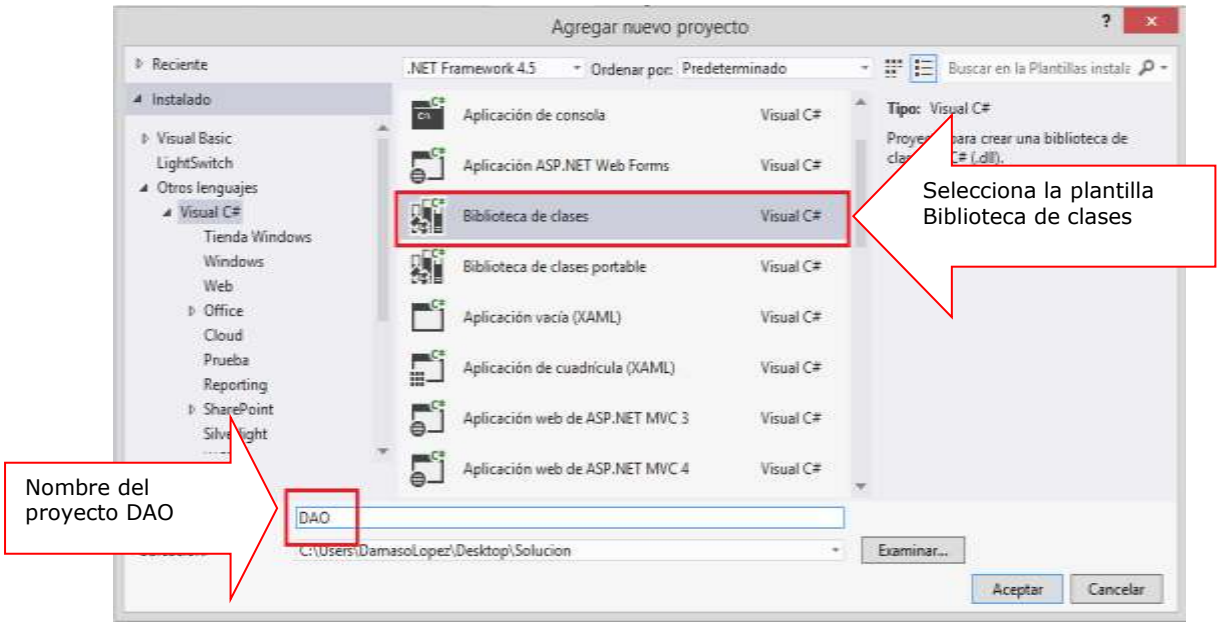
En la clase ClienteBE, como primer paso, vamos a definir los atributos de la clase, los cuales representan a las columnas de la tabla tb_Clientes, tal como se muestra



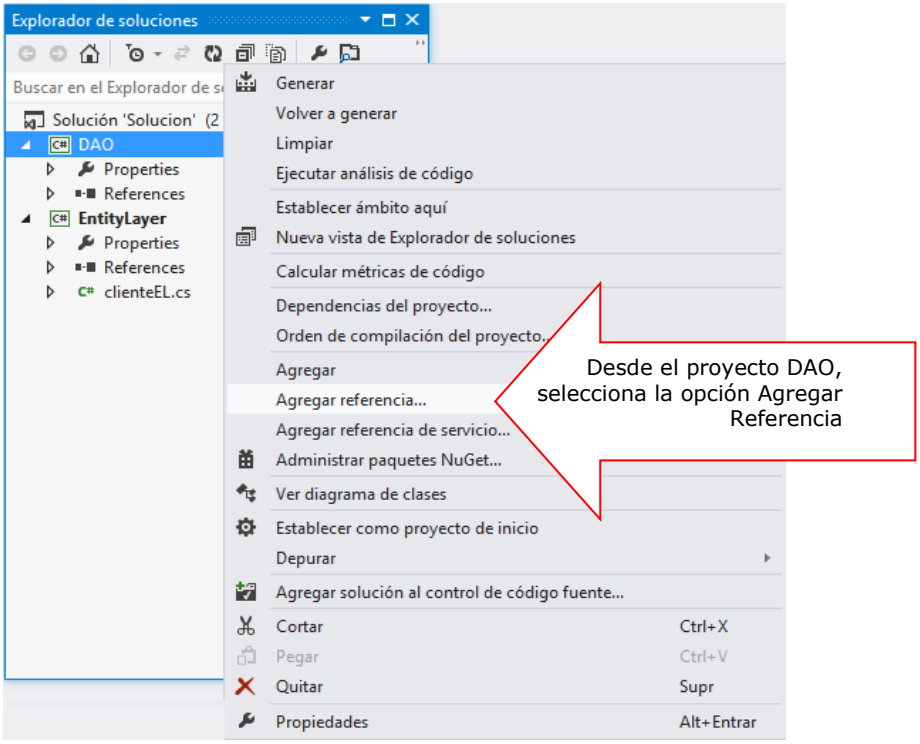
Definida la capa de Datos, a continuación proceda a AGREGAR UN NUEVO PROYECTO a la solución, tal como se muestra

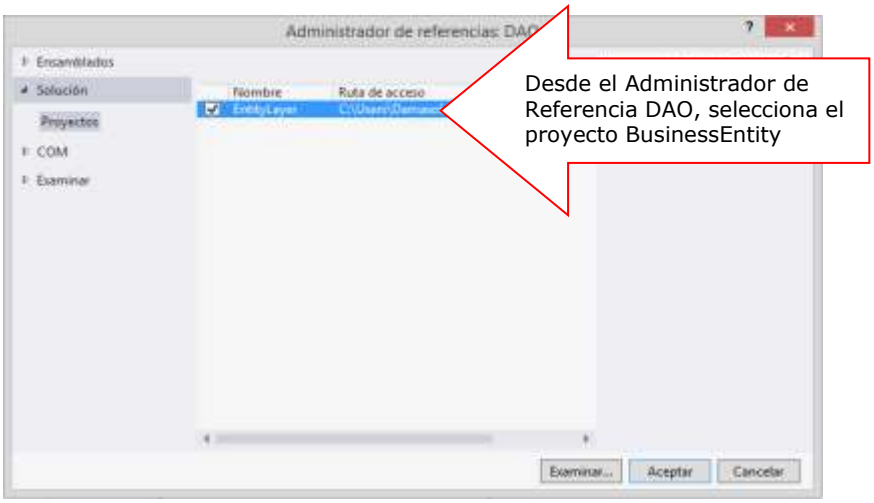


Selecciona el proyecto Biblioteca de Clases, el cual tendrá como nombre DAO (Capa de Acceso a Datos), luego presione el botón ACEPTAR

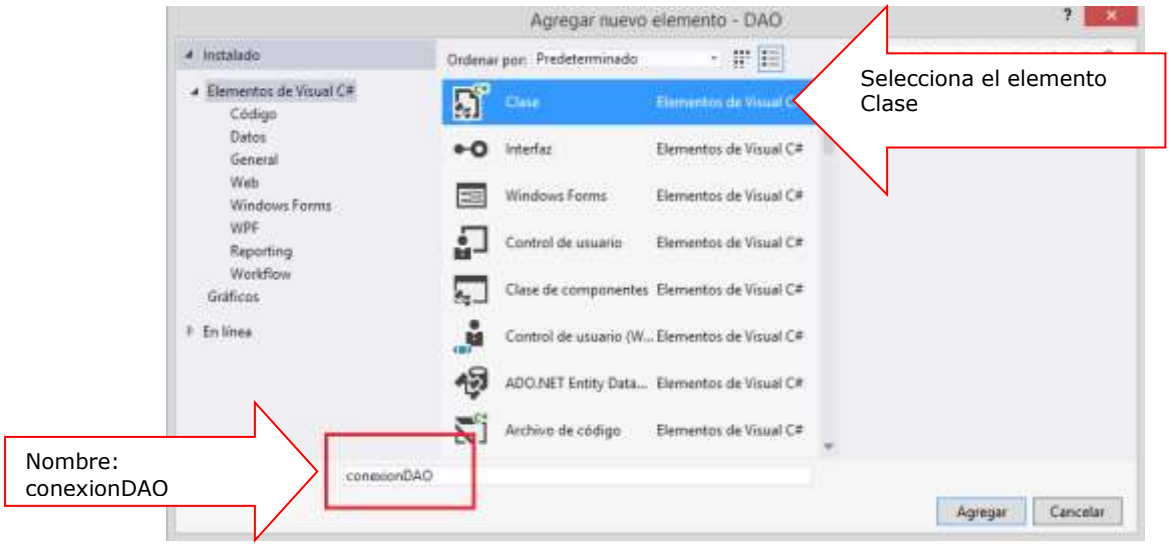


En el proyecto **DAO**, define los métodos del proceso de mantenimiento; para ello requiere el modelo de datos (EntityLayer) ya implementada. Para hacer una referencia hacia el proyecto BusinessEntity, establecer una referencia en el proyecto **DAO**: Click derecho en el proyecto **DAO** y seleccione la opción Agregar Referencia

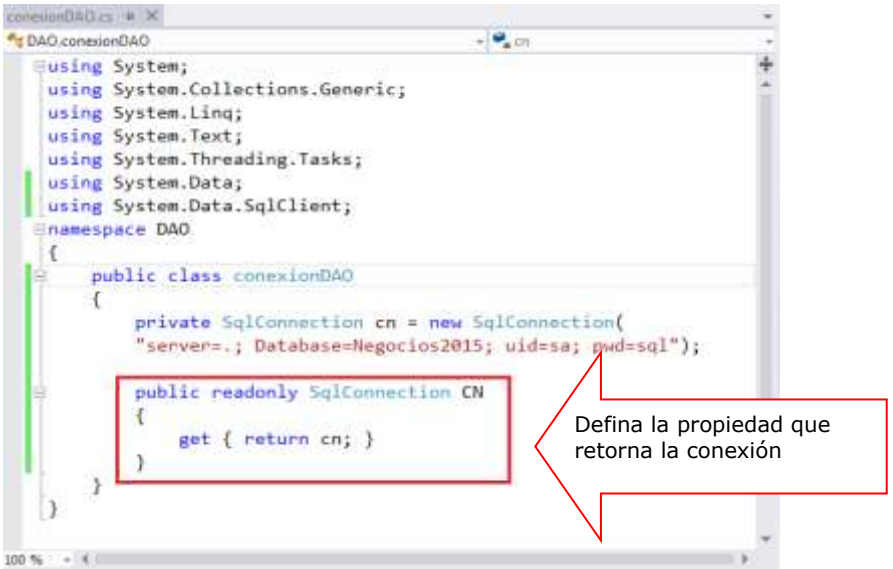




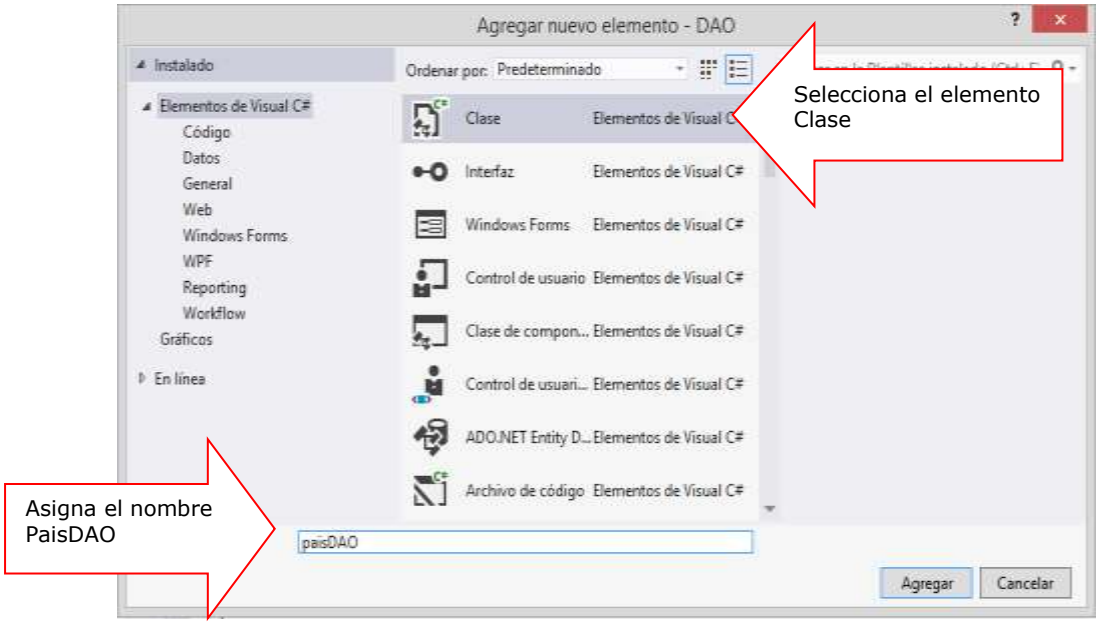
En el proyecto DAO agrega la clase conexionEL, para definir la conexión.



En la Clase conexionDAO defina la conexión a la base de datos.

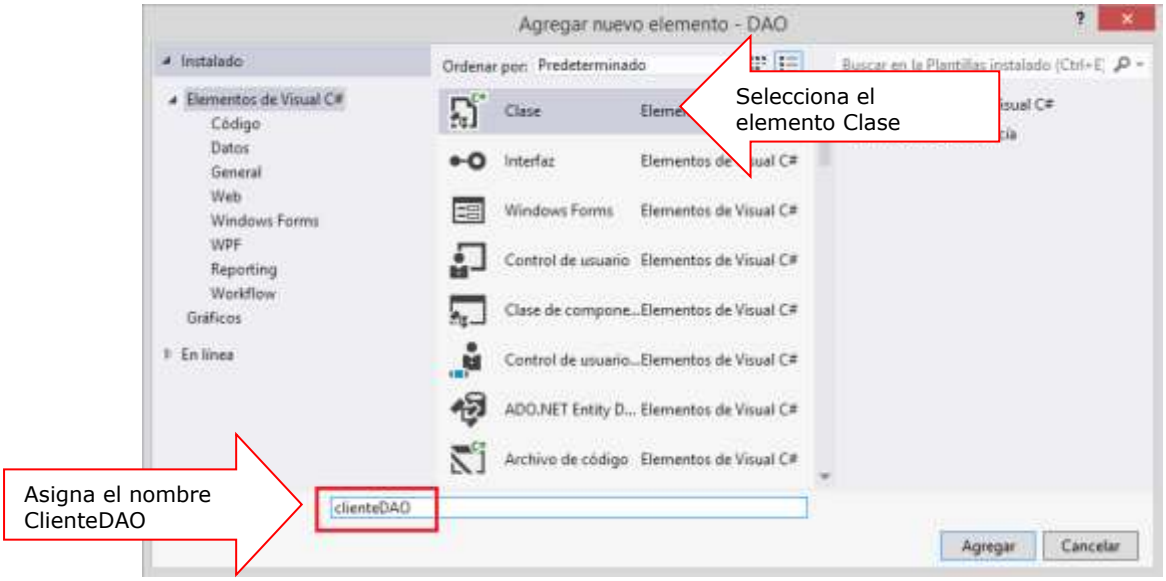


En el proyecto DAO, agrega una clase, llamadá paisDAO, tal como se muestra.

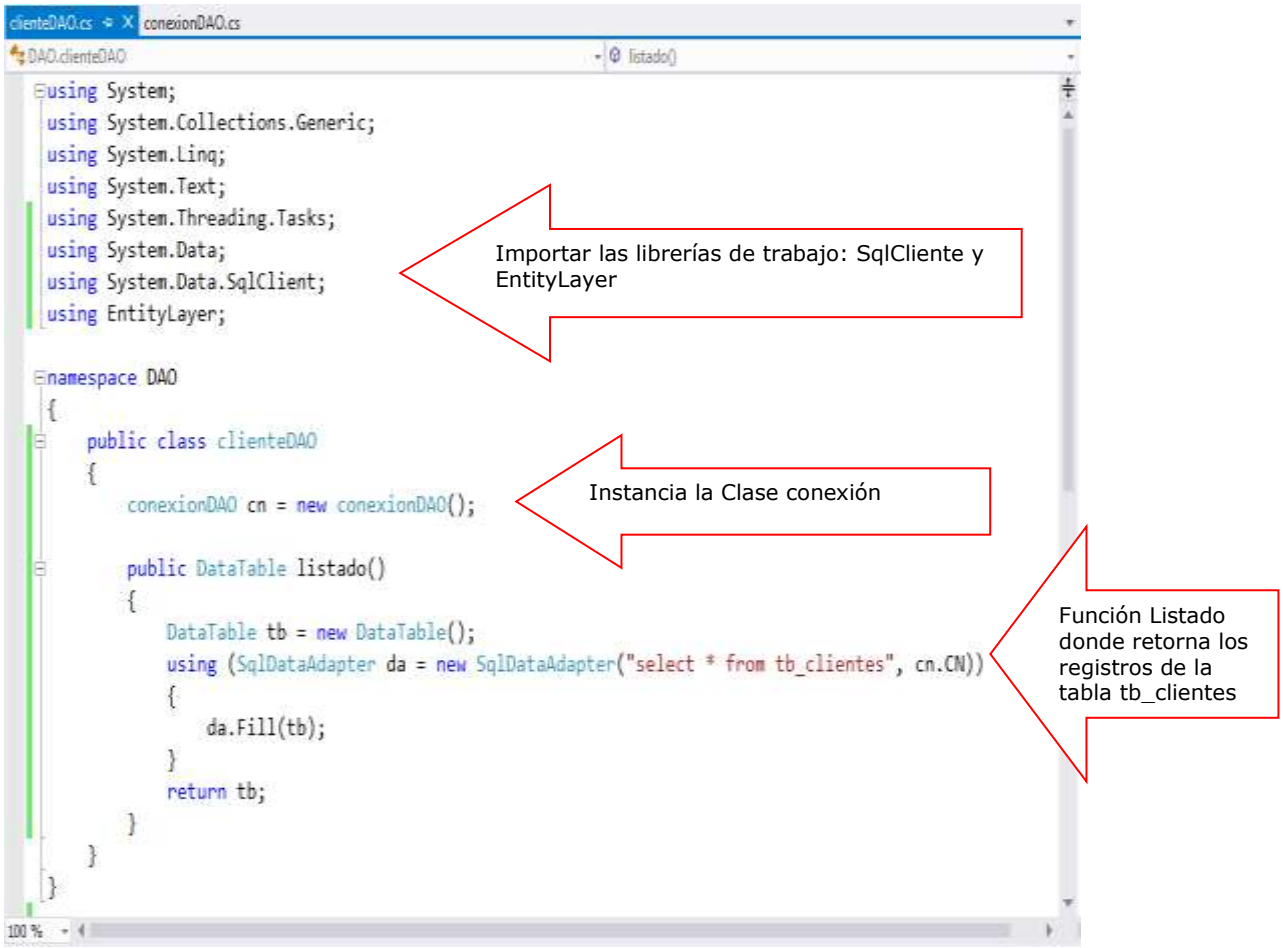


En la Clase PaisDAO defina los métodos a implementars, tal como se muestra

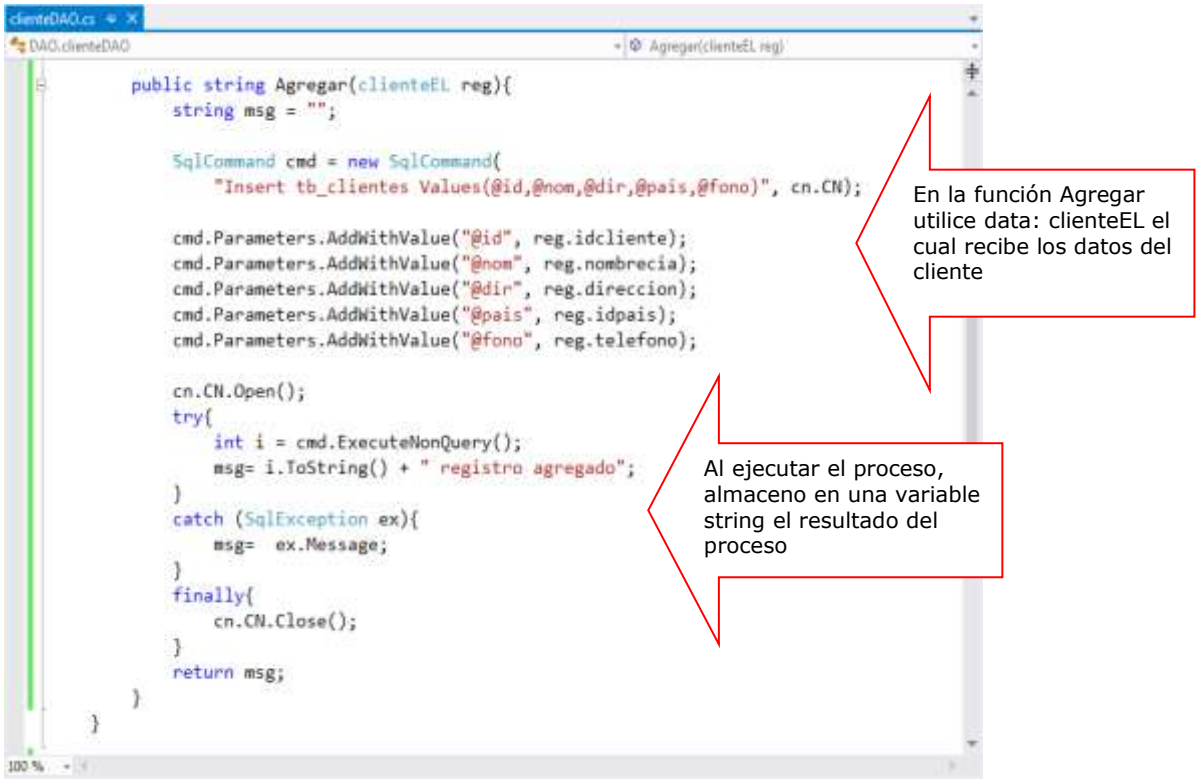
En el proyecto DAO, agrega una clase llamadá clienteDAO, tal como se muestra



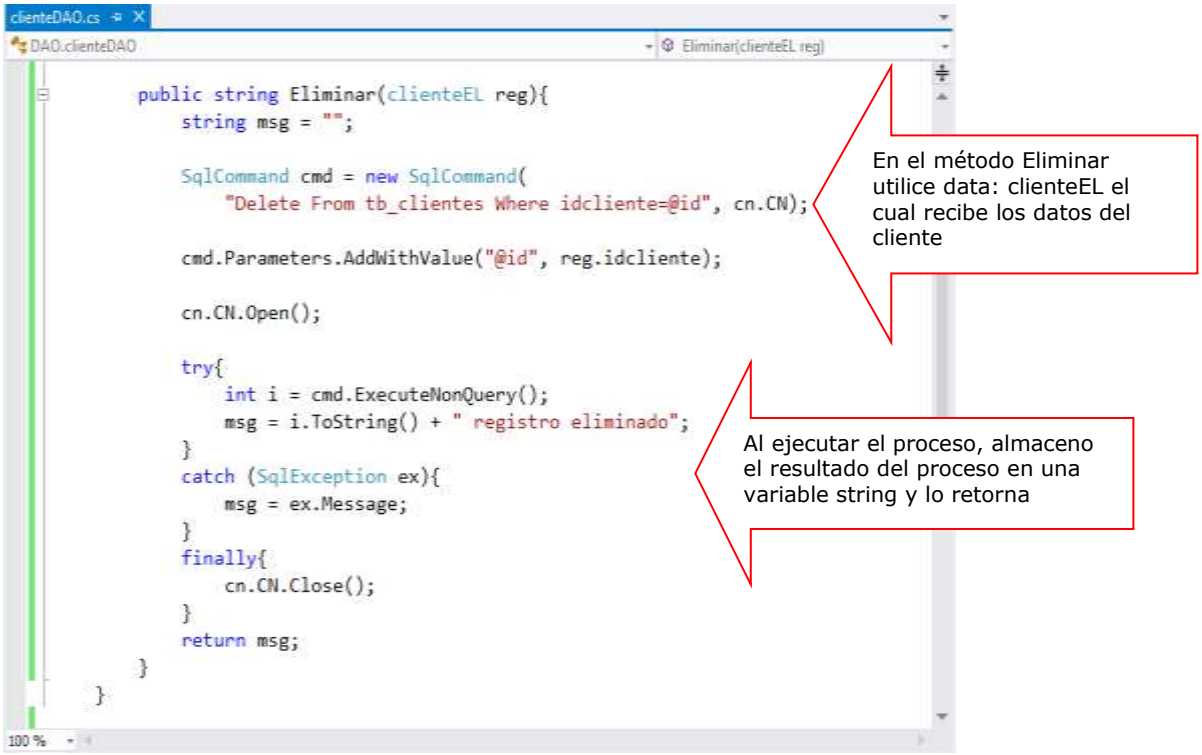
En la Clase **ClienteDAO** importa el proyecto **BusinessEntity** donde utilizamos la clase **ClienteBE**; instancia la conexión, tal como



En la clase clienteDAO defina la función Agregar que permita insertar un registro a la tabla tb_clientes retornando un mensaje de tipo string



En la clase clienteDAO defina el método Eliminar que permita eliminar un registro a la tabla tb_clientes por el campo idcliente



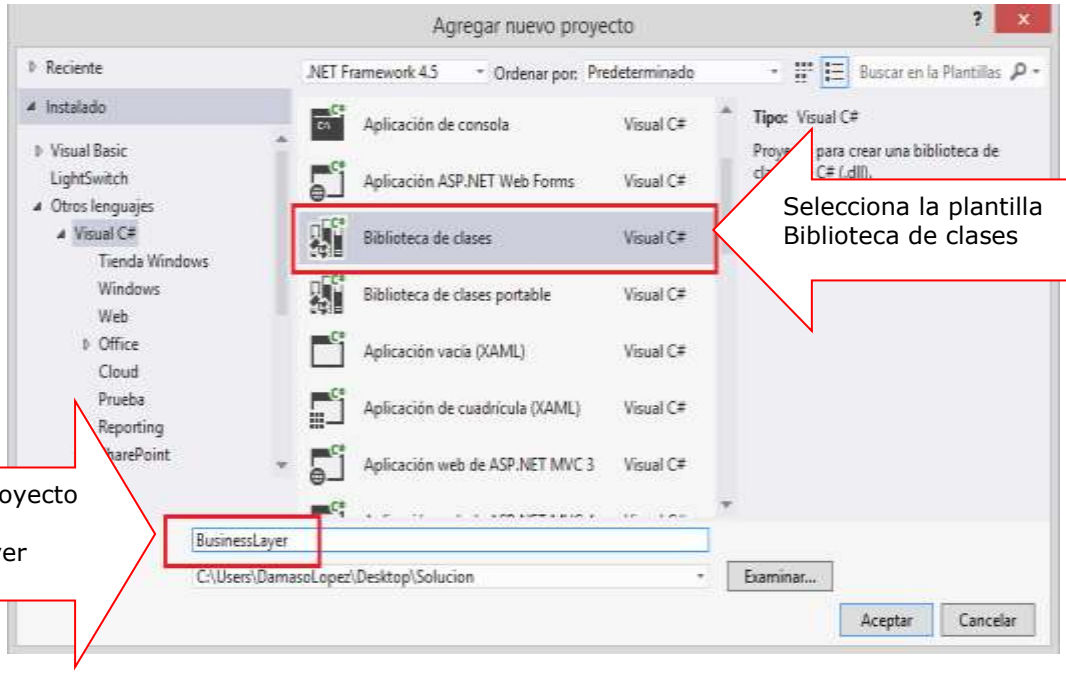
En la clase ClienteDAO defina la función Actualizar que permita actualizar un registro a la tabla tb_clientes por el campo idcliente retornando un mensaje

```
clienteDAO.cs
DAO.clienteDAO                                  Actualizar(clienteEL reg)

        public string Actualizar(clienteEL reg){
            string msg = "";

            SqlCommand cmd = new SqlCommand(
            "Update tb_clientes Set nombrecia=@nom,direccion=@dir,idpais=@pais," +
            "telefono=@fono Where idcliente=@id", cn.CN);

            cmd.Parameters.AddWithValue("@id", reg.idcliente);
            cmd.Parameters.AddWithValue("@nom", reg.nombrecia);
            cmd.Parameters.AddWithValue("@dir", reg.direccion);
            cmd.Parameters.AddWithValue("@pais", reg.idpais);
            cmd.Parameters.AddWithValue("@fono", reg.telefono);

            cn.CN.Open();
            try{
                int i = cmd.ExecuteNonQuery();
                msg = i.ToString() + " registro actualizado";
            }
            catch (SqlException ex){
                msg = ex.Message;
            }
            finally{
                cn.CN.Close();
            }
            return msg;
        }
    }
100 %
```

En la función Actualizar utilice data: clienteEL el cual recibe los datos del cliente

Al ejecutar el proceso, almaceno el resultado en una variable de tipo String y retorno el valor

Definida la capa DAO, a continuación agregamos la capa **BusinessLayer** (Capa Lógica del Negocio), luego presione el botón ACEPTAR

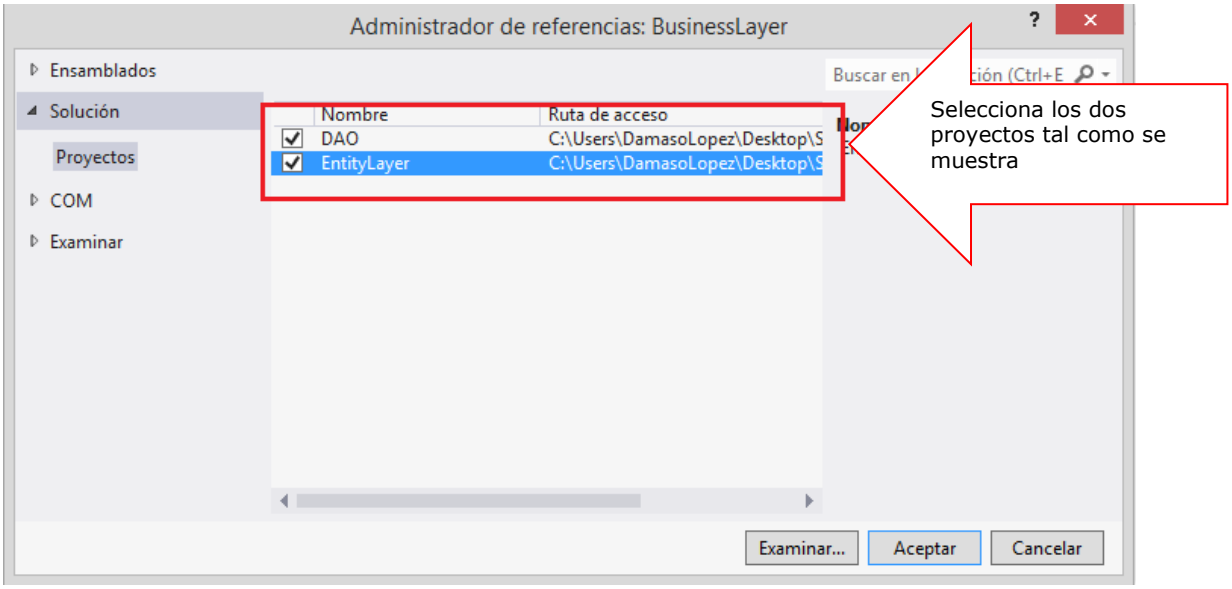


Selecciona la plantilla Biblioteca de clases

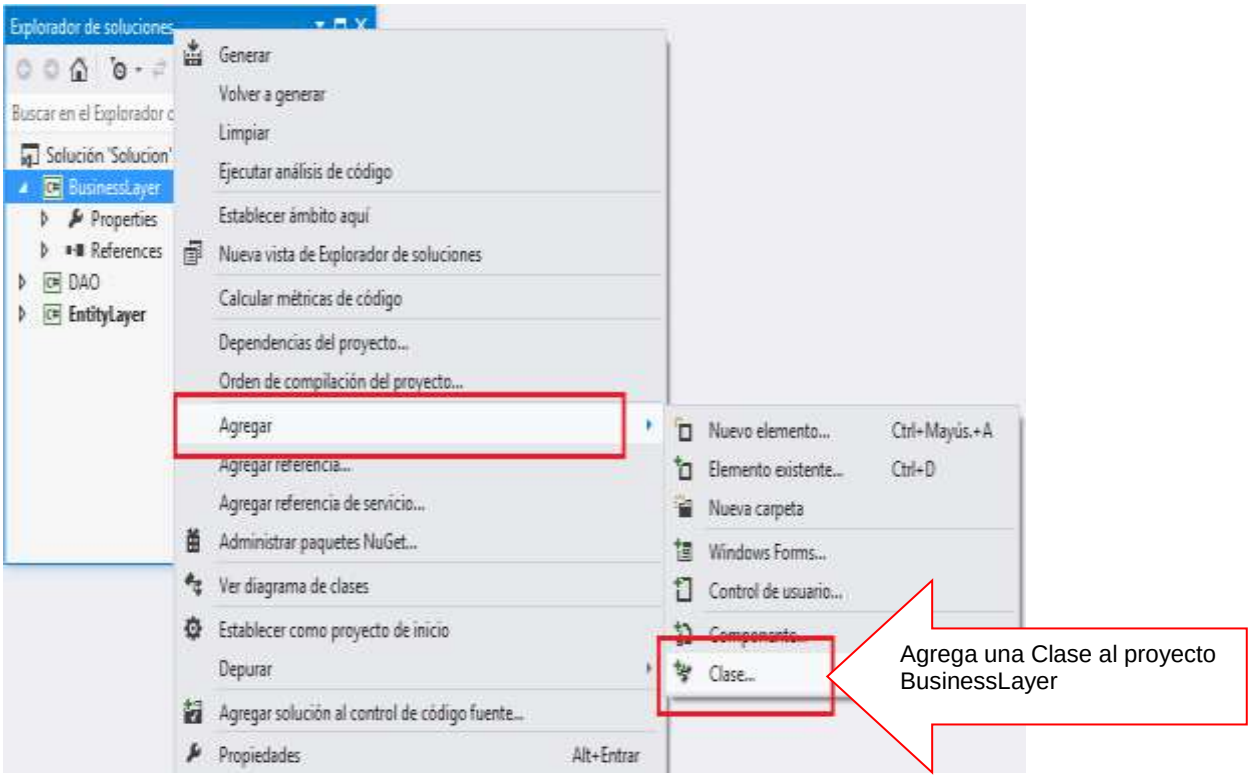
Asigna al proyecto el nombre BusinessLayer

El proyecto BusinessLayer, implementa los métodos definidos en el proyecto DAO; para ello requiere el proyecto **BusinessEntity** y el proyecto DAO.Para hacer una referencia hacia los proyecto, hacer Click derecho en el proyecto **BusinessLayer** y selecciona la opción **Agregar Referencia.**

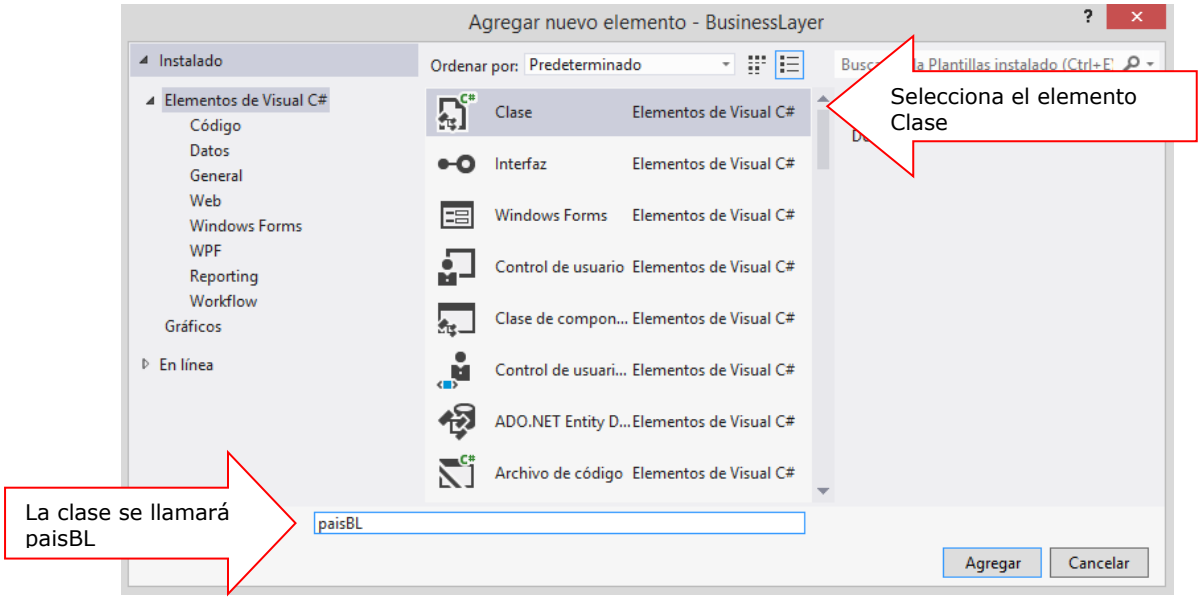
En la ventana Agregar Referencia, seleccione los proyectos: DataLayer y BussinessLayer, tal como se muestra; presione el botón ACEPTAR



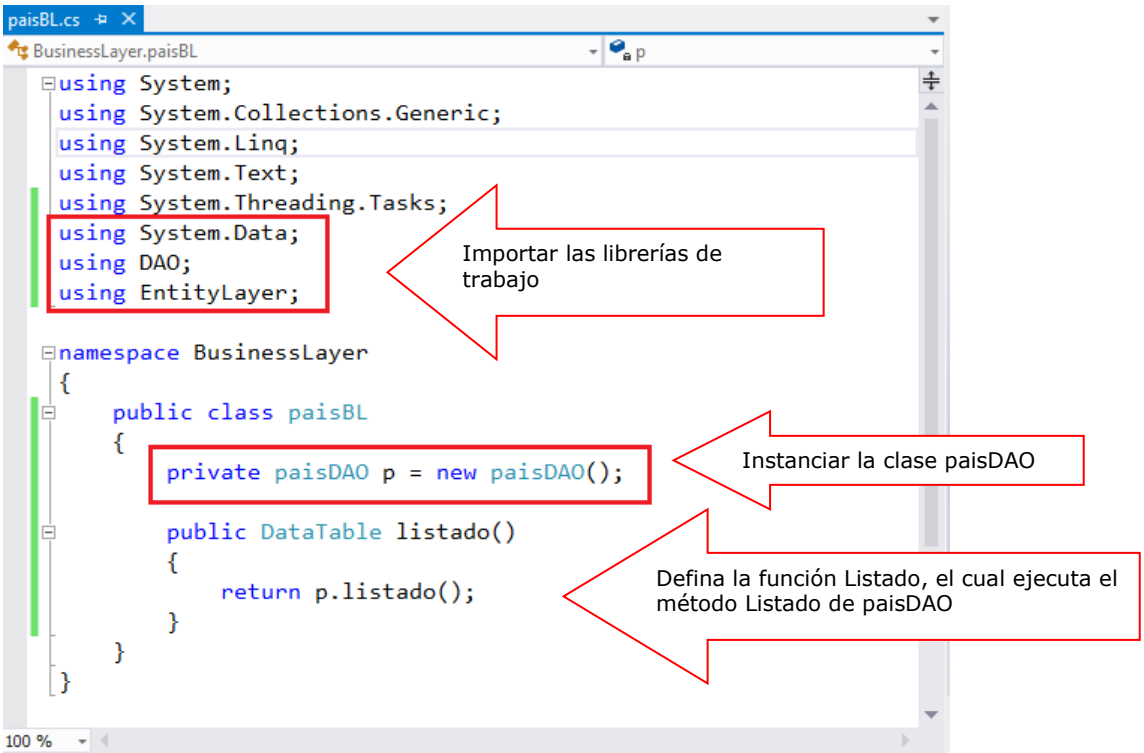
Como siguiente paso, vamos a agregar elementos al proyecto **BusinessLayer** para ejecutar los métodos definidos en el proyectoDAO



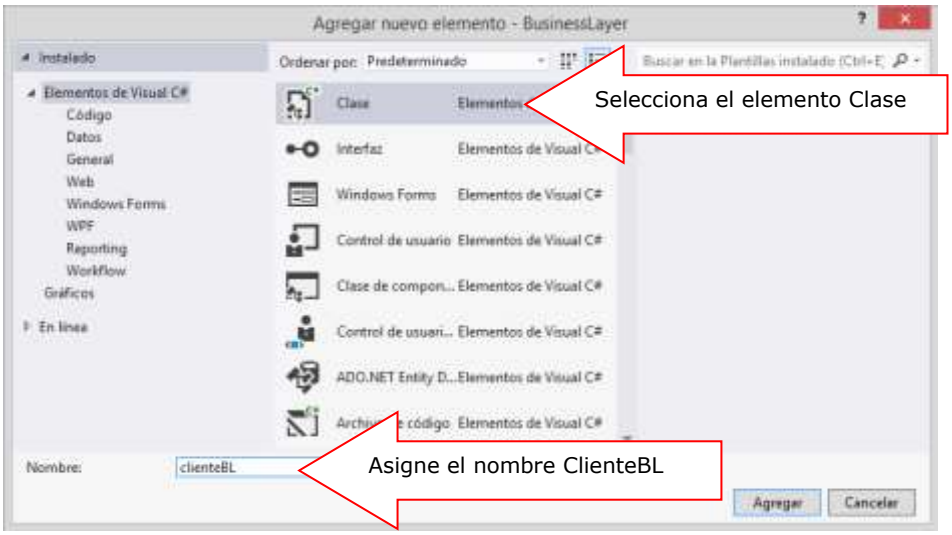
En la ventana AGREGAR ELEMENTO, selecciona el elemento CLASE y asigne un nombre: **paisBL**, a continuación presione el botón AGREGAR



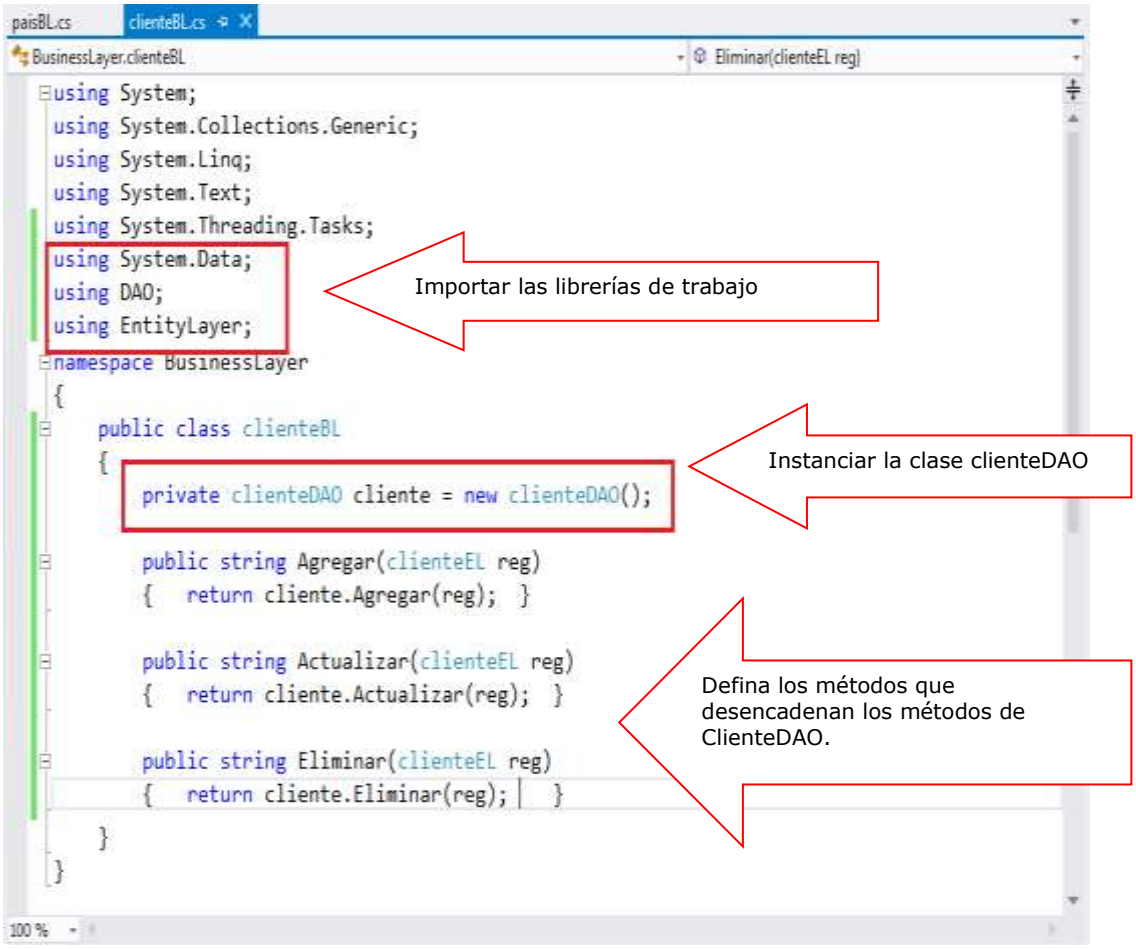
Para implementar la clase **paisBL**, importamos las librerías de trabajo; instanciamos **PaisDAO** y defina la función Listado que ejecuta el método Listado de **PaisDAO**



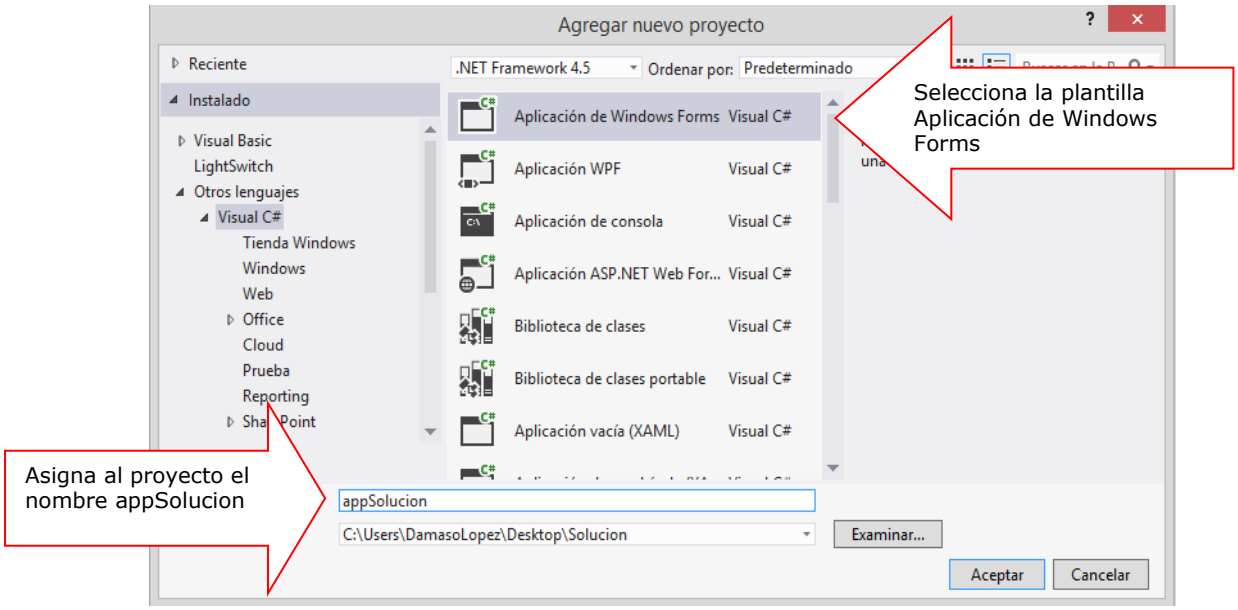
A continuación agrega un nuevo elemento llamado **ClienteBL**, tal como se muestra



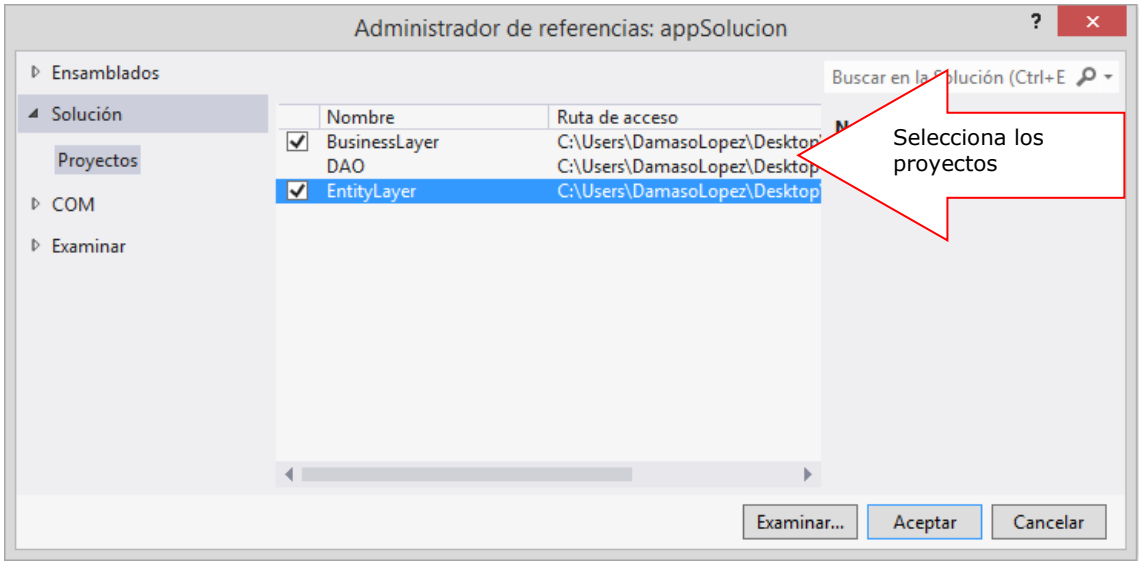
Implementar la clase llamada **ClienteBL**, primero importamos las librerías de trabajo. Defina los métodos los cuales desencadenan los métodos de **ClienteDAO**



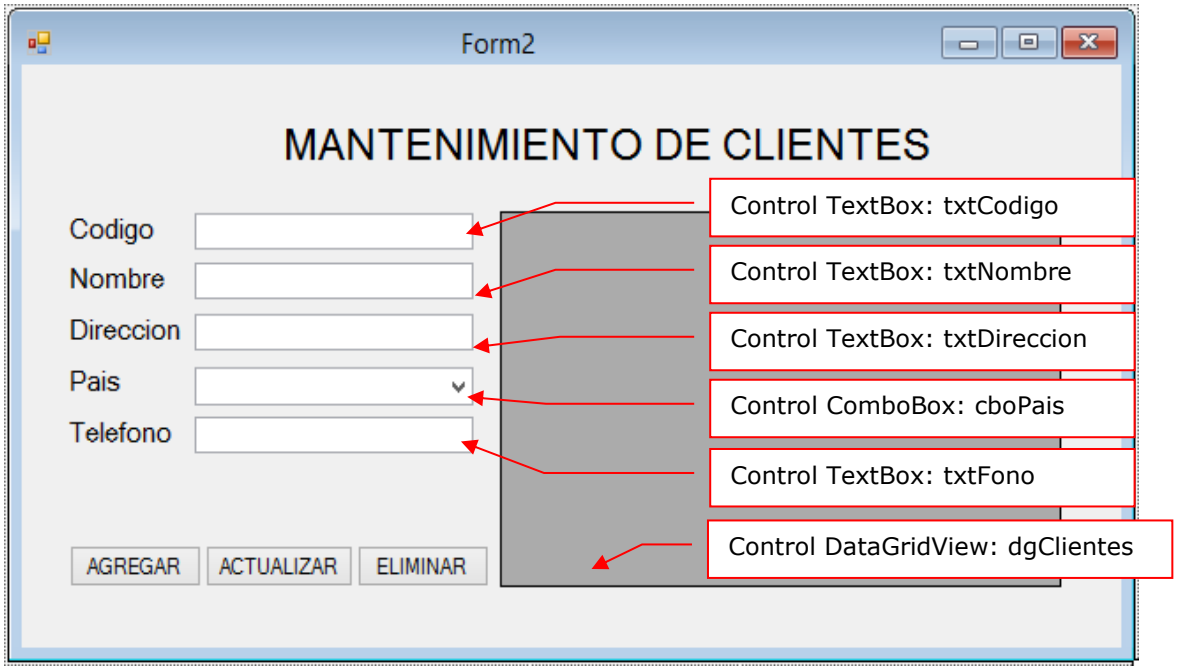
Definida el proyecto **BusinessLayer**, a continuación proceda a AGREGAR UN NUEVO PROYECTO a la solución, tal como se muestra



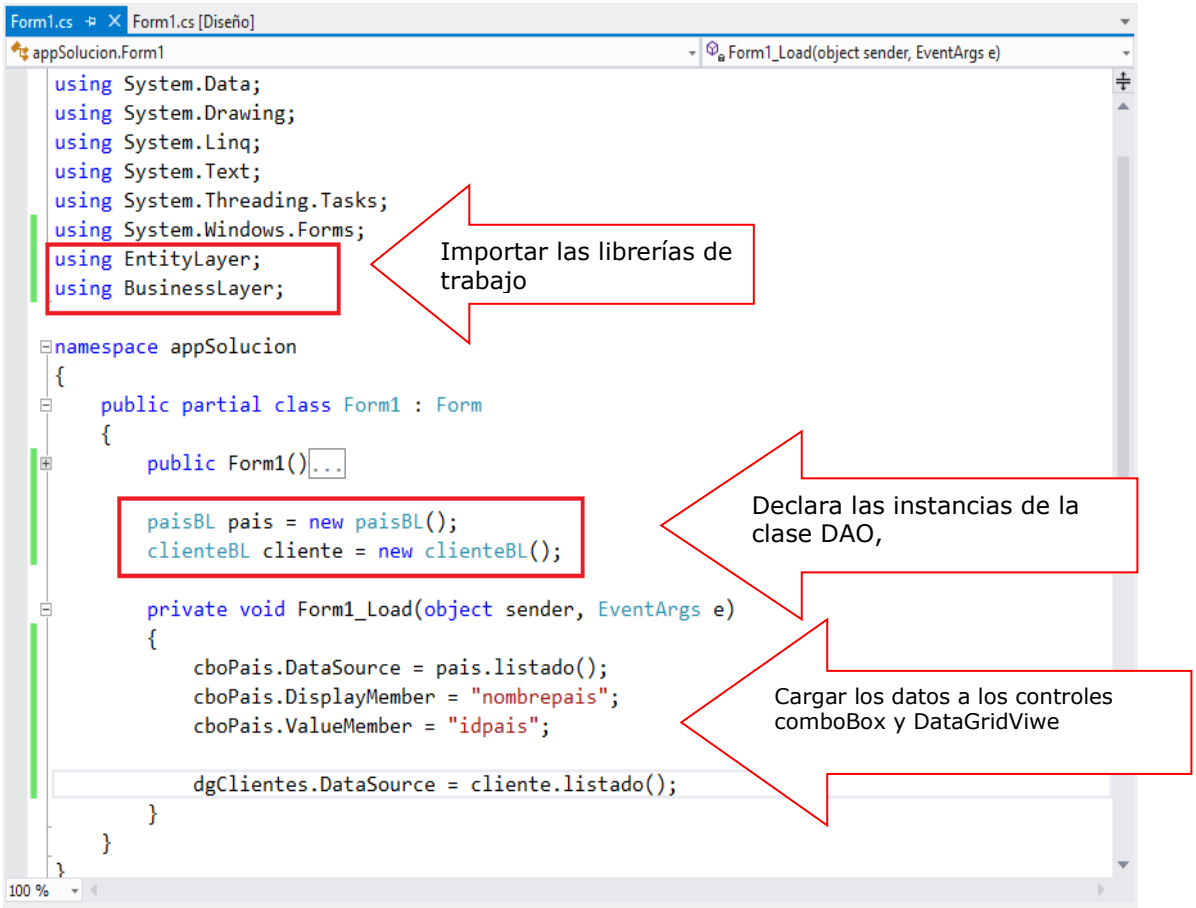
Antes de iniciar, debemos establecer al proyecto **appSolucion** como proyecto de inicio, esto es por ser la interfaz de usuario, es decir, la capa de presentación del proyecto.Selecciona los proyectos **EntityLayer** y **BusinessLayer**, tal como se muestra



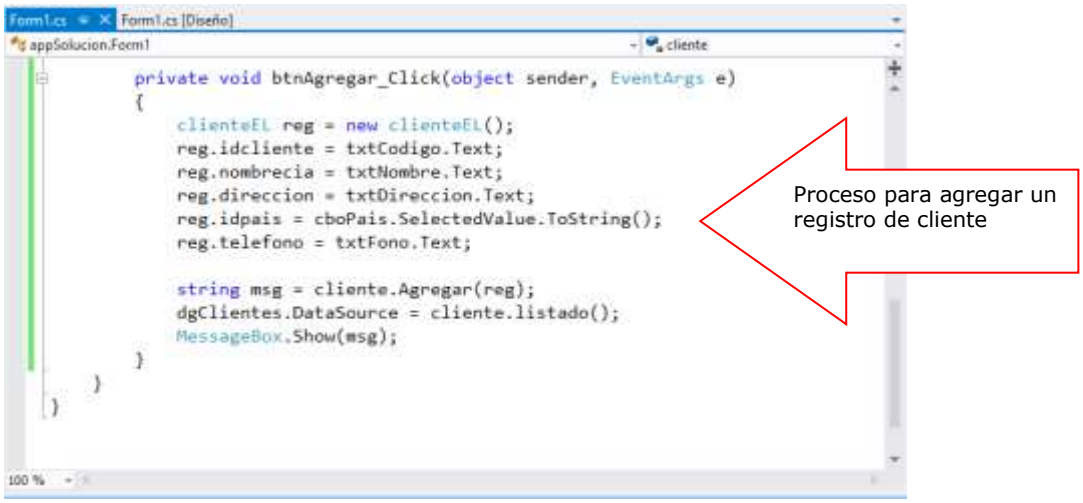
Agrega un Formulario en el proyecto appSolucion. Dibuja la GUI, la como se muestra



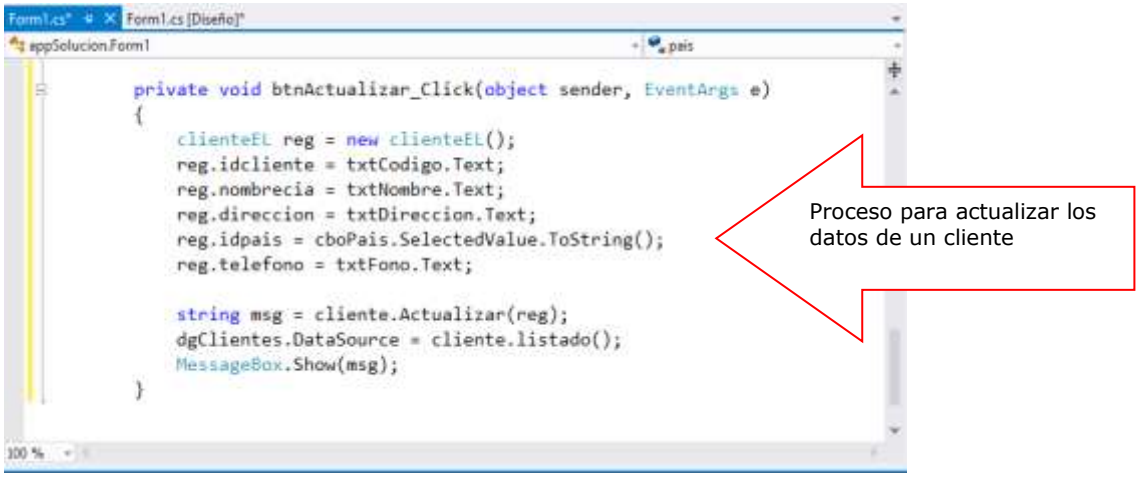
En la ventana de código, importar las librerías: **EntityLayer** y **BusinessLayer**, instanciar las clases de los procesos tal como se muestra



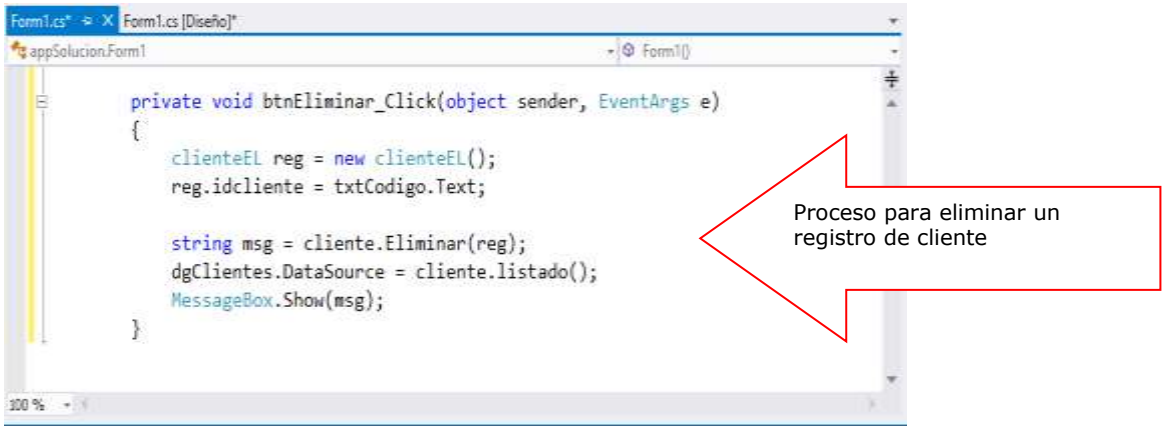
A continuación, programa el evento Click del botón AGREGAR, el cual tiene como objetivo ejecutar el método AGREGA del objeto cliente (clienteBL)



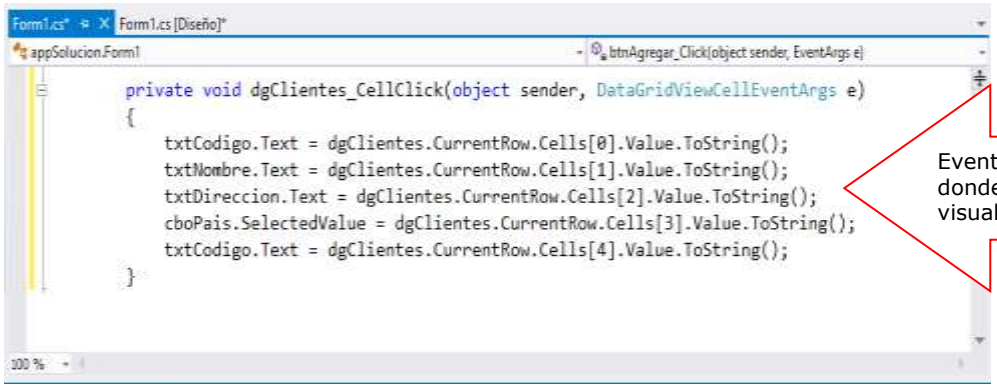
A continuación, programa el evento Click del botón ACTUALIZAR, el cual tiene como objetivo ejecutar el método ACTUALIZAR del objeto cliente (clienteBL)



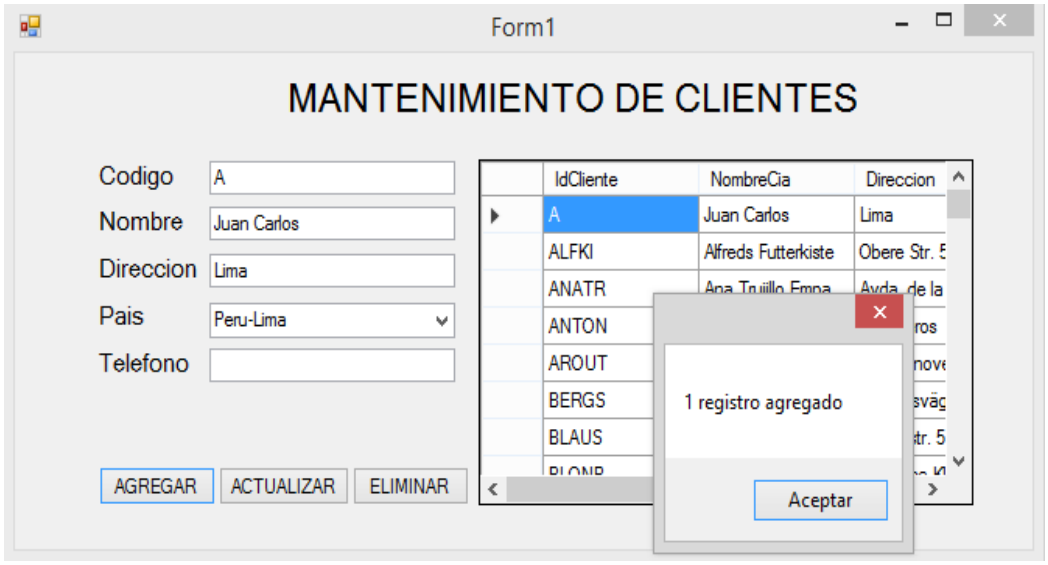
A continuación, programa el evento Click del botón ELIMINAR, el cual tiene como objetivo ejecutar el método ELIMINAR del objeto cliente (ClienteBL)



Programa el evento CellClick del control DataGridView1, donde al seleccionar una fila del control (CurrentRow), visualice los datos en los controles del Formulario



Ejecute el proyecto y ejecuta las operaciones de Agregar, Modificar y eliminar los registro de un cliente.



LABORATORIO 6.2

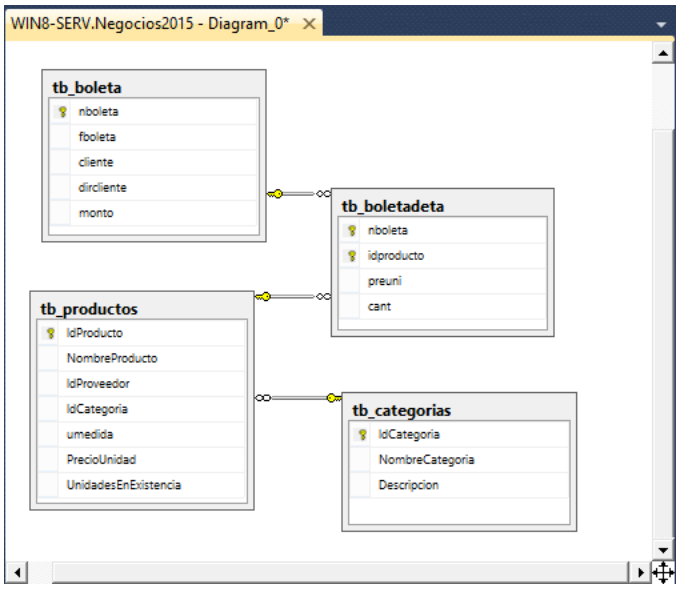
Se desea implementar una solución que permita realizar registrar una boleta de venta utilizando la programación de capas

Se pide:

1. Diseño e implementación de cada una de las capas
2. Diseño de la capa de presentación, formulario para ejecutar los procesos del negocio.

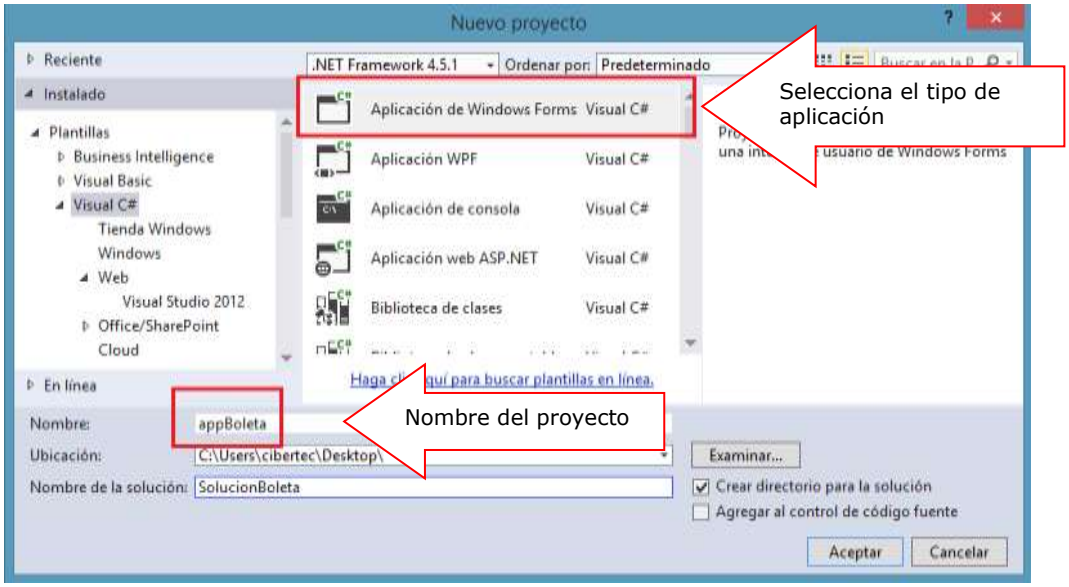
MODELO DE DATOS.

En este proceso trabajaremos con la base de datos Negocios2015. Este es el modelo de datos a utilizar en el proceso

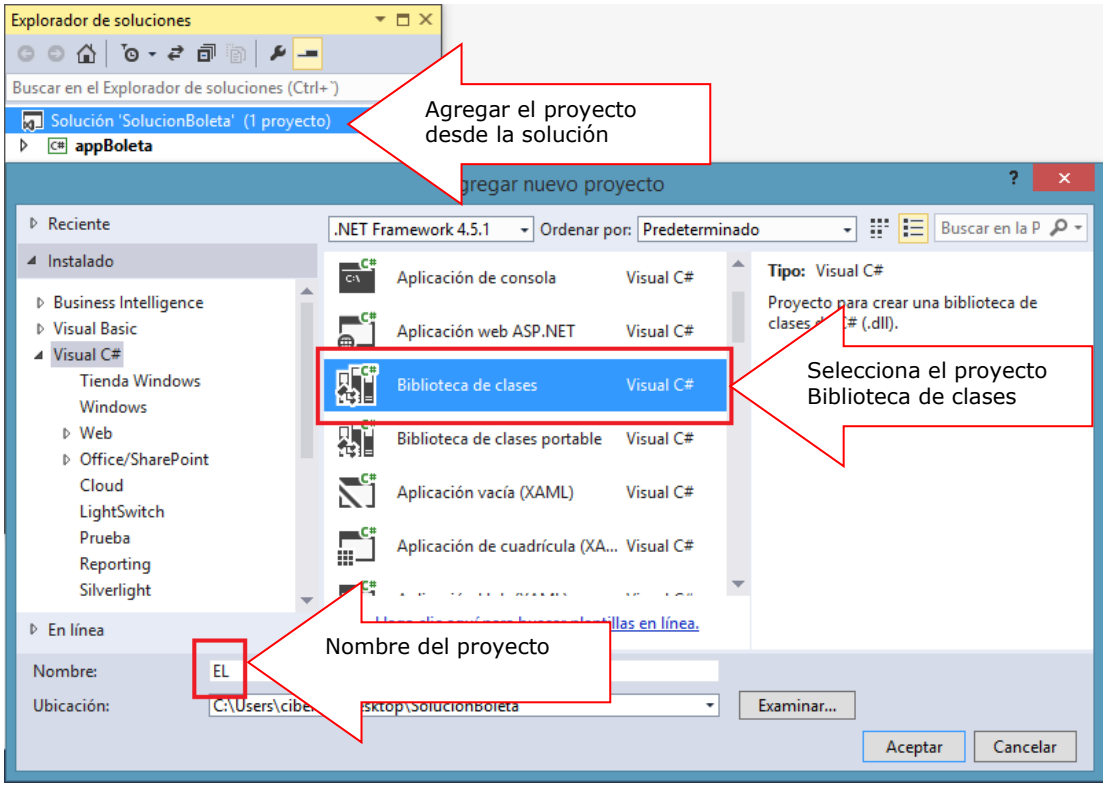


Desarrollando el proyecto

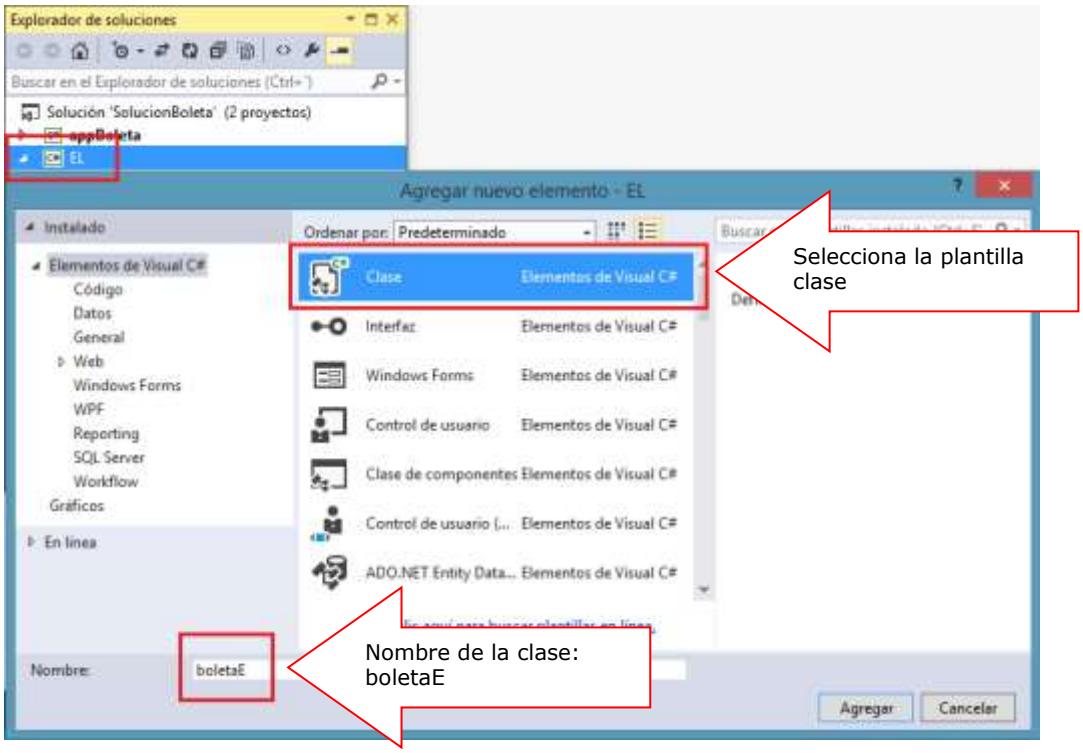
A continuación creamos un nuevo proyecto de tipo Aplicación de Windows Forms en C#, tal como se muestra. El nombre del proyecto es appBoleta



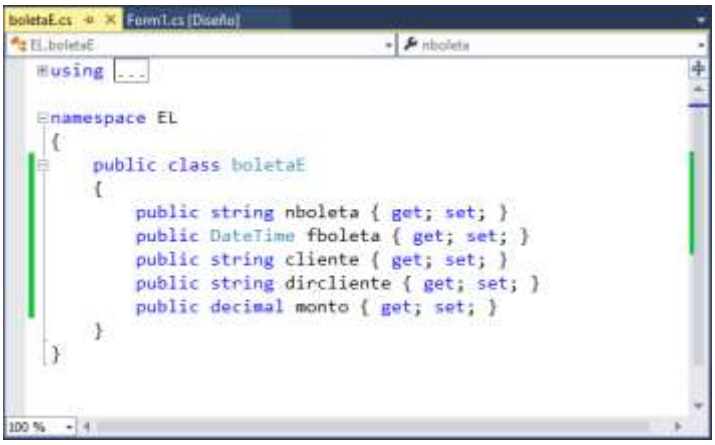
Para iniciar con el proyecto de capas, primero agregamos una biblioteca de clases llamada EL (Capa de Entidades) dentro de la solución, tal como se muestra.



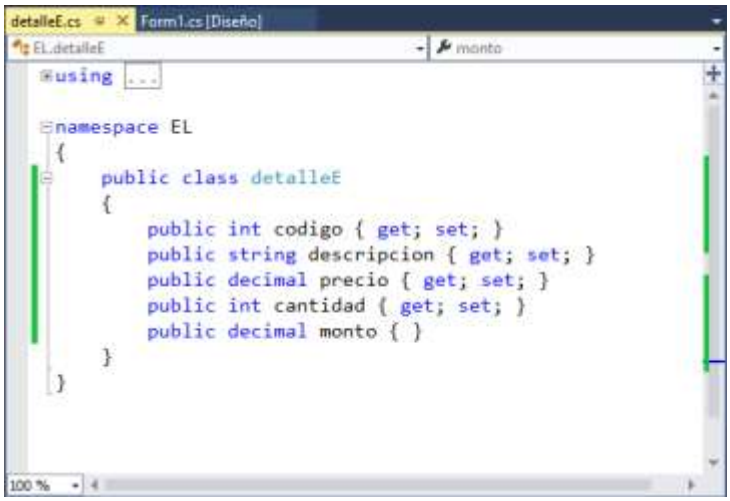
A continuación agregamos las clases a la solución EL (Capa de Entidades)



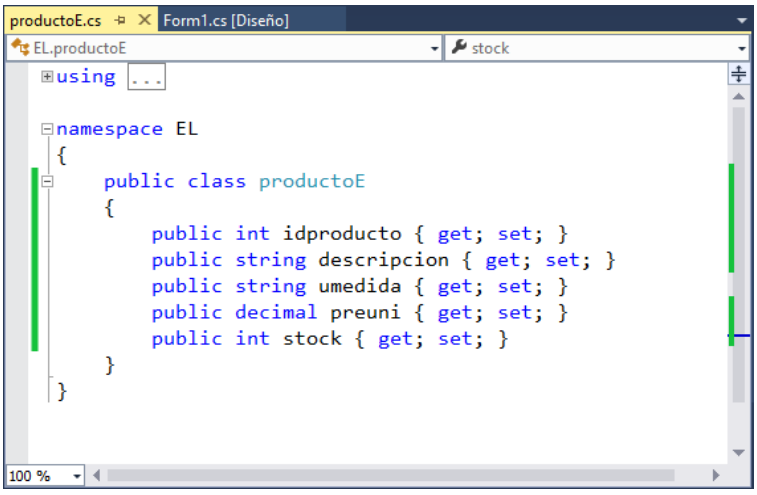
Defina la estructura de la clase boletaE, tal como se muestra



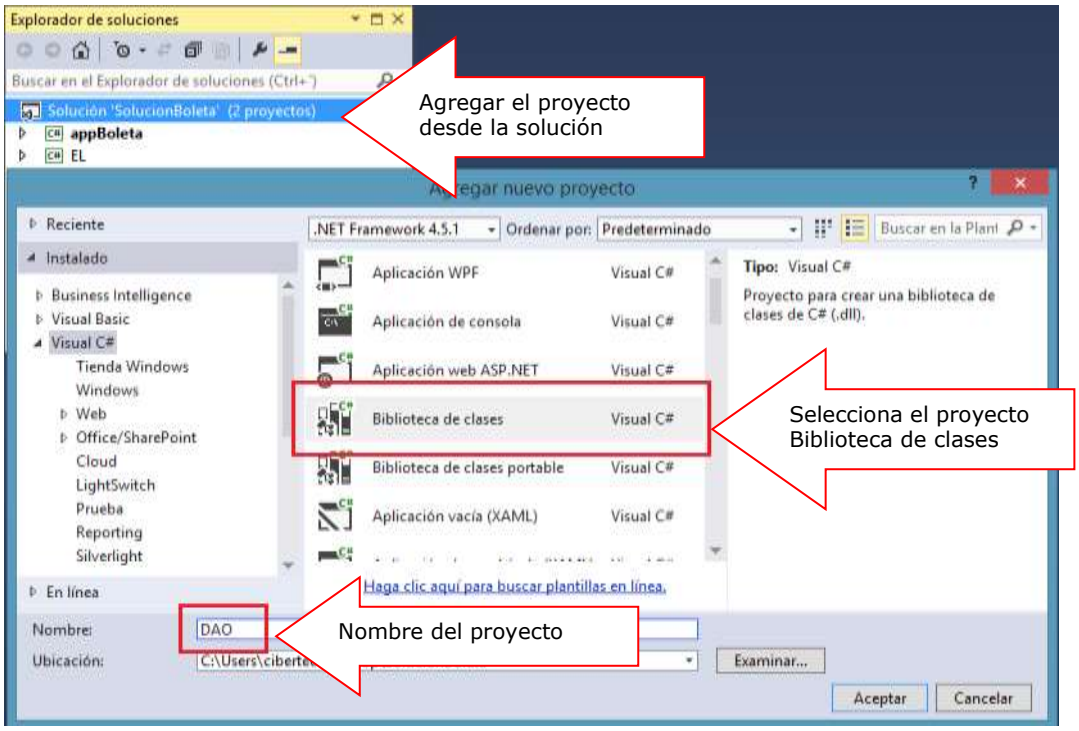
Luego agregamos la clase detalleE, y definimos su estructura, tal como se muestra



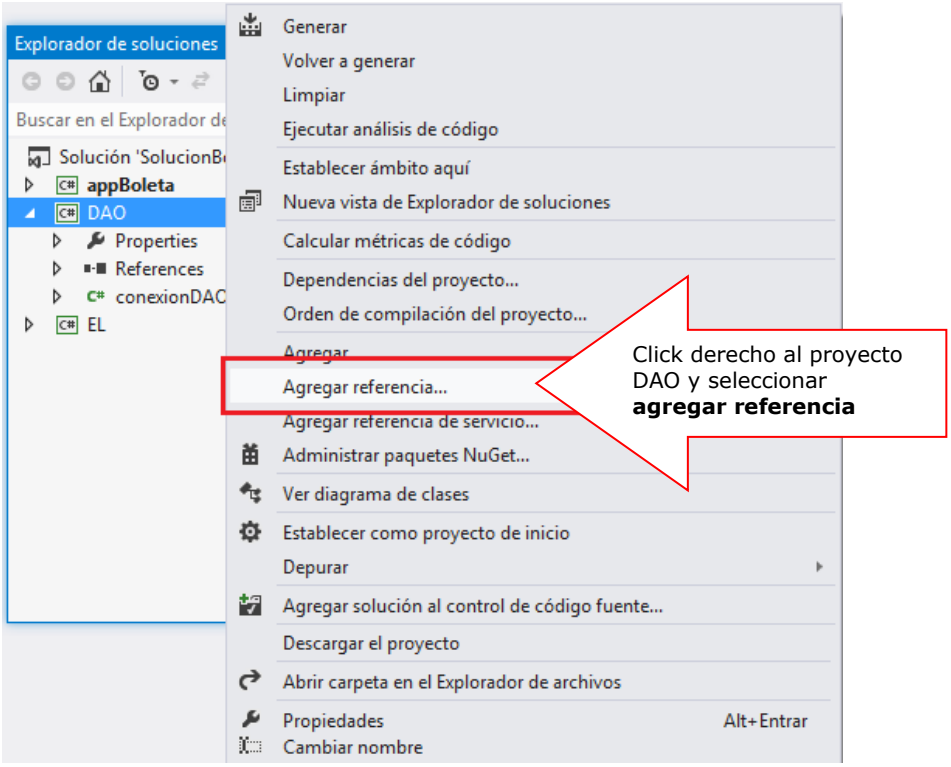
Y por ultimo, agregamos la clase productoE y definimos su estructura, tal como se muestra.



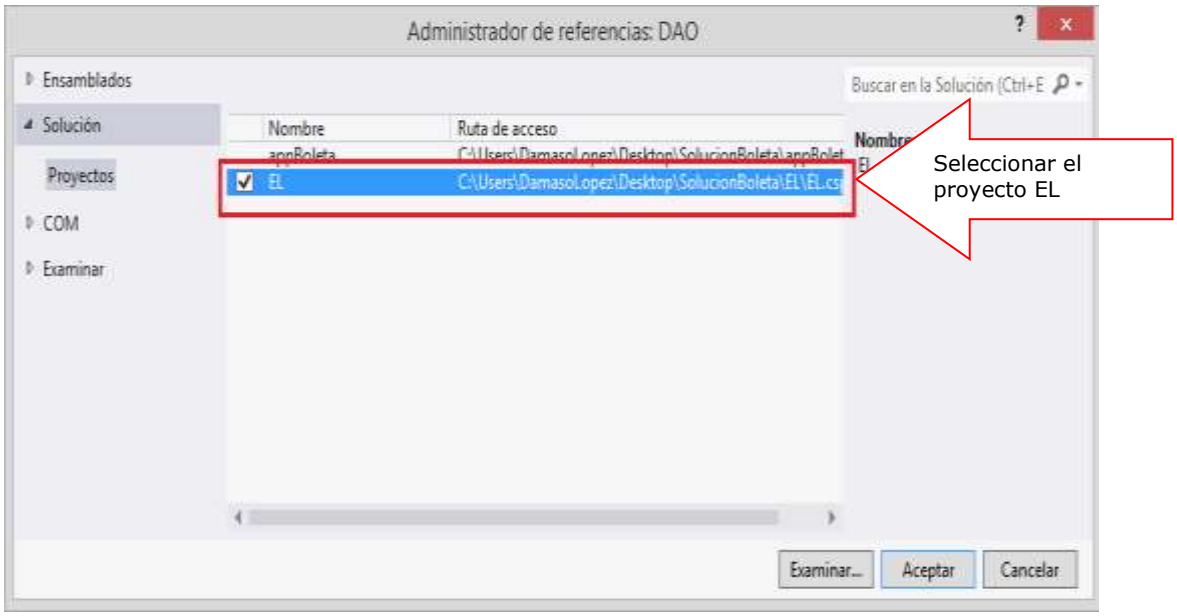
Agregamos, a continuación, la biblioteca de clases llamada DAO (Data Access Object) dentro de la solución, tal como se muestra



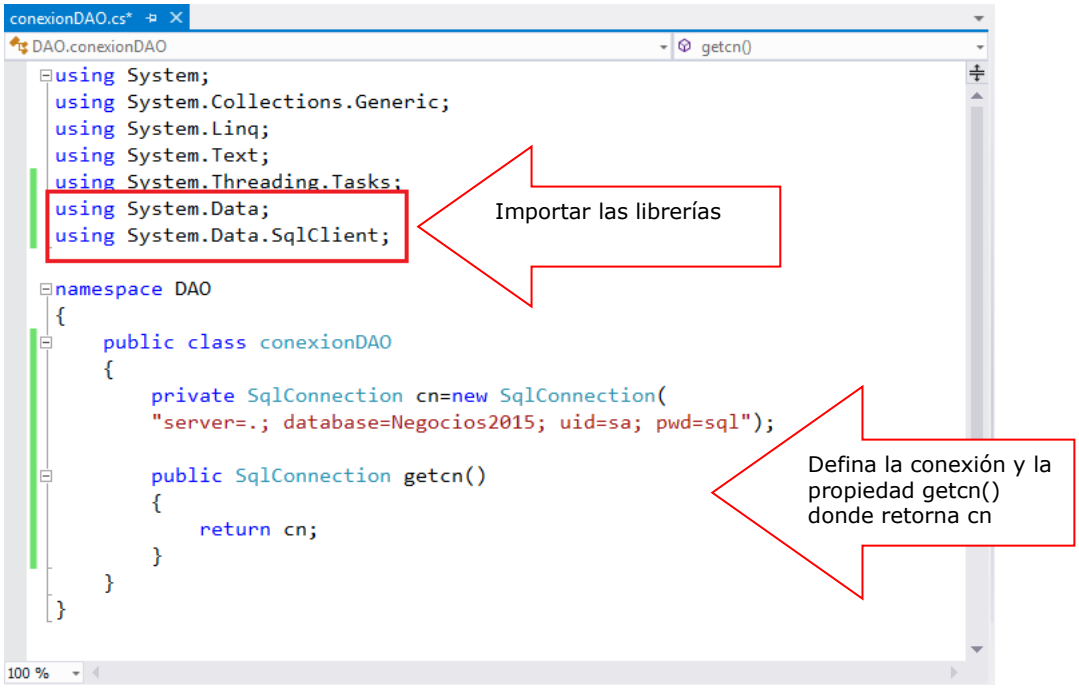
A continuación, en el proyecto DAO agregamos una referencia al proyecto EL, para trabajar con las clases administradas por dicho proyecto.



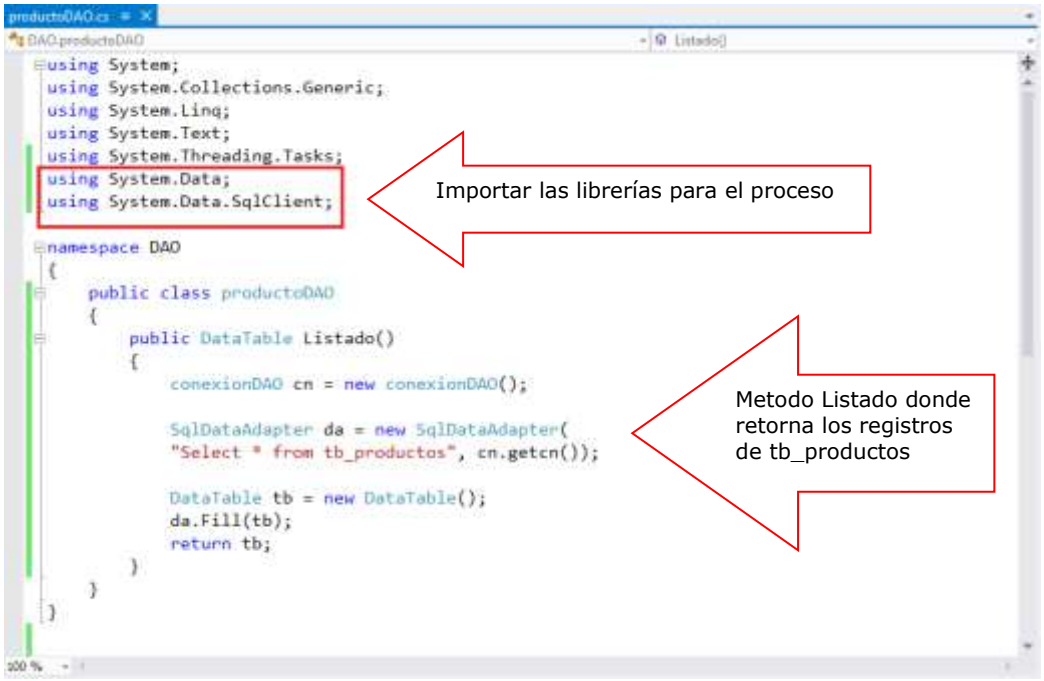
Selecciona el proyecto EL desde la ventana Administrador de referencias DAO, tal como se muestra



A continuación, vamos a trabajar con las clases del proyecto.
En el proyecto DAO, agrega una clase llamada conexionDAO. Codifica la clase tal como se muestra.

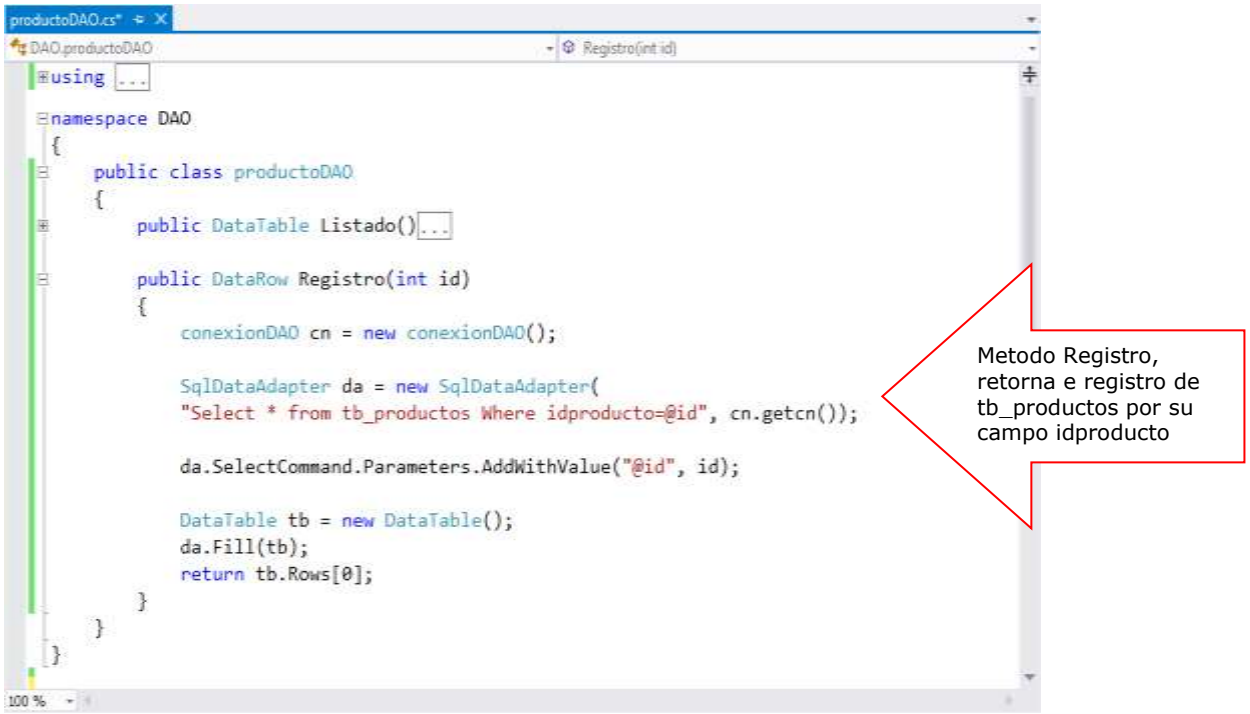


A continuación agrega la clase productoDAO, importar las librerías e implementa el metodo Listado donde retorna los registros de la tabla tb_productos.



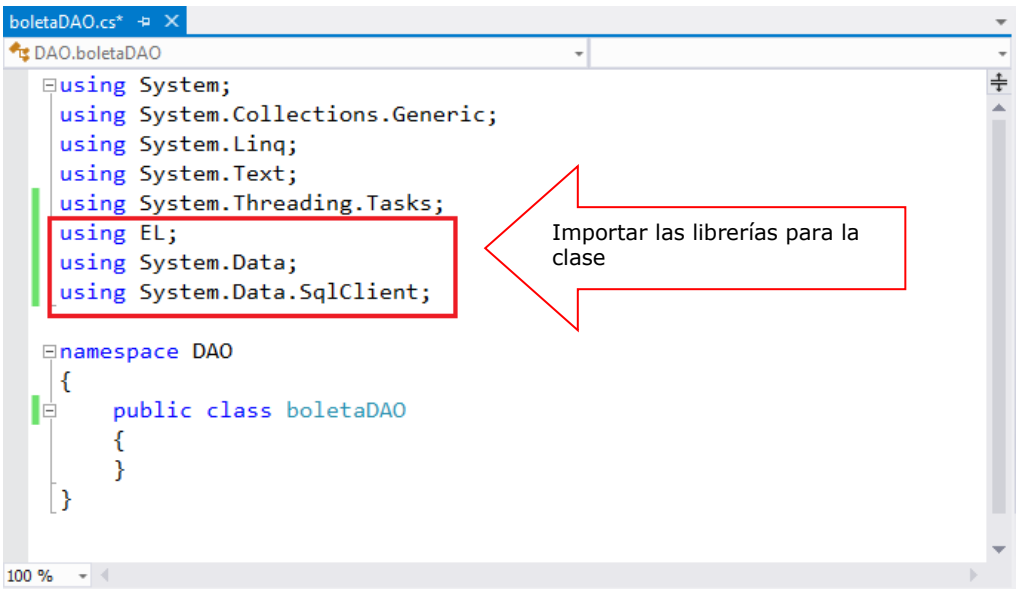

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;

namespace DAO
{
    public class productoDAO
    {
        public DataTable Listado()
        {
            conexionDAO cn = new conexionDAO();

            SqlDataAdapter da = new SqlDataAdapter(
            "Select * from tb_productos", cn.getcn());

            DataTable tb = new DataTable();
            da.Fill(tb);
            return tb;
        }
    }
}
```

En la misma clase defina el metodo Registro, el cual retorna el registro del producto seleccionado por su campo idproducto

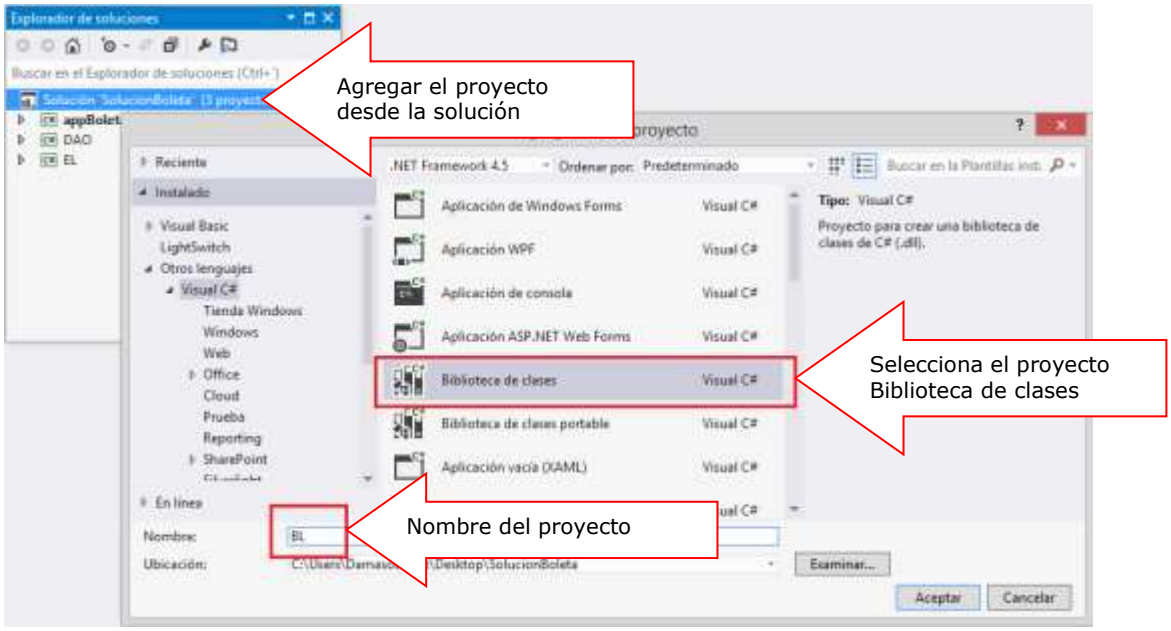



```
using ...

namespace DAO
{
    public class productoDAO
    {
        public DataTable Listado()...

        public DataRow Registro(int id)
        {
            conexionDAO cn = new conexionDAO();

            SqlDataAdapter da = new SqlDataAdapter(
            "Select * from tb_productos Where idproducto=@id", cn.getcn());

            da.SelectCommand.Parameters.AddWithValue("@id", id);

            DataTable tb = new DataTable();
            da.Fill(tb);
            return tb.Rows[0];
        }
    }
}
```

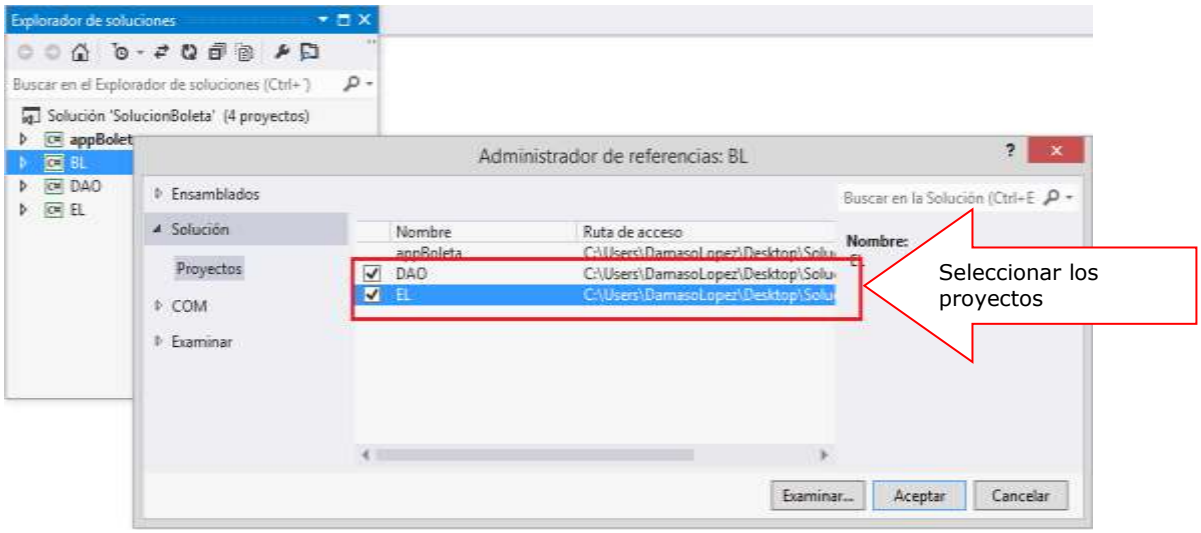
A continuación agregamos al proyecto DAO la clase boletaDAO. Importar las librerías tal como se muestra.



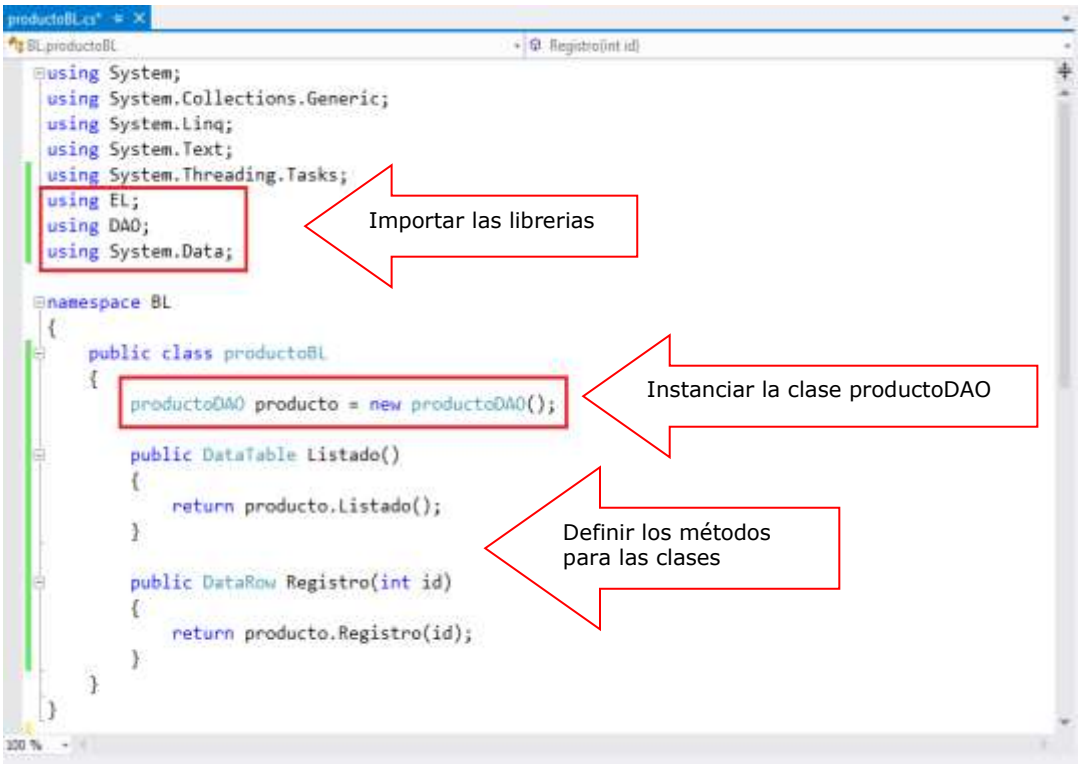
A continuación agregamos a la solución el proyecto BL (capa de Negocio), tal como se muestra.



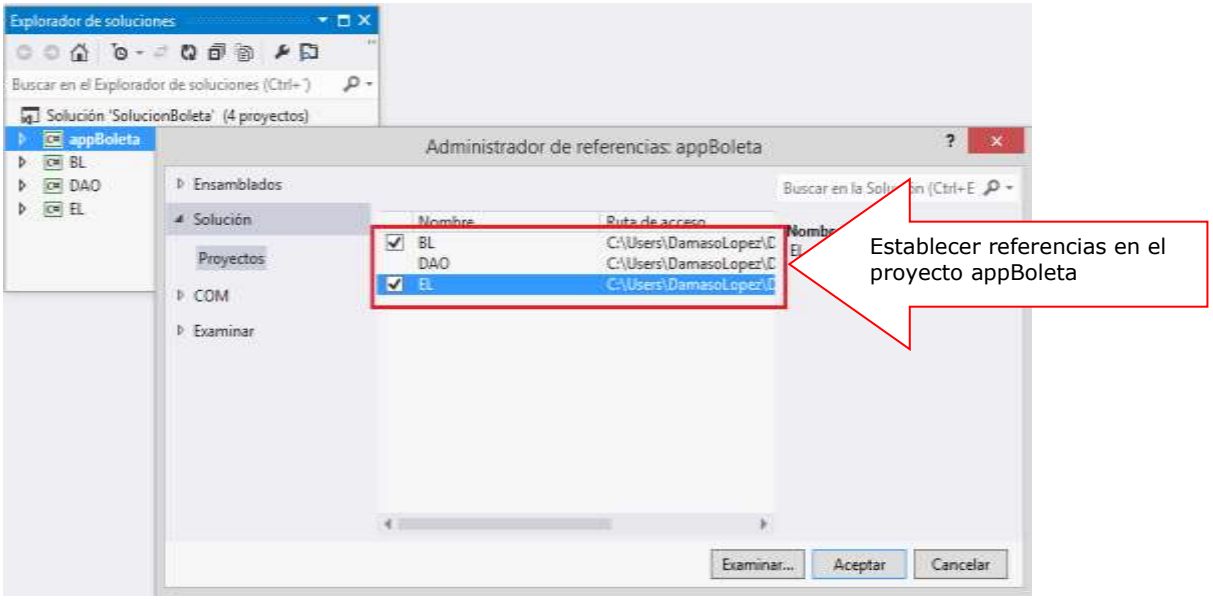
A continuación agregamos las referencias al proyecto BL: click derecho al proyecto y selecciona la opción Agregar Referencia.
En el administrador de referencias BL, selecciona los proyectos DAO y EL, luego presiona Aceptar.



En el proyecto BL agrega la clase productoBL. Importar las librerías, instanciar la clase productoDAO y defina los métodos Listado() y Registro().



En el proyecto appBoleta, primero hacer una referencia hacia los proyectos BL y EL, tal como se muestra.



Resumen

- La programación por capas es una arquitectura cliente-servidor en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.
- La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación sería el modelo de interconexión de sistemas abiertos.
- En la arquitectura en 3 niveles, existe un nivel intermediario. Esto significa que la arquitectura generalmente está compartida por: Un cliente, es decir, el equipo que solicita los recursos, equipado con una interfaz de usuario (generalmente un navegador Web) para la presentación.
- En la arquitectura en 3 niveles, las aplicaciones al nivel del servidor son descentralizadas de uno a otro, es decir, cada servidor se especializa en una determinada tarea, (por ejemplo: servidor web/servidor de bases de datos). La arquitectura en 3 niveles permite: Un mayor grado de flexibilidad Mayor seguridad, ya que la seguridad se puede definir independientemente para cada servicio y en cada nivel Mejor rendimiento, ya que las tareas se comparten entre servidores
- La Capa de presentación es la que ve el usuario (también se la denomina "capa de usuario"), es la representación del sistema al usuario. Esta capa permite mostrar la información; y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato).
- La Capa de negocio es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse
- La Capa de datos es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  http://jtentor.com.ar/post/Arquitectura-de-N-Capas-y-N-Niveles.aspx

  http://iutll-abdd.blogspot.com/2012/05/arquitectura-de-n-capas.html

  http://ronaldcharca.wordpress.com/arquitectura-de-n-capas-y-n-niveles/

UNIDAD

3

CONSUMIENDO Y EXPORTANDO DATOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno implementa servicios de datos desde una aplicación Windows Communication Foundation para consultar y actualizar datos, y exporta los datos de un origen de datos a un archivo de Excel.

Temario

1. **Tema 7: Manejo de servicios en WCF (4 horas)**
   1.1. Definiendo un servicio de datos desde un origen de datos
   1.2. Invocar el servicio en una aplicación Windows Form
   1.3. Realizando operaciones de consulta y actualización de datos

ACTIVIDADES PROPUESTAS

- Los alumnos implementan un servicio para realizar consulta de datos desde un origen de datos remotos.
- Los alumnos crear aplicaciones para consumir servicios de datos remotos.
- Los alumnos desarrollan los laboratorios de la semana.

9. MANEJO DE SERVICIOS EN WCF

Windows Communication Foundation (WCF) es un marco de trabajo para la creación de aplicaciones orientadas a servicios. Con WCF, es posible enviar datos como mensajes asincrónicos de un extremo de servicio a otro. Un extremo de servicio puede formar parte de un servicio disponible continuamente hospedado por IIS, o puede ser un servicio hospedado en una aplicación. Un extremo puede ser un cliente de un servicio que solicita datos de un extremo de servicio. Los mensajes pueden ser tan simples como un carácter o una palabra que se envía como XML, o tan complejos como una secuencia de datos binarios.

A continuación se indican unos cuantos escenarios de ejemplo:

- Un servicio seguro para procesar transacciones comerciales.
- Un servicio que proporciona datos actualizados a otras personas, como un informe sobre tráfico u otro servicio de supervisión.
- Un servicio de chat que permite a dos personas comunicarse o intercambiar datos en tiempo real.
- Una aplicación de panel que sondea los datos de uno o varios servicios y los muestra en una presentación lógica.
- Exponer un flujo de trabajo implementado utilizando Windows Workflow Foundation como un servicio WCF.
- Una aplicación de Silverlight para sondear un servicio en busca de las fuentes de datos más recientes.

Si bien era posible crear tales aplicaciones antes de que existiera WCF, con WCF el desarrollo de extremos resulta más sencillo que nunca. En resumen, WCF se ha diseñado para ofrecer un enfoque manejable para la creación de servicios web y clientes de servicios web.

ARQUITECTURA DE WCF

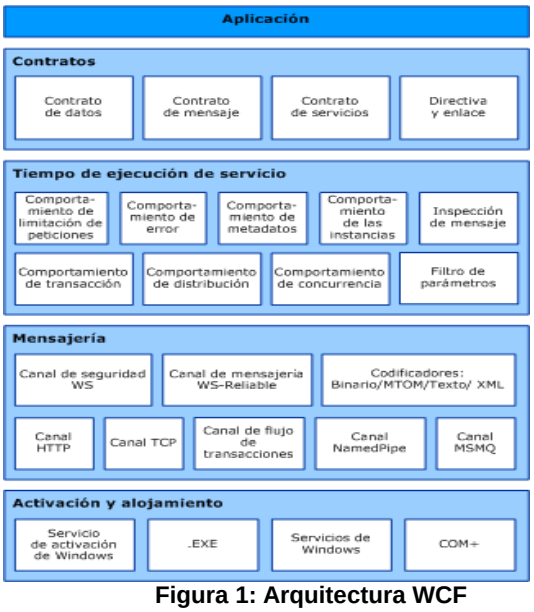


Figura 1: Arquitectura WCF
Referencia: http://msdn.microsoft.com/es-es/library/ms733128(v=vs.110).aspx

Contratos y descripciones

Los contratos definen varios aspectos del sistema de mensajes. El contrato de datos describe cada parámetro que constituye cada mensaje que un servicio puede crear o utilizar. Los documentos de Lenguaje de definición de esquemas XML (XSD) definen los parámetros de mensaje, permitiendo a cualquier sistema que entienda XML procesar los documentos. El contrato del mensaje define partes específicas del mensaje utilizando los protocolos SOAP y permite el control más fino sobre las partes del mensaje, cuando la interoperabilidad exige tal precisión. El contrato de servicios especifica las firmas de método actuales del servicio y se distribuye como una interfaz en uno de los lenguajes de programación compatibles, como Visual Basic o Visual C#.

Las directivas y enlaces estipulan las condiciones exigidas para comunicarse con un servicio. Por ejemplo, el enlace debe especificar (como mínimo) el transporte utilizado (por ejemplo, HTTP o TCP) y una codificación. Las directivas incluyen los requisitos de seguridad y otras condiciones que se deben cumplir para comunicarse con un servicio.

Tiempo de ejecución de servicio

La capa del tiempo de ejecución del servicio contiene los comportamientos que solo se producen durante la operación actual del servicio, es decir, los comportamientos en tiempo de ejecución del servicio. La limitación de peticiones controla cuántos mensajes se procesan que puede variar si la demanda para el servicio crece a un límite preestablecido. Un comportamiento de error especifica lo que sucede cuando se produce un error interno en el servicio, por ejemplo, controlando qué información se comunica al cliente. (Demasiada información puede dar ventaja a un usuario malintencionado para organizar un ataque.)

El comportamiento de los metadatos rige cómo y si los metadatos se ponen a disposición del mundo externo. El comportamiento de la instancia especifica cuántas instancias del servicio se pueden ejecutar (por ejemplo, un singleton especifica solo una instancia para procesar todos los mensajes). El comportamiento de la transacción habilita la recuperación de operaciones de transacción si se produce un error. El comportamiento de la expedición es el control de cómo la infraestructura WCF procesa un mensaje.

La extensibilidad habilita la personalización de procesos en tiempo de ejecución. Por ejemplo, la inspección del mensaje es la facilidad para inspeccionar partes de un mensaje y la filtración de parámetros permite que se realicen acciones preestablecidas basándose en filtros que actúan en encabezados del mensaje.

Mensajería

La capa de la mensajería se crea de canales. Un canal es un componente que procesa un mensaje de alguna manera, por ejemplo, autenticando un mensaje. Un conjunto de canales también se conoce como una pila de canales. Los canales funcionan en los mensajes y encabezados del mensaje. Esto es diferente de la capa en tiempo de ejecución del servicio, que se ocupa principalmente de procesar el contenido de los cuerpos de los mensajes.

Hay dos tipos de canales: canales de transporte y canales de protocolo.
Los canales de transporte leen y escriben mensajes de la red (o algún otro punto de la comunicación con el mundo externo). Algunos transportes utilizan un codificador para convertir los mensajes (que se representan como conjuntos de información XMLs) hacia y desde la representación de la secuencia de bytes

utilizada por la red. Son ejemplos de transportes HTTP, canalizaciones con nombre, TCP y MSMQ. Son ejemplos de codificaciones XML y binario optimizado.

La capa de la mensajería muestra los posibles formatos y modelos de intercambio de los datos. WS-Security es una implementación de la especificación WS-Security que habilita la seguridad en la capa del mensaje. El canal de mensajería WS-Reliable habilita la garantía de entrega del mensaje. Los codificadores presentan una variedad de codificaciones que se pueden utilizar para satisfacer las necesidades del mensaje. El canal HTTP especifica que el Protocolo de transporte de hipertexto se utiliza para la entrega del mensaje. El canal TCP especifica de manera similar el protocolo TCP. El canal de flujo de transacciones rige los modelos de mensajes de transacción. El canal de la canalización con nombre habilita la comunicación entre procesos. El canal de MSMQ habilita la interoperación con aplicaciones MSMQ.

Alojamiento y activación

En su forma final, un servicio es un programa. Como otros programas, un servicio se debe ejecutar en un ejecutable. Esto se conoce como un servicio con host propio.

Los servicios también se pueden hospedaro ejecutar en un ejecutable administrado por un agente externo, como IIS o Servicio de activación de Windows (WAS). WAS permite activar automáticamente aplicaciones WCF cuando se implementan en un equipo que ejecuta WAS. Los servicios también se pueden ejecutar manualmente como ejecutables (archivos .exe). Un servicio también se puede ejecutar automáticamente como un servicio de Windows. Los componentes COM+ también se pueden hospedar como servicios WCF.

7.1 DEFINIENDO UN SERVICIO DE DATOS DESDE UN ORIGEN

WCF es un tiempo de ejecución y un conjunto de API para la creación de sistemas que envíen mensajes entre servicios y clientes. La misma infraestructura y API se utilizan para crear aplicaciones que se comuniquen con otras aplicaciones en el mismo sistema del equipo o en un sistema que resida en otra compañía y a la que se obtenga acceso a través de Internet.

Mensajería y extremos.

WCF se basa en la noción de comunicación basada en mensajes, y cualquier cosa que se pueda modelar como un mensaje (por ejemplo, una solicitud HTTP o un transporte de cola de mensajes (también conocido como MSMQ)) se puede representar de manera uniforme en el modelo de programación. Esto habilita una API unificada en todos los mecanismos de transporte diferentes.

El modelo distingue entre clientes, que son aplicaciones que inician la comunicación y servicios, que son aplicaciones que esperan a que los clientes se comuniquen con ellos y respondan a esa comunicación. Una única aplicación puede actuar como cliente y como servicio. Para obtener ejemplos, vea Servicios dúplex y Conexión de redes punto a punto.

Los mensajes se envían entre extremos. Los extremos son los lugares donde los mensajes se envían o reciben (o ambos), y definen toda la información requerida para el intercambio de mensajes. Un servicio expone uno o más extremos de la

aplicación (y a cero o más extremos de la infraestructura), y el cliente genera un extremo que es compatible con uno de los extremos del servicio.

Un extremo describe de una manera basada en estándar dónde se deberían enviar los mensajes, cómo se deberían enviar y qué aspecto deberían tener los mensajes. Un servicio puede exponer esta información como metadatos que los clientes pueden procesar para generar clientes WCF adecuados y pilasde comunicación.

Protocolos de comunicaciones

Un elemento requerido de la pila de la comunicación es el protocolo de transporte. Los mensajes se pueden enviar a través de intranets e Internet utilizando transportes comunes, como HTTP y TCP. Otros transportes incluidos admiten la comunicación con aplicaciones Message Queuing (MSMQ) y nodos en una malla de redes del mismo nivel. Se pueden agregar más mecanismos de transporte utilizando los puntos de la extensión integrados de WCF.

Otro elemento necesario en la pila de comunicación es la codificación que especifica cómo se da formato a cualquier mensaje determinado. WCF proporciona las codificaciones siguientes:

Codificación de texto, una codificación interoperable.

Codificación Mecanismo de optimización de transmisión de mensajes (MTOM), que es una manera interoperable de enviar eficazmente datos binarios no estructurados a y desde un servicio.

Codificación binaria para una transferencia eficaz.

Se pueden agregar más mecanismos de codificación (por ejemplo, una codificación de compresión) utilizando los puntos de extensión integrados de WCF.

Patrones de mensajes

WCF admite varios patrones de mensajería, incluida la comunicación de solicitud-respuesta unidireccional y dúplex. Los transportes diferentes admiten patrones de mensajería diferentes y, por consiguiente, afectan a los tipos de interacciones que admiten. El tiempo de ejecución y las API de WCF también le ayudan a enviar mensajes de manera segura y fiable.

7.2 INVOCAR EL SERVICIO DESDE UNA APLICACIÓN WINDOWS

El consumo de servicios WCF le habilita para agregar servicios WCF existentes al proceso empresarial. Es posible agregar varios servicios WCF a una única orquestación.

Por medio del Asistente para consumición del Servicio WCF de BizTalk, puede consumir (llamar) a un Servicio WCF desde la orquestación. Este Asistente crea los tipos de mensaje y de puerto necesarios para consumir servicios WCF. Además, el Asistente para consumición del Servicio WCF de BizTalk crea dos archivos de enlace que se pueden importar utilizando el asistente o la herramienta de línea de comandos de desarrollo con el fin de configurar puertos de envío con

el adaptador personalizado de WCF y los adaptadores de WCF proporcionados por el sistema. El asistente permite usar encabezados SOAP con un servicio WCF consumido, cambiar el URI de un servicio WCF consumido y establecer de forma dinámica el WCF para un servicio web consumido. Los adaptadores de envío WCF consumen servicios WCF y las publicaciones de recepción WCF publican servicios WCF.

Entre los adaptadores de envío WCF de BizTalk se incluyen cinco adaptadores de envío físicos que representan los enlaces predefinidos de WCF BasicHttpBinding, WsHttpBinding, NetTcpBinding, NetNamedPipeBinding y NetMsmqBinding. Asimismo, los adaptadores de WCF de BizTalk incorporan los adaptadores personalizados que permiten configurar libremente la información de comportamiento y enlace de WCF en un puerto de envío. Para obtener más información acerca de cómo configurar un puerto de envío y un controlador de envío WCF, vea para los temas de procedimientos para los adaptadores de WCF en Adaptadores de WCF.

7.3 REALIZANDO OPERACIONES DE CONSULTA Y ACTUALIZACION DE DATOS

Cuando se usa la biblioteca de cliente de Servicios de datos de WCF para usar una fuente de Open Data Protocol (OData), la biblioteca traduce las entradas de la fuente en instancias de clases del servicio de datos del cliente. Se realiza el seguimiento de estas clases del servicio de datos mediante el uso de la clase DataServiceContext a la que pertenece la clase DataServiceQuery. El cliente realiza el seguimiento de los cambios en las entidades que se notifican utilizando métodos de la clase DataServiceContext.

Estos métodos permiten al cliente realizar el seguimiento de las entidades agregadas y eliminadas y también de los cambios que se realizan en los valores de propiedad o en las relaciones entre instancias de entidad. Estos cambios se devuelven al servicio de datos como operaciones basadas en REST al llamar al método SaveChanges.

LABORATORIO 7.1

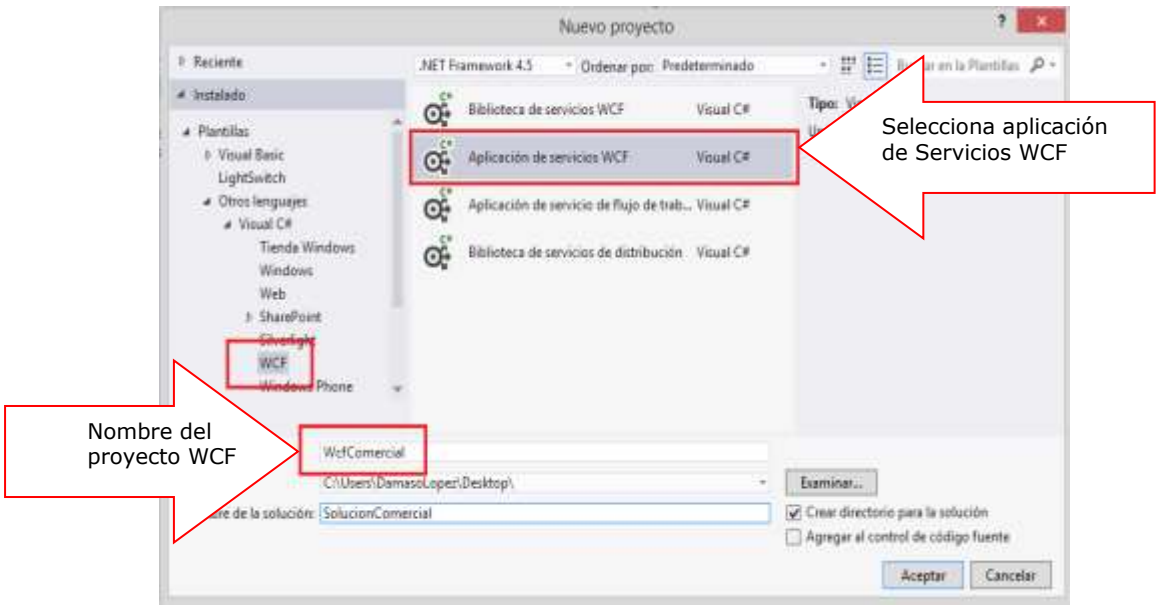
Se desea implementar un servicio de datos, que permita realizar la consulta de Pedidos por un determinado cliente. Implemente procedimientos almacenados para ejecutar dicha consulta

1. DESARROLLANDO LOS PROCEDIMIENTOS ALMACENADOS

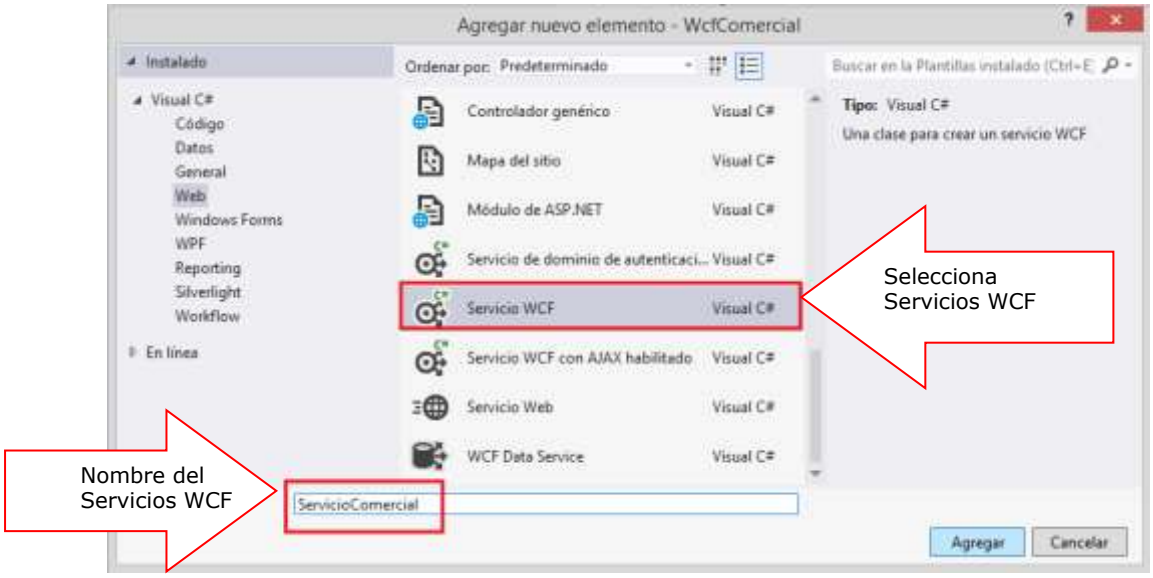
```
use Comercial2012
go

--Procedure donde liste los clientes
Create Procedure usp_Clientes
As
Select idcliente, nombreCia, Direccion, Telefono
from tb_clientes
go

--Procedure donde liste los registros de PedidosCabe
--por un determinado Cliente
Create Procedure usp_PedidosCliente
@cli varchar(5)
As
Select idpedido,FechaPedido,Destinatario,DireccionDestinatario
from tb_pedidoscabe
Where IdCliente=@cli
go
```

2. TRABAJANDO CON WINDOWS COMMUNICATION FOUNDATION

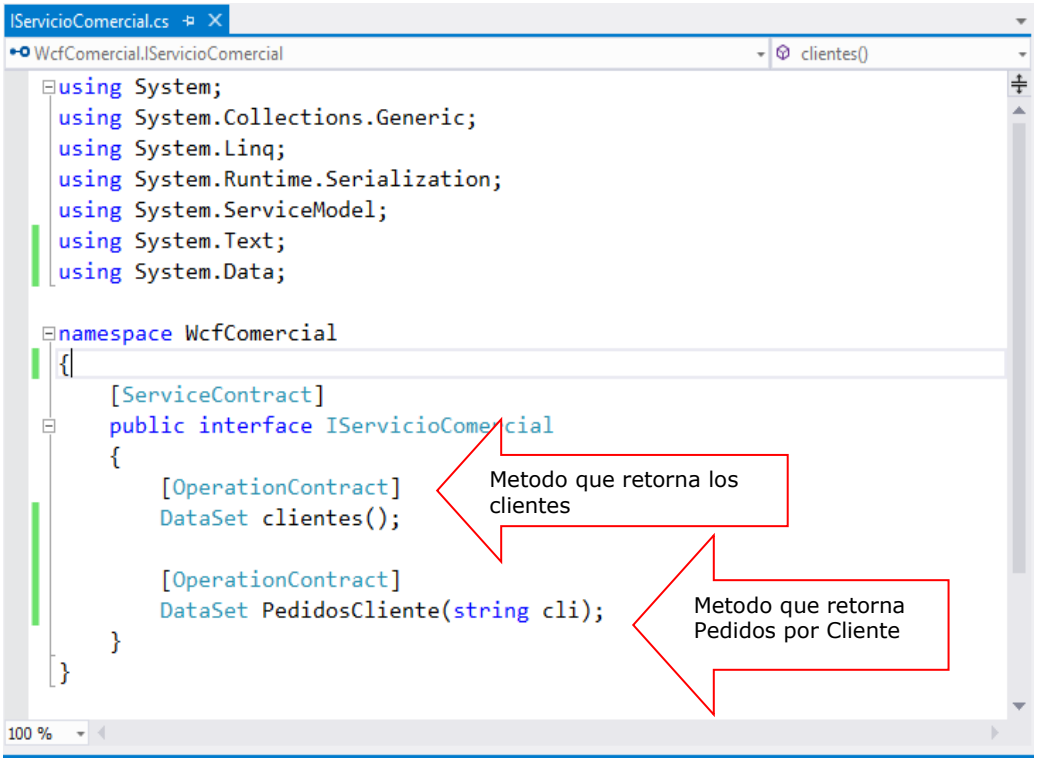
Agregar a la solución (appSolucion) un proyecto Visual C# de tipo WCF para definir un servicio de datos.



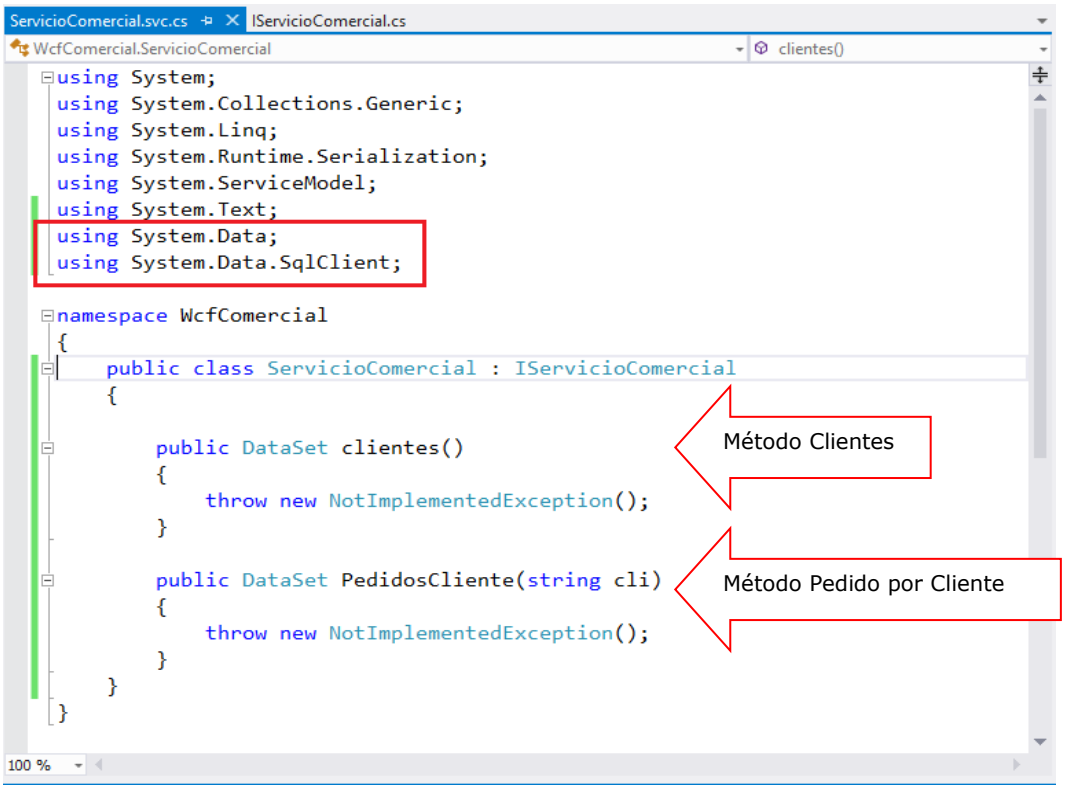
A continuación agregue al proyecto WCF un nuevo elemento: Servicio WCF



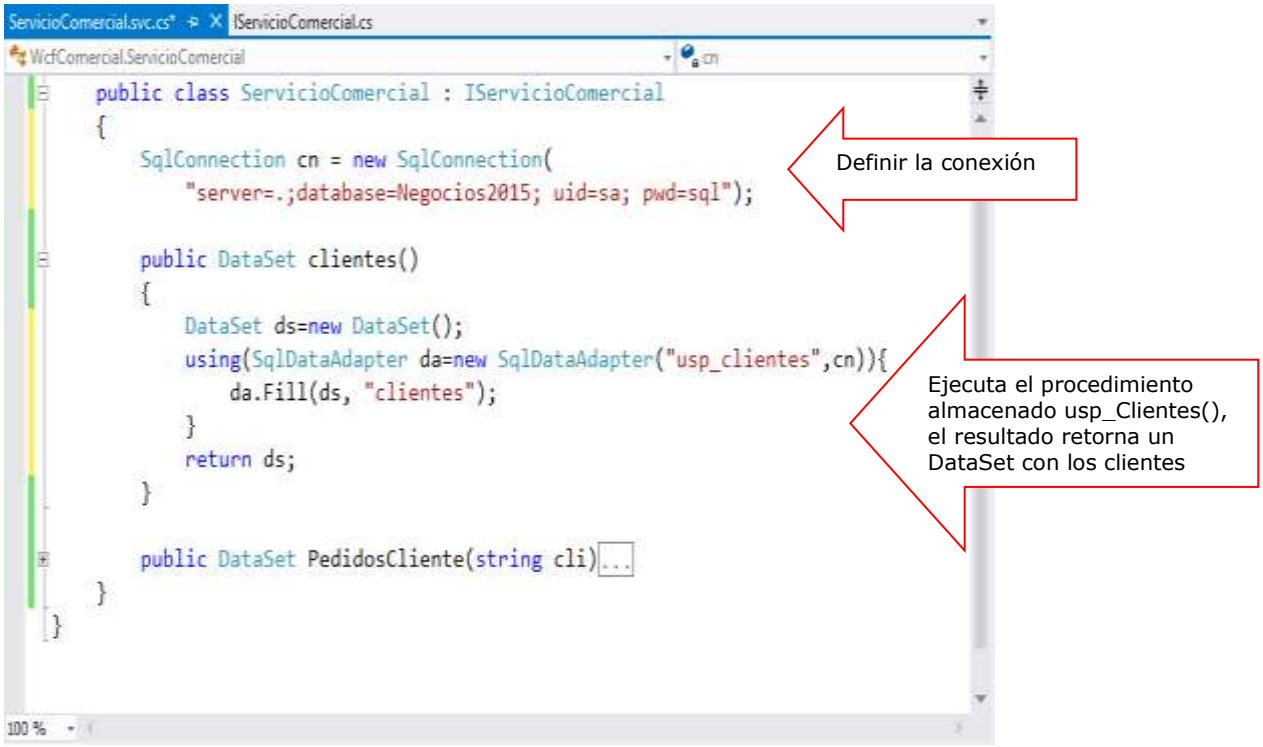
En la interface del ServicioComercial, defina los métodos de datos (retorna DataSet) que retorna la información de los clientes y los registros de pedidos por Cliente seleccionado.



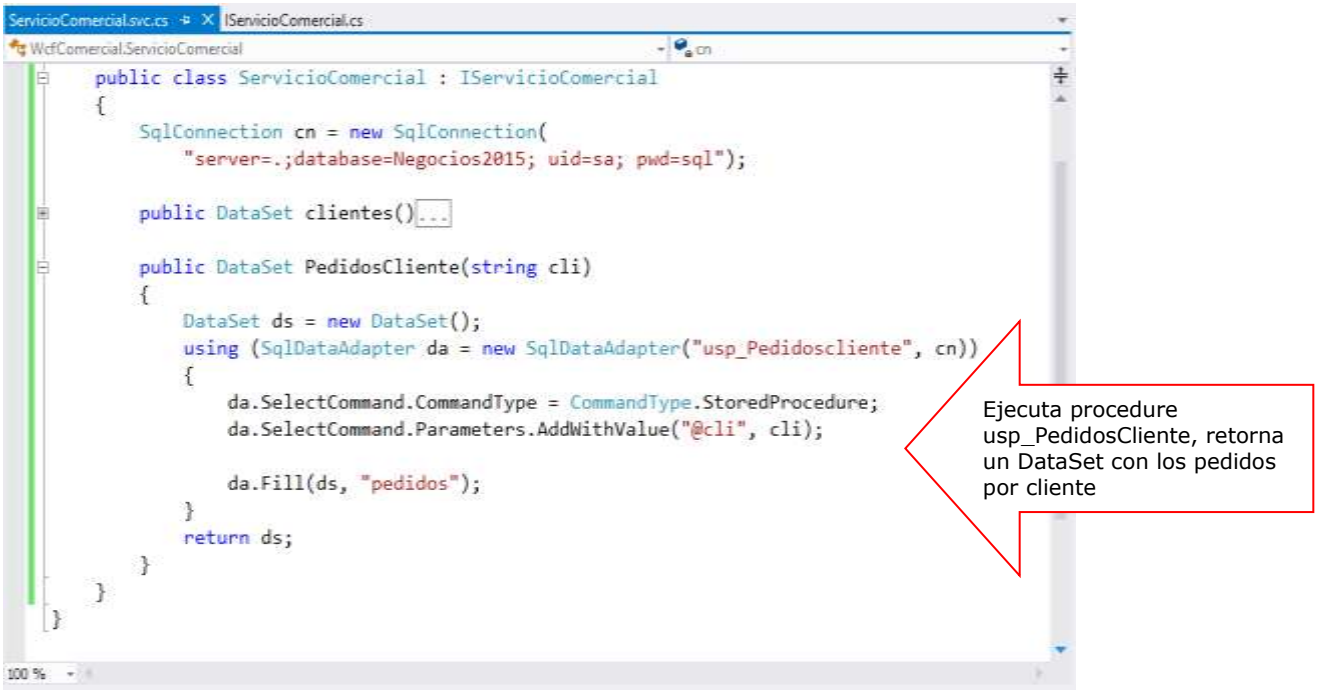
En ServicioComercial.cs, implemente los métodos definidos en la Interface del servicio, visualice los métodos a implementar.



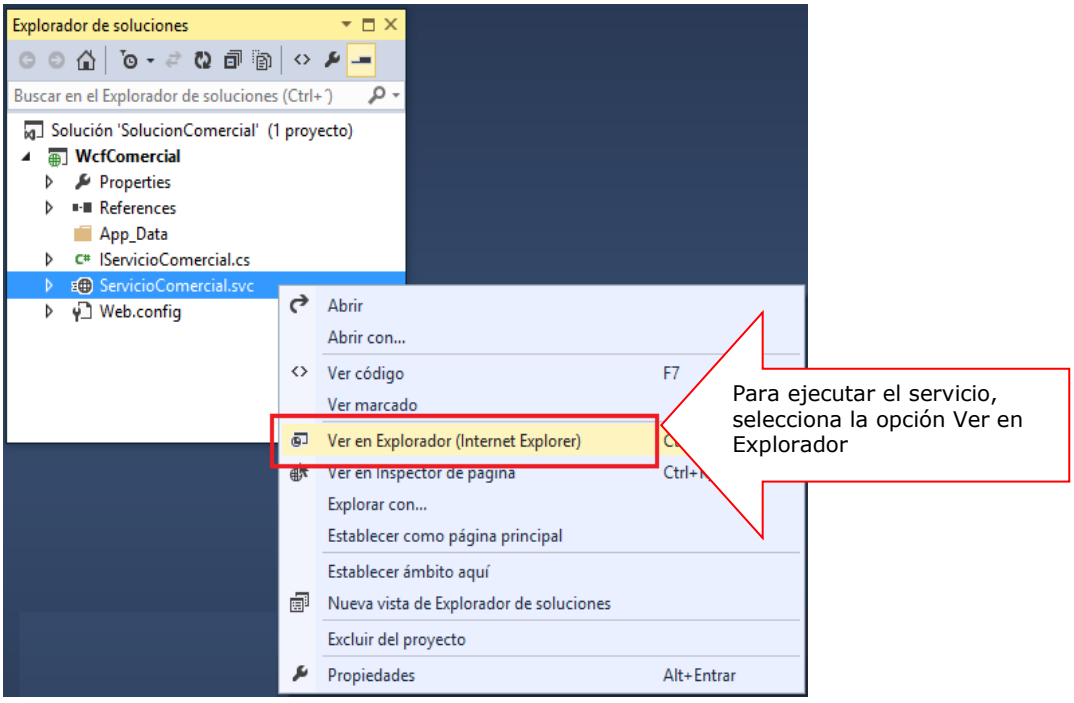
A continuación, defina las librerías para trabajar con SQL Server. En el método instancia la conexión a la base de datos Negocios2015. Implemente el método Clientes, ejecuta su Stored Procedure y retorna un DataSet



Implemente el método Pedidos por Cliente, donde ejecute el Stored Procedure y reciba su parámetro de entrada



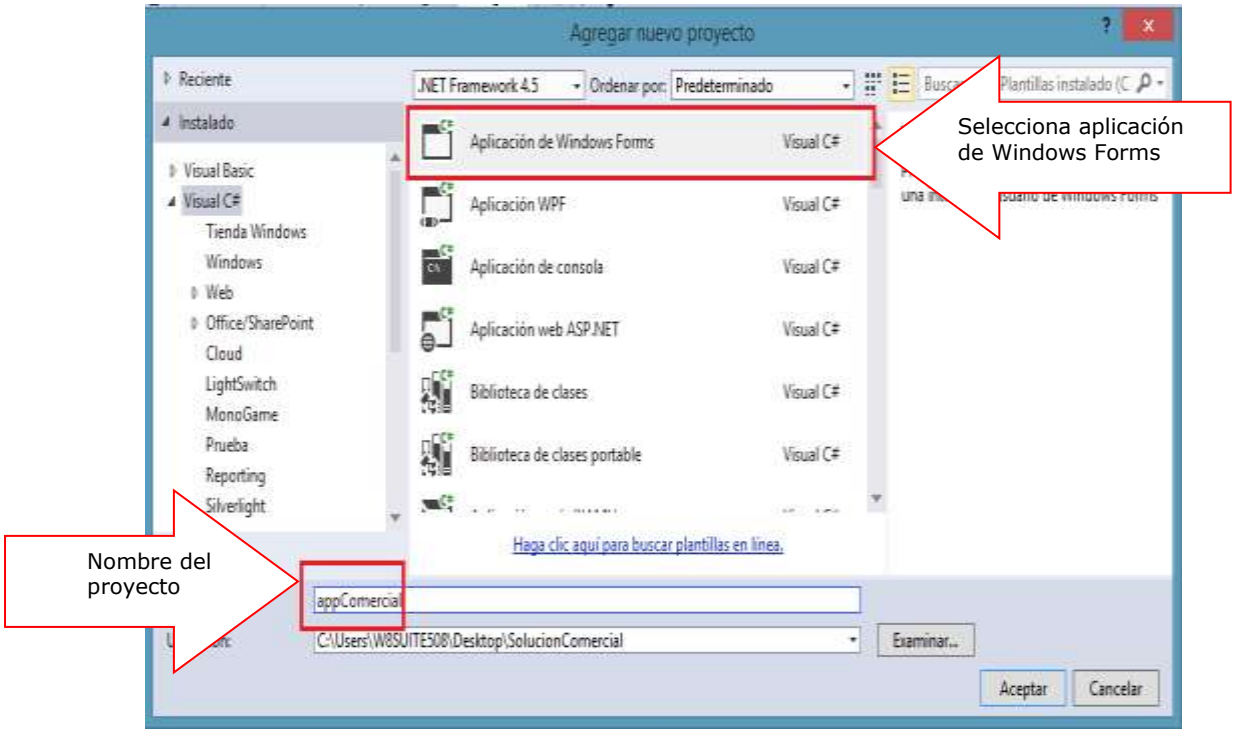
Terminado la implementación de los métodos, compilar el proyecto. A continuación, ejecutar el Servicio en un Explorador: Hacer click derecho al servicio y selecciona la opción Ver en Explorador



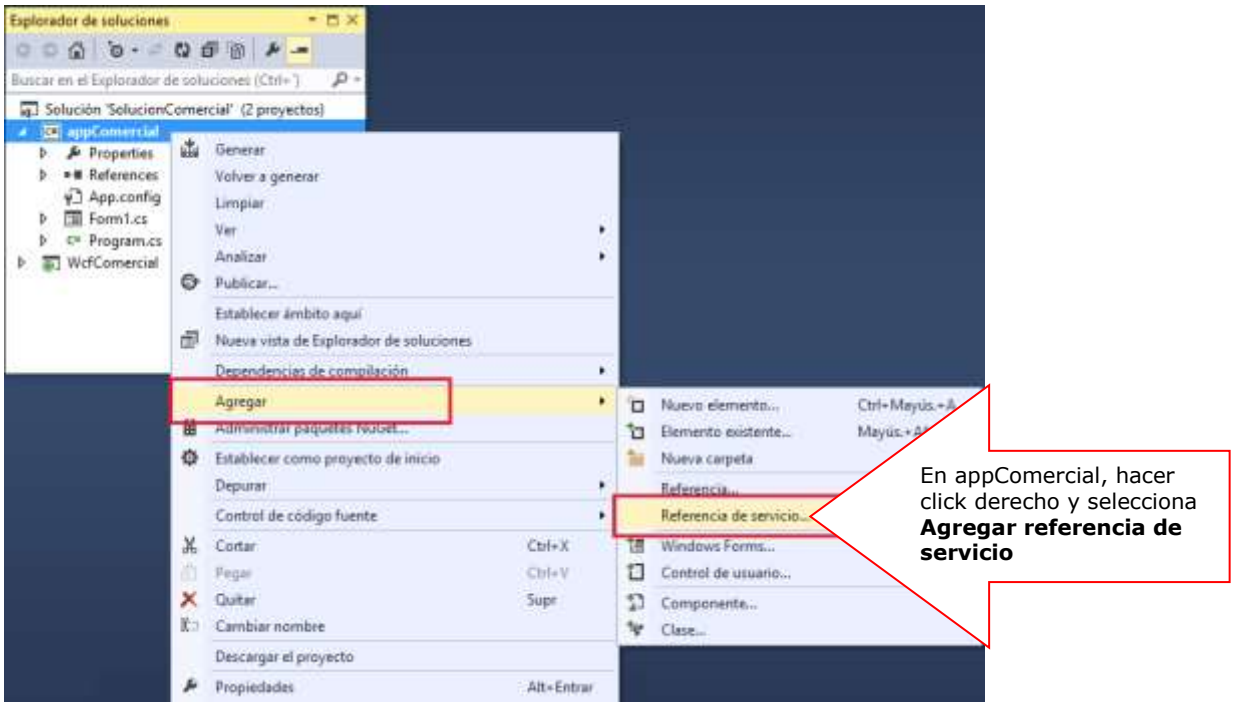
Al ver en el Explorador se crea una dirección donde se puede ejecutar el servicio de datos, copiar la dirección que esta sombreada.

3. TRABAJANDO CON LA APLICACIÓN DE WINDOWS FORMS

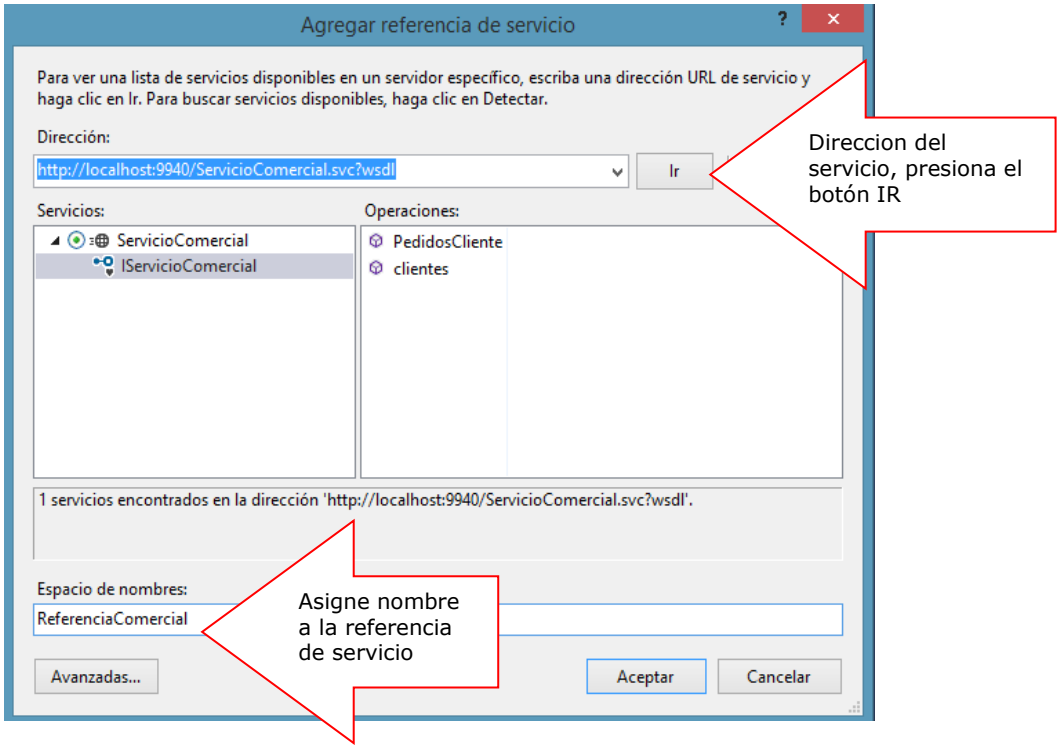
Agregar a la solución un proyecto Windows Form de C#. Selecciona el proyecto de tipo aplicación de Windows Forms.



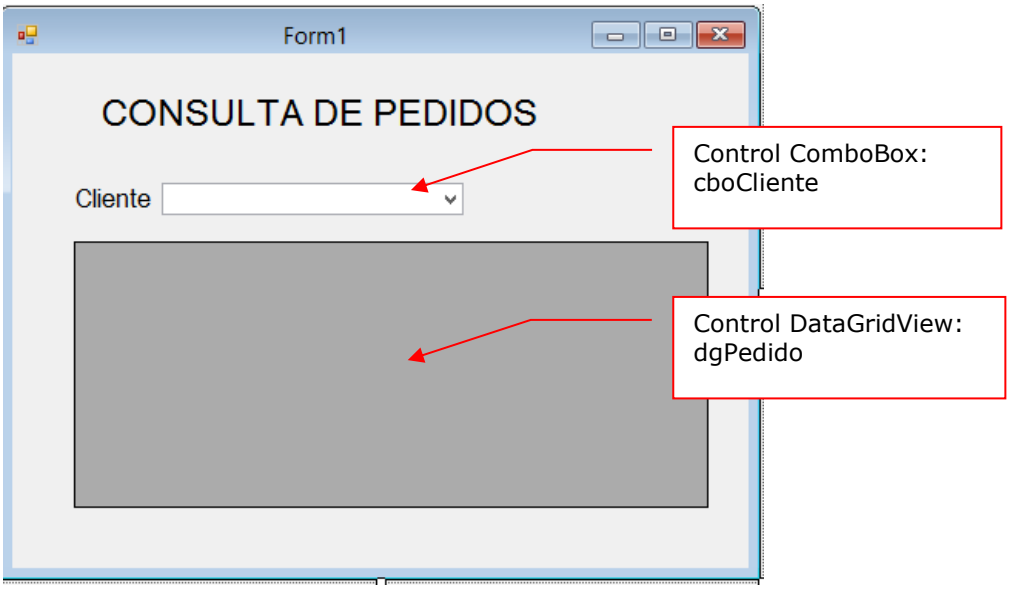
Para que el proyecto de Windows consuma el servicio de datos; desde el proyecto debe Agregar una referencia de Servicio, tal como se muestra



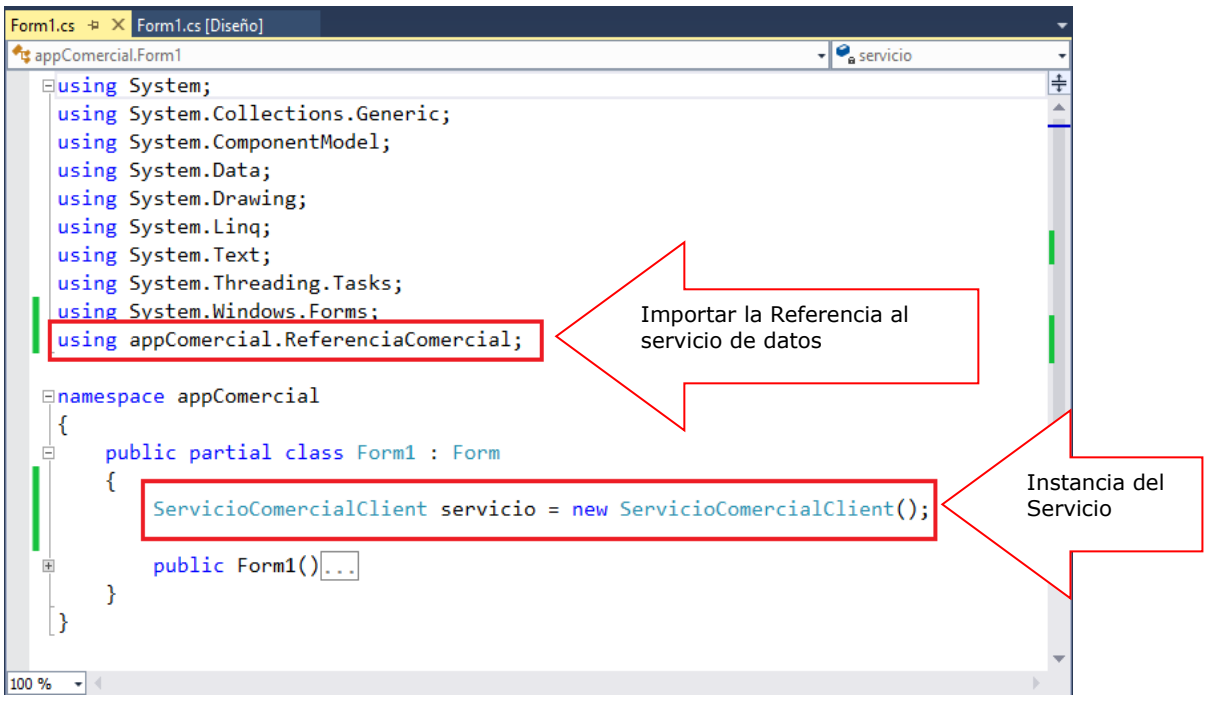
Pegar la dirección del servicio que vamos a consumir y presiona el botón IR para visualizar el nombre del servicio y sus métodos. Asigne un nombre a la referencia de Servicio



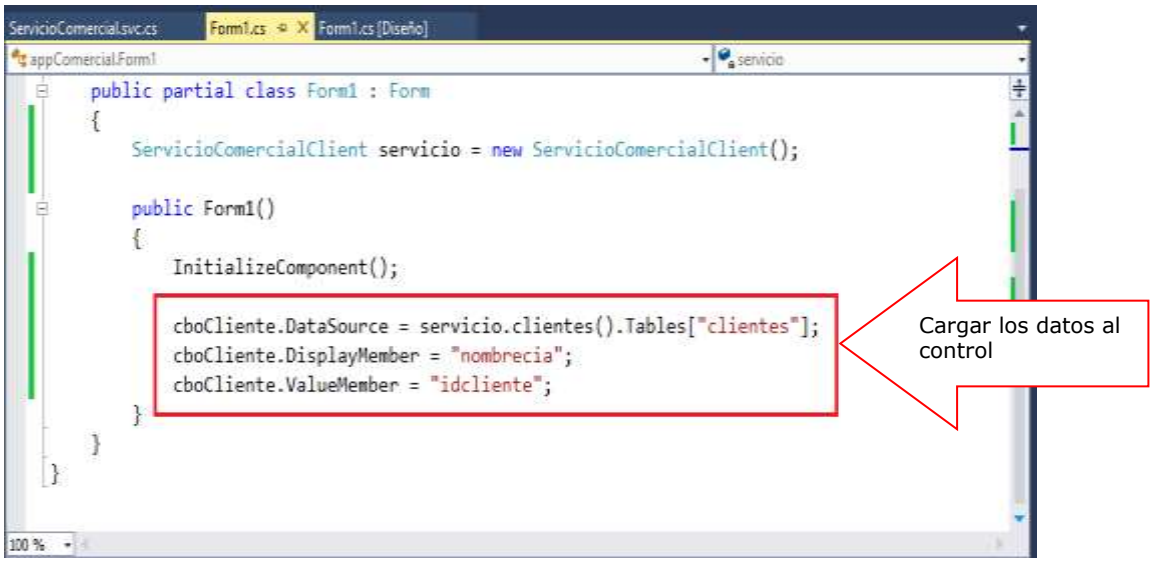
Dibuja el formulario, tal como se muestra



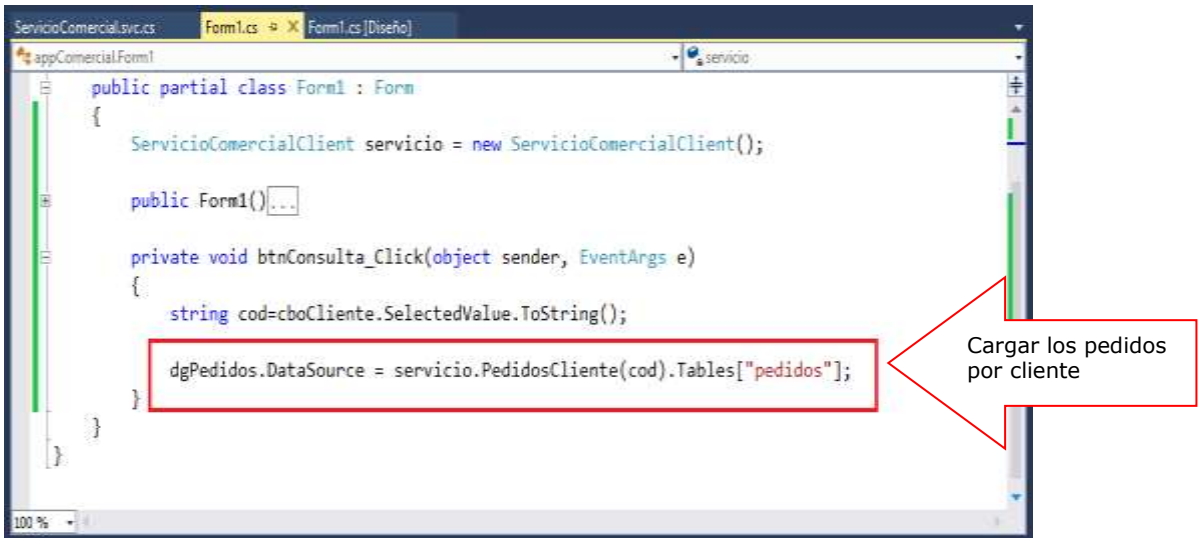
En la ventana de código, primero importar la referencia de servicio: appComercial.ReferenciaComercial. En la clase Form1 instanciar el servicio de datos



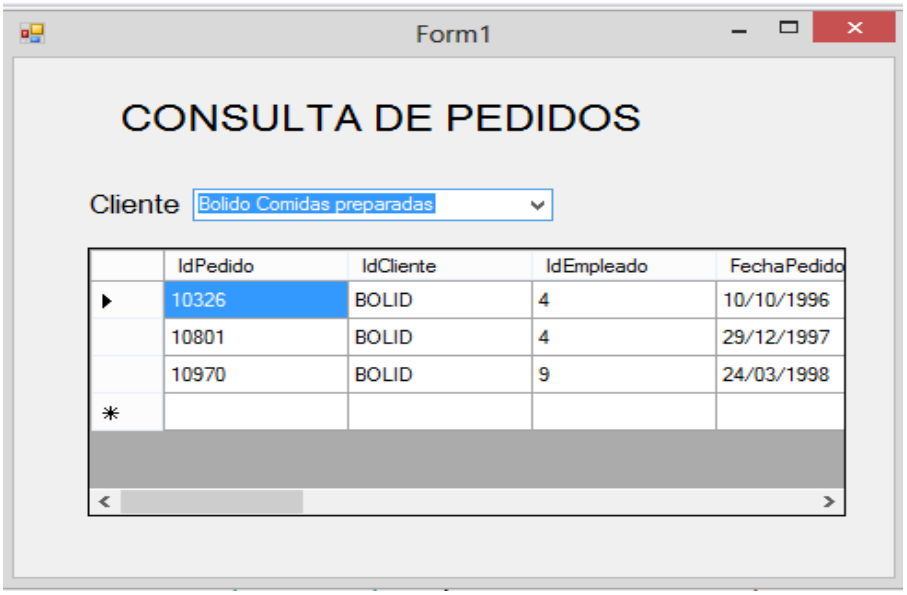
En el constructor defina las sentencias para cargar los datos de los clientes en el control cboCliente.



Programa el botón Consulta, selecciona un cliente, al presionar el botón listar los pedidos por cliente seleccionado



Presiona la tecla F5 para ejecutar el proyecto, selecciona un cliente, desde el control comboBox, se visualiza los pedidos del cliente seleccionado

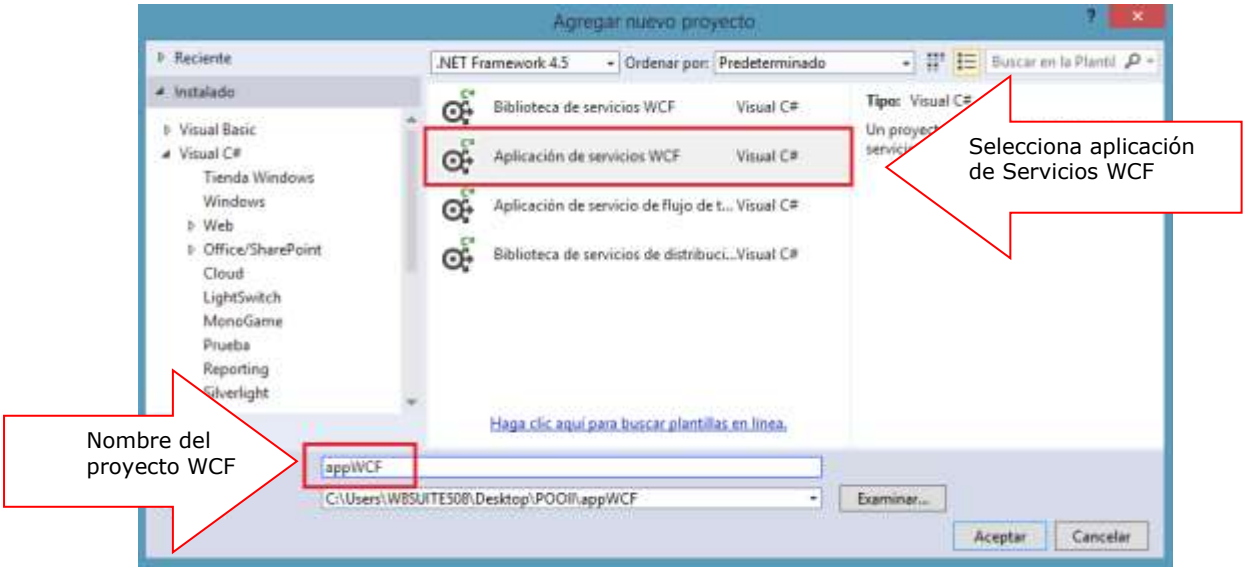


LABORATORIO 7.2

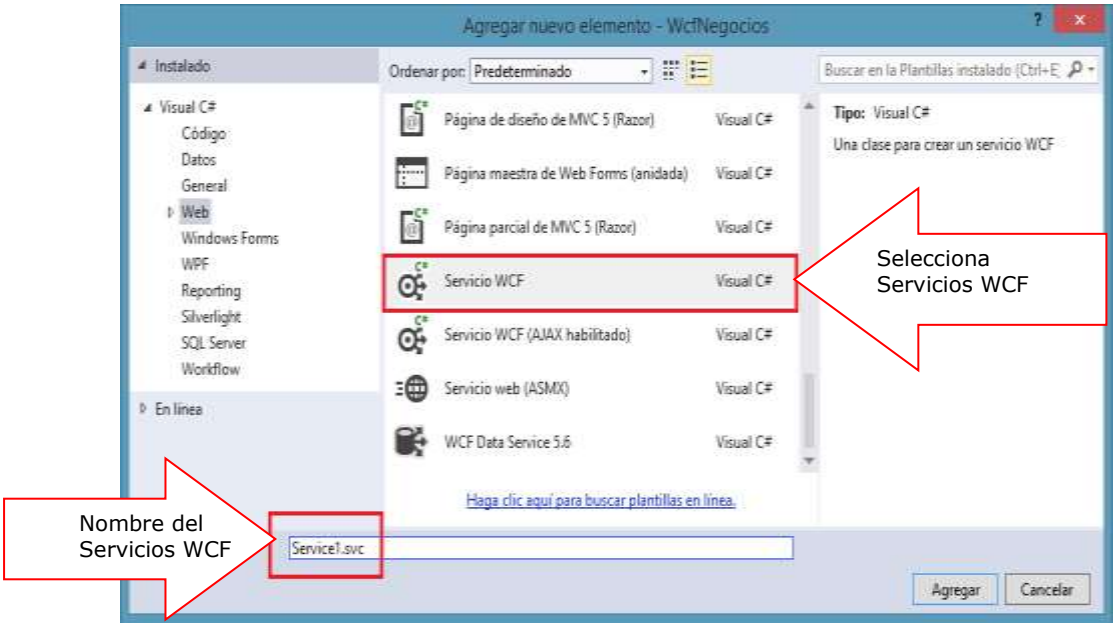
Se desea implementar un servicio de datos, que permita realizar un CRUD sonbre la tabla tb_clientes.

1. TRABAJANDO CON WINDOWS COMMUNICATION FOUNDATION

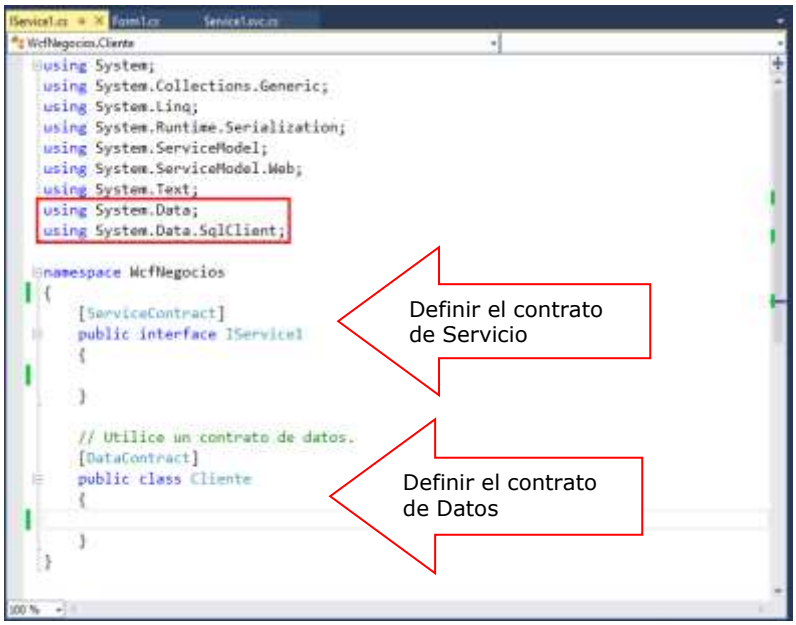
Agregar a la solución (appSolucion) un proyecto Visual C# de tipo WCF para definir un servicio de datos.



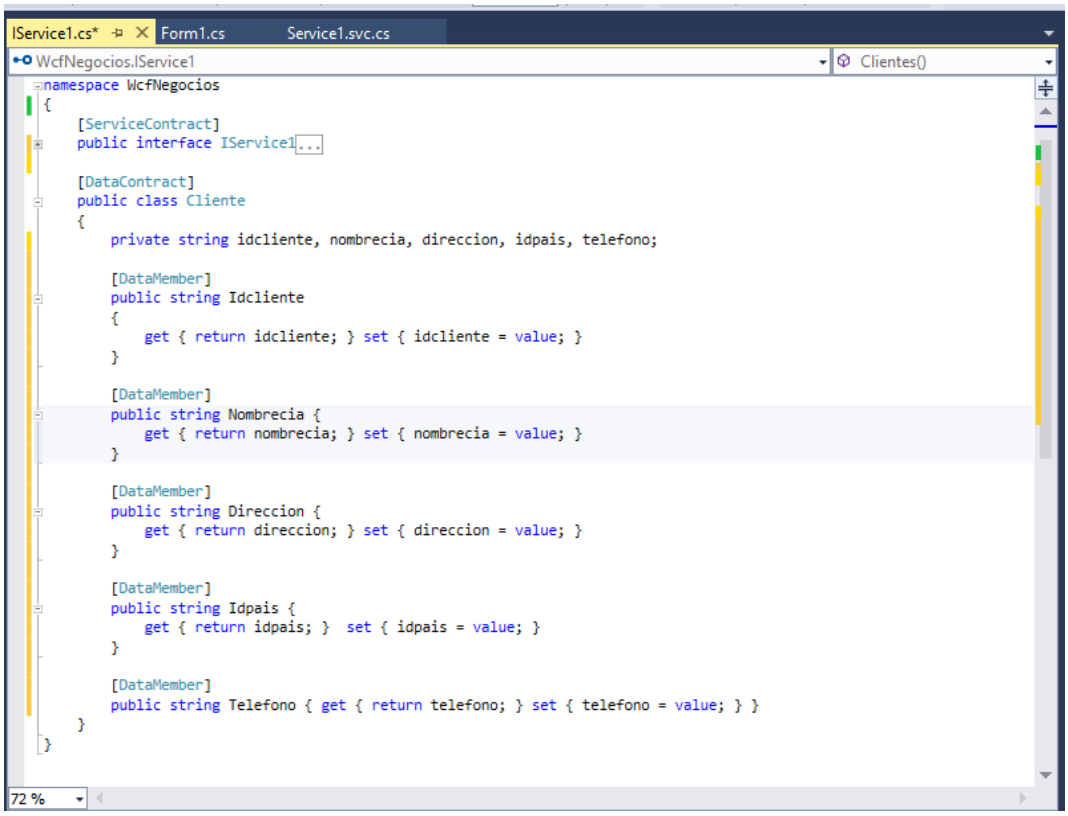
A continuación agregue al proyecto WCF un nuevo elemento: Servicio WCF



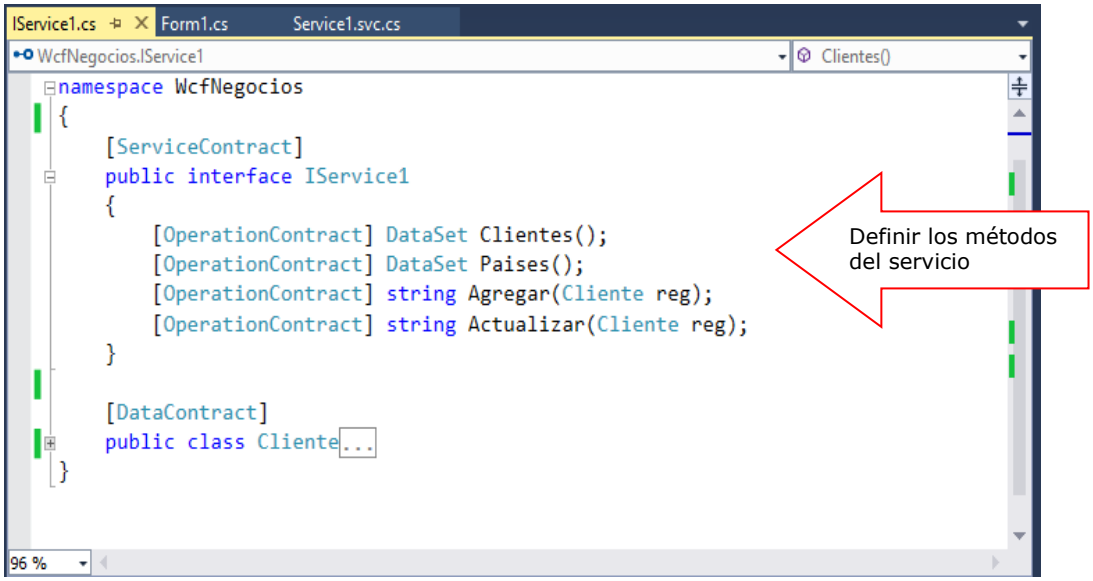
Defina en la interface IService1, las librerías, el contrato de Servicio y el contrato de datos



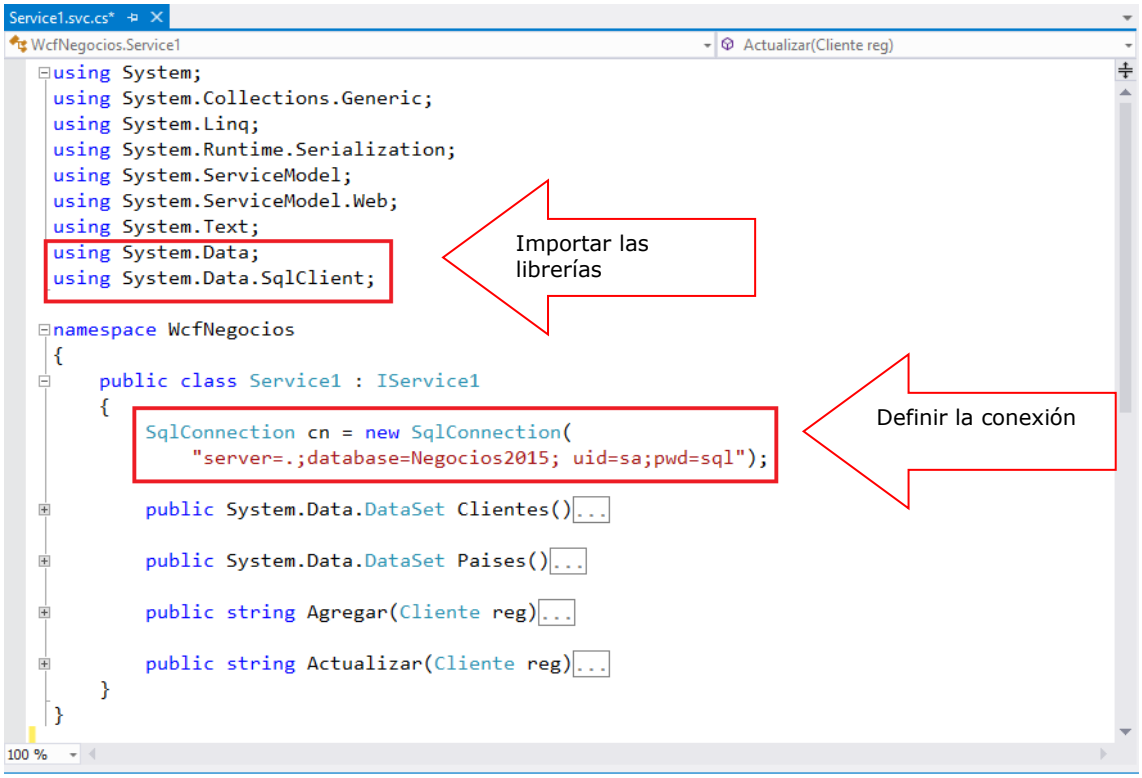
En el DataContract Cliente, definimos la estructura de datos, tal como se muestra



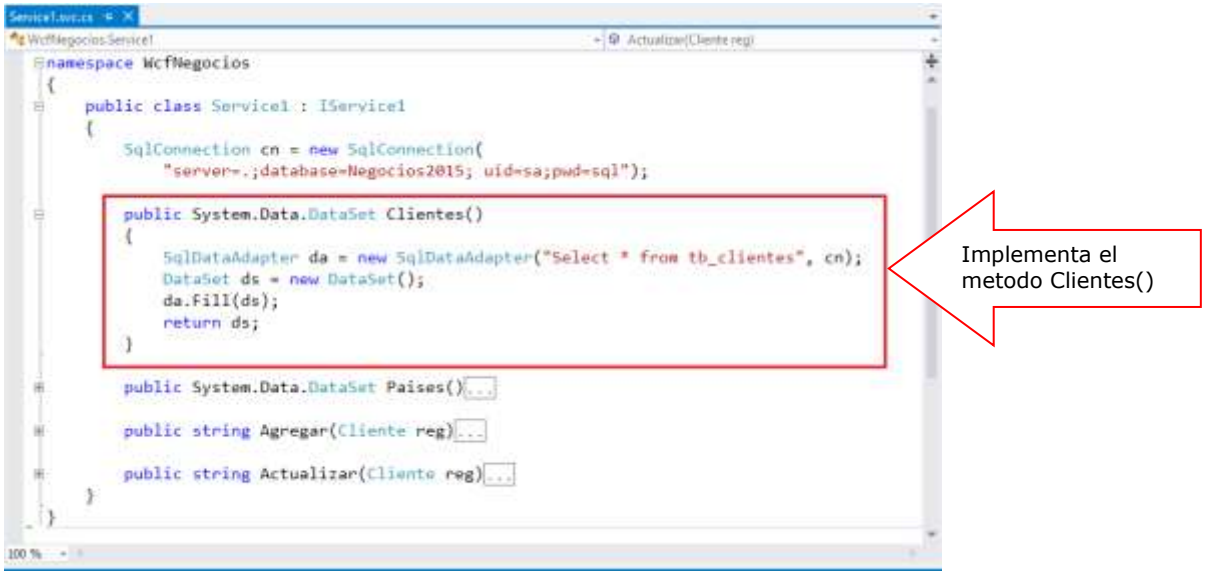
En el ServiceContract, defina los métodos a implementar en el Servicio, tal como se muestra.



Ubicarse en Service1 e implementar los métodos definidos en IService1, tal como se muestra



A continuación implementamos cada uno de los métodos de Service1

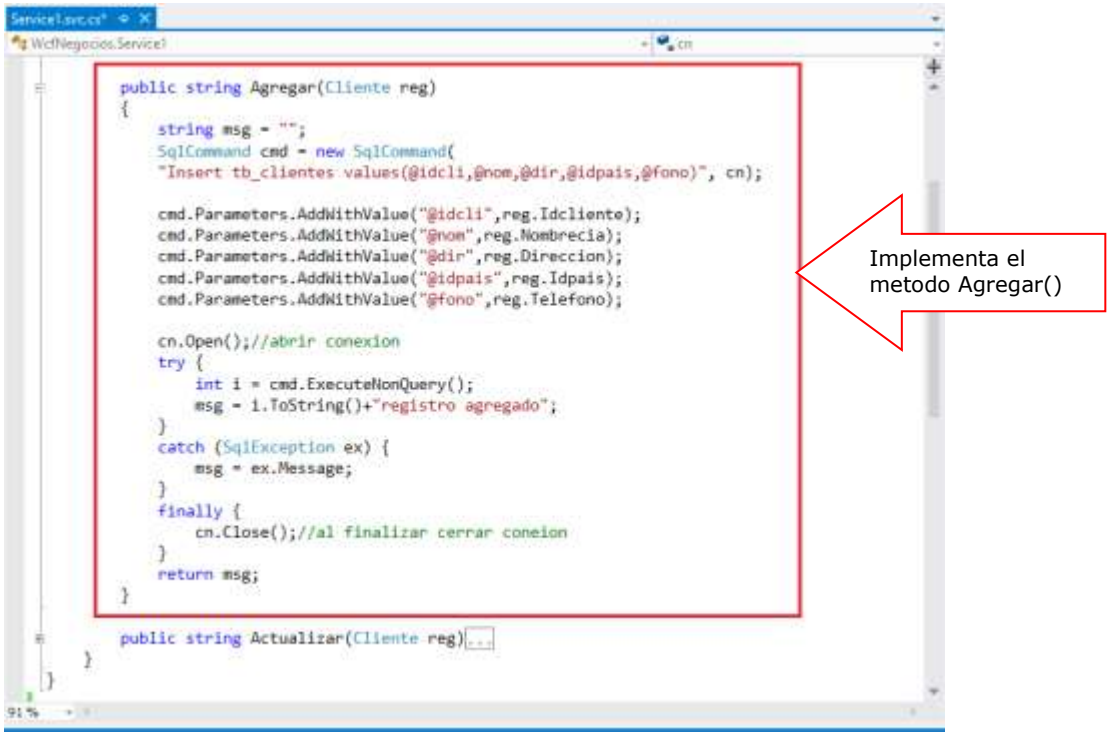



```
namespace WcfNegocios
{
    public class Service1 : IService1
    {
        SqlConnection cn = new SqlConnection(
            "server=.;database=Negocios2015; uid=sa;pwd=sql");

        public System.Data.DataSet Clientes()
        {
            SqlDataAdapter da = new SqlDataAdapter("Select * from tb_clientes", cn);
            DataSet ds = new DataSet();
            da.Fill(ds);
            return ds;
        }

        public System.Data.DataSet Paises()...

        public string Agregar(Cliente reg)...

        public string Actualizar(Cliente reg)...
    }
}
```

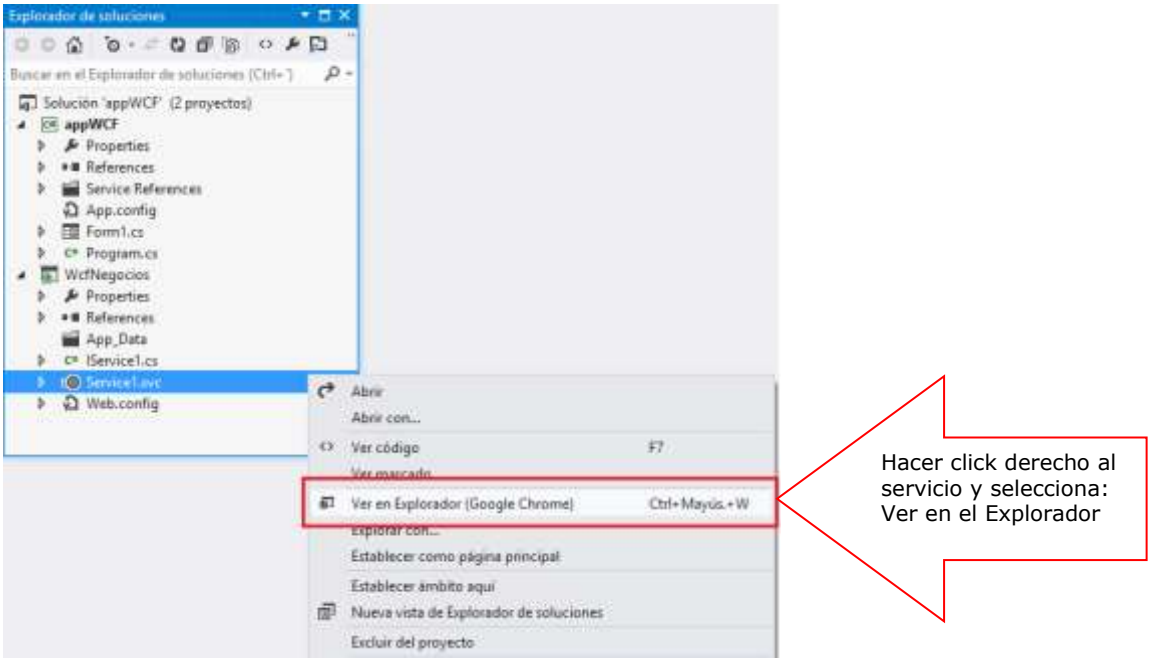
A continuación implementamos el metodo Paises()


```
namespace WcfNegocios
{
    public class Service1 : IService1
    {
        SqlConnection cn = new SqlConnection(
            "server=.;database=Negocios2015; uid=sa;pwd=sql");

        public System.Data.DataSet Clientes()...

        public System.Data.DataSet Paises()
        {
            SqlDataAdapter da = new SqlDataAdapter("Select * from tb_paises", cn);
            DataSet ds = new DataSet();
            da.Fill(ds);
            return ds;
        }

        public string Agregar(Cliente reg)...

        public string Actualizar(Cliente reg)...
    }
}
```

Implementando el metodo Agregar, definiendo el parámetro reg de tipo Cliente,

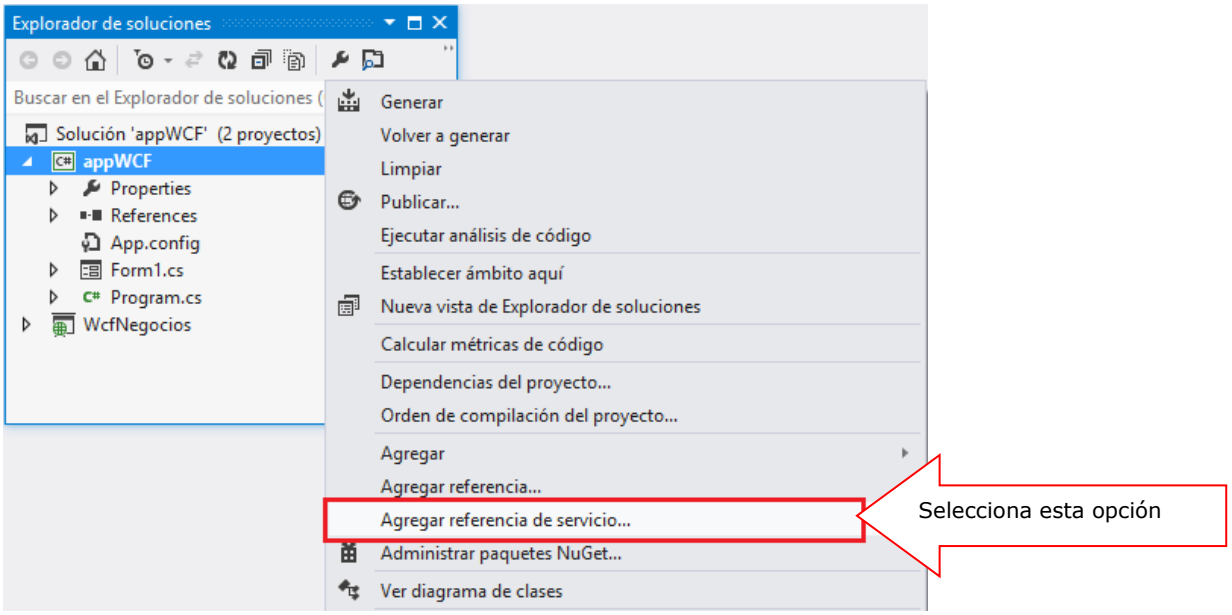


Terminado de implementar los métodos, vamos a ejecutar el servicio en un navegador para que obtener una dirección por la cual se ejecuta el servicio

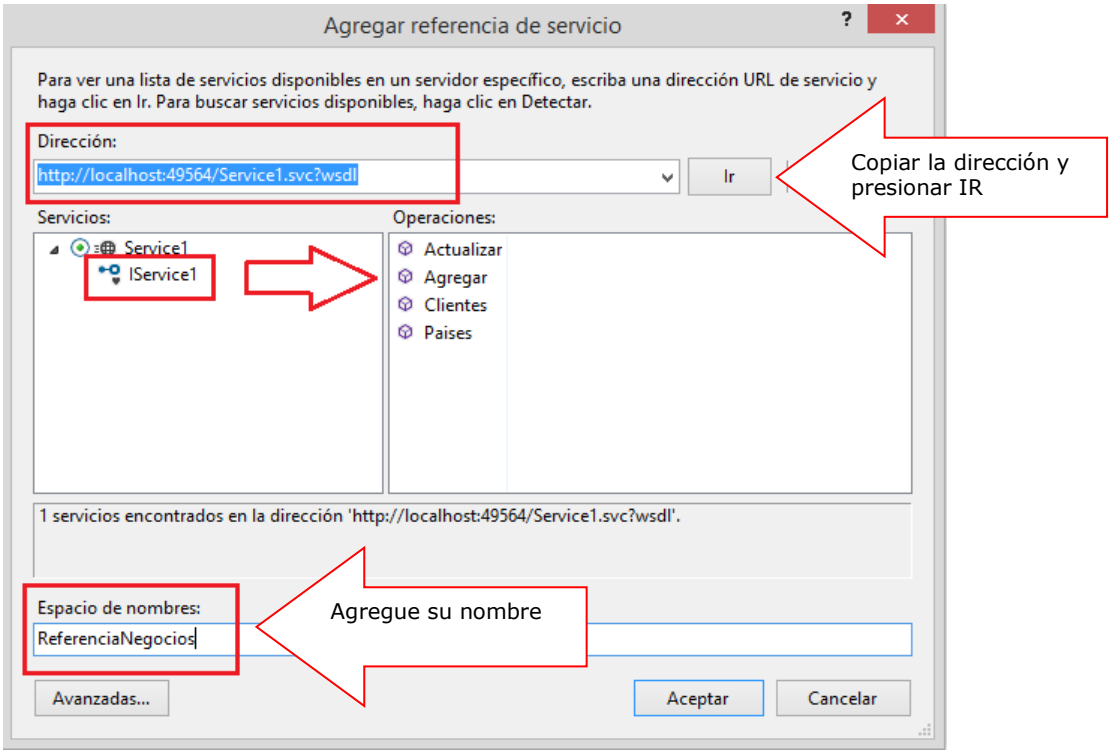


El navegador nos dara una dirección la cual se copiará en el proyecto

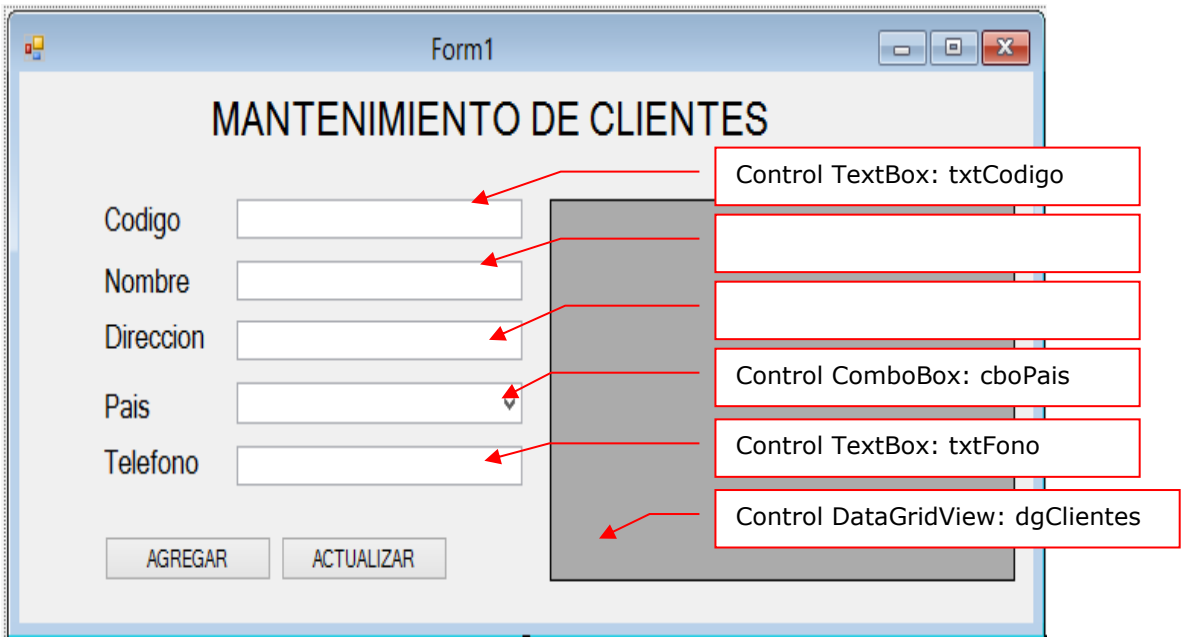
A continuación, en el proyecto de Windows, agregamos una referencia de servicio. Hacer click derecho al proyecto, selecciona **Agregar referencia de servicio**



En la ventana, copiar la dirección y presionar el boton Ir, donde nos visualiza el servicio. Desplegar el servicio y visualizamos su Interface. Para terminar asigne un nombre a la referencia: referenciaNegocios

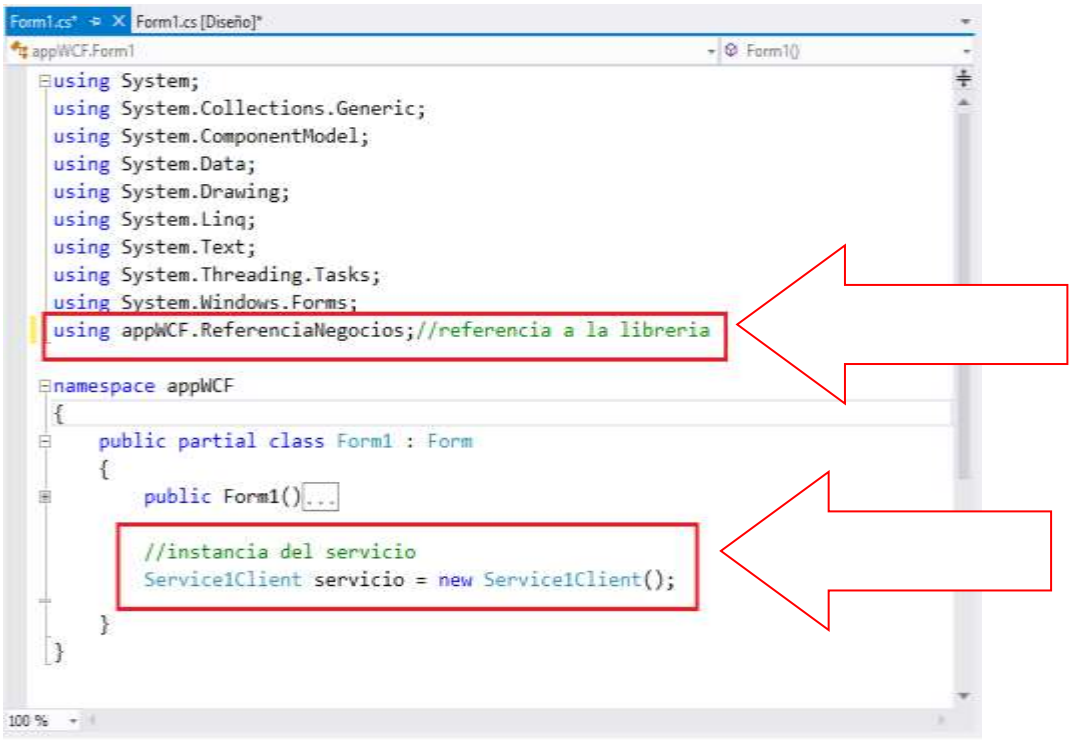


A continuación dibuja la GUI

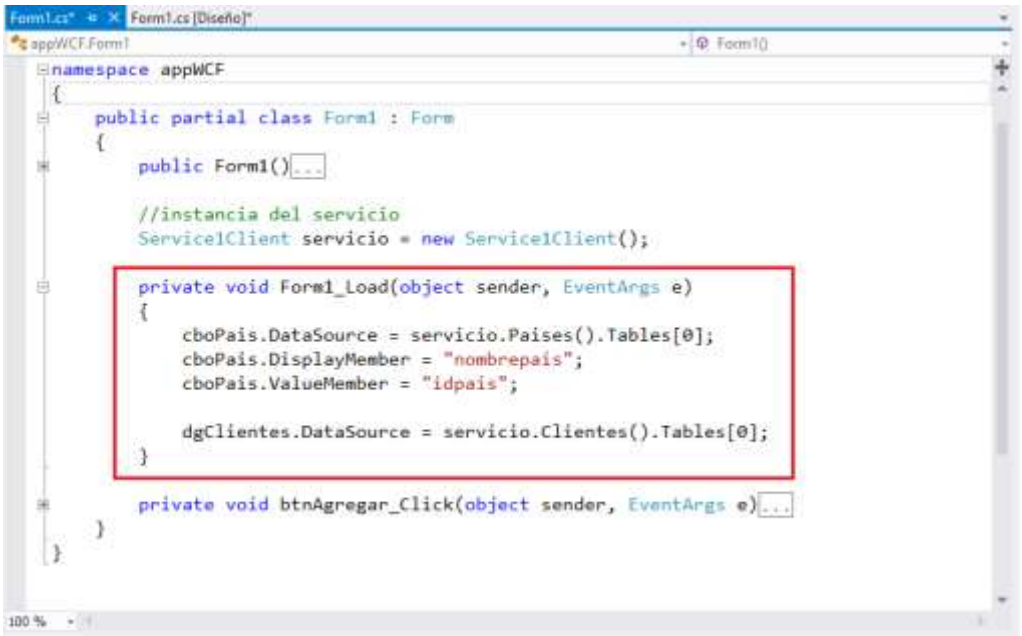


Programación

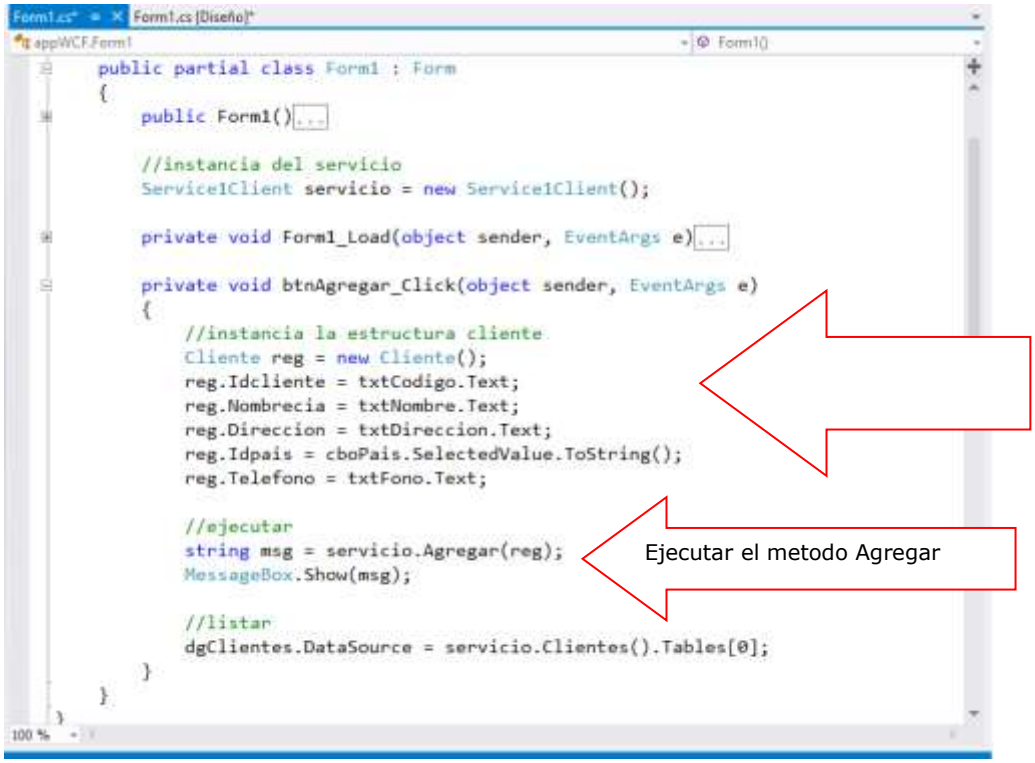
Importar la librería del servicio, y luego instanciar el servicio en el formulario



Programa el evento Load del formulario para cargar los datos a los controles, tal como se muestra

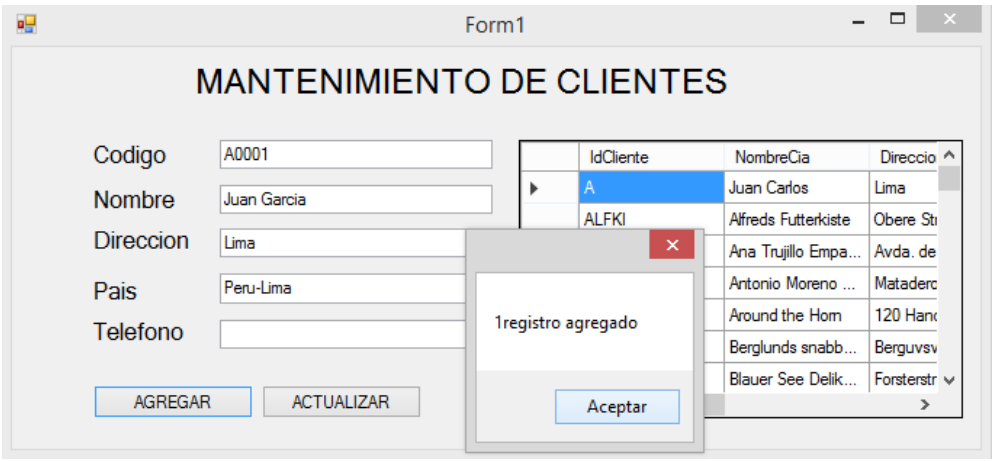


Programa el botón Agregar: instanciar la estructura Cliente, ingresar los datos y ejecutar el metodo AGREGAR de servicio, visualizando un mensaje




```
public partial class Form1 : Form
{
    public Form1()...

    //instancia del servicio
    Service1Client servicio = new Service1Client();

    private void Form1_Load(object sender, EventArgs e)...

    private void btnAgregar_Click(object sender, EventArgs e)
    {
        //instancia la estructura cliente
        Cliente reg = new Cliente();
        reg.Idcliente = txtCodigo.Text;
        reg.Nombrecia = txtNombre.Text;
        reg.Direccion = txtDireccion.Text;
        reg.Idpais = cboPais.SelectedValue.ToString();
        reg.Telefono = txtFono.Text;

        //ejecutar
        string msg = servicio.Agregar(reg);
        MessageBox.Show(msg);

        //listar
        dgClientes.DataSource = servicio.Clientes().Tables[0];
    }
}
```

Ejecuta el proyecto, ingresa los datos en el Formulario, al presionar el boton Agregar, se ejecuta el proceso visualizando un mensaje y actualiza la lista de clientes.



Resumen

- Windows Communication Foundation (WCF) es un marco de trabajo para la creación de aplicaciones orientadas a servicios. Con WCF, es posible enviar datos como mensajes asincrónicos de un extremo de servicio a otro.
- Un extremo puede ser un cliente de un servicio que solicita datos de un extremo de servicio. Los mensajes pueden ser tan simples como un carácter o una palabra que se envía como XML, o tan complejos como una secuencia de datos binarios.
- Los contratos definen varios aspectos del sistema de mensajes. El contrato de datos describe cada parámetro que constituye cada mensaje que un servicio puede crear o utilizar. Los documentos de Lenguaje de definición de esquemas XML (XSD) definen los parámetros de mensaje, permitiendo a cualquier sistema que entienda XML procesar los documentos.
- El contrato de servicios especifica las firmas de método actuales del servicio y se distribuye como una interfaz en uno de los lenguajes de programación compatibles, como Visual Basic o Visual C#.
- La capa del tiempo de ejecución del servicio contiene los comportamientos que solo se producen durante la operación actual del servicio, es decir, los comportamientos en tiempo de ejecución del servicio. La limitación de peticiones controla cuántos mensajes se procesan que puede variar si la demanda para el servicio crece a un límite preestablecido.
- El comportamiento de los metadatos rige cómo y si los metadatos se ponen a disposición del mundo externo. El comportamiento de la instancia especifica cuántas instancias del servicio se pueden ejecutar (por ejemplo, un singleton especifica solo una instancia para procesar todos los mensajes).
- La capa de la mensajería se crea de canales. Un canal es un componente que procesa un mensaje de alguna manera, por ejemplo, autenticando un mensaje. Un conjunto de canales también se conoce como una pila de canales
- Hay dos tipos de canales: canales de transporte y canales de protocolo. Los canales de transporte leen y escriben mensajes de la red (o algún otro punto de la comunicación con el mundo externo). La capa de la mensajería muestra los posibles formatos y modelos de intercambio de los datos. WS-Security es una implementación de la especificación WS-Security que habilita la seguridad en la capa del mensaje.
- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

  http://msdn.microsoft.com/es-es/library/ms731082%28v=vs.110%29.aspx

  http://geeks.ms/blogs/jnunez/archive/2007/08/10/tutorial-wcf-1-de-5.aspx

  http://msdn.microsoft.com/es-es/library/dd456779%28v=vs.110%29.aspx

UNIDAD

3

CONSUMIENDO Y EXPORTANDO DATOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno implementa servicios de datos desde una aplicación Windows Communication Foundation para consultar y actualizar datos, y exporta los datos de un origen de datos a un archivo de Excel.

Temario

1. **Tema 8: Manejo de datos en un archivo de Excel (4 horas)**
   1.1. Trabajando con archivos de Excel
   1.2. Almacenar los datos hacia un archivo Excel
   1.3. Cargar datos desde un archivo de Excel

ACTIVIDADES PROPUESTAS

- Los alumnos crea una aplicación para realizar consulta de datos a un origen de datos y almacenar los datos en archivo de Excel.
- Los alumnos crea una aplicación para cargar datos desde un archivo de Excel y realizar operaciones de consulta.
- Los alumnos desarrollan los laboratorios de la semana.

8. MANEJO DE DATOS EN UN ARCHIVO DE EXCEL

Algunas veces como programadores nos toca manejar datos de excel en nuestras aplicaciones, pues ahora veremos como hacerlo usando ado.net.

En este capítulo vamos a describir cómo puede utilizar ADO.NET para recuperar datos de un libro de Microsoft Excel, modificar los datos de un libro existente o agregar datos a un nuevo libro. Para tener acceso a libros de Excel con ADO.NET, puede utilizar el proveedor OLE DB de Jet; en este artículo se proporciona la información necesaria para que pueda utilizar el proveedor OLE DB de Jet cuando Excel es el origen de datos de destino.

8.1 TRABAJANDO CON ARCHIVOS DE EXCEL

El motor de base de datos de Microsoft Jet puede tener acceso a los datos en otros formatos de archivo de base de datos, como libros de Excel, a través de los controladores instalables de método de acceso secuencial indizado (ISAM).

La información que se pasa al proveedor para la conexión es la ruta + nombre del libro y la versión. Como versión de proveedor se debe usar la 4.0, pues la 3.51 no admite controladores Jet ISAM. Como versión de Excel se pasa:

- Excel 5.0: para Excel 95
- Excel 8.0: para Excel 2000/XP/2003/2007
- Excel 12.0: para Excel 2007

Cuando queremos abrir un libro de Excel de versión 2002 o anterior, es necesario que establezcamos como proveedor Microsoft.JET.OLEDB.4.0; si queremos abrir un libro de Excel 2007, es necesario que establezcamos como proveedor Microsoft.ACE.OLEDB.12.0.

Conexión a un archivo de Excel de versión 2002

```
  Dim cn As New OleDbConnection(
  "data Source=archivo; provider=Microsoft.Jet.OLEDB.4.0;" +
  "Extended properties='Excel 8.0, HDR=Yes'")
```

Conexión a un archivo

```
  Dim cn As New OleDbConnection(
  "data Source=archivo; provider=Microsoft.ACE.OLEDB.12.0;"
+
  "Extended properties='Excel 12.0, HDR=Yes'")
```

8.2 ALMACENAR DATOS HACIA UN ARCHIVO DE EXCEL

Puede recuperar los registros de una base de datos mediante uno de estos dos enfoques en ADO.NET: con un **conjunto de datos** o con un objeto **DataReader**.

Un **conjunto de datos** es una memoria caché de registros recuperados de un origen de datos. Los datos en el conjunto de datos son ser una versión mucho más reducida de lo que está en la base de datos. Sin embargo, puede trabajar con él de la misma forma que trabaja con los datos reales y estar desconectado de la

base de datos real. Además de la recuperación de datos, también puede utilizar un conjunto de datos para realizar operaciones de actualización en la base de datos subyacente.

Como alternativa, puede utilizar un **DataReader** para recuperar una secuencia de sólo lectura, sólo hacia delante de los datos de una base de datos. Cuando utilice el programa **DataReader**, un aumento del rendimiento y es la sobrecarga del sistema disminuye porque sólo una fila a la vez está siempre en la memoria. Si tiene una gran cantidad de datos que desea recuperar y no piensa realizar cambios en la base de datos, **DataReader** es una opción mejor que un conjunto de datos.

Recuperar Registros

Con ADO.NET se puede insertar y actualizar registros en un libro de Excel:

- Ejecuta directamente un comando para insertar o actualizar registros de uno en uno. Para ello, puede crear un objeto **OLEDbCommand** en su conexión y establecer su propiedad **CommandText** en un comando y a continuación llame el método **ExecuteNonQuery**

Comando para insertar

```
INSERT INTO [Sheet1$] (F1, F2) values ('111', 'ABC')
```

Comando para actualizar registros

```
UPDATE [Sheet1$] SET F2 = 'XYZ' WHERE F1 = '111'
```

- Realizar cambios en un **conjunto de datos** que haya rellenado con una tabla o consulta desde un libro de Excel y, a continuación, llamar al método **Update** de **DataAdapter** se resuelven los cambios del **conjunto de datos** al libro. Sin embargo, para utilizar el método **Update** para cambiar resolución que debe establecer comandos parametrizados para de **DataAdapter InsertCommand**

Comando para insertar

```
INSERT INTO [Sheet1$] (F1, F2) values (?,?)
```

Comando para actualizar registros

```
UPDATE [Sheet1$] SET F2 =? WHERE F1 =?
```

Se precisan los comandos parametrizado INSERT y UPDATE porque **OleDbDataAdapter** no proporcionan información clave o índice para libros de Excel; sin campos de clave o índice, **CommandBuilder** no puede generar automáticamente los comandos para usted

- Exportar datos a partir de los datos de otro origen en un libro de Excel, siempre que el otro origen de datos puede utilizarse con el proveedor OLE DB de Jet. Orígenes de datos que puede utilizar con el proveedor OLE DB de Jet de esta manera son archivos de texto, bases de datos de Microsoft Access y, por supuesto, otros libros de Excel. Con un solo comando INSERT INTO, puede exportar los datos de otra tabla o consulta en el libro:

```
INSERT INTO [Sheet1$] IN 'C:\Book1.xls' 'Excel 8.0;' SELECT * FROM
MyTable"
```

INSERT INTO requiere que la tabla de destino (o la hoja de cálculo) ya existe; datos se anexan a la tabla de destino.

Aunque el proveedor OLE DB de Jet le permite insertar y actualizar registros en un libro de Excel, no permite las operaciones de eliminación. Si intenta realizar una operación de eliminación de uno o más registros, recibirá el siguiente mensaje de error: Este ISAM no admite la eliminación de datos en una tabla vinculada

8.3 CARGAR DATOS DESDE UN ARCHIVO DE EXCEL

Cuando queremos abrir un libro de Excel de versión 2002 o anterior, es necesario que establezcamos como proveedor Microsoft.JET.OLEDB.4.0; si queremos abrir un libro de Excel 2007, es necesario que establezcamos como proveedor Microsoft.ACE.OLEDB.12.0.

Conexión a un archivo de Excel de versión 2002

```
Dim cn As New OleDbConnection(
"data Source=archivo; provider=Microsoft.Jet.OLEDB.4.0;" +
"Extended properties='Excel 8.0, HDR=Yes'")
```

Conexión a un archivo

```
Dim cn As New OleDbConnection(
"data Source=archivo; provider=Microsoft.ACE.OLEDB.12.0;" +
"Extended properties='Excel 12.0, HDR=Yes'")
```

Tipos de Datos

Hay varias maneras que puede hacer referencia a una tabla (o rango) en un libro de Excel:

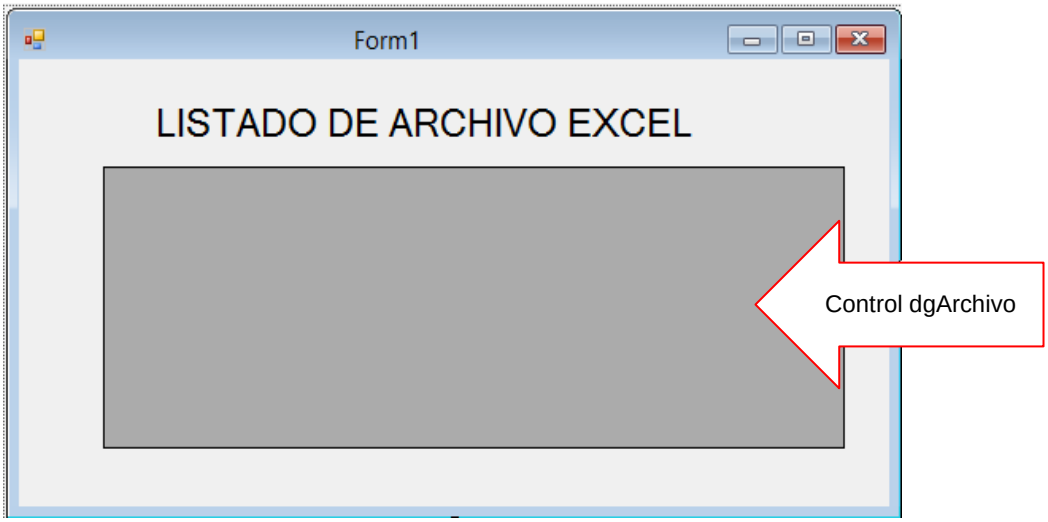
- Utilice el nombre de la hoja seguido de un signo de dólar (por ejemplo, [Hoja1$] o [Mi hoja de cálculo$]).
- Select * from [Hoja1$]
- Utilice un rango con un nombre definido (por ejemplo, [MyNamedRange]):
- Seleccionar * desde [MyNamedRange]
- Utilice un rango con una dirección específica (por ejemplo, [Hoja1$ A1: B10]):
- Seleccionar * desde [Hoja1$ A1: B10]

LABORATORIO 8.1

Se desea implementar un programa que permita consultar los datos almacenados en un archivo de Excel, cuya hoja se llama “INVENTARIO”

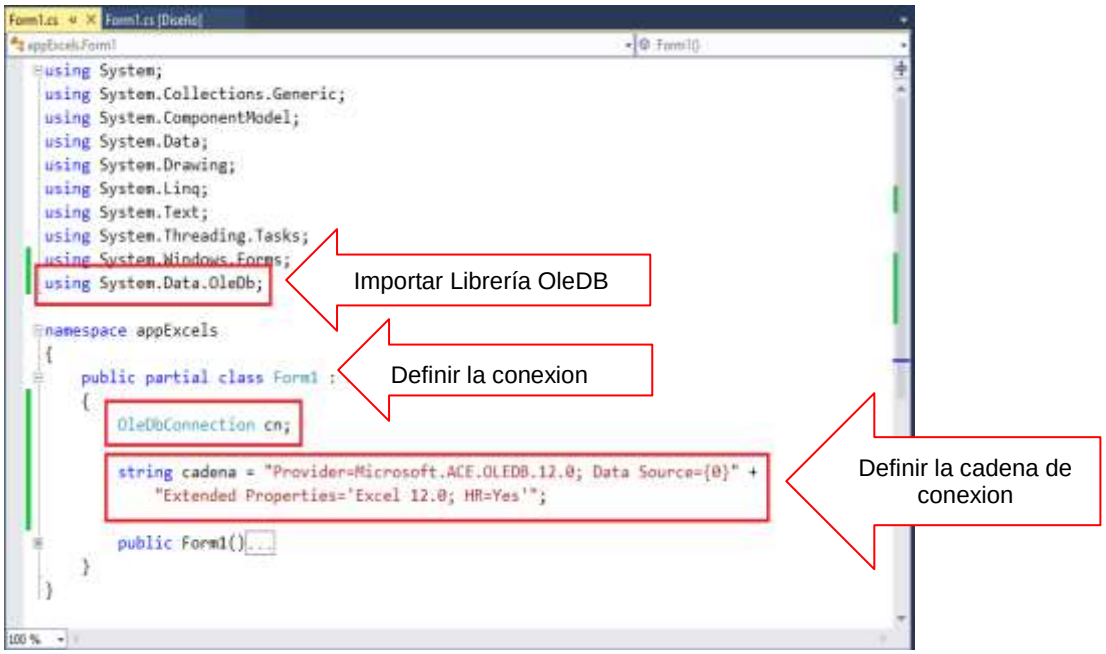
1. DISEÑO DEL FORMULARIO.

Diseña el formulario para el listado de archivos de Excel, tal como se muestra

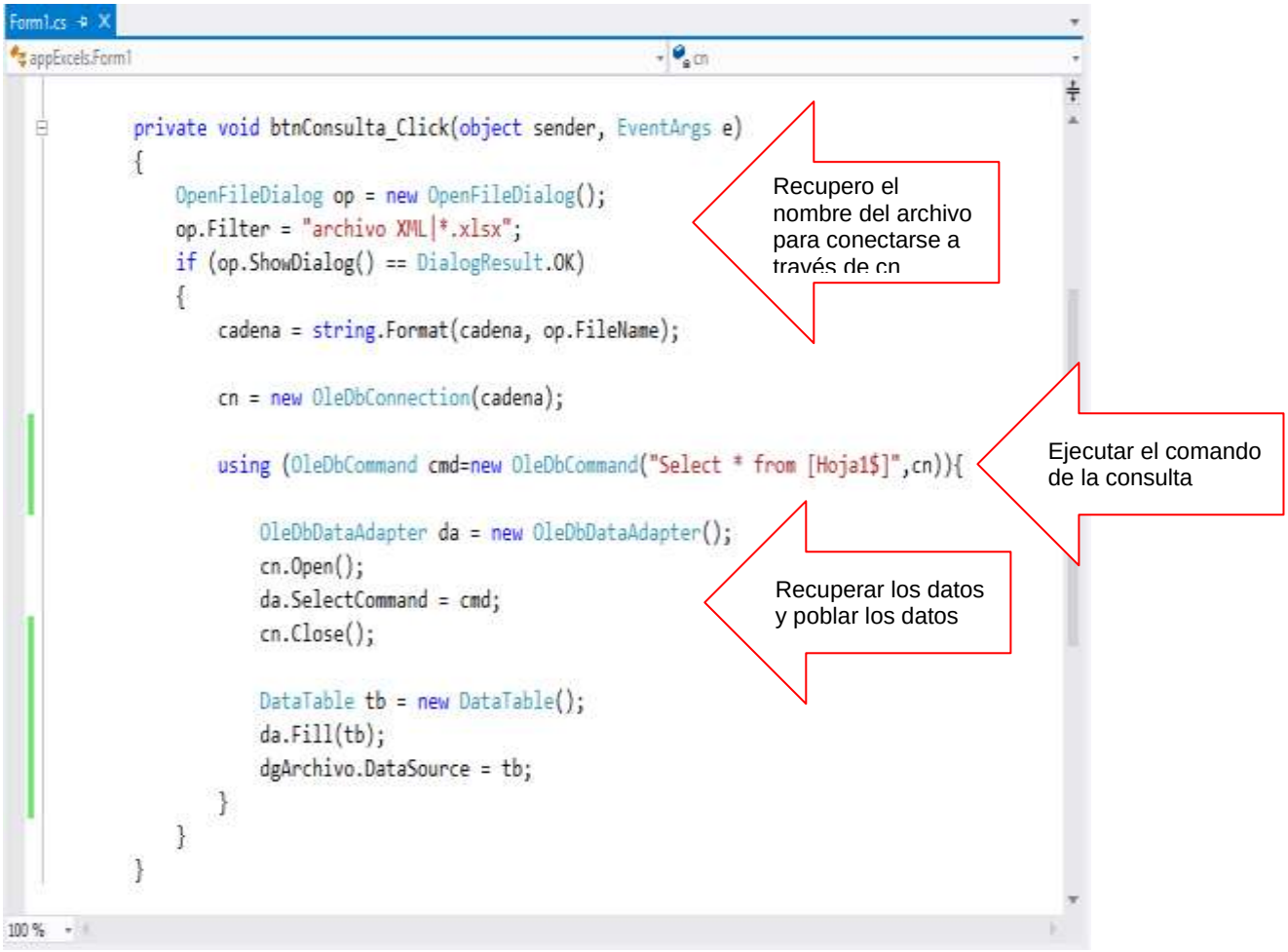


2. PROGRAMACIÓN.

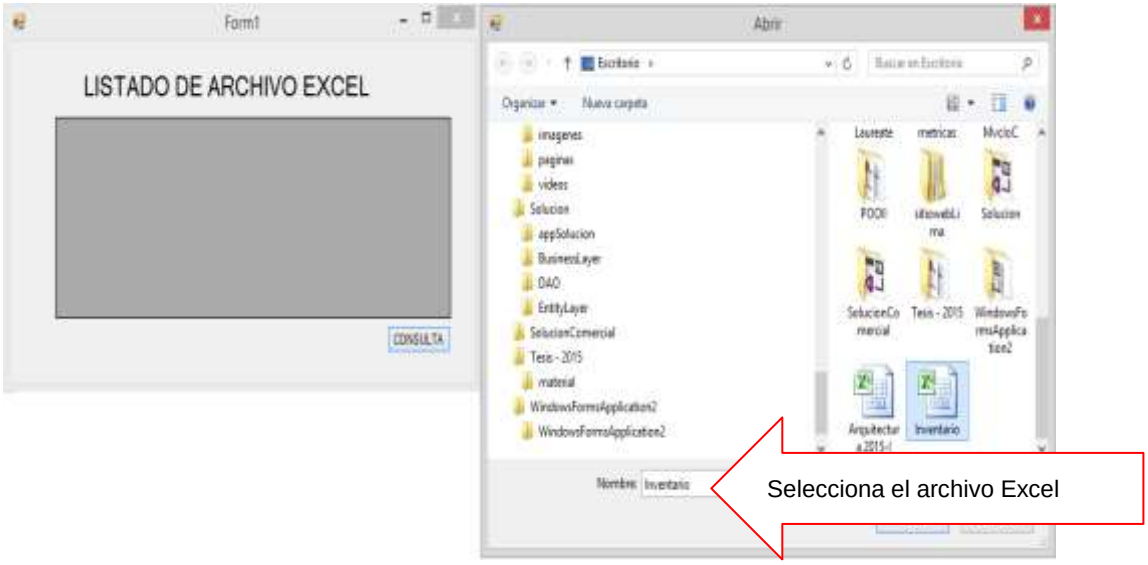
En la ventana de Programación, primero hacemos referencia a la librería System.Data.OleDB.



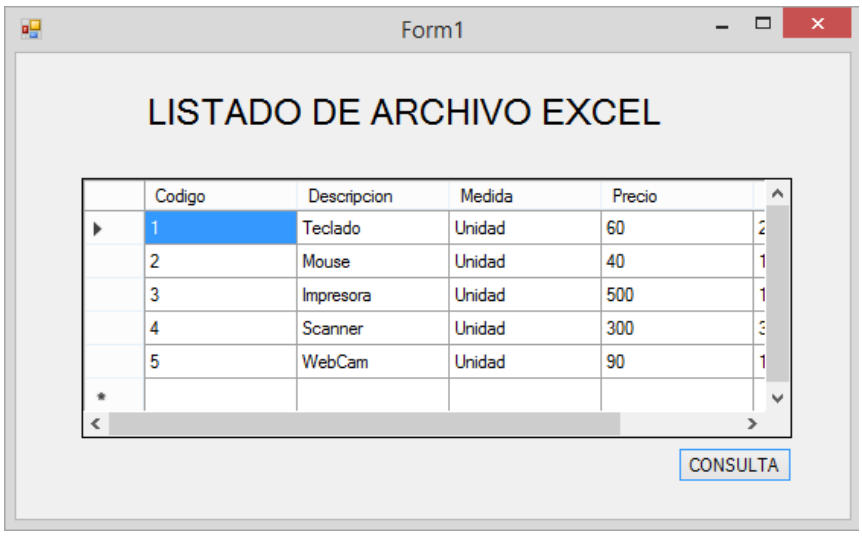
En el evento Clic del botón Consultal, procedemos a abrir el archivo de Excel; para luego conectarse a dicho origen de datos y luego cargar los datos al control



Al finalizar, presione F5, presiona el botón Consulta, donde se muestra un cuadro de dialogo para seleccionar el archivo, una vez seleccionado presiona el botón Abrir, donde se visualizar el contenido de la Hoja1 del Libro.



Visualizamos el contenido del archivo de Excel

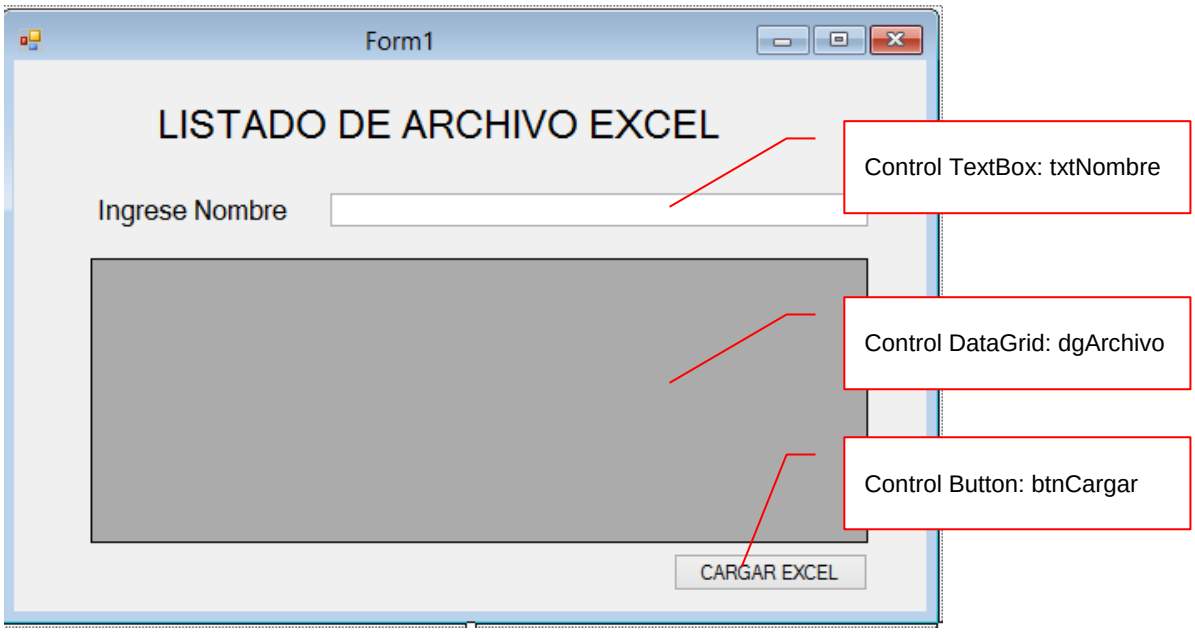


LABORATORIO 8.2

Se desea implementar un programa que permita consultar los datos almacenados en un archivo de Excel, cuya hoja se llama “INVENTARIO”; donde busque los registros por la inicial del nombre del producto

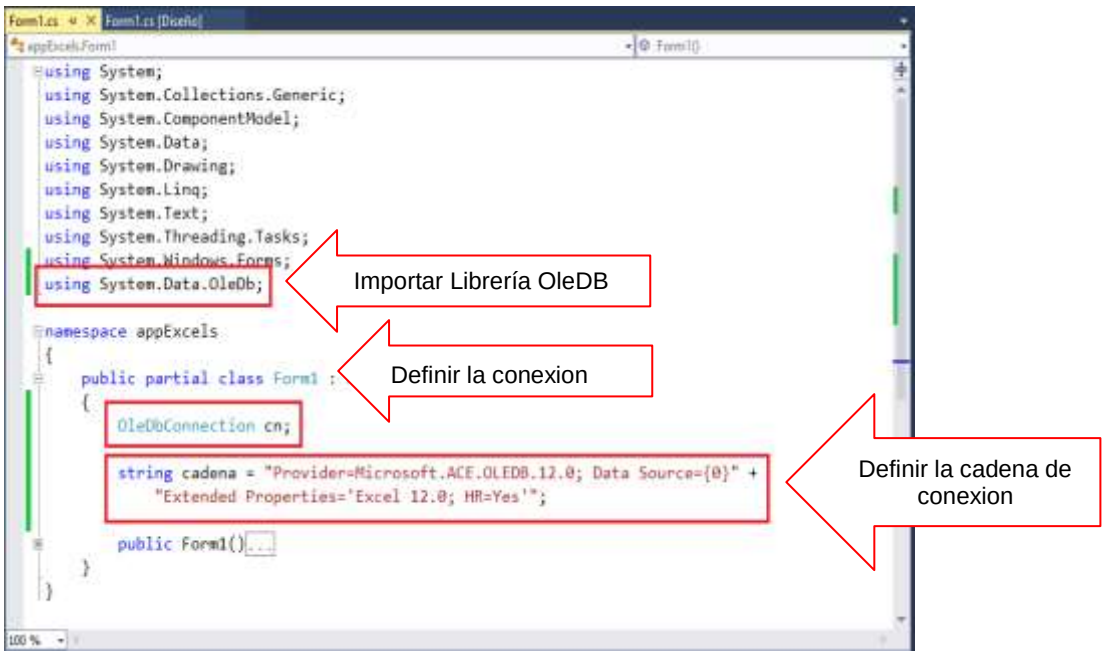
1. DISEÑO DEL FORMULARIO.

Diseña el formulario para el listado de archivos de Excel, tal como se muestra

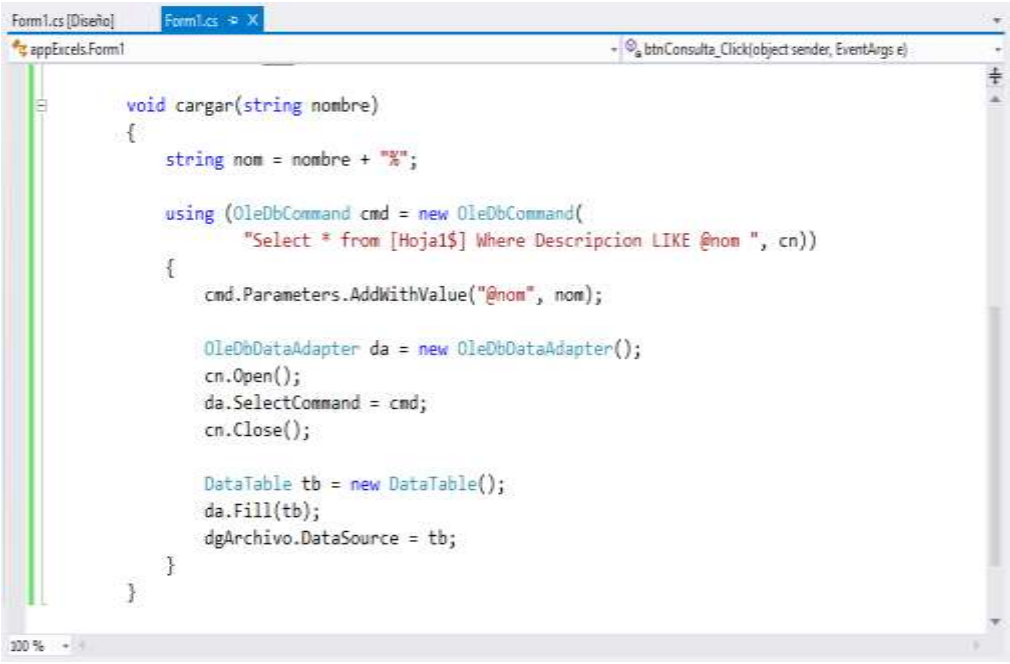


2. PROGRAMACIÓN.

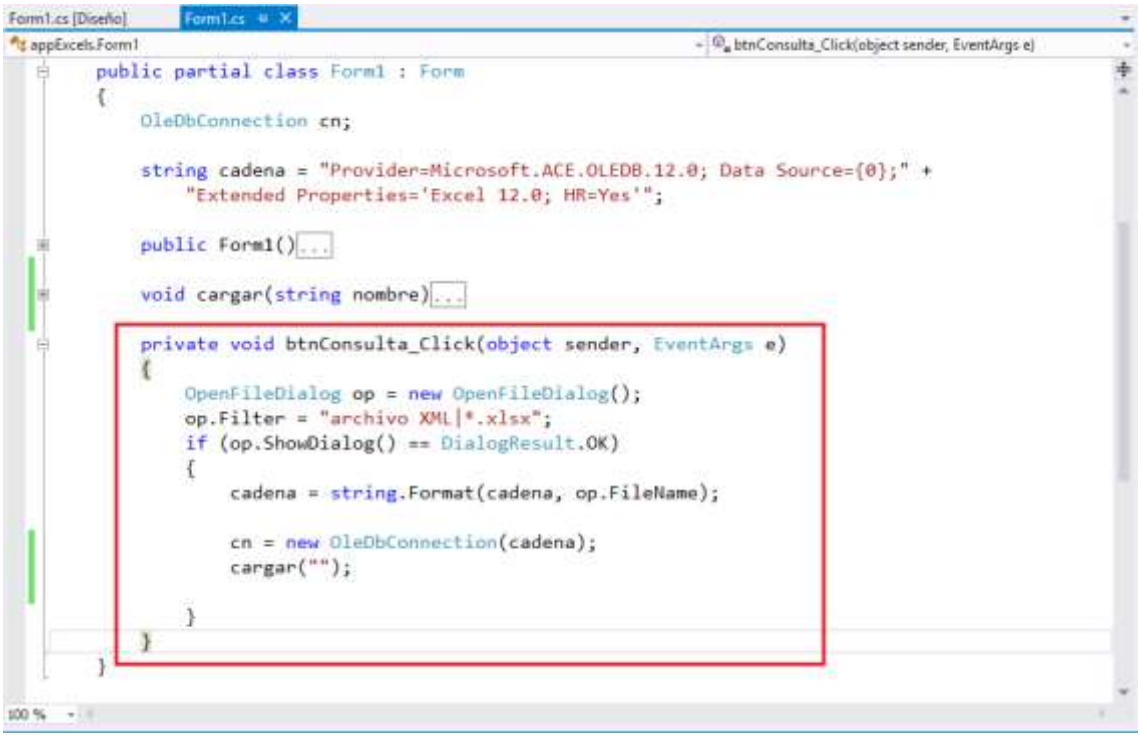
En la ventana de Programación, primero hacemos referencia a la librería System.Data.OleDB.



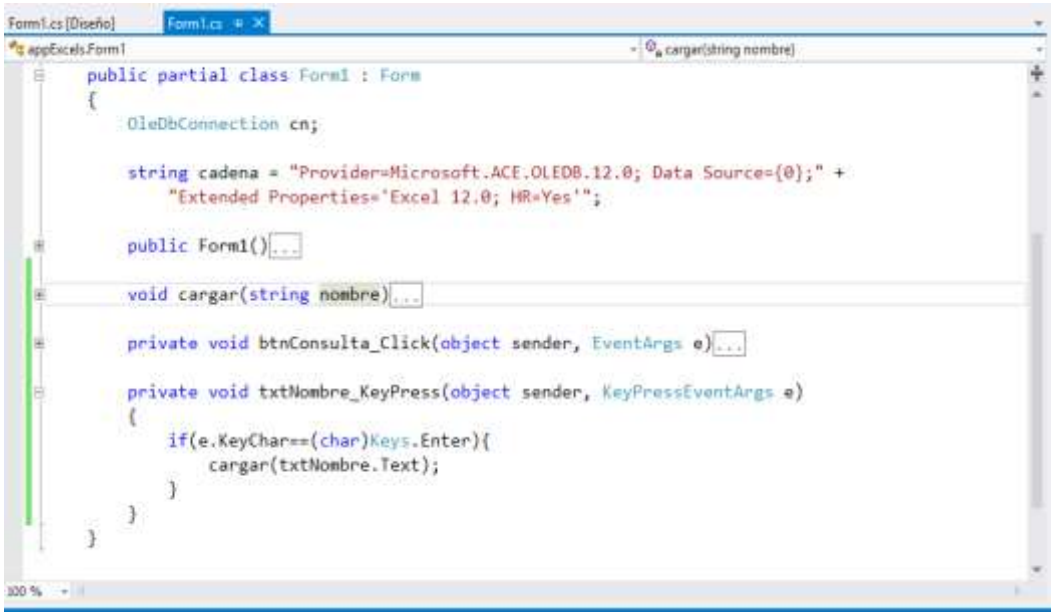
Defina el metodo Cargar, el cual muestra los registros de la búsqueda



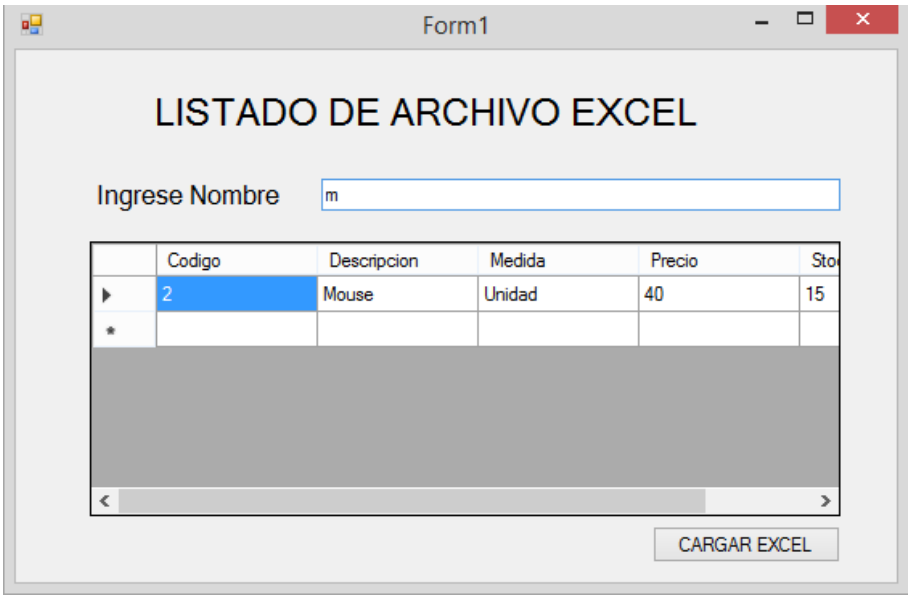
En el evento Clic del botón Consultal, procedemos a abrir el archivo de Excel; para luego conectarse a dicho origen de datos y luego cargar los datos al control



Programa el evento KeyPress de txtNombre, donde al presionar ENTER, listar los registros de inventario cuyo campo Descripcion empiece con las iniciales ingresadas desde el textBox



Presiona la tecla F5, cargue los datos al datagridView. Ingrese las iniciales del campo Descripcion en el textBox, al presionar ENTER se visualiza los registros que coincida con la condición.

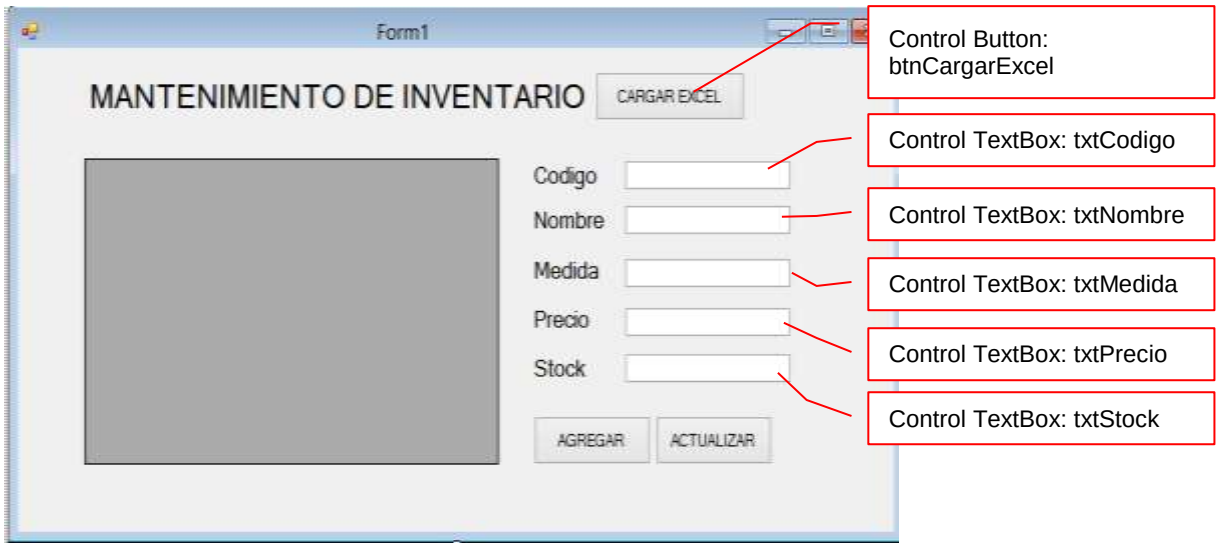


LABORATORIO 8.3

Se desea implementar un programa que permita agregar, actualizar y consultar los datos almacenados en un archivo de Excel, cuya hoja se llama “INVENTARIO”.

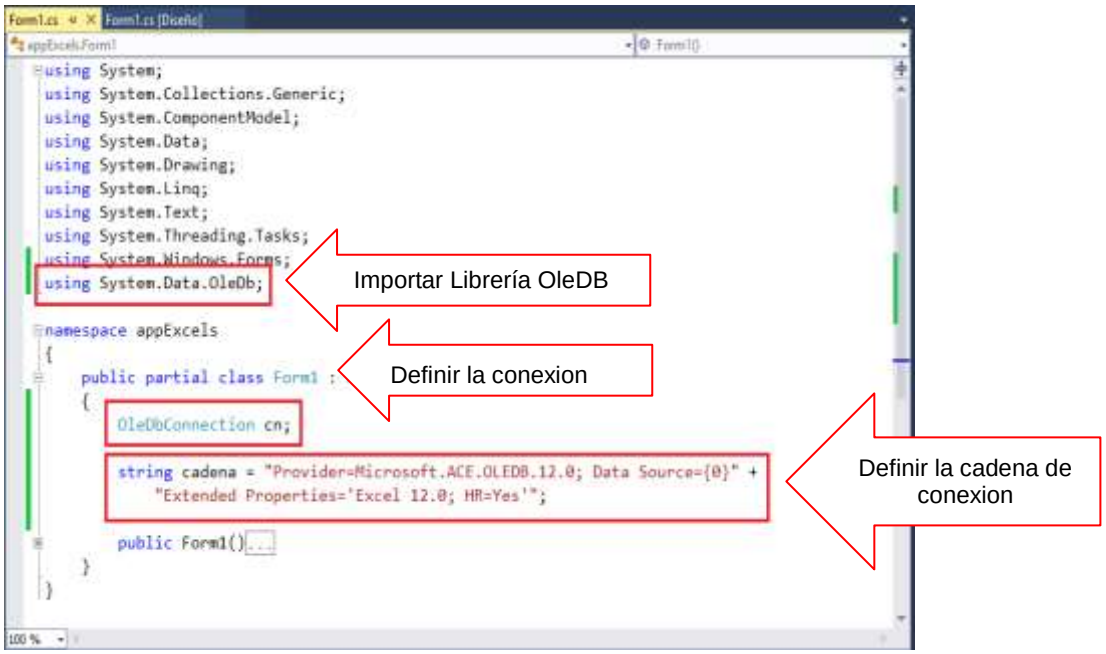
1. DISEÑO DEL FORMULARIO.

A continuación diseñe el formulario para el manejo de un archivo secuencial

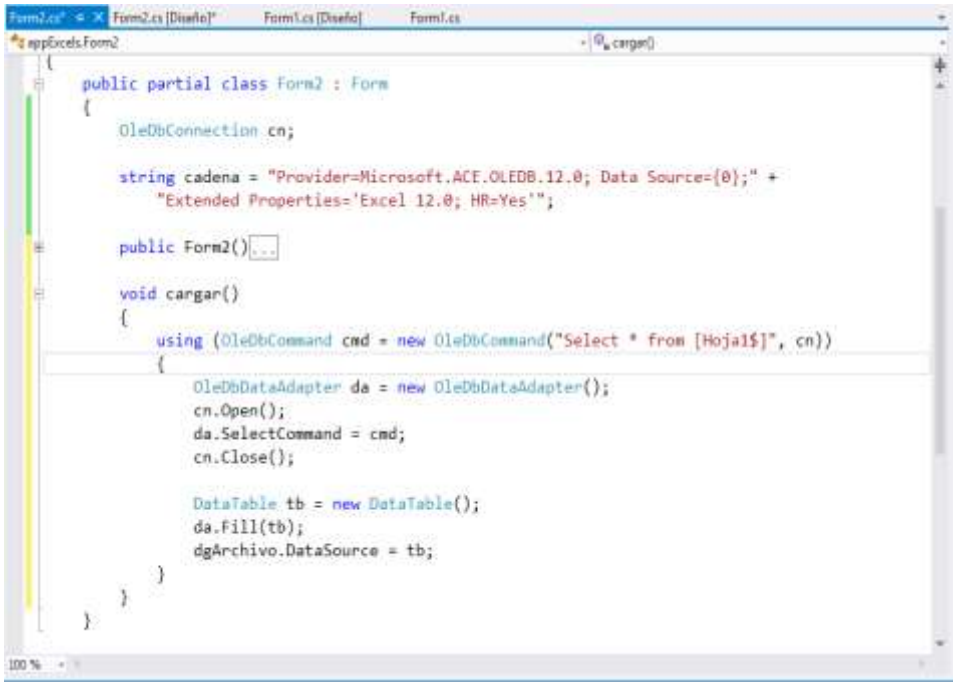


2. PROGRAMACIÓN.

En la ventana de Programación, primero hacemos referencia a la librería System.Data.OleDB. Dentro de la clase Form2, defina la conexión y su cadena de conexión para un Libro de Excel

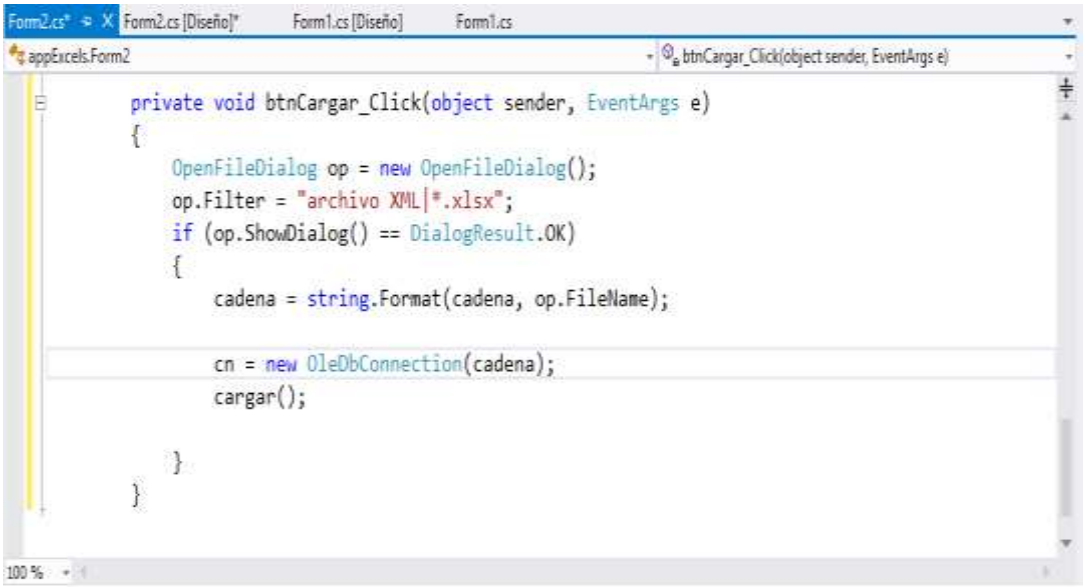


Defina el metodo Cargar() el cual permite abrir un archivo de Excel y visualizar el contenido en el control DataGridView.



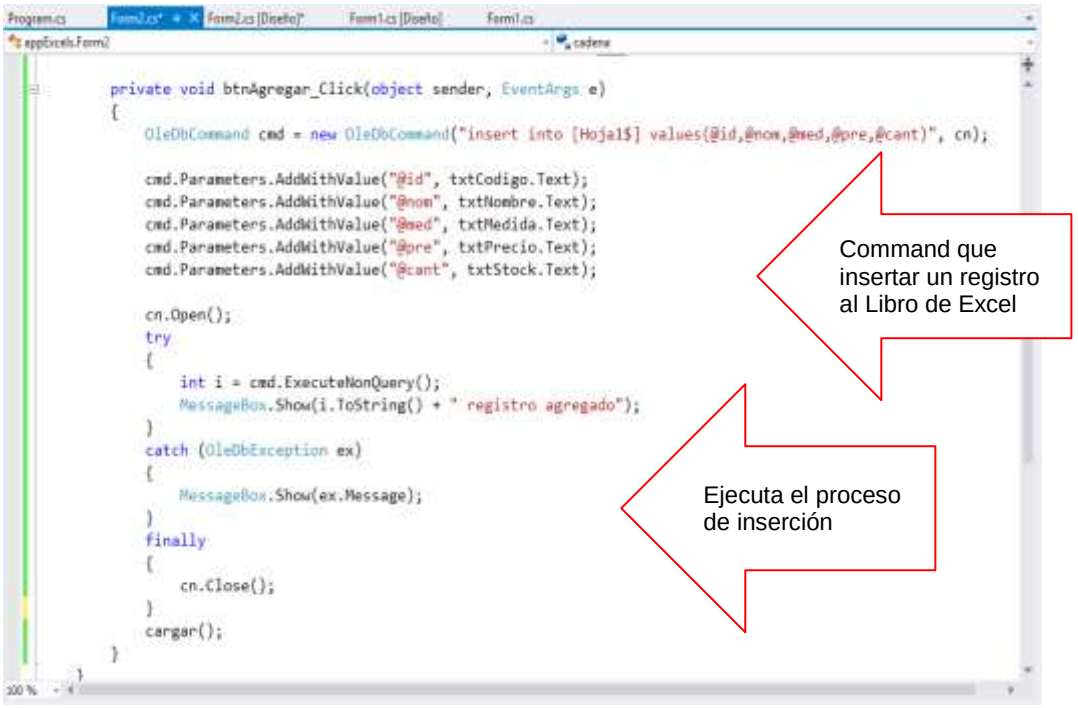
```
{
    public partial class Form2 : Form
    {
        OleDbConnection cn;

        string cadena = "Provider=Microsoft.ACE.OLEDB.12.0; Data Source={0};" +
            "Extended Properties='Excel 12.0; HR=Yes'";

        public Form2()...

        void cargar()
        {
            using (OleDbCommand cmd = new OleDbCommand("Select * from [Hoja1$]", cn))
            {
                OleDbDataAdapter da = new OleDbDataAdapter();
                cn.Open();
                da.SelectCommand = cmd;
                cn.Close();

                DataTable tb = new DataTable();
                da.Fill(tb);
                dgArchivo.DataSource = tb;
            }
        }
    }
```

En el evento Clic del botón CargarExcel, procedemos a abrir el archivo de Excel; para luego conectarse a dicho origen de datos y luego ejecutar el metodo Cargar().



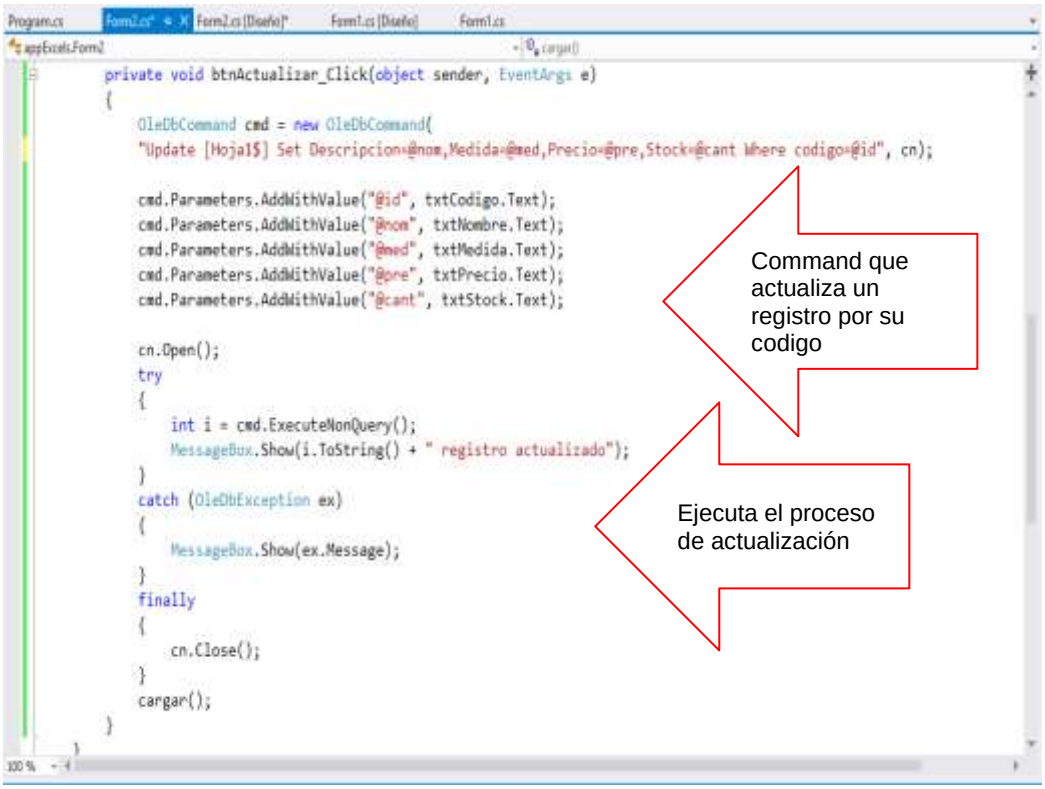
```
        private void btnCargar_Click(object sender, EventArgs e)
        {
            OpenFileDialog op = new OpenFileDialog();
            op.Filter = "archivo XML|*.xlsx";
            if (op.ShowDialog() == DialogResult.OK)
            {
                cadena = string.Format(cadena, op.FileName);

                cn = new OleDbConnection(cadena);
                cargar();

            }
        }
```

Programa el botón Agregar, donde permita insertar un registro al Libro Inventario de Excel



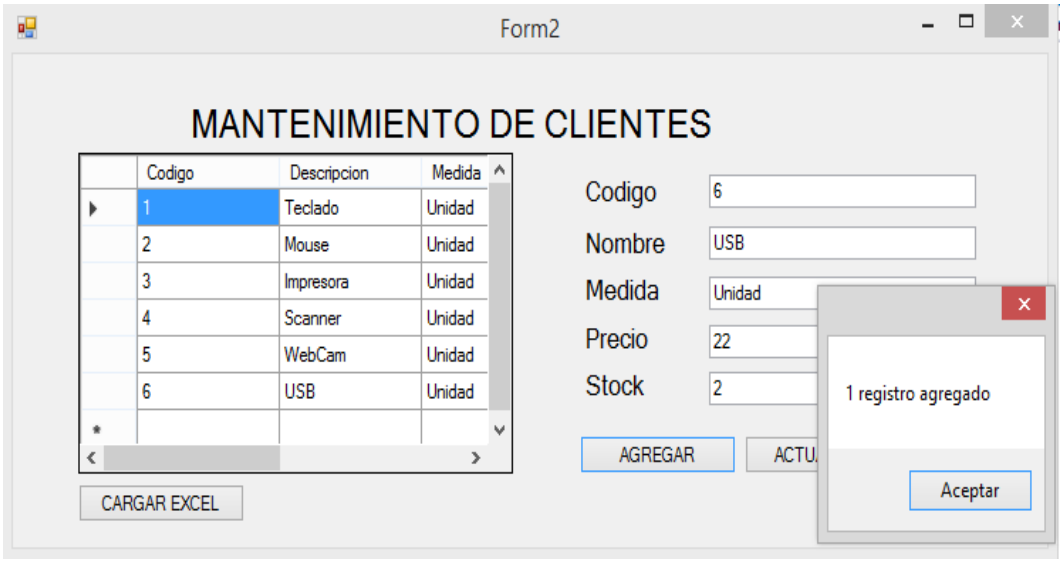

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    OleDbCommand cmd = new OleDbCommand("insert into [Hoja1$] values(@id,@nom,@med,@pre,@cant)", cn);

    cmd.Parameters.AddWithValue("@id", txtCodigo.Text);
    cmd.Parameters.AddWithValue("@nom", txtNombre.Text);
    cmd.Parameters.AddWithValue("@med", txtMedida.Text);
    cmd.Parameters.AddWithValue("@pre", txtPrecio.Text);
    cmd.Parameters.AddWithValue("@cant", txtStock.Text);

    cn.Open();
    try
    {
        int i = cmd.ExecuteNonQuery();
        MessageBox.Show(i.ToString() + " registro agregado");
    }
    catch (OleDbException ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        cn.Close();
    }
    cargar();
}
```

Programa el botón Actualizar, donde permita actualizar los datos de un registro al Libro Inventario por su columna código




```
private void btnActualizar_Click(object sender, EventArgs e)
{
    OleDbCommand cmd = new OleDbCommand(
    "Update [Hoja1$] Set Descripcion=@nom,Medida=@med,Precio=@pre,Stock=@cant Where codigo=@id", cn);

    cmd.Parameters.AddWithValue("@id", txtCodigo.Text);
    cmd.Parameters.AddWithValue("@nom", txtNombre.Text);
    cmd.Parameters.AddWithValue("@med", txtMedida.Text);
    cmd.Parameters.AddWithValue("@pre", txtPrecio.Text);
    cmd.Parameters.AddWithValue("@cant", txtStock.Text);

    cn.Open();
    try
    {
        int i = cmd.ExecuteNonQuery();
        MessageBox.Show(i.ToString() + " registro actualizado");
    }
    catch (OleDbException ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        cn.Close();
    }
    cargar();
}
```

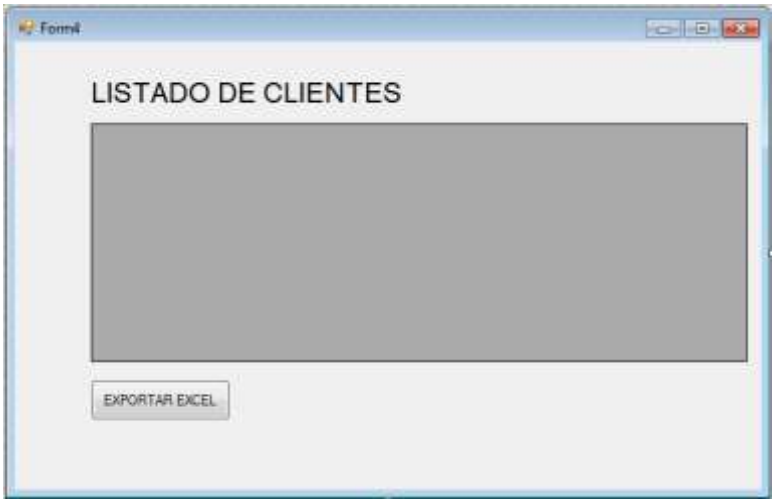
Presiona la tecla F5 para ejecutar el proceso de inserción y actualización de datos de los registros del Libro de Excel



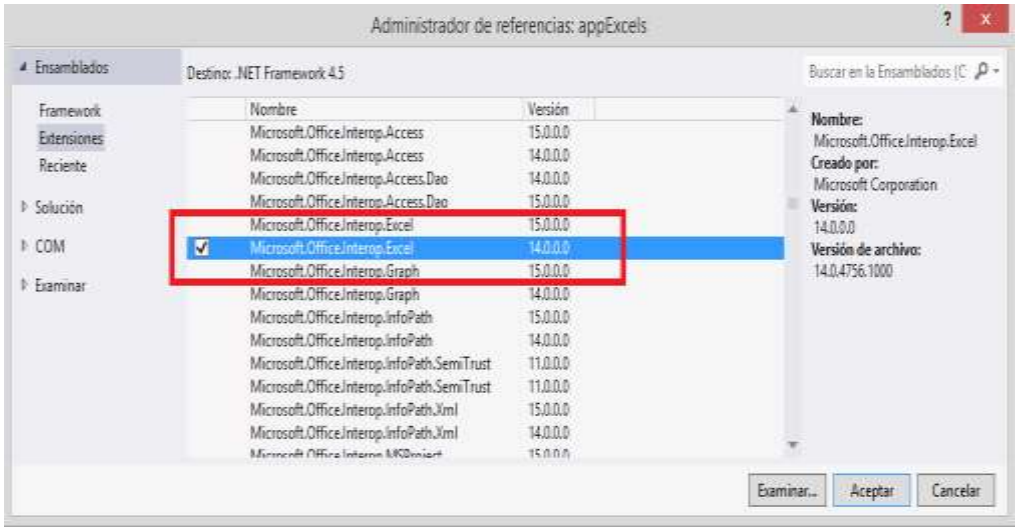
LABORATORIO 8.4

Se desea implementar un programa que permita exportar los datos desde una fuente de datos hacia un archivo de Excel.

DISEÑO DE FORMULARIO

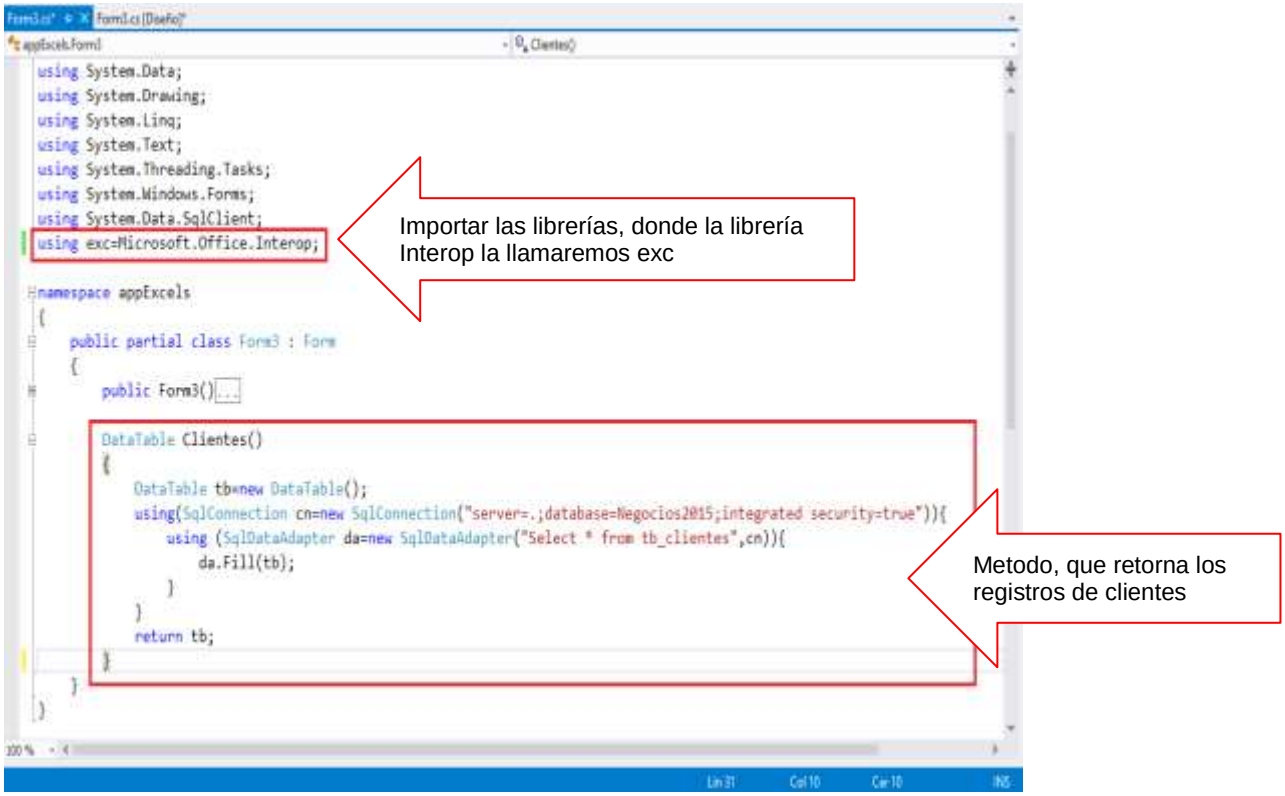


Agregar una referencia al proyecto, el ensamblado: Microsoft.Office.Interop.Excel, tal como se muestra; luego presione el botón Aceptar

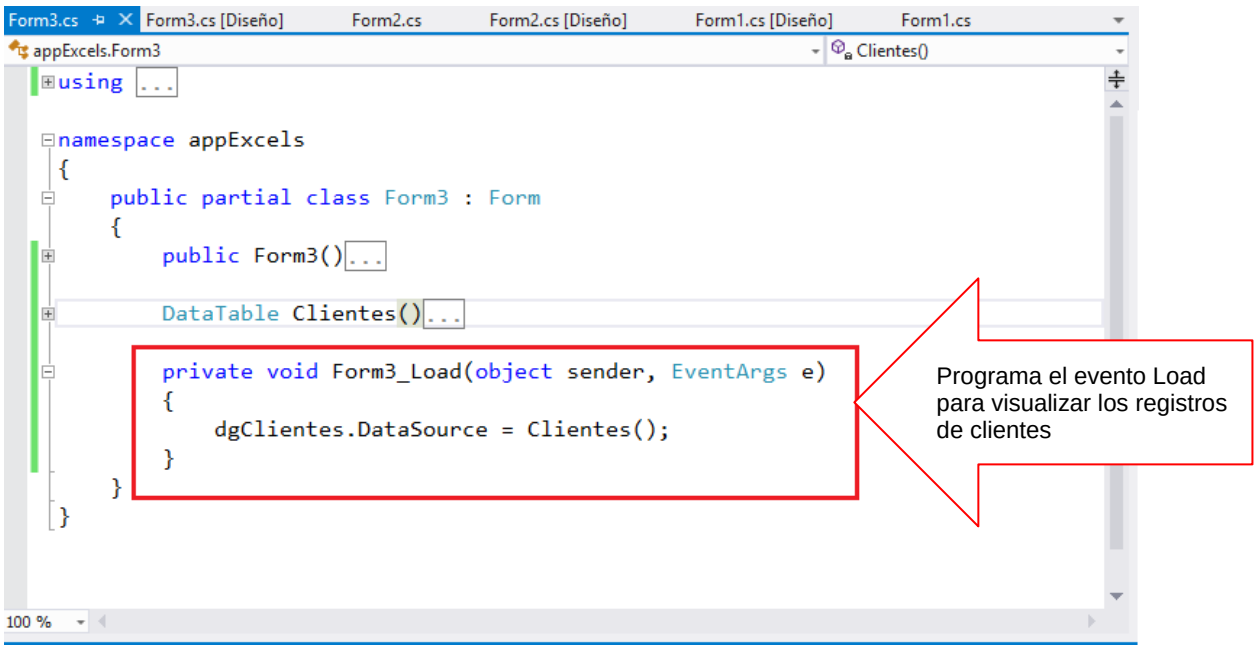


PROGRAMACION

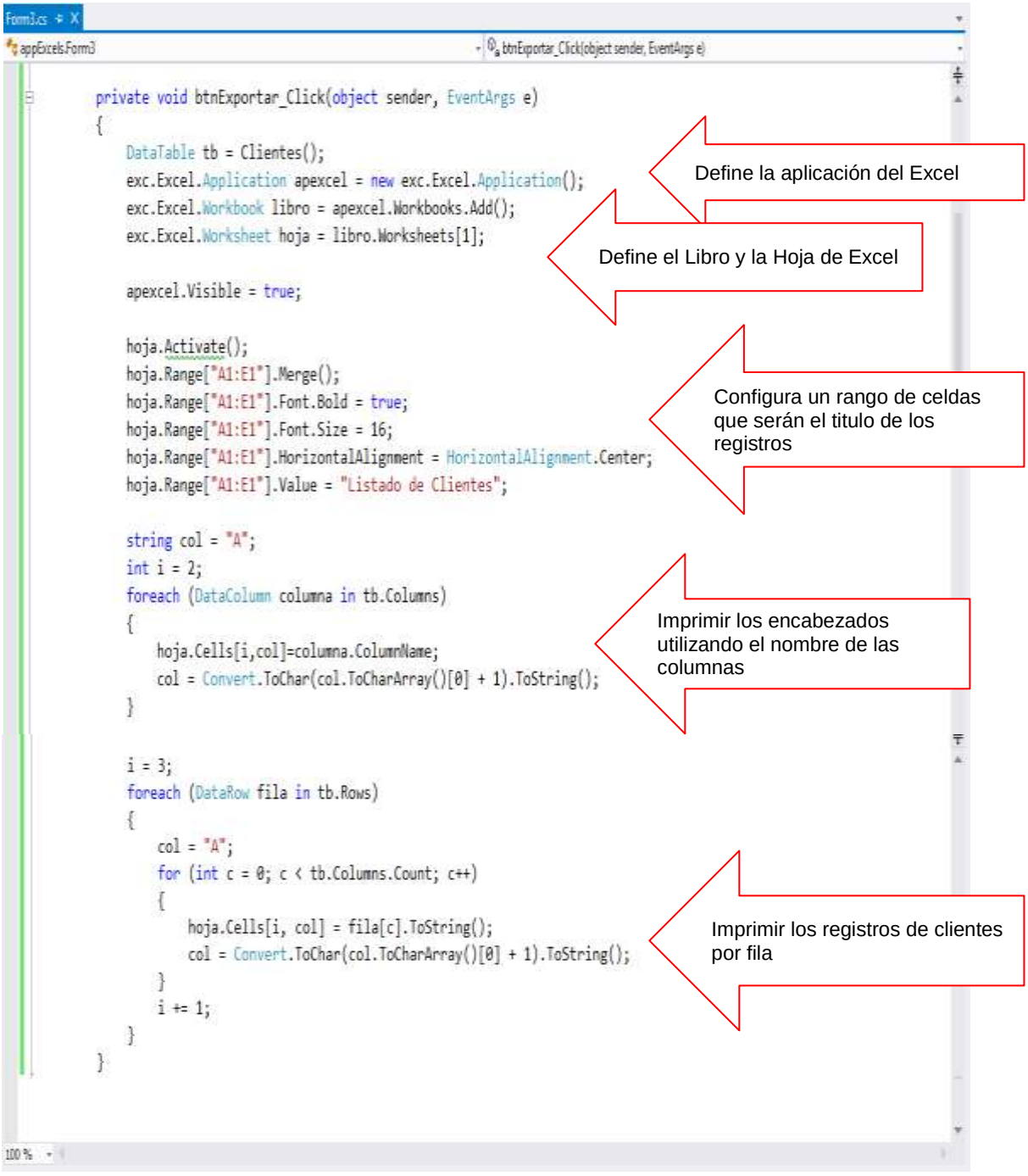
Importar las librerías de trabajo: System.Data.SqlClient y Microsoft.Office.Interop, tal como se muestra. Defina el metodo Listar donde retorna los registros de los clientes



Programa el evento Load del Formulario, para ejecutar el metodo Listar() visualizando los registros de la tabla tb_clientes en el control dgClientes, tal como se muestra

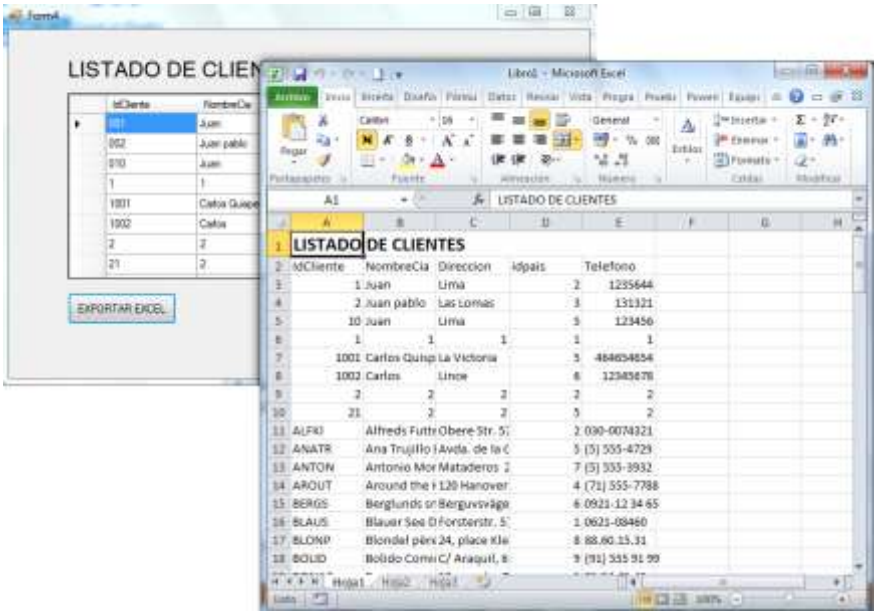


Programa el evento Exportar Excel, donde los registros de los clientes, serán almacenados en un archivo de Excel, el codigopara realizar este proceso se muestra a continuación.




```
private void btnExportar_Click(object sender, EventArgs e)
{
    DataTable tb = Clientes();
    exc.Excel.Application apexcel = new exc.Excel.Application();
    exc.Excel.Workbook libro = apexcel.Workbooks.Add();
    exc.Excel.Worksheet hoja = libro.Worksheets[1];

    apexcel.Visible = true;

    hoja.Activate();
    hoja.Range["A1:E1"].Merge();
    hoja.Range["A1:E1"].Font.Bold = true;
    hoja.Range["A1:E1"].Font.Size = 16;
    hoja.Range["A1:E1"].HorizontalAlignment = HorizontalAlignment.Center;
    hoja.Range["A1:E1"].Value = "Listado de Clientes";

    string col = "A";
    int i = 2;
    foreach (DataColumn columna in tb.Columns)
    {
        hoja.Cells[i,col]=columna.ColumnName;
        col = Convert.ToChar(col.ToCharArray()[0] + 1).ToString();
    }

    i = 3;
    foreach (DataRow fila in tb.Rows)
    {
        col = "A";
        for (int c = 0; c < tb.Columns.Count; c++)
        {
            hoja.Cells[i, col] = fila[c].ToString();
            col = Convert.ToChar(col.ToCharArray()[0] + 1).ToString();
        }
        i += 1;
    }
}
```

Presiona la tecla F5, donde se visualiza los registros de clientes. Al presionar elboton Exportar, los registros se visualizan en una hoja de Excel



Resumen

- El motor de base de datos de Microsoft Jet puede tener acceso a los datos en otros formatos de archivo de base de datos, como libros de Excel, a través de los controladores instalables de método de acceso secuencial indizado (ISAM).
- La información que se pasa al proveedor para la conexión es la ruta + nombre del libro y la versión. Como versión de proveedor se debe usar la 4.0, pues la 3.51 no admite controladores Jet ISAM. Como versión de Excel se pasa:
  - Excel 5.0: para Excel 95
  - Excel 8.0: para Excel 2000/XP/2003/2007
  - Excel 12.0: para Excel 2007
- Cuando queremos abrir un libro de Excel de versión 2002 o anterior, es necesario que establezcamos como proveedor Microsoft.JET.OLEDB.4.0; si queremos abrir un libro de Excel 2007, es necesario que establezcamos como proveedor Microsoft.ACE.OLEDB.12.0.
- Puede recuperar los registros de una base de datos mediante uno de estos dos enfoques en ADO.NET: con un conjunto de datos o con un objeto DataReader.
- Con ADO.NET se puede insertar y actualizar registros en un libro de Excel: Ejecuta directamente un comando para insertar o actualizar registros de uno en uno. Para ello, puede crear un objeto **OLEDbCommand** en su conexión y establecer su propiedad **CommandText** en un comando y a continuación llame el método **ExecuteNonQuery**
- Aunque el proveedor OLE DB de Jet le permite insertar y actualizar registros en un libro de Excel, no permite las operaciones de eliminación. Si intenta realizar una operación de eliminación de uno o más registros, recibirá el siguiente mensaje de error: Este ISAM no admite la eliminación de datos en una tabla vinculada

- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

http://msmvps.com/blogs/cwalzer/pages/exceladoaspnet.aspx

http://www.devjoker.com/contenidos/catss/484/LecturaEscritura-de-archivos-Excel-con-ADONET.aspx

http://support.microsoft.com/kb/316934/es

http://ripertec.wordpress.com/2011/04/23/exportar-datagridview-a-excel/#more-81

UNIDAD

4

MANIPULACION DE DATOS: MODELO RELACIONAL DE OBJETOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno realiza consultas y actualización de datos a través del lenguaje de consulta integrado utilizando la plataforma ORM en un entorno de una aplicación Windows.

Temario

Tema 9: LINQ (4 horas)

1. LINQ
   1.1. Plataforma de acceso a Datos
   1.2. LINQ to SQL: arquitectura, DataContext
   1.3. Operaciones de consulta y actualización de datos utilizando LINQ

ACTIVIDADES PROPUESTAS

- Los alumnos conocer el modelo de datos orientado a objetos
- Los alumnos realizan operaciones de consulta utilizando el modelo de clase LINQ to SQL
- Los alumnos realizan operaciones de actualización utilizando el modelo de clase LINQ to SQL

9. LINQ

Actualmente, muchos programadores empresariales deben usar dos (o más) lenguajes de programación: un lenguaje de alto nivel para las capas de presentación y lógica empresarial (como Visual C# o Visual Basic) y un lenguaje de consulta para interactuar con la base de datos (como Transact-SQL). Esto requiere que el programador tenga conocimientos de varios idiomas para ser efectivo y también causa discrepancias de idiomas en el entorno de desarrollo. Por ejemplo, una aplicación que utiliza API de acceso a datos para ejecutar una consulta en una base de datos especifica la consulta como un literal de cadena usando comillas. Esta cadena de consulta es ilegible y no se comprueba si contiene errores, tales como una sintaxis no válida o si existen las columnas o las filas a las que hace referencia. No hay ninguna comprobación de tipo de los parámetros de consulta y tampoco hay compatibilidad con IntelliSense.

Language-Integrated Query (LINQ) permite a los programadores formar consultas basadas en conjuntos en el código de su aplicación, sin tener que usar un lenguaje de consulta independiente. Se puede escribir consultas de LINQ en varios orígenes de datos enumerables, como estructuras de datos en memoria, documentos XML, bases de datos SQL y objetos DataSet. Aunque esos orígenes de datos enumerables se implementan de varias formas, todos revelan las mismas construcciones de lenguaje y sintaxis. Como las consultas se pueden formar en el lenguaje de programación mismo, no es necesario utilizar otro lenguaje de consultas que esté incrustado como literales de cadena que el compilador no pueda entender o comprobar. La integración de consultas en el lenguaje de programación también permite a los programadores de Visual Studio ser más productivos proporcionando comprobación de sintaxis y tipo en tiempo de compilación e IntelliSense. Estas características reducen la necesidad de depuración y corrección de errores de consultas.

9.1 PLATAFORMA DE ACCESO A DATOS

Existen tres tecnologías Language-Integrated Query (LINQ) de ADO.NET independientes: LINQ to DataSet, LINQ to SQL y LINQ to Entities. LINQ to DataSet proporciona consultas más ricas y optimizadas de DataSet, LINQ to SQL permite consultar directamente los esquemas de base de datos de SQL Server y LINQ to Entities permite consultar un Entity Data Model. El siguiente diagrama proporciona una visión general de cómo se relacionan las tecnologías ADO.NET LINQ con lenguajes de programación de alto nivel y orígenes de datos habilitados para LINQ.

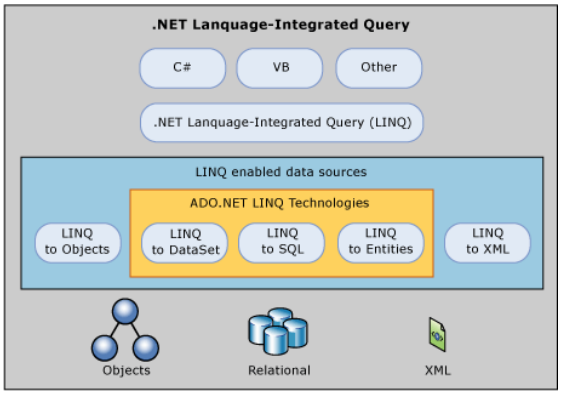


Figura 1: LINQ y ADO.NET
Referencia: http://msdn.microsoft.com/en-us/library/bb399365(v=vs.110).aspx

9.2 LINQ to SQL: ARQUITECTURA, DATACONTEXT

Language-Integrated Query (LINQ) es una innovación introducida en Visual Studio 2008 y .NET Framework versión 3.5 que elimina la distancia que separa el mundo de los objetos y el mundo de los datos.

Tradicionalmente, las consultas con datos se expresan como cadenas sencillas, sin comprobación de tipos en tiempo de compilación ni compatibilidad con IntelliSense. Además, es necesario aprender un lenguaje de consultas diferente para cada tipo de origen de datos: bases de datos SQL, documentos XML, servicios Web diversos, etc. LINQ convierte una consulta en una construcción de lenguaje de primera clase en C# y Visual Basic. Las consultas se escriben para colecciones de objetos fuertemente tipadas, utilizando palabras clave del lenguaje y operadores con los que se está familiarizado. La ilustración siguiente muestra una consulta LINQ parcialmente completada en una base de datos SQL Server en C#, con comprobación de tipos completa y compatibilidad con IntelliSense.

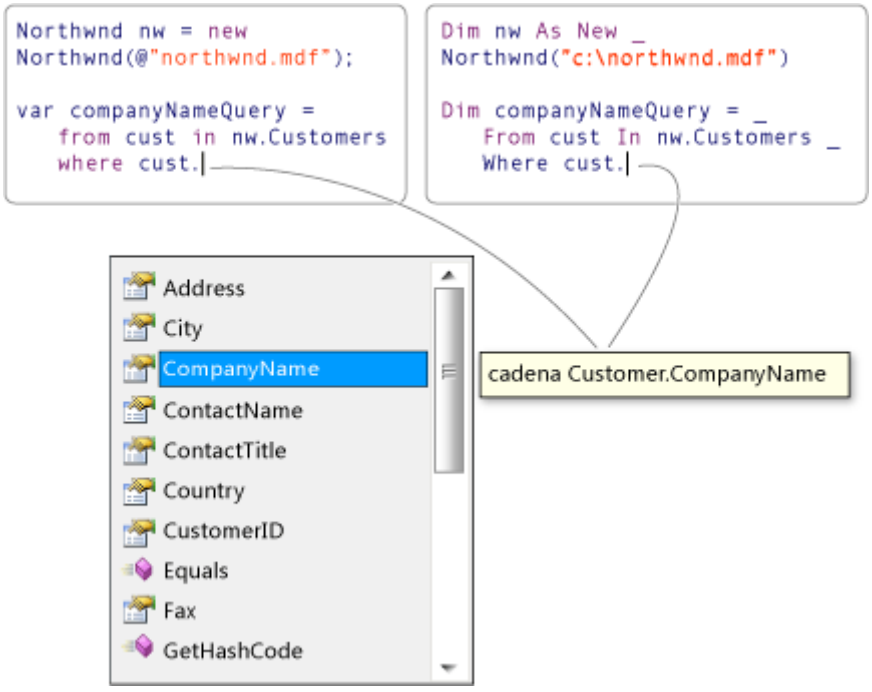


Figura 2: LINQ
Referencia: http://msdn.microsoft.com/es-es/library/bb397897(v=vs.110).aspx

En Visual Studio se pueden escribir consultas LINQ en Visual Basic o en C# con bases de datos SQL Server, documentos XML, conjuntos de datos ADO.NET y cualquier colección de objetos que admita IEnumerable o la interfaz genérica IEnumerable(Of T). También se ha previsto la compatibilidad de LINQ con ADO.NET Entity Framework, y otros fabricantes se encuentran escribiendo proveedores LINQ para muchos servicios Web y otras implementaciones de bases de datos.

Puede utilizar consultas LINQ en proyectos nuevos o junto a consultas que no son LINQ en proyectos existentes. El único requisito es que el proyecto esté orientado a .NET Framework 3.5 o posterior.

Arquitectura de LINQ to SQL

Figura 3: Arquitectura LINQ to SQL
Referencia: http://slideplayer.es/slide/1677515/

DataContext.

DataContext es el origen de todas las entidades asignadas en una conexión de base de datos. Realiza un seguimiento de los cambios realizados en todas las entidades recuperadas y mantiene una "memoria caché de identidad" que garantiza que las entidades que se recuperan más de una vez se representan utilizando la misma instancia de objeto.

En general, una instancia de DataContext está diseñada para que dure una "unidad de trabajo", sea cual sea la definición de ese término en la aplicación. DataContext es un objeto ligero cuya creación no es costosa. Una aplicación LINQ to SQL típica crea instancias de DataContext en el ámbito de métodos o como miembro de clases de duración corta que representan un conjunto lógico de operaciones de base de datos relacionadas.

9.3 Operaciones de consulta y actualización de datos utilizando LINQ

En LINQ to SQL, el modelo de datos de una base de datos relacional se asigna a un modelo de objetos expresado en el lenguaje de programación del programador. Cuando la aplicación se ejecuta, LINQ to SQL convierte a SQL las consultas integradas en el lenguaje en el modelo de objetos y las envía a la base de datos para su ejecución. Cuando la base de datos devuelve los resultados, LINQ to SQL los vuelve a convertir en objetos con los que pueda trabajar en su propio lenguaje de programación.

Los desarrolladores de Visual Studio normalmente utilizan el Object Relational Designer, que proporciona una interfaz de usuario para implementar muchas de las características de LINQ to SQL.

La documentación que se incluye con esta versión de LINQ to SQL describe las unidades de creación básicas, los procesos y las técnicas que necesita para crear aplicaciones LINQ to SQL.

Operaciones con LINQ

1. **Selección**: Para la selección (proyección), basta con que escriba una consulta LINQ en su propio lenguaje de programación y, después, la ejecute para recuperar los resultados. LINQ to SQL traduce automáticamente todas las operaciones necesarias a las operaciones SQL correspondientes con la que ya está familiarizado. Para obtener más información, vea LINQ a SQL [LINQ to SQL].
2. **Inserción**: Para realizar una operación Insert de SQL, basta con que agregue objetos al modelo de objetos que ha creado y llame a SubmitChanges en DataContext.
3. **Actualización**: Para realizar una operación Update en una entrada de base de datos, primero recupere el elemento y modifíquelo directamente en el modelo de objetos. Después de haber modificado el objeto, llame a SubmitChanges en DataContext para actualizar la base de datos.
4. **Eliminación**: Para realizar una operación Delete en un elemento, quite el elemento de la colección a la que pertenece y, a continuación, llame al método SubmitChanges en DataContext para confirmar el cambio.

Expresiones Lambda y el lenguaje LINQ

Las expresiones lambda junto a las instrucciones que proporciona LINQ (Lenguaje de Consulta Integrado) resultan en una dupla complementarias en materia de desarrollo de software. La inclusión y el uso de expresiones lambda en consultas de base de datos utilizando LINQ son elementales y poderosos.

Cabe señalar que mediante LINQ resulta posible construir suficiente y eficiente código para, no solamente poder acceder bases de datos, tablas, consultas, procedimientos almacenados, triggers, etc., sino que también resulta posible manipular dichos datos en dichos centros de datos de forma directa y eficaz. No se requiere de mucho esfuerzo y su principal ventaja es que no es necesario aprender por completo el lenguaje SQL y todos sus artilugios, de manera que de esta forma, se pueda explotar todas sus virtudes. LINQ ofrece una forma menos exhaustiva de desarrollar software y para ello recurre al uso de expresiones lambda entre otras técnicas.

Operadores de LINQ

Las clases del espacio de nombres System.Linq y el resto de los espacios de nombres que admiten consultas LINQ incluyen métodos a los que puede llamar para crear y refinar consultas basándose en las necesidades de la aplicación.

Operador	Descripción
Selección	Select<expr>, SelectMany<expr>
Filtro	Where<expr>, Distinct
Condición	Any(<expr>), All(<expr>), Contains(<expr>)
Unión	<expr> Join <expr> On <expr> Equals <expr>
Agrupación	Group By <expr>, <expr> Into <expr>, <expr> Group Join <decl> On <expr> Equals <expr> Into <expr>
Agregación	Count([<expr>]), Sum(<expr>), Min(<expr>), Max(<expr>), Avg(<expr>)
Partición	Skip [While] <expr>, Take [While] <expr>
Conjunto	Union, Intersect, Except
Ordenación	Order By <expr>, <expr> [Ascending | Descending]

Figura 3: Operadores de LINQ

From (Cláusula): Se necesita una cláusula From o Aggregate para iniciar una consulta. Una cláusula From especifica una colección de origen y una variable de iteración de una consulta. Por ejemplo:

```
Dim names = From cust In customers _
        Where cust.State = "WA" _
        Select cust.CompanyName
```

Select (Cláusula): Opcional. Declara un conjunto de variables de iteración de una consulta. Por ejemplo:

```
Dim customerList = From cust In customers _
                Select cust.CompanyName, cust.CustomerID
```

Where (Cláusula): Opcional. Especifica una condición de filtrado de una consulta. Por ejemplo:

```
Dim names = From product In products _
        Where product.Category = "Beverages" _
        Select product.Name
```

Order By (Cláusula): Opcional. Especifica el criterio de ordenación de las columnas de una consulta. Por ejemplo:

```
Dim titlesAscendingPrice = From b In books  Order By b.price
```

Distinct (Cláusula): Opcional. Restringe los valores de la variable de iteración actual para eliminar los valores duplicados de los resultados de la consulta. Por ejemplo:

```
Dim cities = From item In customers _
         Select item.City Distinct
```

LINQ y procedimientos almacenados

LINQ to SQL usa métodos del modelo de objetos para representar procedimientos almacenados en la base de datos. Los métodos se designan como procedimientos almacenados aplicando el atributo FunctionAttribute y, si es necesario, el atributo ParameterAttribute.

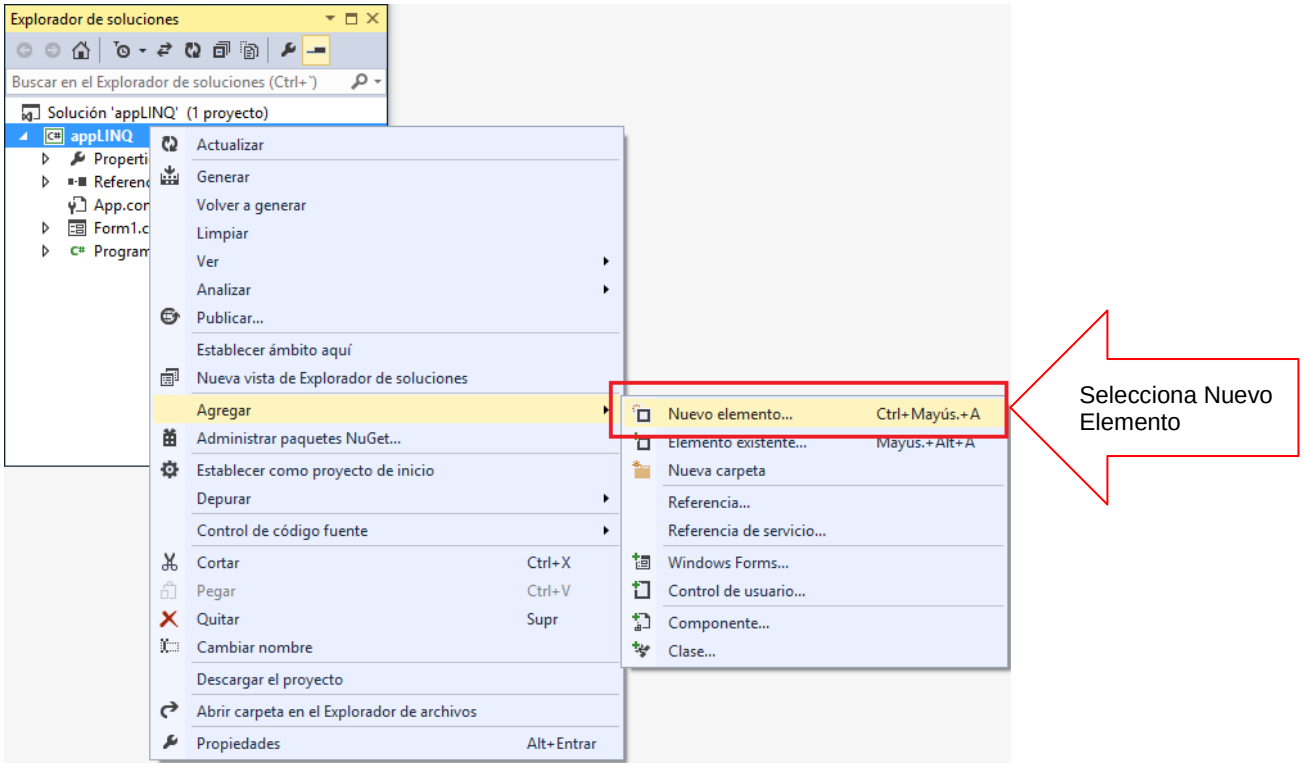
Los desarrolladores que utilizan Visual Studio emplearían normalmente Object Relational Designer para asignar procedimientos almacenados. Los temas de esta sección muestran cómo formar y llamar a estos métodos en la aplicación si escribe su propio código.

LABORATORIO 9.1
Consulta utilizando LINQ to SQL

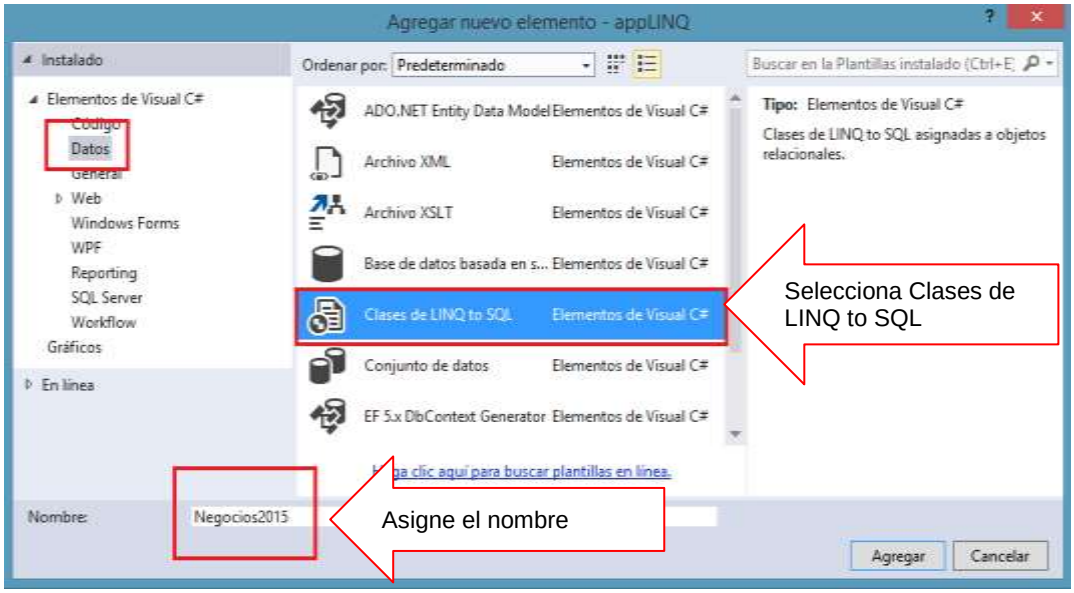
Se desea implementar una aplicación que permite realizar una consulta de la tabla tb_clientes utilizando el modelo LINQ to SQL

DISEÑO DEL DATACONTEXT.

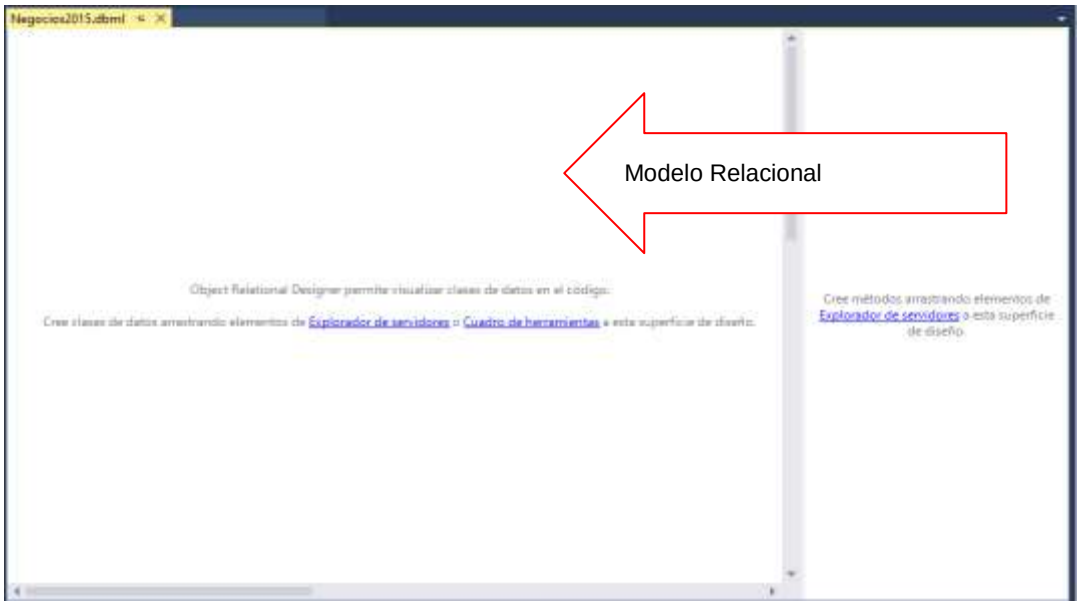
Antes de comenzar a utilizar el LINQ to SQL, debemos diseñar el DataContext. Desde el proyecto selecciona la opción NUEVO ELEMENTO, tal como se muestra.



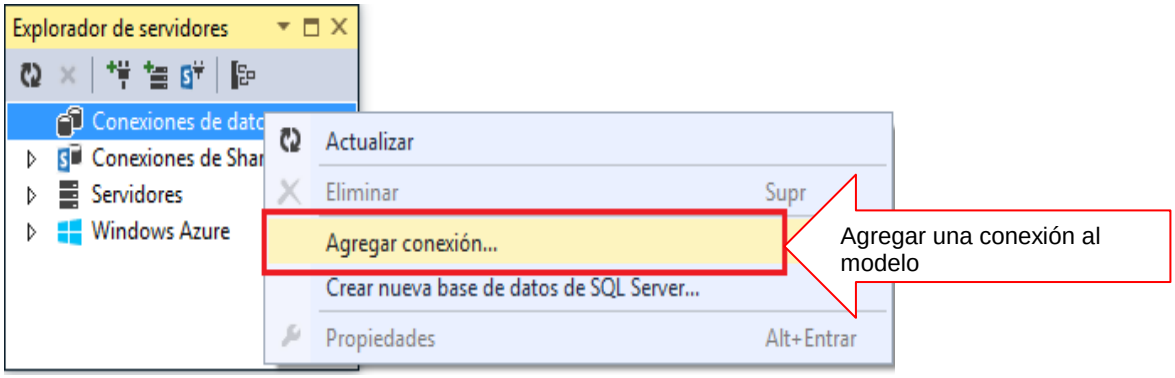
Selección desde la opción Datos, la plantilla de LINQ to SQL; asigne un nombre.



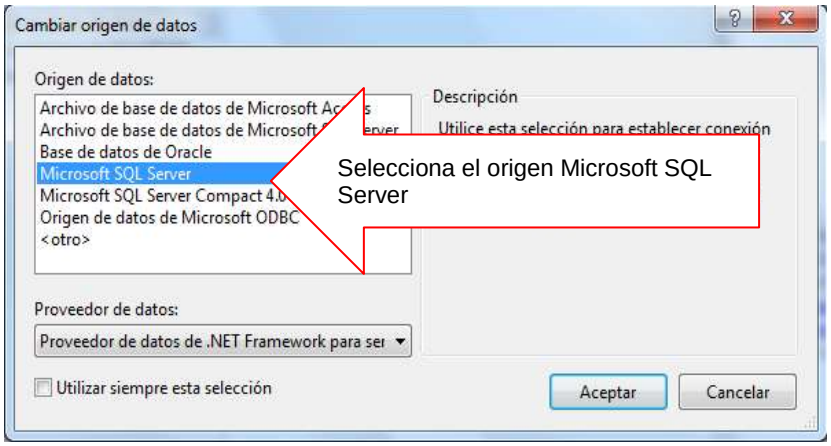
Modelo de Contexto de Negocios2015.



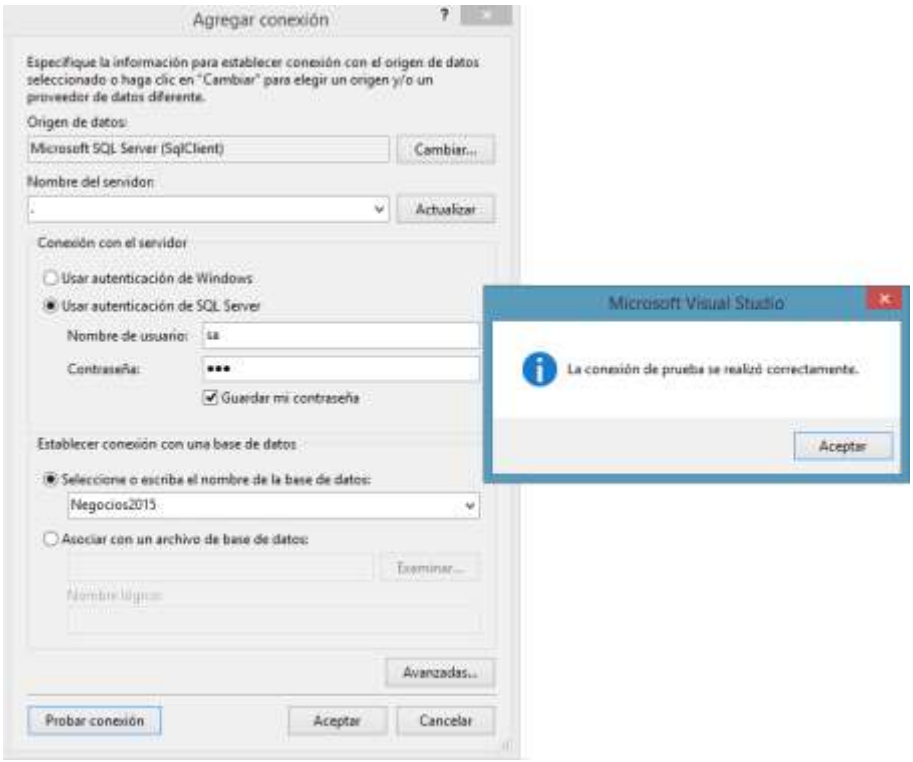
Para agregar una conexión, desde la ventana Explorador de servidores, selecciona la opción **Agregar conexión**… tal como se muestra.



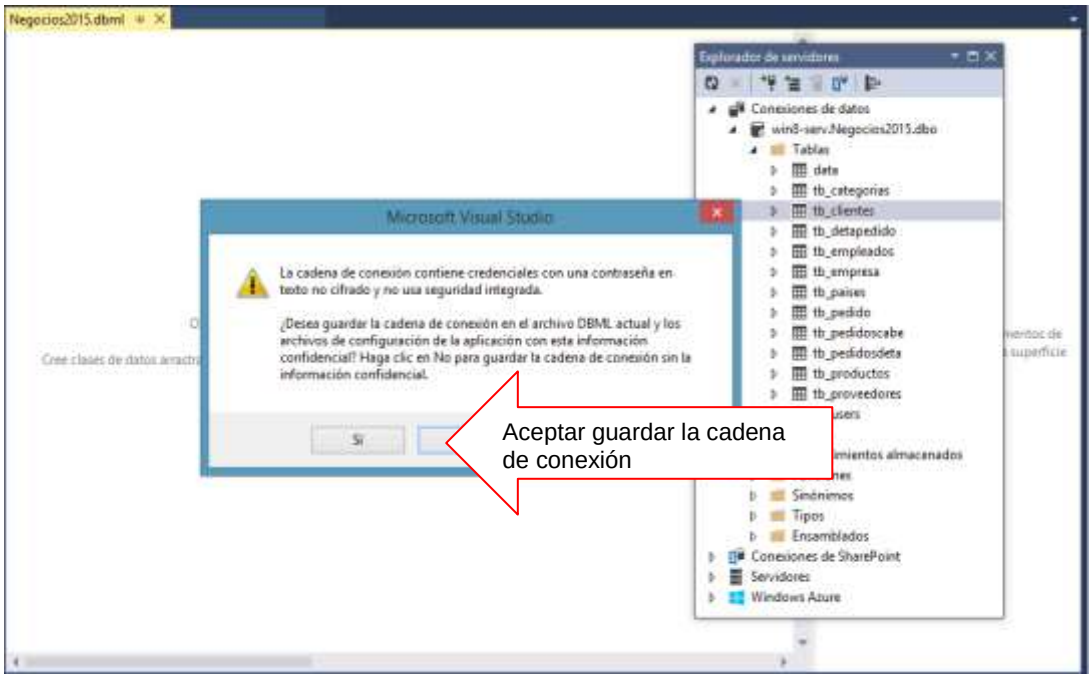
Selecciona el origen de datos: Microsoft SQL Server



Defina la conexión: nombre del servidor, Autenticacion por SQL Server: usuario y clave; y selecciona la base de datos: Negocios2015. Presiona el botón Aceptar



Teniendo la conexión agregada, arrastre las tablas de la conexión a la base de datos al Contexto. Al arrastar la primera tabla se visualiza un mensaje de credenciales, ACEPTAR el mensaje



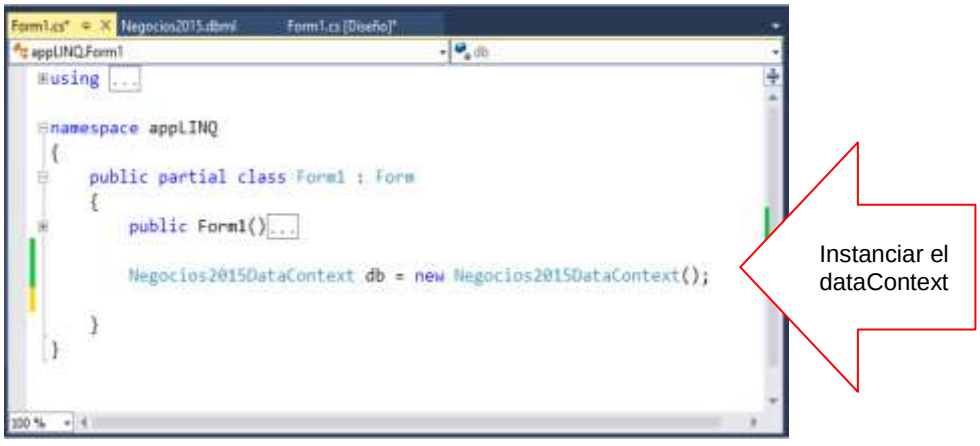
Agregada las tablas, el modelo se muestra.

DISEÑO DE FORMULARIO

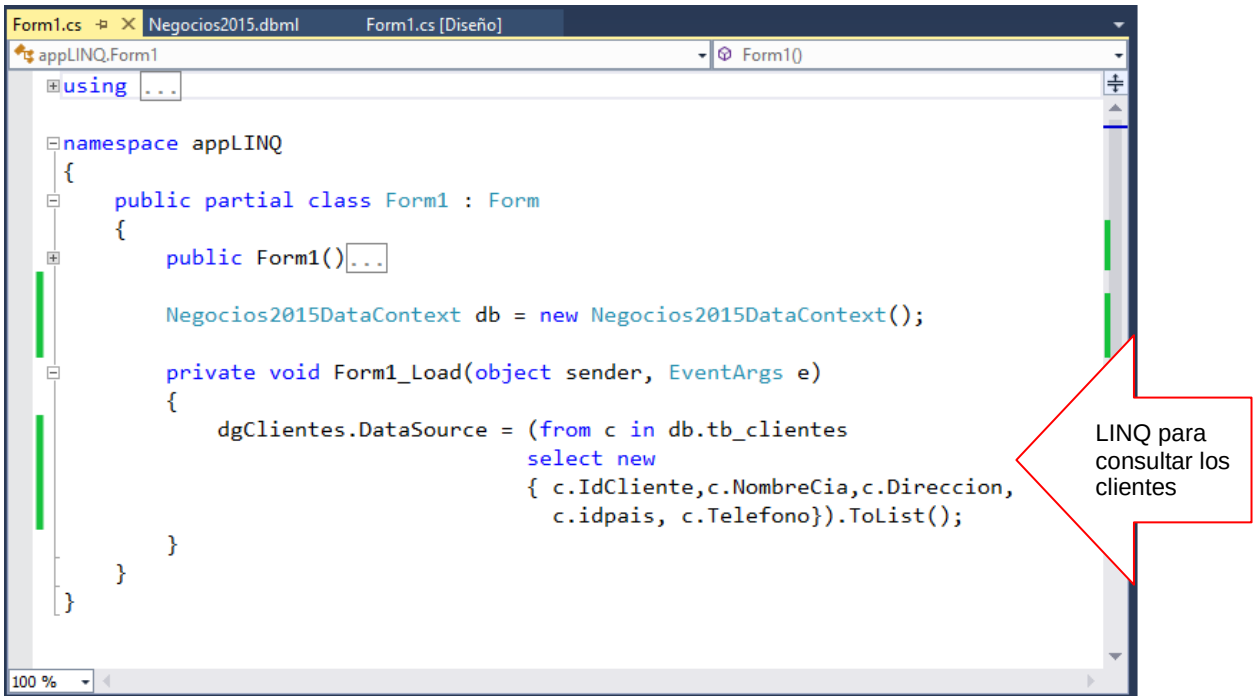
Diseña el formulario tal como se muestra

PROGRAMACION

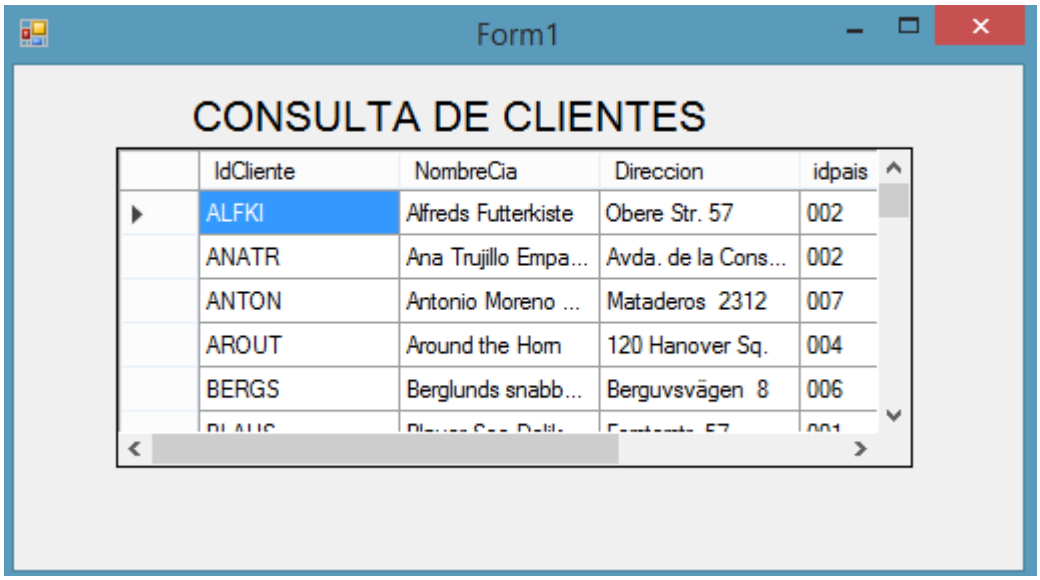
Instanciar el dataContext en el Formulario



En el evento Load, ejecutar la consulta LINQ to SQL para listar los clientes, visualizándolos en el control dgClientes



Presiona la tecla F5, para visualizar la consulta

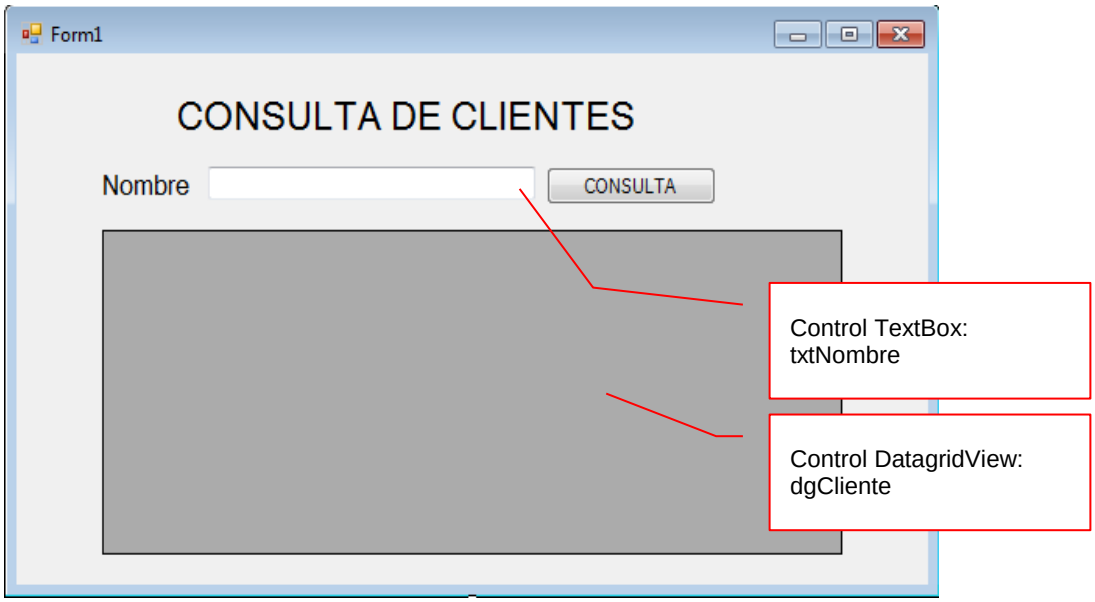


LABORATORIO 9.2

Se desea implementar un programa que permita filtrar los clientes por las iniciales de su nombre ingresados desde un TextBox.

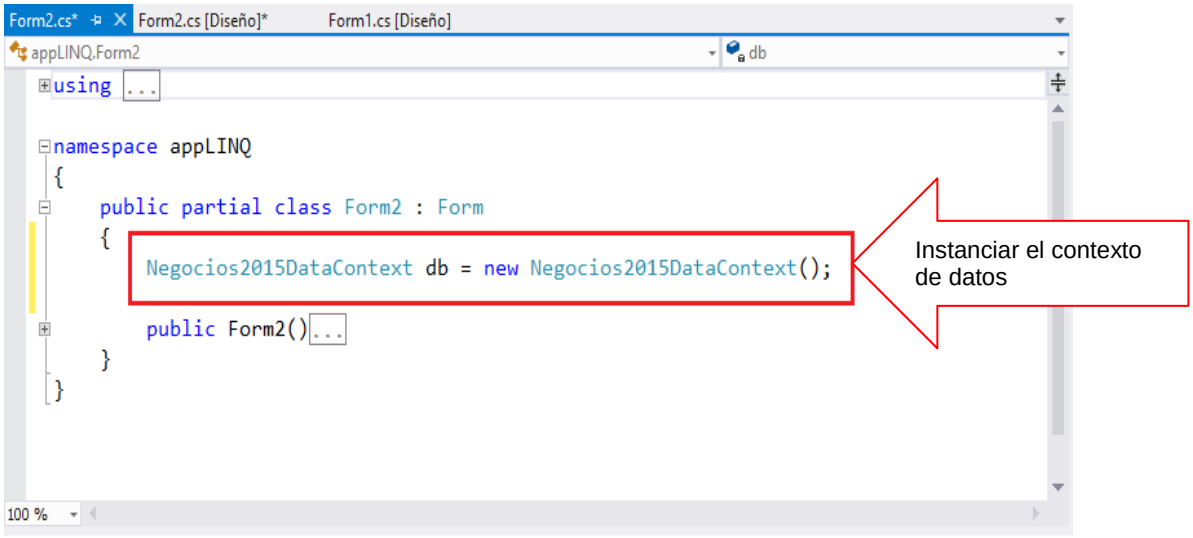
DISEÑO DE FORMULARIO.

A continuación diseñe el formulario tal como se muestra

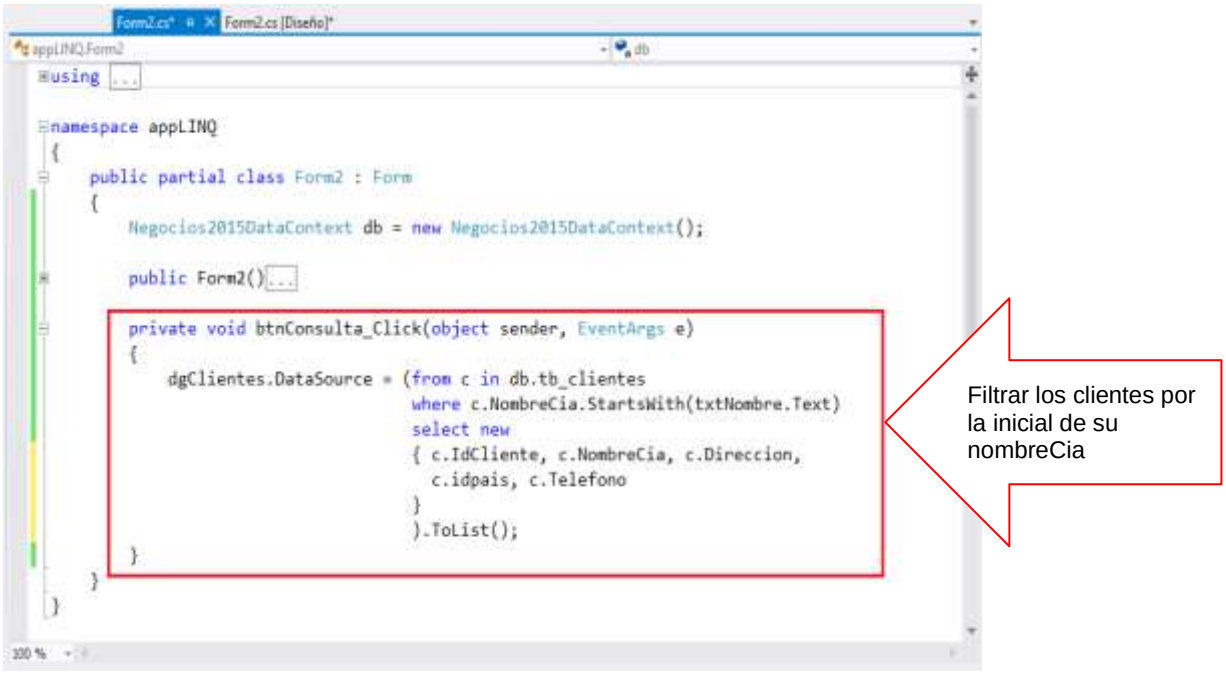


PROGRAMACIÓN.

Instancia el contexto de Datos a nivel Formulario



Programa el evento del botón Consulta, donde permite filtrar los clientes por la inicial de su nombre




```
using ...

namespace appLINQ
{
    public partial class Form2 : Form
    {
        Negocios2015DataContext db = new Negocios2015DataContext();

        public Form2()...

        private void btnConsulta_Click(object sender, EventArgs e)
        {
            dgClientes.DataSource = (from c in db.tb_clientes
                                     where c.NombreCia.StartsWith(txtNombre.Text)
                                     select new
                                     { c.IdCliente, c.NombreCia, c.Direccion,
                                       c.idpais, c.Telefono
                                     }
                                     ).ToList();
        }
    }
}
```

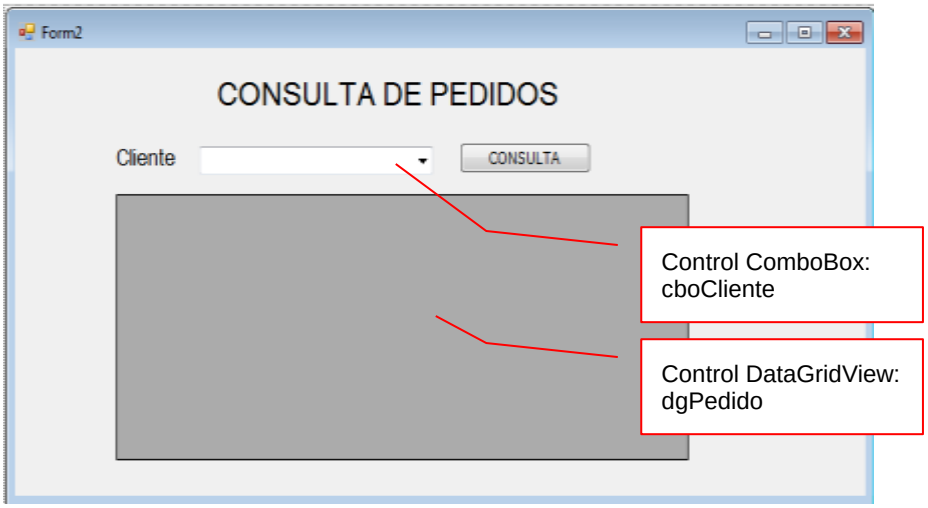
Presiona la tecla F5, ingrese la inicial del nombre en el TextBox, al presionar el botón Consulta, listamos los clientes que coincidan con las iniciales de su nombre.

LABORATORIO 9.3

Se desea implementar un programa que permita filtrar los registros de pedidoscabe por un determinado Cliente seleccionado desde un control ComboBox.

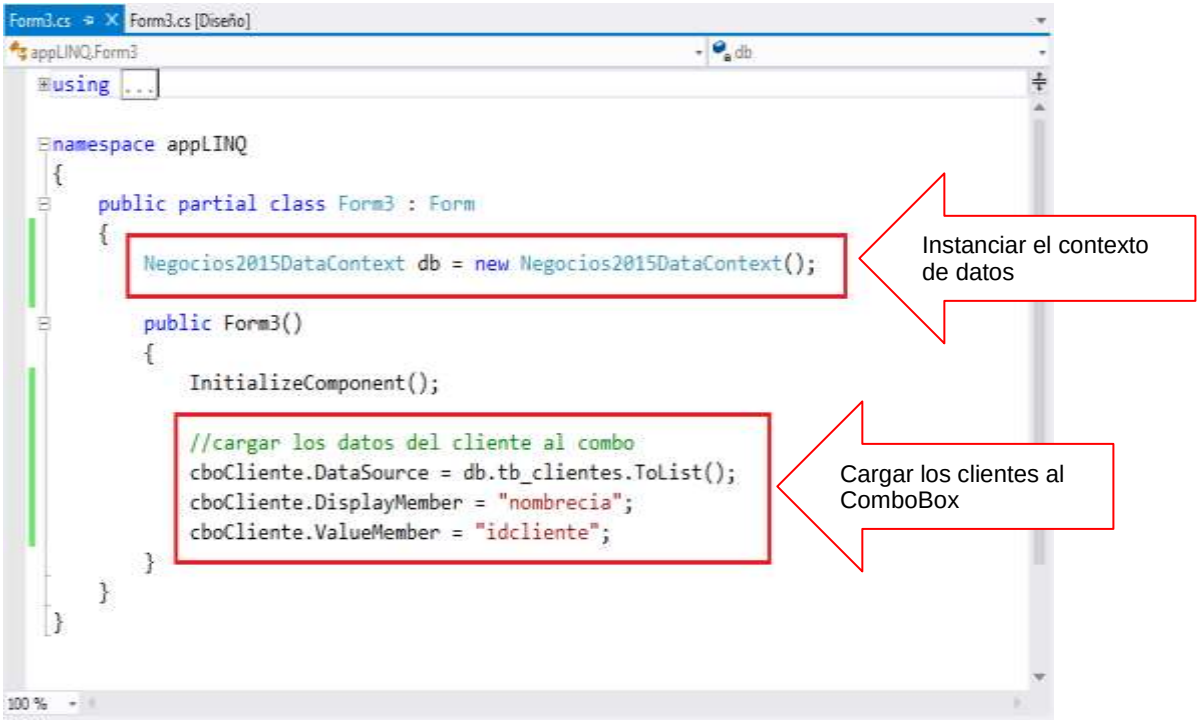
DISEÑO DE FORMULARIO.

A continuación diseñe el formulario tal como se muestra

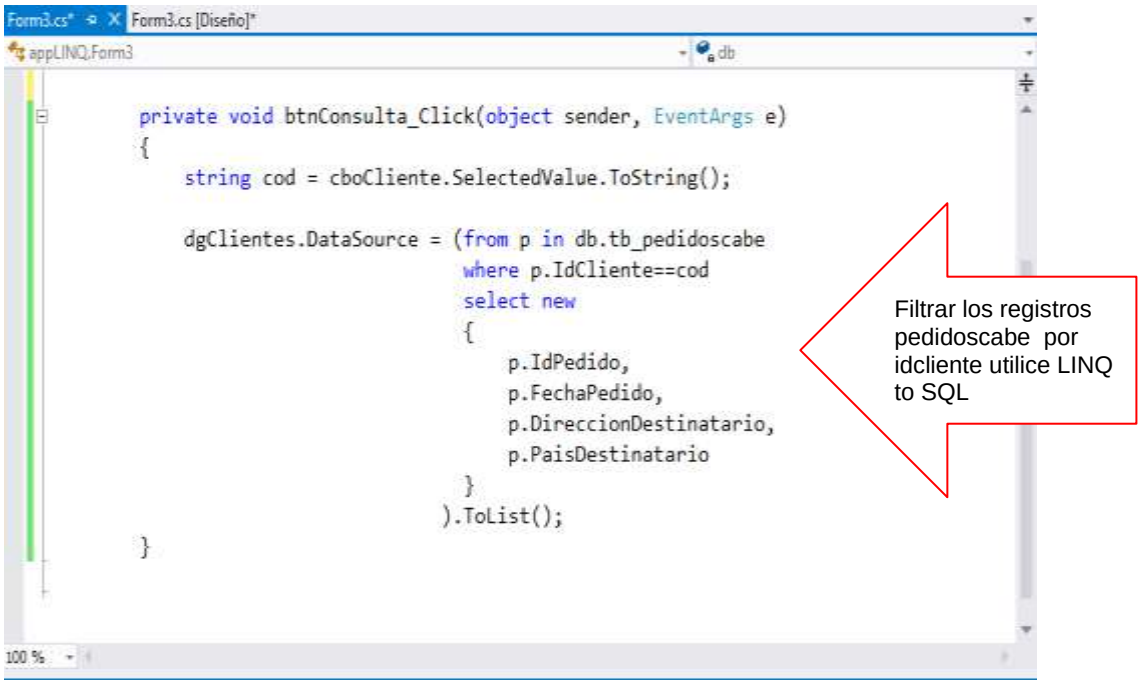


PROGRAMACIÓN.

Instancia el contexto de Datos a nivel Formulario. Programa el evento Load para cargar los clientes al comboBox, utilice LINQ to SQL para obtener el listado de clientes



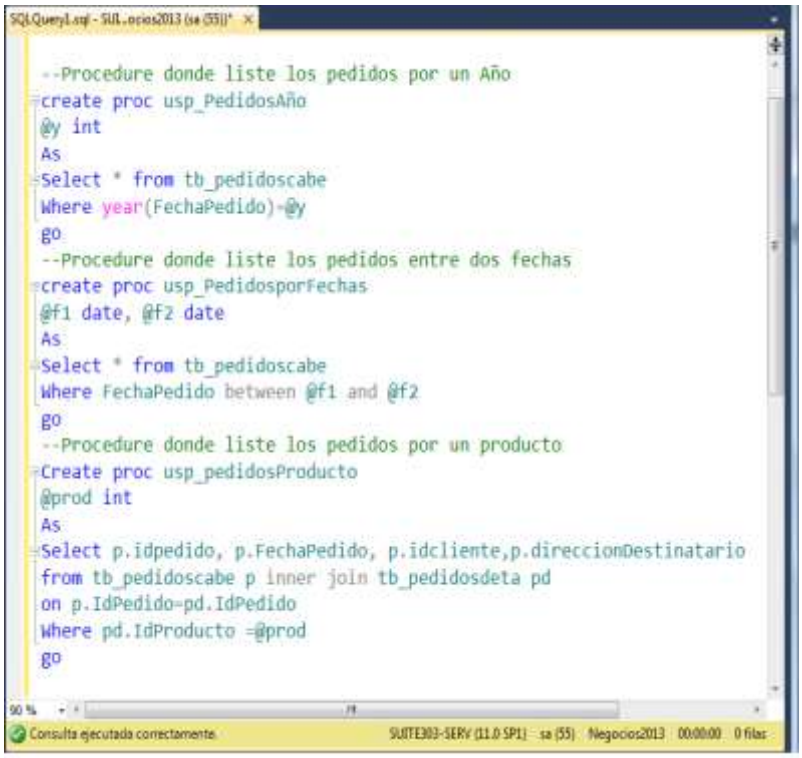
Programe el evento Clic del botón Consulta, donde filtre los registros de pedidoscabe por un determinado cliente seleccionado desde el ComboBox utilizando LINQ to SQL.



Presiona la tecla F5, selecciona un cliente desde el comboBox, al presionar el botón Consulta, listamos los pedidoscabe por el cliente seleccionado.

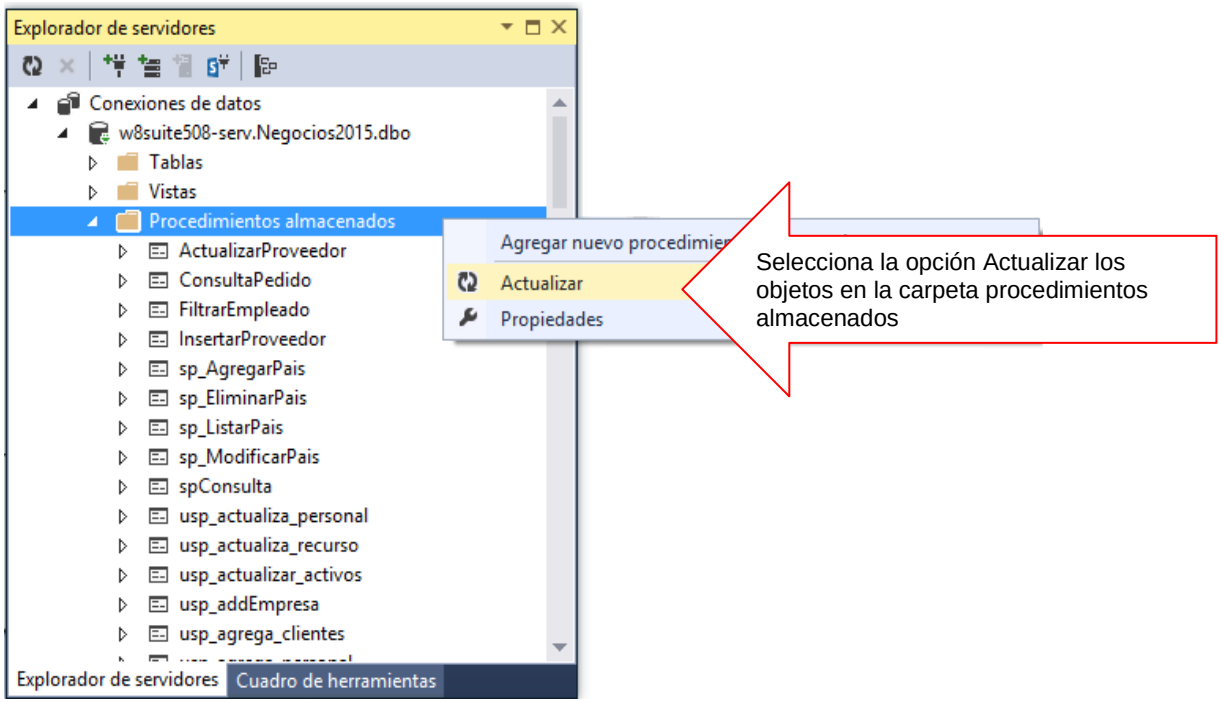
MANEJO DE PROCEDIMIENTOS ALMACENADOS EN EL LINQ TO SQL

Se tiene los siguientes procedimientos almacenados definidos en la base de datos.

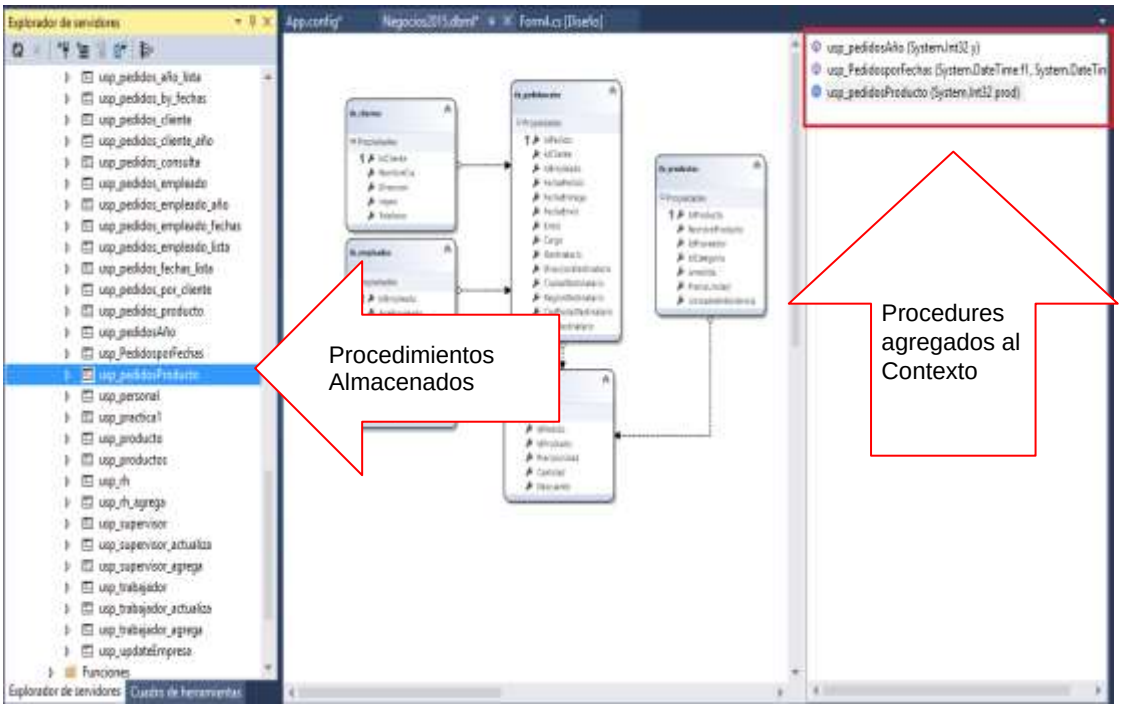


```
--Procedure donde liste los pedidos por un Año
create proc usp_PedidosAño
@y int
As
Select * from tb_pedidoscabe
Where year(FechaPedido)=@y
go
--Procedure donde liste los pedidos entre dos fechas
create proc usp_PedidosporFechas
@f1 date, @f2 date
As
Select * from tb_pedidoscabe
Where FechaPedido between @f1 and @f2
go
--Procedure donde liste los pedidos por un producto
Create proc usp_pedidosProducto
@prod int
As
Select p.idpedido, p.FechaPedido, p.idcliente,p.direccionDestinatario
from tb_pedidoscabe p inner join tb_pedidosdeta pd
on p.IdPedido=pd.IdPedido
Where pd.IdProducto =@prod
go
```

Creado los procedimientos almacenados, en el proyecto, actualice la carpeta Procedimientos Almacenados tal como se muestra



Los procedimientos almacenados que se encuentran en la carpeta, serán arrastrados al Contexto, tal como se muestra en la figura.

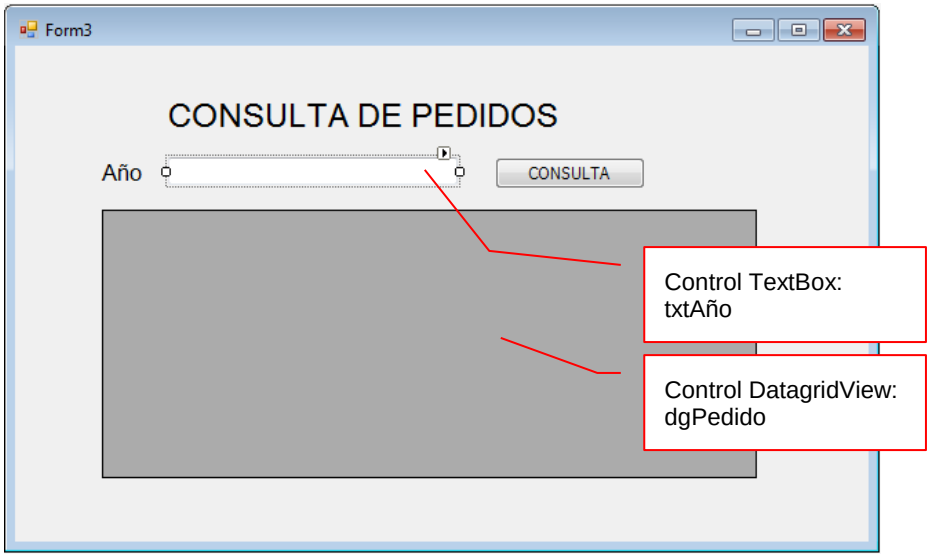


LABORATORIO 9.4

Se desea implementar un programa que permita filtrar los registros de pedidoscabe por un determinado Año ingresado desde un control TextBox. Ejecute el procedimiento almacenado del proceso.

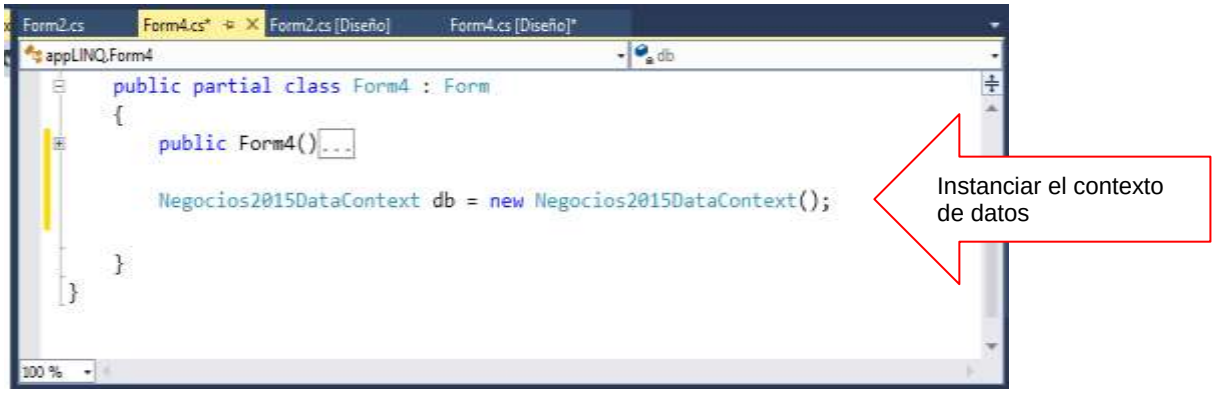
DISEÑO DE FORMULARIO.

A continuación diseñe el formulario tal como se muestra

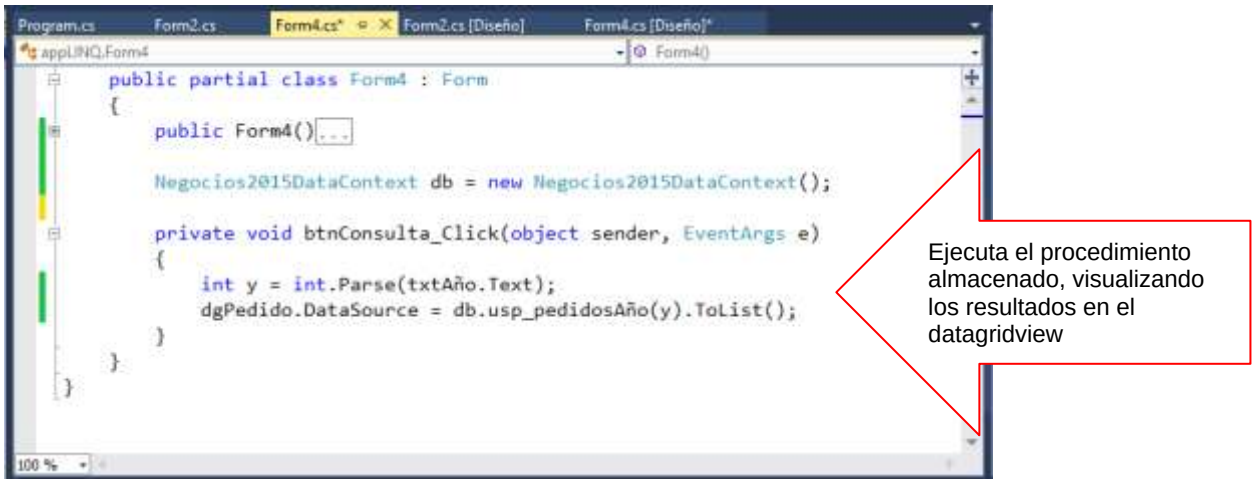


PROGRAMACIÓN.

Instancia el contexto de Datos a nivel Formulario



Programa el evento del botón Consulta, donde ejecuta el procedimiento almacenado usp_PedidosAño, visualizando los resultados en el DataGridView



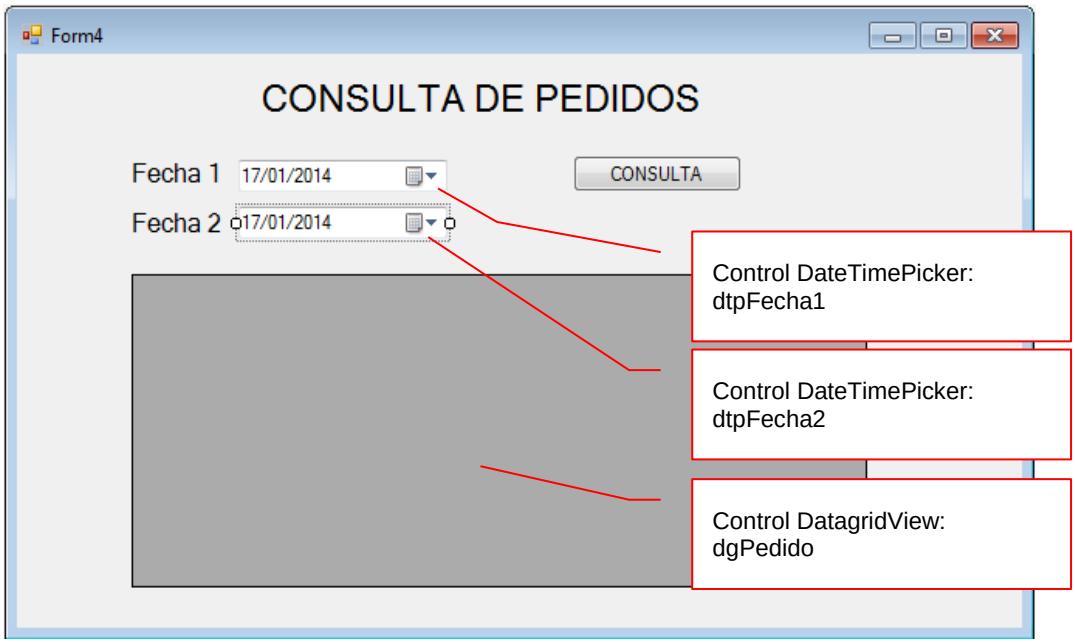
Al finalizar, presione F5 para ejecutar el Formulario, ingrese el año desde el control TextBox, al presionar el botón Consulta listar los registros de pedidoscabe por dicho año de Fechapedido.

LABORATORIO 9.5

Se desea implementar un programa que permita filtrar los registros de pedidoscabe por un rango de Fechas seleccionada desde controles DataTime. Ejecute el procedimiento almacenado del proceso.

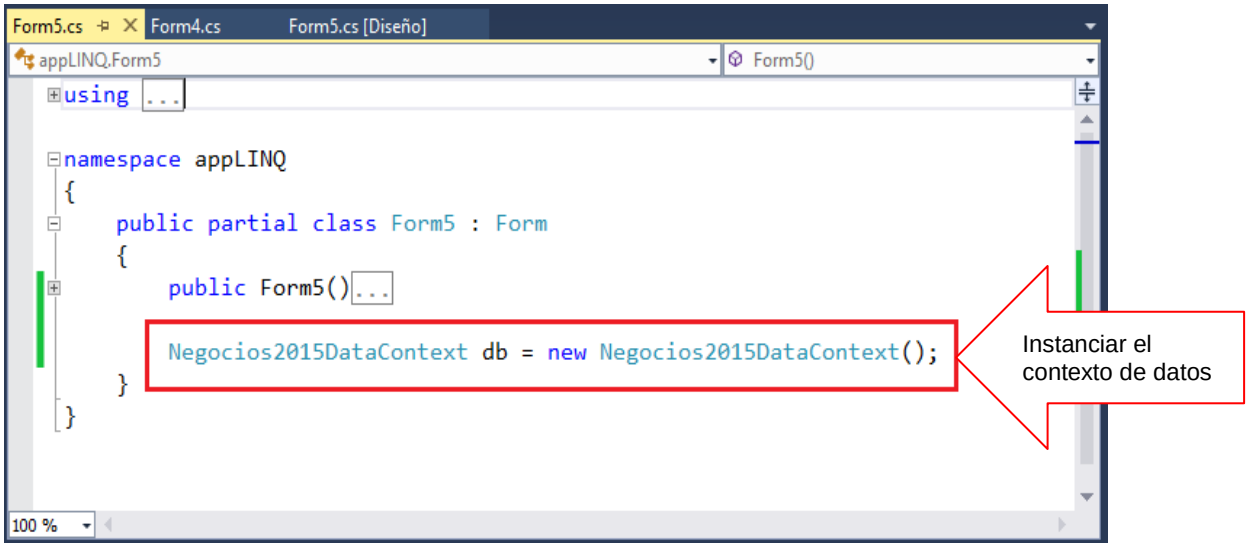
DISEÑO DE FORMULARIO.

A continuación diseñe el formulario tal como se muestra

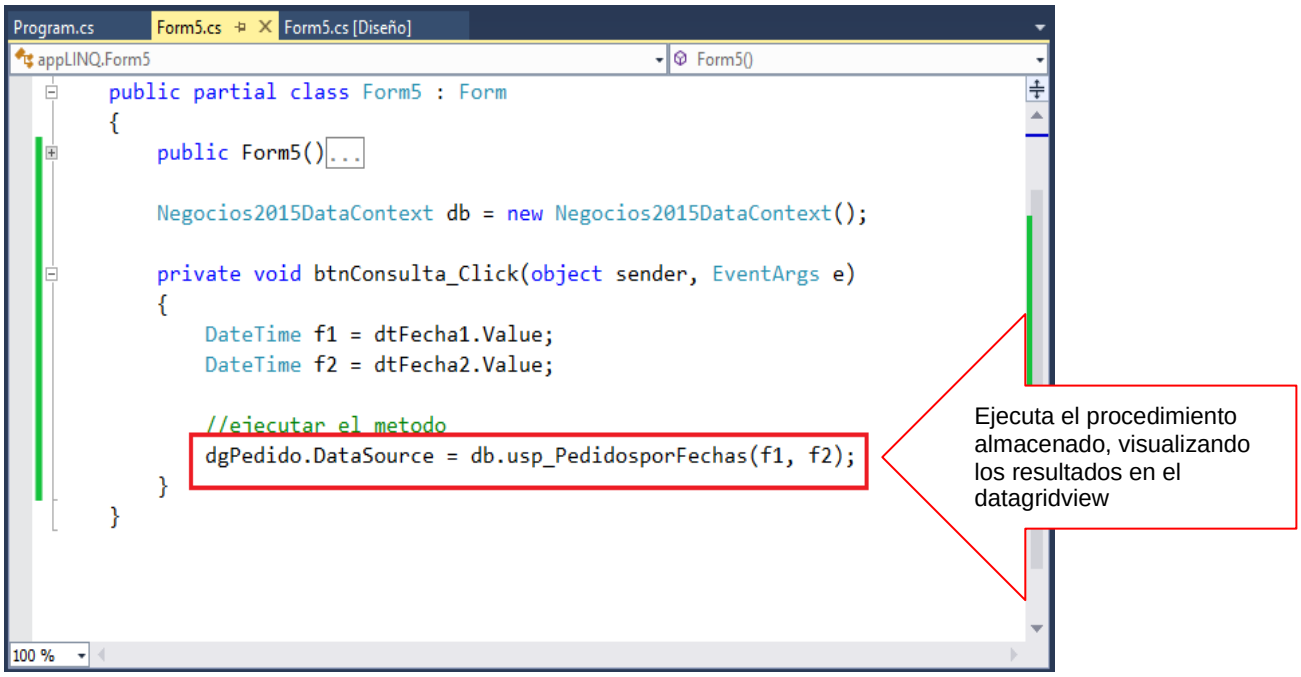


PROGRAMACIÓN.

Instancia el contexto de Datos a nivel Formulario



Programa el evento del botón Consulta, donde ejecuta el procedimiento almacenado usp_PedidosporFechas, visualizando los resultados en el DataGridView



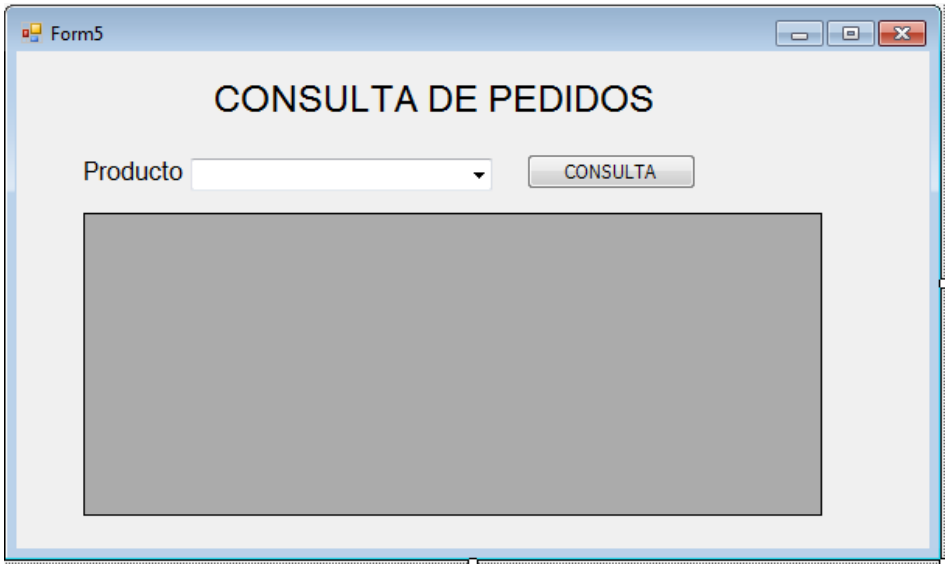
Al finalizar, presione F5 para ejecutar el Formulario, selecciona las fechas desde los controles DateTimePicker, al presionar el botón Consulta listar los registros de pedidoscabe entre el rango de las fechas por fechaPedido.

DESARROLLO PRACTICO DE LABORATORIO

Se desea implementar un programa que permita filtrar los registros de pedidoscabe por un producto seleccionado desde el control ComboBox. Ejecute el procedimiento almacenado del proceso.

DISEÑO DE FORMULARIO.

A continuación diseñe el formulario tal como se muestra



Resumen

- Language-Integrated Query (LINQ) permite a los programadores formar consultas basadas en conjuntos en el código de su aplicación, sin tener que usar un lenguaje de consulta independiente. Se puede escribir consultas de LINQ en varios orígenes de datos enumerables, como estructuras de datos en memoria, documentos XML, bases de datos SQL y objetos DataSet
- Existen tres tecnologías Language-Integrated Query (LINQ) de ADO.NET independientes: LINQ to DataSet, LINQ to SQL y LINQ to Entities. LINQ to DataSet proporciona consultas más ricas y optimizadas de DataSet, LINQ to SQL permite consultar directamente los esquemas de base de datos de SQL Server y LINQ to Entities permite consultar un Entity Data Model.
- Language-Integrated Query (LINQ) es una innovación introducida en Visual Studio 2008 y .NET Framework versión 3.5 que elimina la distancia que separa el mundo de los objetos y el mundo de los datos
- En Visual Studio se pueden escribir consultas LINQ en Visual Basic o en C# con bases de datos SQL Server, documentos XML, conjuntos de datos ADO.NET y cualquier colección de objetos que admita IEnumerable o la interfaz genérica IEnumerable(Of T). También se ha previsto la compatibilidad de LINQ con ADO.NET Entity Framework, y otros fabricantes se encuentran escribiendo proveedores LINQ para muchos servicios Web y otras implementaciones de bases de datos.
- Cuando la aplicación se ejecuta, LINQ to SQL convierte a SQL las consultas integradas en el lenguaje en el modelo de objetos y las envía a la base de datos para su ejecución. Cuando la base de datos devuelve los resultados, LINQ to SQL los vuelve a convertir en objetos con los que pueda trabajar en su propio lenguaje de programación.
- Las expresiones lambda junto a las instrucciones que proporciona LINQ (Lenguaje de Consulta Integrado) resultan en una dupla complementarias en materia de desarrollo de software. La inclusión y el uso de expresiones lambda en consultas de base de datos utilizando LINQ son elementales y poderosos.
- El moderno concepto de OODBMS (Sistema de Manejador de Base de Datos Orientado a Objetos) utiliza como eje central de desarrollo el uso del lenguaje LINQ y adiciona a este compendio el uso de expresiones lambda como un modelo de desarrollo excepcional.
- LINQ to SQL usa métodos del modelo de objetos para representar procedimientos almacenados en la base de datos. Los métodos se designan como procedimientos almacenados aplicando el atributo FunctionAttribute y, si es necesario, el atributo ParameterAttribute.

- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.
  - http://msdn.microsoft.com/es-es/library/bb387007(v=vs.110).aspx
  - http://msdn.microsoft.com/es-es/library/bb386946(v=vs.110).aspx
  - http://alexjimenez.wordpress.com/2007/09/10/linq-definicion-como-usarlo/
  - http://msdn.microsoft.com/es-es/library/vstudio/bb386955(v=vs.100).aspx

UNIDAD

4

MANIPULACION DE DATOS: MODELO RELACIONAL DE OBJETOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno realiza consultas y actualización de datos a través del lenguaje de consulta integrado utilizando la plataforma ORM en un entorno de una aplicación Windows.

Temario

Tema 10: ENTITIES (6 horas)

1.1. ADO.NET y Entities
1.2. Modelo de datos de Entidades, operaciones de consulta utilizando sentencias de consulta de LINQ to ENTITIES
1.3. Acceso a datos utilizando EntityClientProvider
  1.3.1 Clase EntityConnection
  1.3.2 Clase EntityCommand y EntityTransaction
  1.3.3 Clase EntityDataReader

ACTIVIDADES PROPUESTAS

- Los alumnos conocer el modelo de datos orientado a objetos
- Los alumnos realizan operaciones de consulta utilizando el modelo de clase LINQ to ENTITY
- Los alumnos realizan operaciones de actualización utilizando el modelo de clase ENTITY FRAMEWORK

10. ENTITY FRAMEWORK

Entity Framework es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos. Los arquitectos y programadores de aplicaciones orientadas a datos se han enfrentado a la necesidad de lograr dos objetivos muy diferentes. Deben modelar las entidades, las relaciones y la lógica de los problemas empresariales que resuelven, y también deben trabajar con los motores de datos que se usan para almacenar y recuperar los datos. Los datos pueden abarcar varios sistemas de almacenamiento, cada uno con sus propios protocolos; incluso las aplicaciones que funcionan con un único sistema de almacenamiento deben equilibrar los requisitos del sistema de almacenamiento con respecto a los requisitos de escribir un código de aplicación eficaz y fácil de mantener.

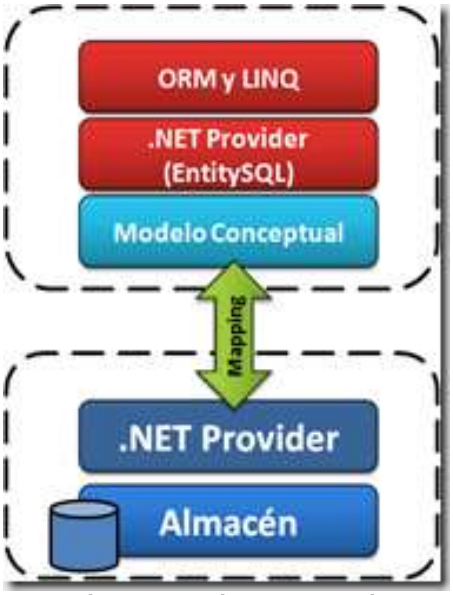


Figura 1: Entity Framework
Referencia: http://geeks.ms/blogs/ciin/archive/2008/01/25/ado-net-entity-framework-linq-to-entities-entity-sql-y-entity-services-i.aspx

Entity Framework permite a los desarrolladores trabajar con datos en forma de objetos y propiedades específicos del dominio, como clientes y direcciones de cliente, sin tener que preocuparse por las tablas y columnas de la base de datos subyacente donde se almacenan estos datos. Con Entity Framework, los desarrolladores pueden trabajar en un nivel mayor de abstracción cuando tratan con datos, y pueden crear y mantener aplicaciones orientadas a datos con menos código que en las aplicaciones tradicionales. Dado que Entity Framework es un componente de .NET Framework, las aplicaciones de Entity Framework se pueden ejecutar en cualquier equipo en el que esté instalado .NET Framework a partir de la versión 3.5 SP1.

CARACTERISTICAS DEL ENTITY FRAMEWORK

Una aplicación de Entity Framework requiere crear un modelo conceptual que defina las entidades y las relaciones, un modelo lógico que represente el modelo relacional subyacente y las asignaciones entre los dos. A continuación, se genera un modelo de objetos programable a partir del modelo conceptual.
Las características y componentes siguientes de Entity Framework trabajan conjuntamente para proporcionar un entorno de programación de un extremo a otro.

- El Entity Data Model (EDM) es la pieza central de Entity Framework. Especifica el esquema de diseño, que se usa para generar las clases programables que usa el código de la aplicación. Las estructuras de almacenamiento de los datos conservados se representan en un esquema de almacenamiento y una especificación de asignación conecta el esquema de diseño con el esquema de almacenamiento.
- El componente Object Services permite a los programadores trabajar con las clases de Common Language Runtime (CLR) generadas a partir del modelo conceptual. También proporcionan compatibilidad de infraestructura con Entity Framework, con servicios como administración de estados, seguimiento de cambios, resolución de identidad, relaciones de carga y navegación, propagación de cambios de objeto a modificaciones de base de datos y compatibilidad con consultas para Entity SQL.
- LINQ to Entities proporciona compatibilidad con Language-Integrated Query (LINQ) para consultar las entidades. LINQ to Entities permite a los programadores escribir consultas con la base de datos utilizando uno de los lenguajes de programación de .NET Framework, como Visual Basic o C#.
- Entity SQL es un lenguaje independiente del almacenamiento que es similar a SQL y que se ha diseñado para la consulta y manipulación de gráficos enriquecidos de objetos basados en el modelo Entity Data Model (EDM).
- El proveedor EntityClient extiende el modelo de proveedor de ADO.NET teniendo acceso a los datos en lo que respecta a las entidades conceptuales y relaciones. Ejecuta consultas que usan Entity SQL. Entity SQL proporciona el lenguaje de consulta subyacente que permite a EntityClient comunicarse con la base de datos.

DIAGRAMA DE LA ARQUITECTURA

El diagrama siguiente muestra cómo se relacionan las diversas interfaces de programación de usuario accesibles en Entity Framework.

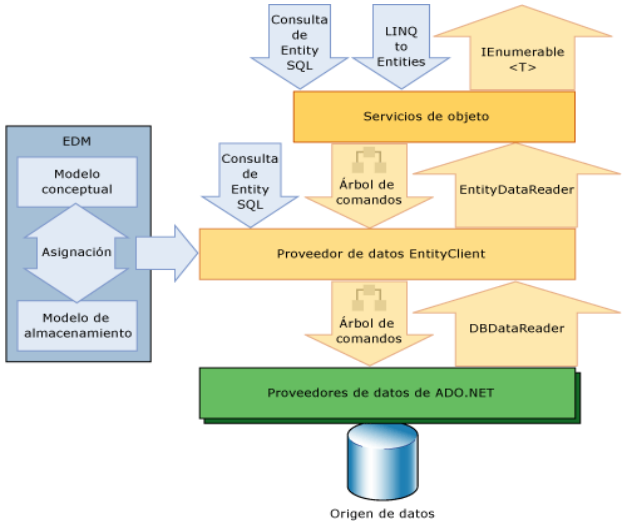


Figura 2: Arquitectura
Referencia: http://msdn.microsoft.com/es-es/library/bb896338(v=vs.90).aspx

Una flecha descendente indica una consulta en el origen de datos, y una flecha ascendente indica los datos devueltos. Servicios de objeto genera un árbol de comandos canónico que representa a LINQ to Entities o una operación de Entity SQL

con el modelo conceptual. El proveedor de EntityClient transforma este árbol de comandos canónico, basado en el modelo EDM, en un nuevo árbol de comandos canónico que es una operación equivalente en el origen de datos.

10.1 ADO.NET Y ENTITY

El Entity Data Model (EDM) es un modelo de entidad relación. El EDM define los datos en un formato neutro que no está restringido por la estructura de los lenguajes de programación o las bases de datos relacionales. Los esquemas EDM se usan para especificar los detalles de las entidades y las relaciones, y para implementarlos como estructuras de datos.

Una entidad es una parte del dominio de una aplicación que se debe representar mediante datos. Algunos ejemplos de entidades y relaciones se pueden encontrar en una aplicación de línea de negocio (LOB, Line Of Business) típica. Entre las entidades del dominio de una aplicación LOB se podrían incluir los clientes, pedidos, líneas de pedido, proveedores, productos, vendedores, suministradores, facturas, etcétera. El EntityType del EDM es la especificación de un tipo de datos que representa la entidad en el dominio de la aplicación.

Una relación es la conexión lógica entre entidades: por ejemplo, la conexión lógica entre el pedido de una mercancía y el cliente que lo realiza. Dado que un cliente puede tener asociados muchos pedidos, la relación entre un cliente y sus pedidos es de uno a varios. Los productos y los suministradores podrían tener una relación varios a varios.

El EDM modela las entidades y sus relaciones mediante dos tipos básicos.

- **EntityType**: es la especificación abstracta de los detalles de una estructura de datos en el dominio de la aplicación.

- **AssociationType**: es la conexión lógica entre los tipos.

Figura 3: Modelo de Entidades
Referencia: http://www.albloguera.es/index.php/2008/04/06/adonet-entity-framework/

El esquema de diseño de un EDM define la estructura, semántica, restricciones y relaciones de las entidades en el dominio de una aplicación. En la implementación del EDM de servicios de objeto, el esquema conceptual está asignado a otro esquema que contiene metadatos que describen el modelo de almacenamiento, normalmente las tablas en una base de datos. El esquema conceptual se usa para generar las clases de un modelo de objetos programables que se usará en el código de la aplicación. Los esquemas conceptual y de almacenamiento también se usan en el

Entity Framework para validar, consultar y actualizar los datos de la aplicación en tiempo de ejecución.

10.2 MODELO DE DATOS DE ENTIDADES: LINQ TO ENTITIES

Las tres técnicas principales que pueden usar para interactuar con un EDM:

- **Entity SQL con el proveedor EntityClient**: el proveedor EntityClient define las clases y mapea los datos para interactuar con modelos de entidades de datos. EntityClient transforma las operaciones de las entidades hacia operaciones directas a las base de datos.
- La escritura de código con la API de EntityClient le ofrece el control màs granular de las tres técnicas. Por crear un objeto EntityConnection para conectar al EDM, escribir una consulta SQL con un EntityCommand y devolver resultados a través de un EntityDataReader
- **Entity SQL con servicio de Objeto**: esta técnica se aleja de la interacción directa con el proveedor de EntityClient. Usted usarìa ObjectContext y ObjectQuery<T> para publicar las consultas en el EDM.
- **Entity SQL con LINQ**: la versión de ADO.NET incluye una capa que permite exponer los datos de la base de datos como objetos .NET, permitiendo a los desarrolladores formular consultas en un origen de datos directamente desde el lenguaje de consulta LINQ to Entities.

10.3 ACCESO A DATOS: ENTITYPROVIDERCLIENT

El proveedor de EntityClient es un proveedor de datos que usan las aplicaciones de Entity Framework para tener acceso a los datos descritos en un modelo conceptual.

Principales clases

Clase	Descripción
EntityConnection	Contiene una referencia a un modelo conceptual y una conexión a un origen de datos. Esta clase no puede heredarse.
EntityCommand	Representa un comando para el nivel conceptual.
EntityDataReader	Lee una secuencia sólo hacia delante de filas de un origen de datos.
EntityParameter	Representa un parámetro utilizado en EntityCommand.
EntityTransaction	Especifica la transacción para EntityCommand.
EntityProviderFactory	Representa un conjunto de métodos para crear instancias de la implementación de un proveedor de las clases de origen de datos.

10.4 ADMINISTRACION DE LOS OBJETOS ENTITY

EntityClient utiliza otros proveedores de datos .NET Framework para tener acceso al origen de datos. Por ejemplo, EntityClient utiliza el Proveedor de datos .NET Framework para SQL Server (SqlClient) al tener acceso a una base de datos de SQL Server.

10.4.1 Conexión a un origen de datos: EntityConnection

La clase EntityConnection permite conectarse con un determinado origen de datos mediante una cadena de conexión en la que proporciona la información de autenticación necesaria.

Cuando implementamos el modelo de entidades de datos EDM, a través del asistente, se crea una cadena de conexión, la cual se almacena en el archivo app.config:

Figura 4: Conexion Entidades
Referencia: http://sankarsan.wordpress.com/2011/10/22/entityconnection-metadata-in-entityframework/

A partir de la publicación de la conexión, definimos al objeto de conexión EntityConnection

```
Imports System.Data.EntityClient

Dim cn as new EntityConnection("Name=AdventureWorksLT2008Entities")
```

10.4.2 Trabajar con EntityCommand

El proveedor **EntityClient** posee una clase que permite ejecutar consultas sobre el modelo de entidad de datos: **EntityCommand**. La clase **EntityCommand** tiene 4 metodos que permite utilizar para ejecutar sentencia SQL:

Metodo	Descripcion
ExecuteScalar	Ejecuta una consulta donde retorna un valor.
ExecuteReader	Ejecuta una consulta donde retorna un conjunto de datos a un DataReader
ExecuteNonQuery	Ejecuta una sentencia para actualizar datos, donde retorna una expresión numérica que indica el numero de procesos afectados.

10.4.3 Trabajar con EntityReader

El proveedor EntityClient posee una clase que permite recuperar los datos de un origen de datos **EntityDataReader**.

Un **EntityDataReader** no tiene un constructor público. Solo se puede obtener a través de una sobrecarga del método **EntityCommand**.**ExecuteReader**.

SQL Server coloca los parámetros de salida de procedimientos almacenados al final de la secuencia resultante, después de que todos los conjuntos de resultados. Por consiguiente, para obtener los valores de los parámetros de salida, una aplicación debe consumir todos los registros en todos los conjuntos de resultados. Si la aplicación cierra **EntityDataReader** (con lo que también se cerraría DbDataReader), es posible que no se llenen los parámetros de salida.

EntityDataReader no consume implícitamente conjuntos de resultados para hacer que los parámetros de salida estén disponibles. Por lo tanto, considere los siguientes aspectos:

- **EntityDataReader** llama al método **DbDataReader**.**NextResult** solo cuando se llama explícitamente al método **EntityDataReader**.**NextResult**. Si **DbDataReader**.**NextResult** produce una excepción, [EntityDataReader] la encapsulará en **EntityException** (o una excepción derivada).

- Close solo cierra **DbDataReader**, sin consumir conjuntos de resultados o registros pendientes.

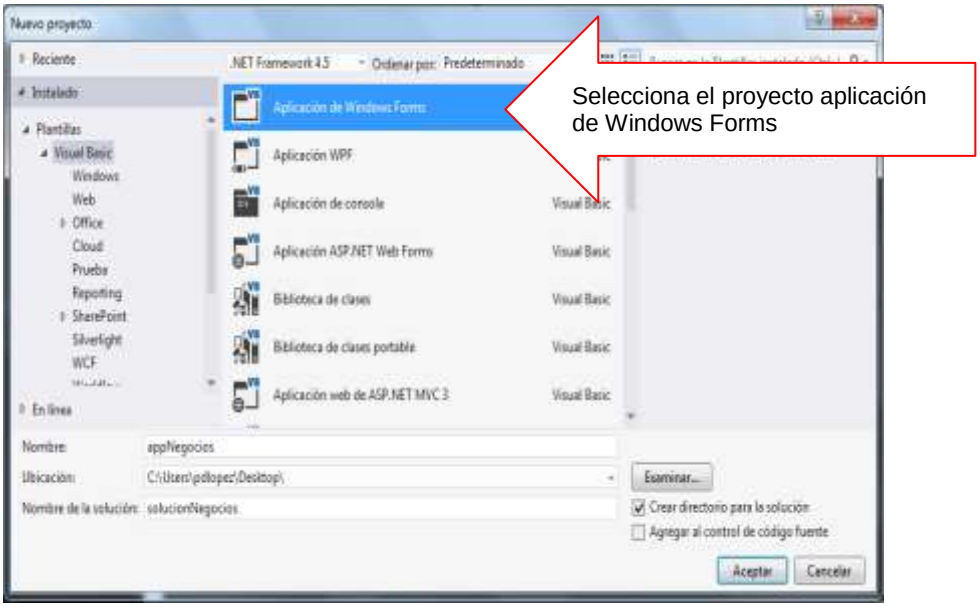
- Dispose solo elimina **DbDataReader**, sin consumir conjuntos de resultados o registros pendientes.

TRABAJANDO CON LINQ TO ENTITIES

LABORATORIO 10.1

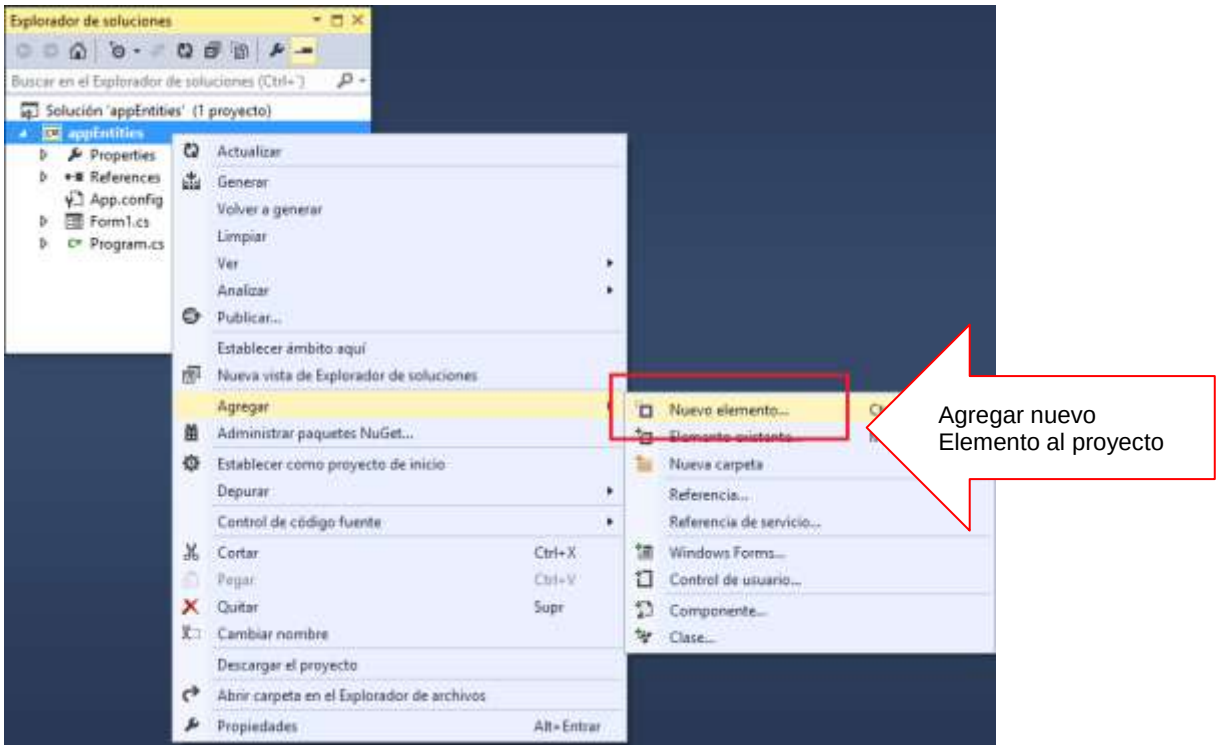
Se desea implementar un programa que liste los registros de la tabla tb_clientes.

Creando la aplicación

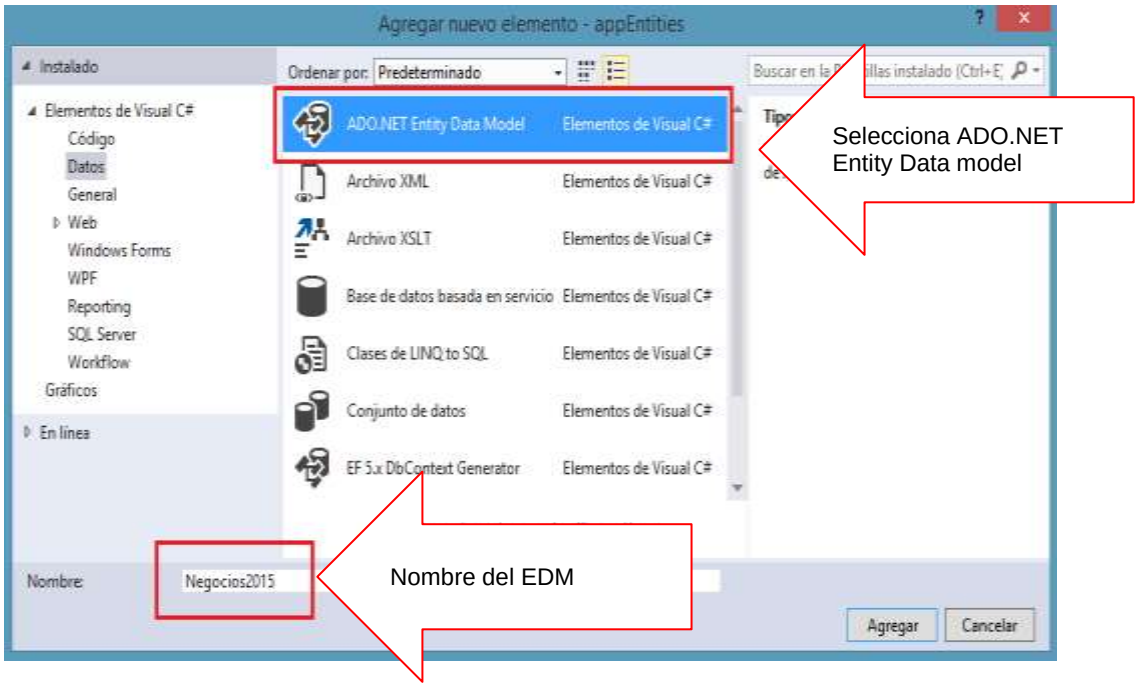


Agregando el Contexto ENTITY DATA MODEL.

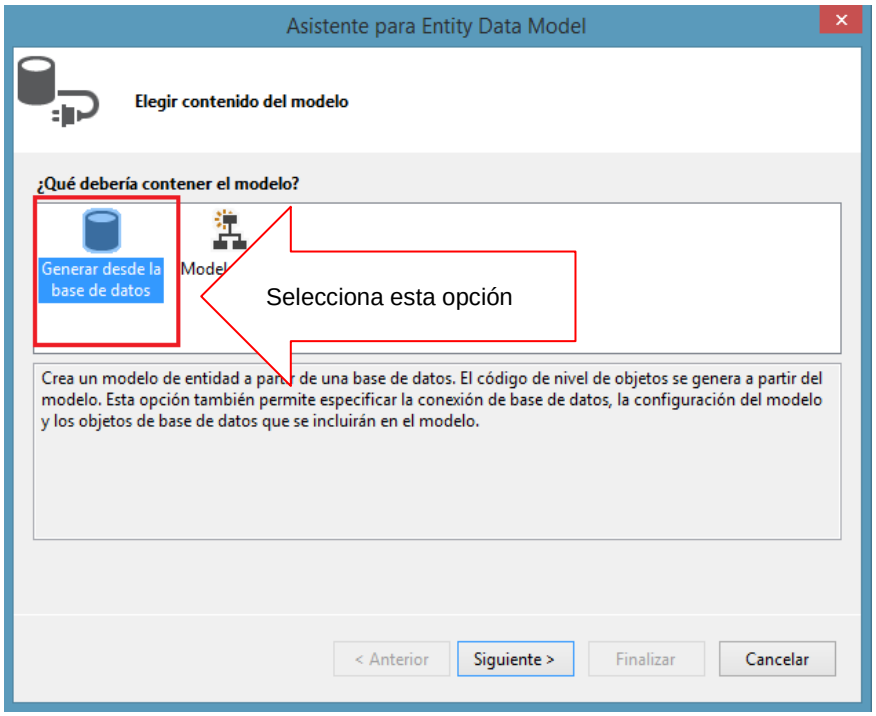
Desde el proyecto, selecciona la opción AGREGAR NUEVO ELEMENTO



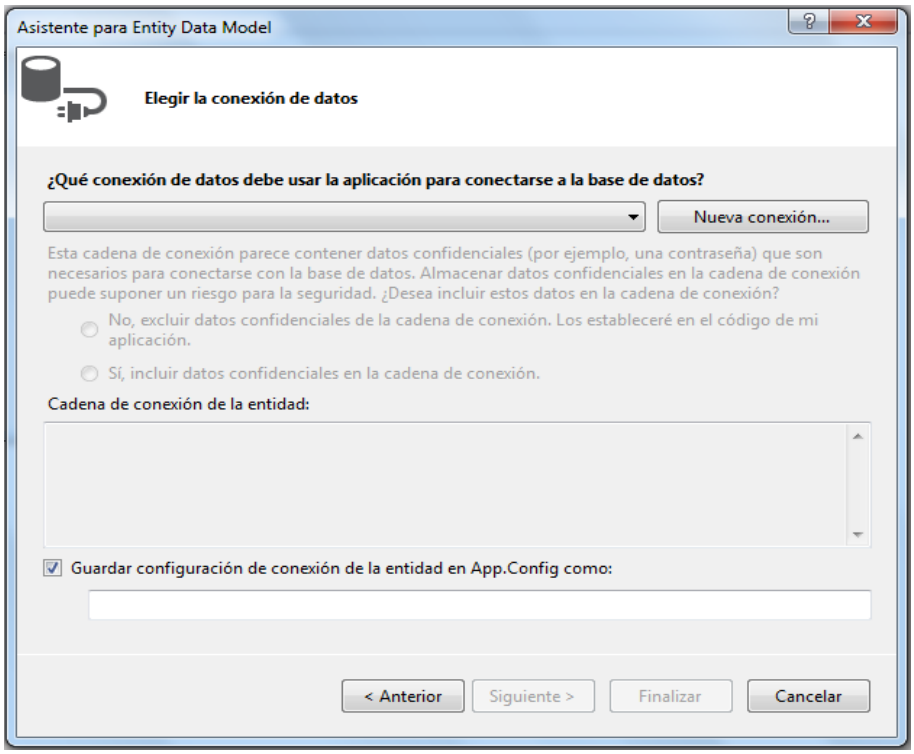
Selecciona ADO.NET Entity Data Model, asigne el nombre Negocios2015



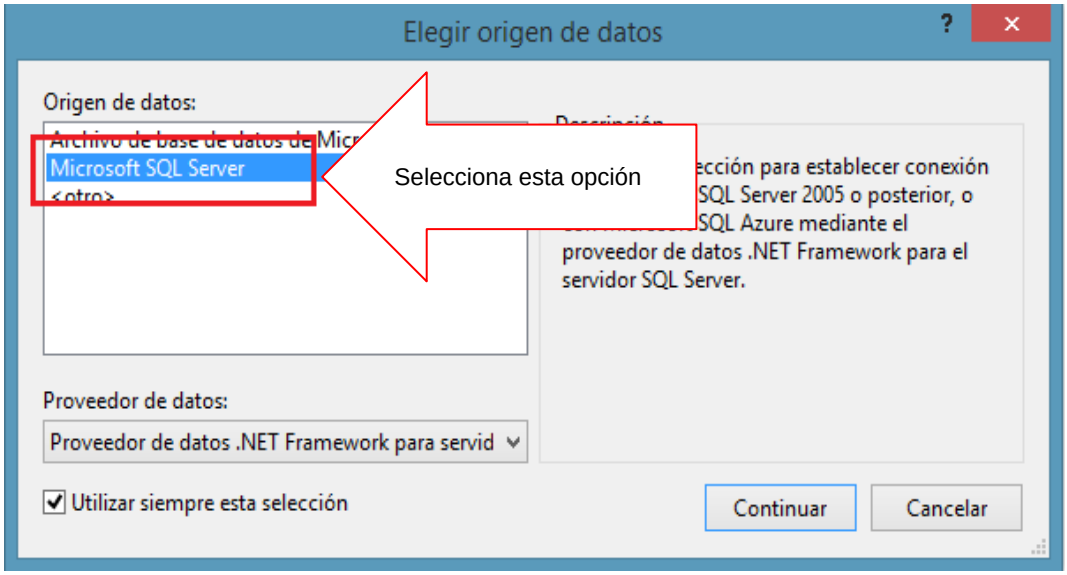
Elija el contenido del modelo: generar desde la base de datos, presiona SIGUIENTE



En esta ventana del asistente, para crear una nueva conexión, presiona la opción **Nueva Conexión**.



Selecciona el origen de datos: Microsoft SQL Server, presiona el botón Continuar.

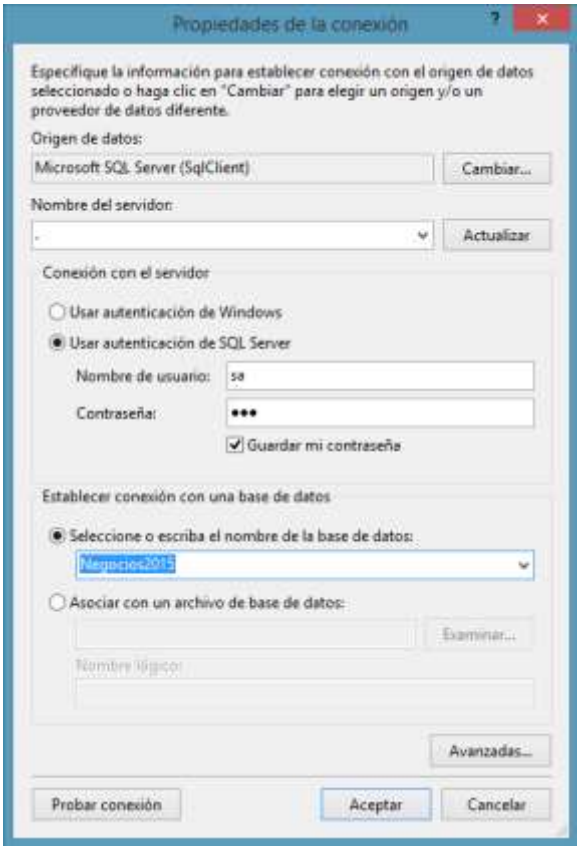


Trabajar con la conexión a la base de datos.

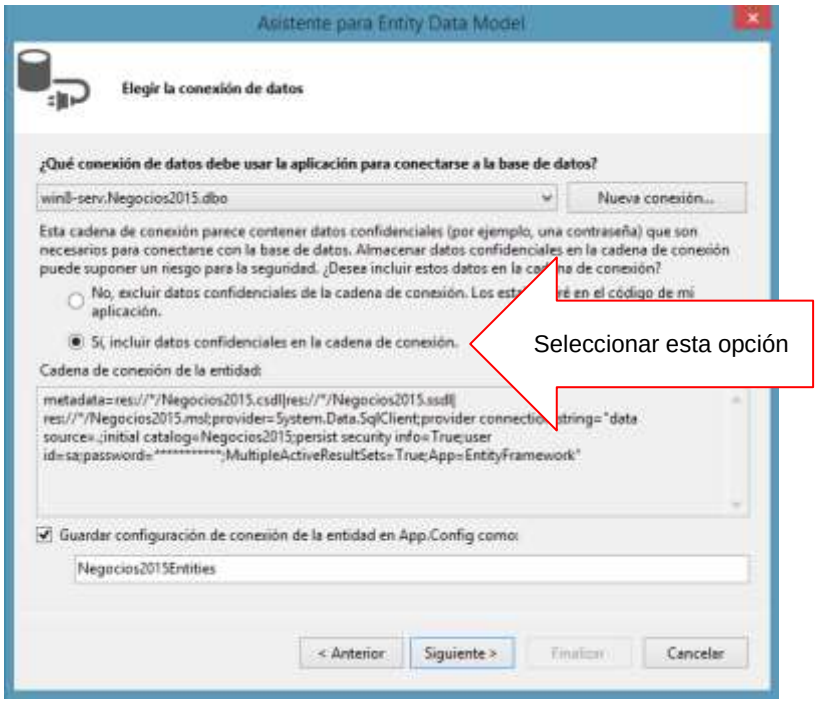
En las propiedades de la conexión:

1. selecciona el nombre del servidor
2. selecciona la autenticación (Sql Server)
3. Ingrese el usuario y su clave
4. Marque la opción Guardar mi contraseña
5. Selecciona la base de datos.

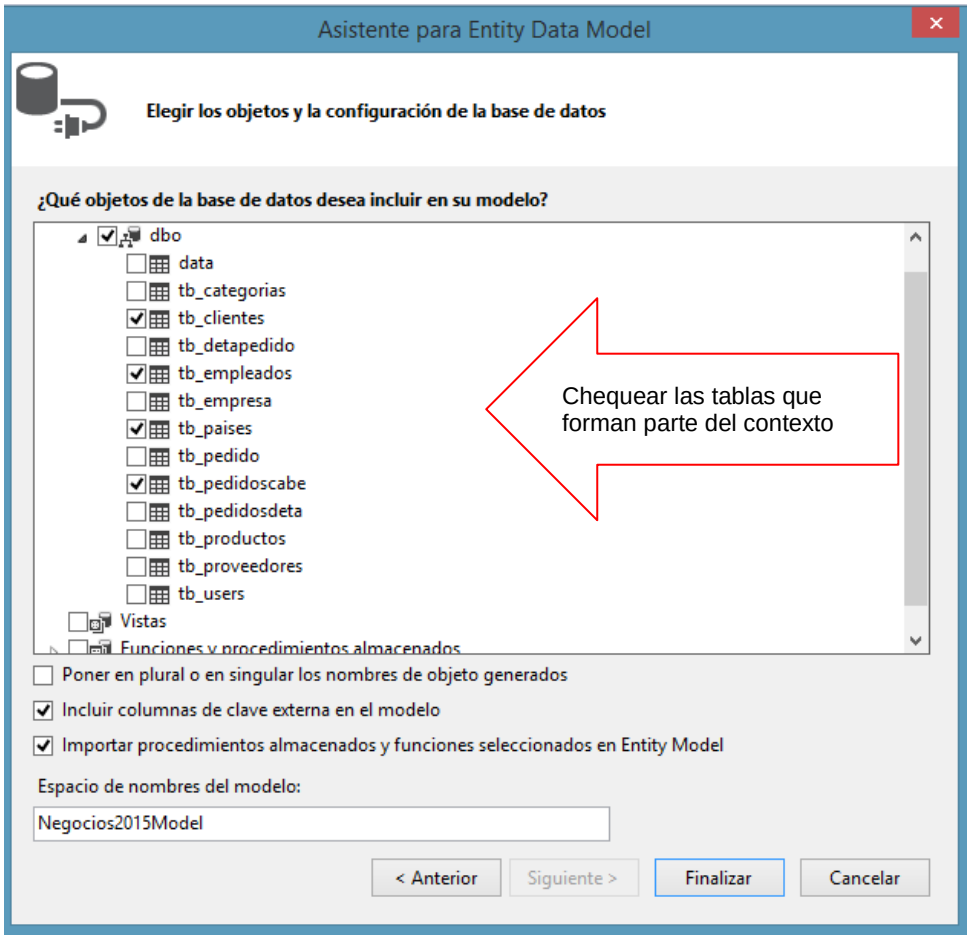
Realizada las operaciones, presiona el botón ACEPTAR



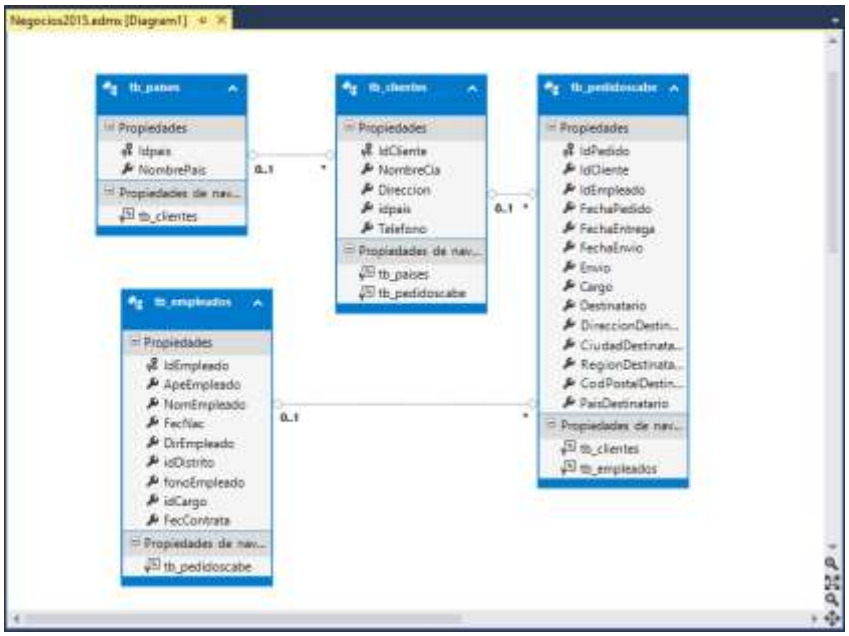
Generada la conexión, selecciona la opción SI, incluir datos confidenciales en la cadena de conexión, presiona el botón SIGUIENTE



Selecciona las tablas que se visualizan en el Contexto y presionar el botón Finalizar

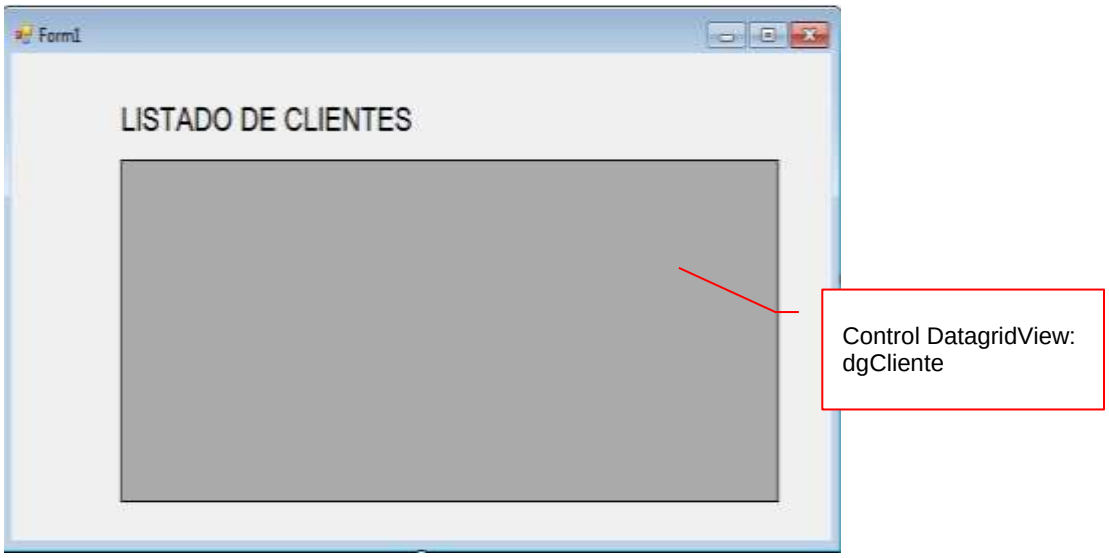


Al finalizar se visualiza el contexto. Compile el proyecto: Ctrl + May + B



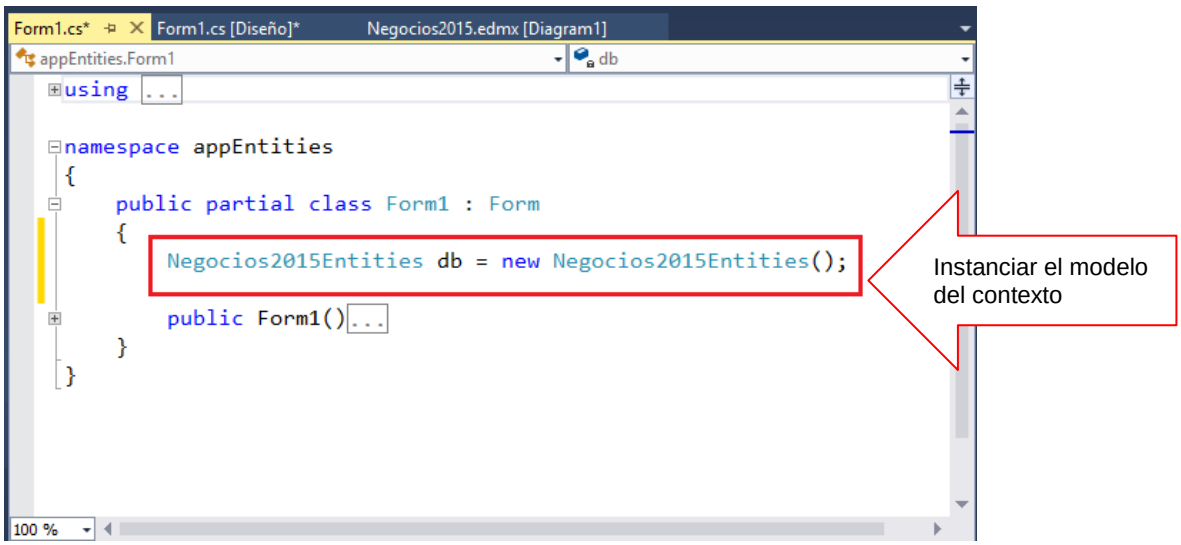
DISEÑO DEL FORMULARIO.

A continuación diseñe el formulario

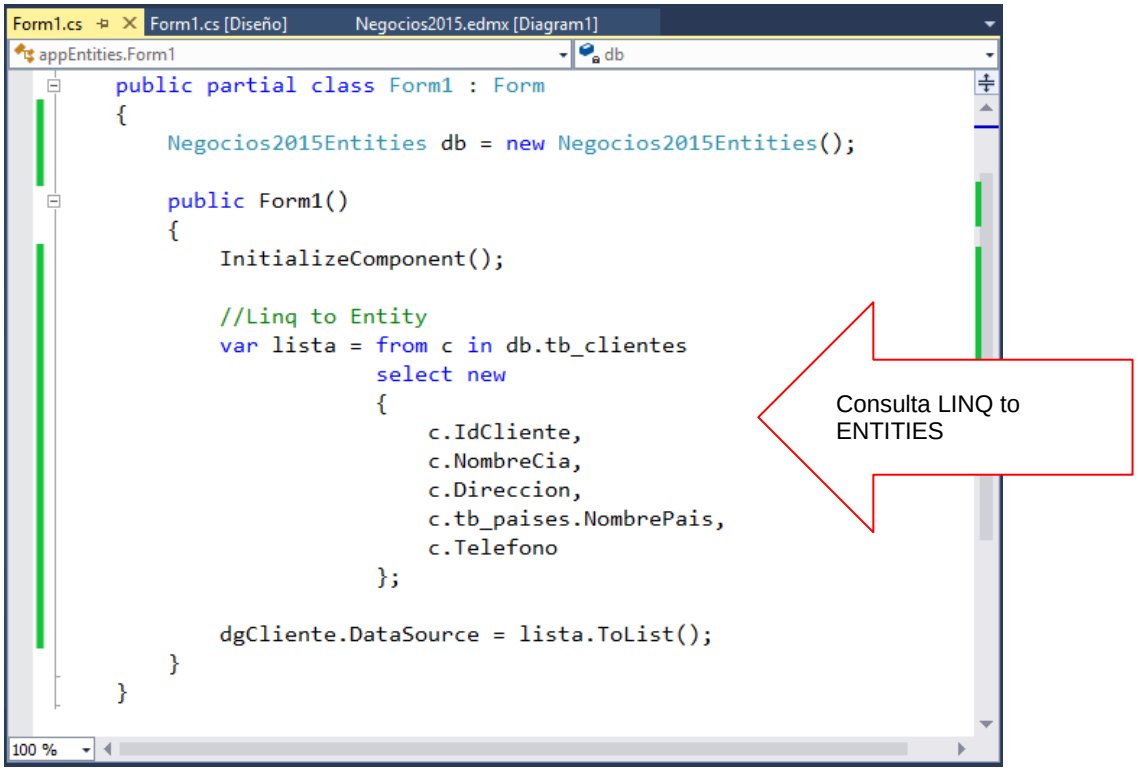


PROGRAMACION.

Instanciar el contexto a nivel formulario, tal como se muestra



Programa el constructor del Formulario, defina una consulta LINQ to ENTITIES, donde liste los registros de tb_clientes, visualizando los resultados en el DataGridView.



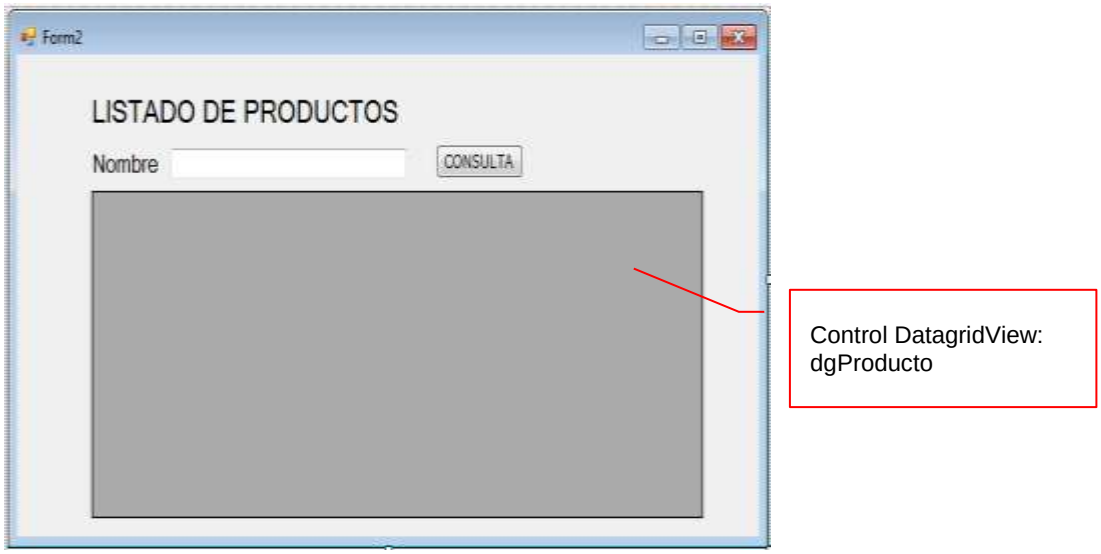
Para ejecutar el Formulario, presiona la tecla F5, donde se visualizan los registros

LABORATORIO 10.2

Se desea implementar un programa donde liste los registros de productos filtrando por la inicial de su campo NombreProducto.

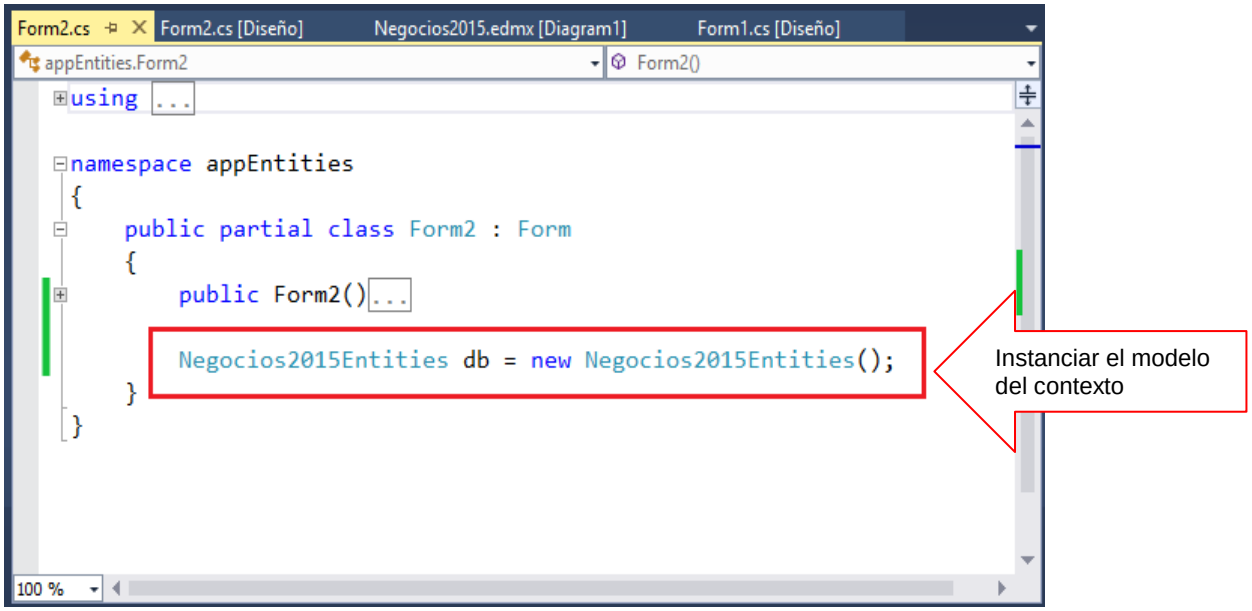
DISEÑO DEL FORMULARIO.

A continuación diseñe el formulario

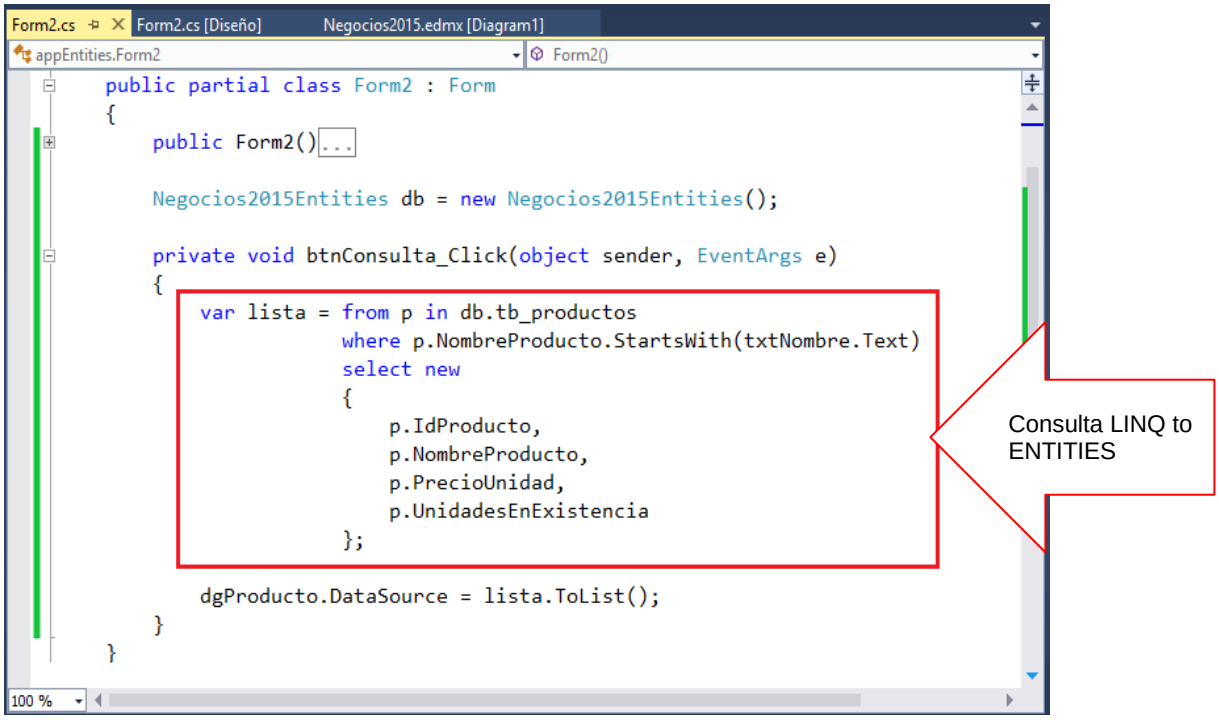


PROGRAMACION.

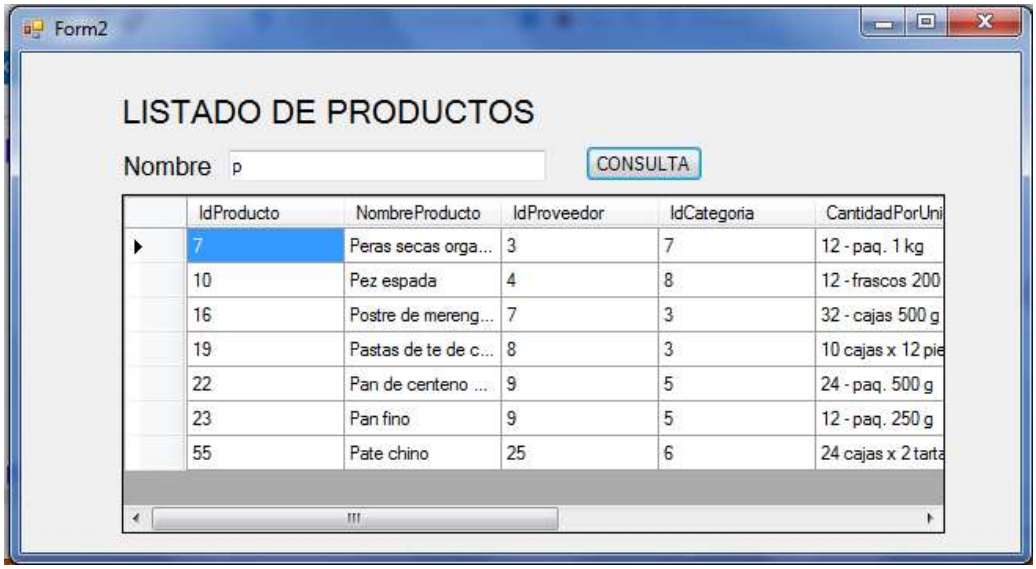
Instanciar el contexto a nivel formulario, tal como se muestra



Programe el evento Click del botón, donde liste los productos por la inicial de su nombre. Ejecute LINQ to ENTITIES



Para ejecutar el Formulario, presiona la tecla F5, ingrese la inicial del nombre del producto, al presionar el botón CONSULTA, se visualizan los registros por dicha condición

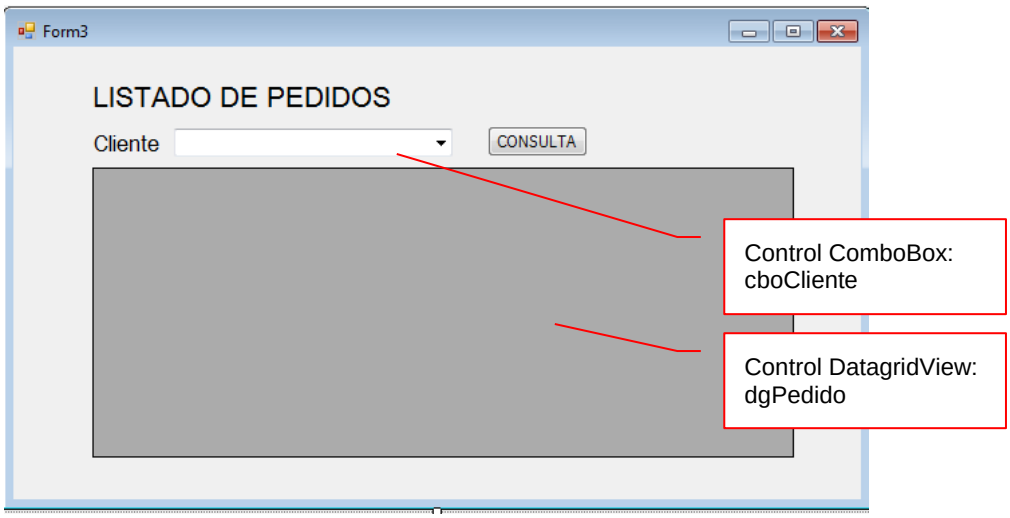


LABORATORIO 10.3

Se desea implementar un programa donde liste los registros de pedidoscabe filtrando por un cliente seleccionado desde un ComboBox.

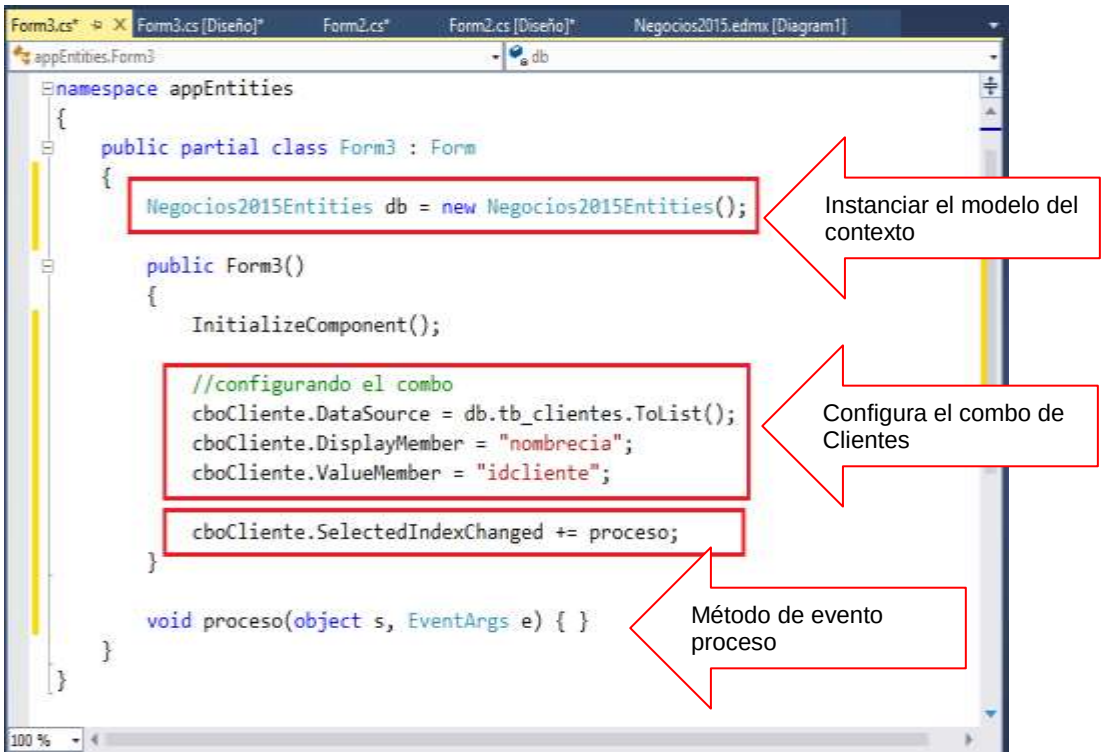
DISEÑO DEL FORMULARIO.

A continuación diseñe el formulario

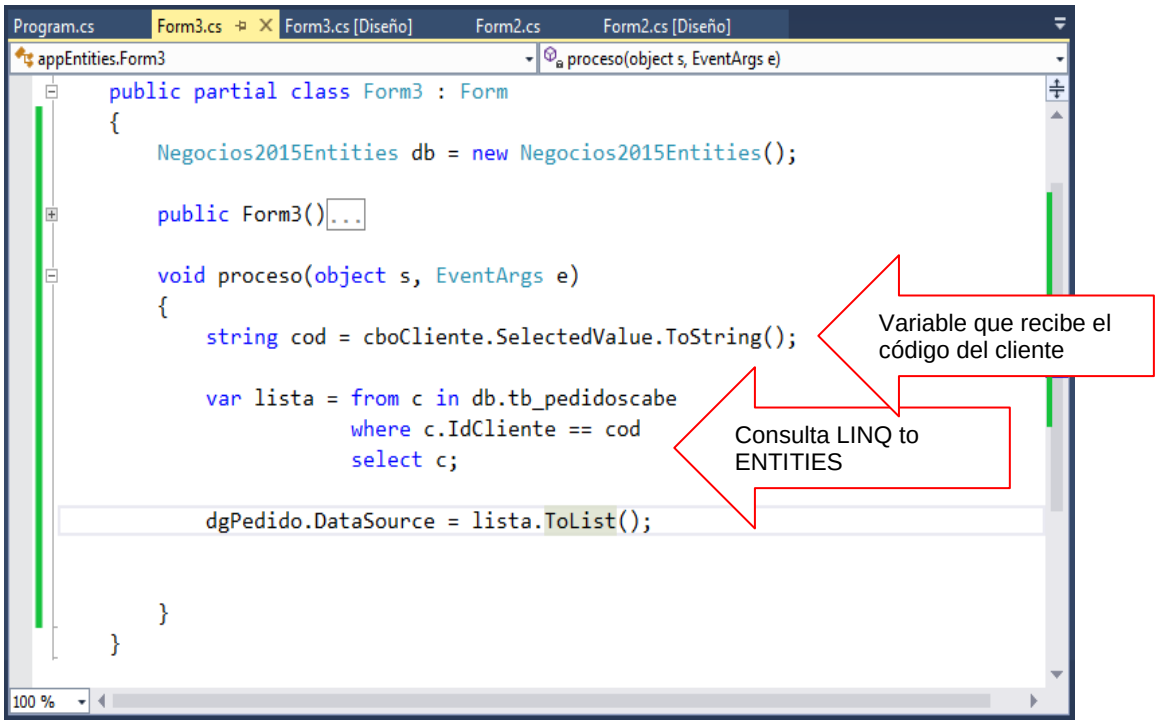


PROGRAMACION.

Instanciar el contexto a nivel formulario. Programa el evento Load del Formulario, donde configura el comboBox de Cliente.



Programe el evento Click del botón, donde liste los pedidos por un cliente seleccionado desde el control ComboBox. Ejecute LINQ to ENTITIES



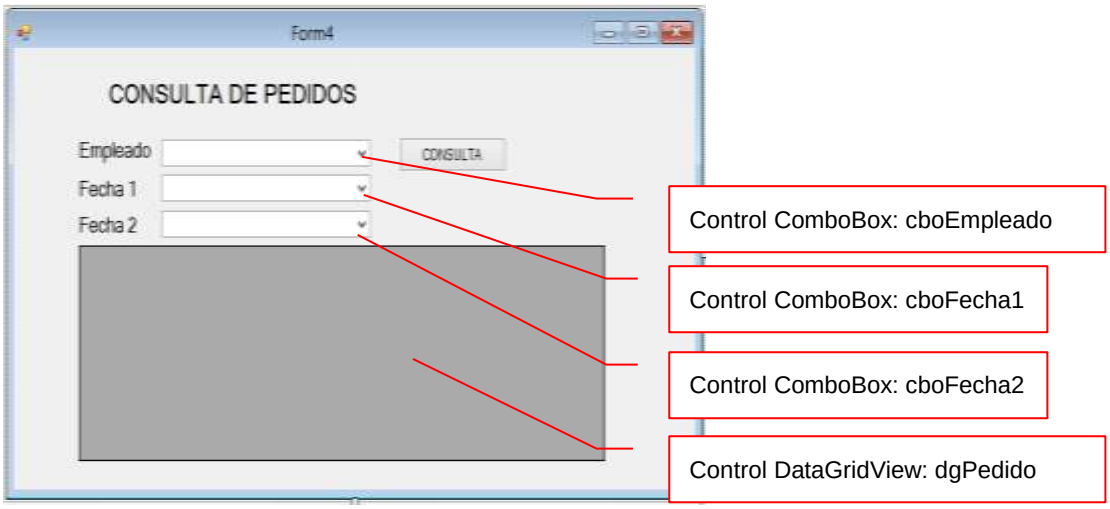
Para ejecutar el Formulario, presiona la tecla F5, selecciona el cliente, al presionar el botón CONSULTA, se visualizan los registros de pedidos por cliente seleccionado

LABORATORIO 10.4

Se desea implementar un programa donde liste los registros de pedidoscabe filtrando por un cliente seleccionado desde un ComboBox.

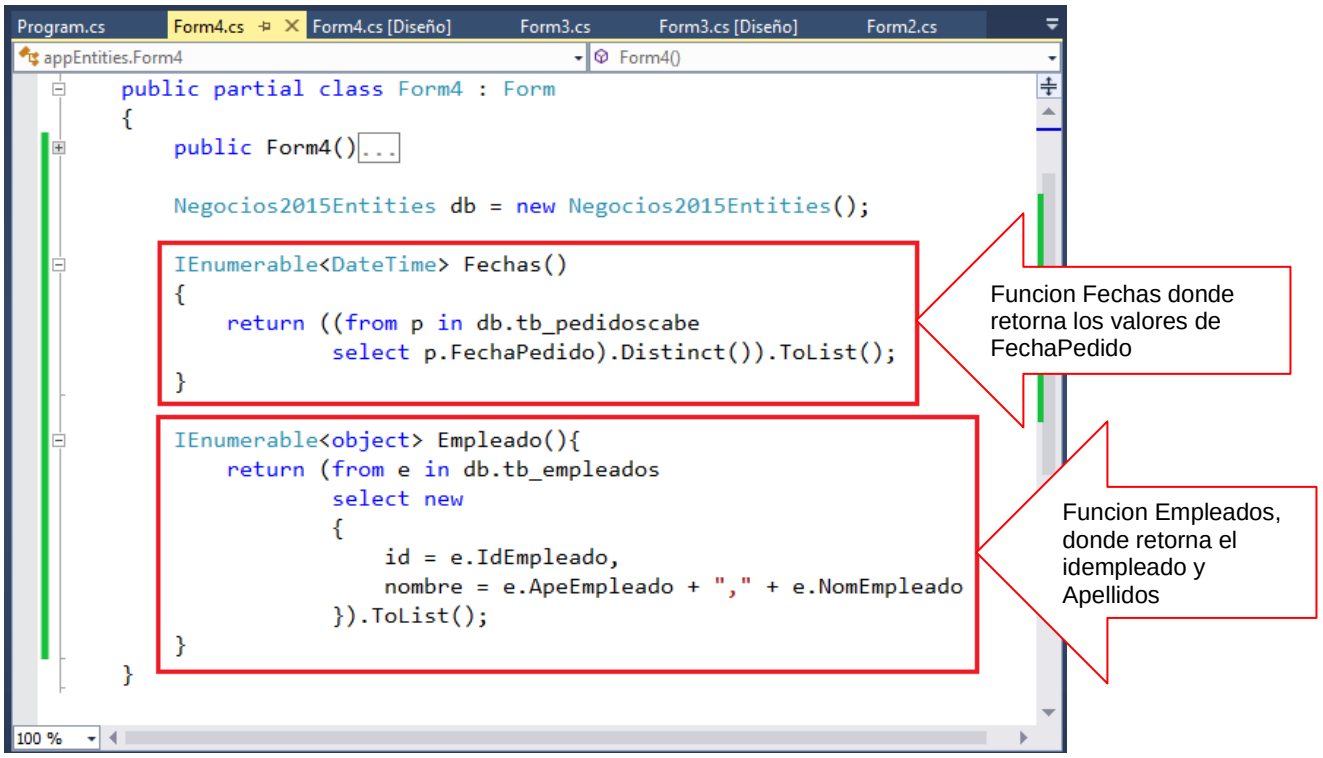
DISEÑO DEL FORMULARIO.

A continuación diseñe el formulario

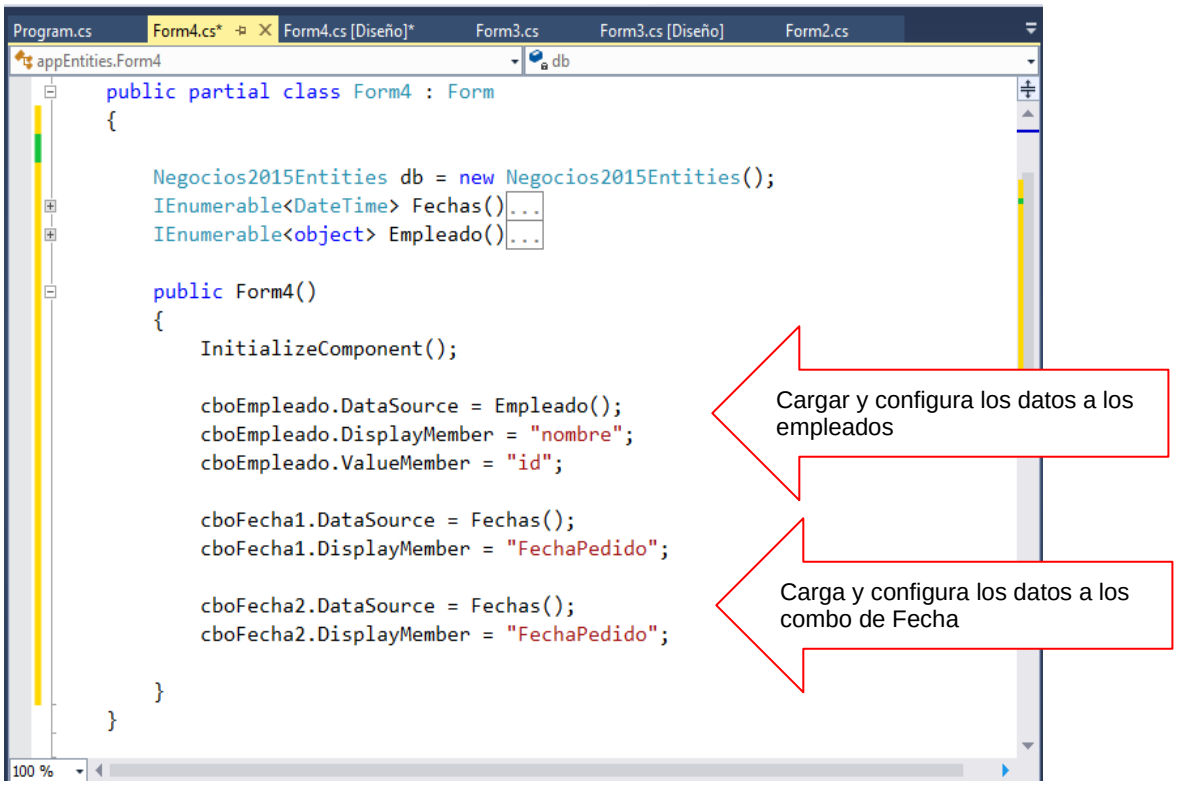


PROGRAMACION.

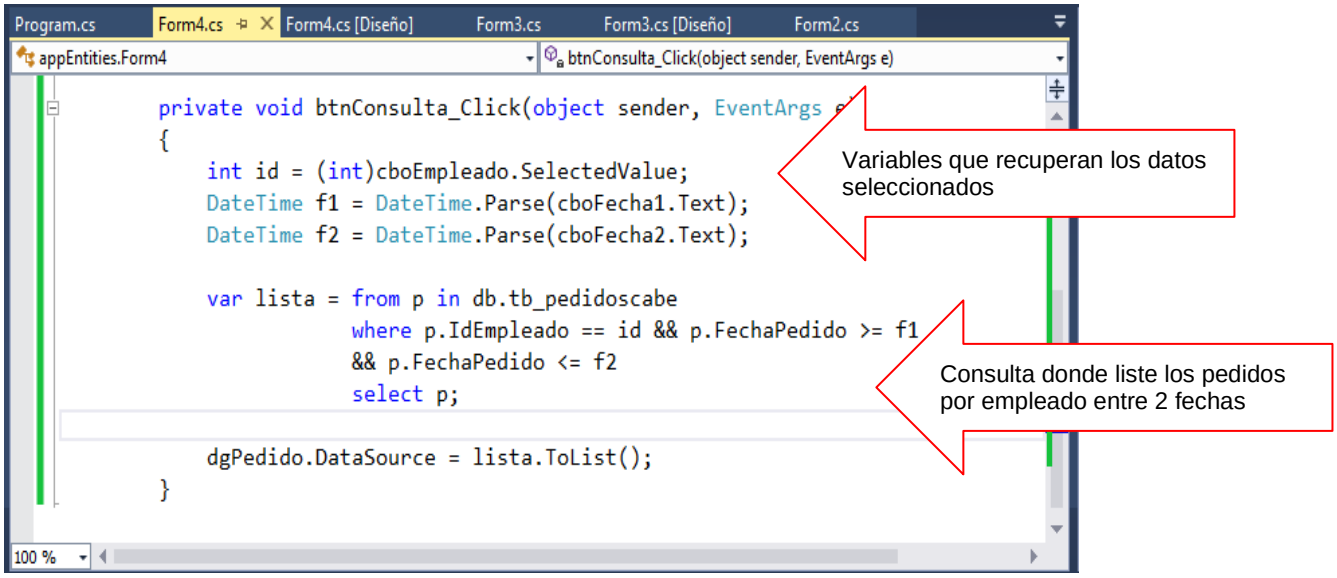
Instanciar el contexto a nivel formulario. Programa los métodos Empleados y Fechas, donde retornan una colección de datos



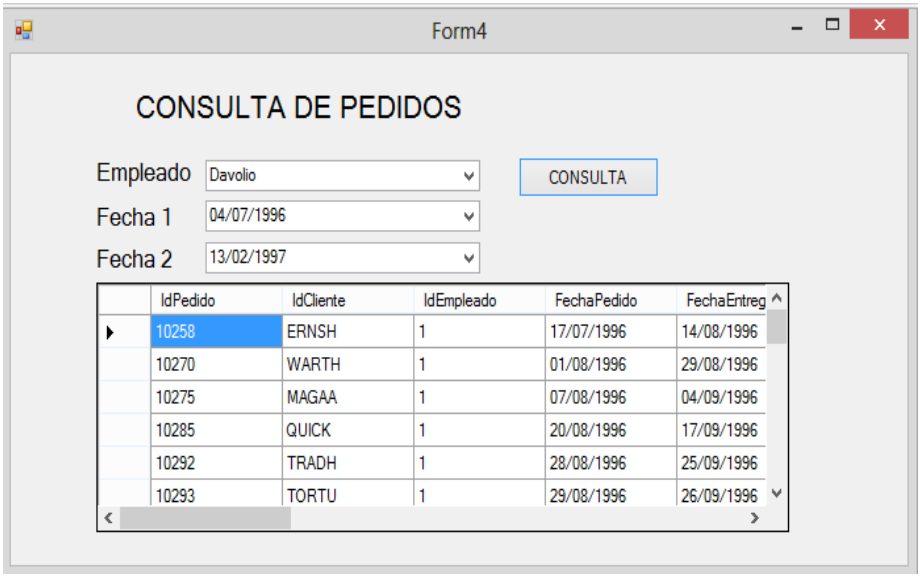
Programa el evento Load del Formulario, donde se ejecuta los métodos definidos anteriormente para cargar los datos a los controles ComboBox



Programa el botón Consulta, donde liste los pedidos poe empleado seleccionado entre un rango de dos fechas seleccionada desde comboBox



Presiona la tecla F5, selecciona el empleado y un rango de fechas para visualizar los pedidos por dicha condiciones

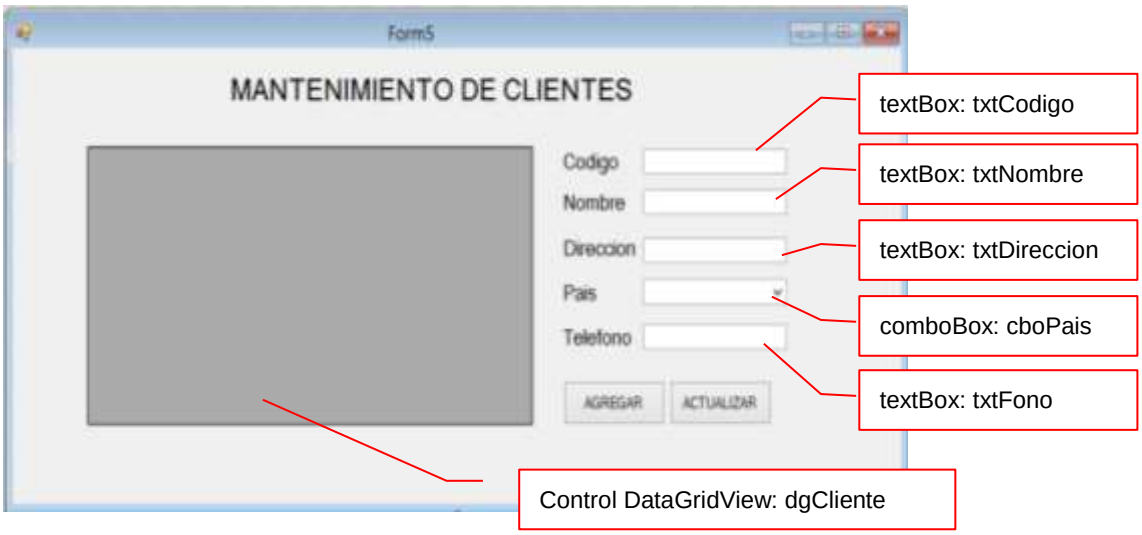


LABORATORIO 10.5

Se desea implementar un programa donde realice el mantenimiento a la tabla de tb_clientes

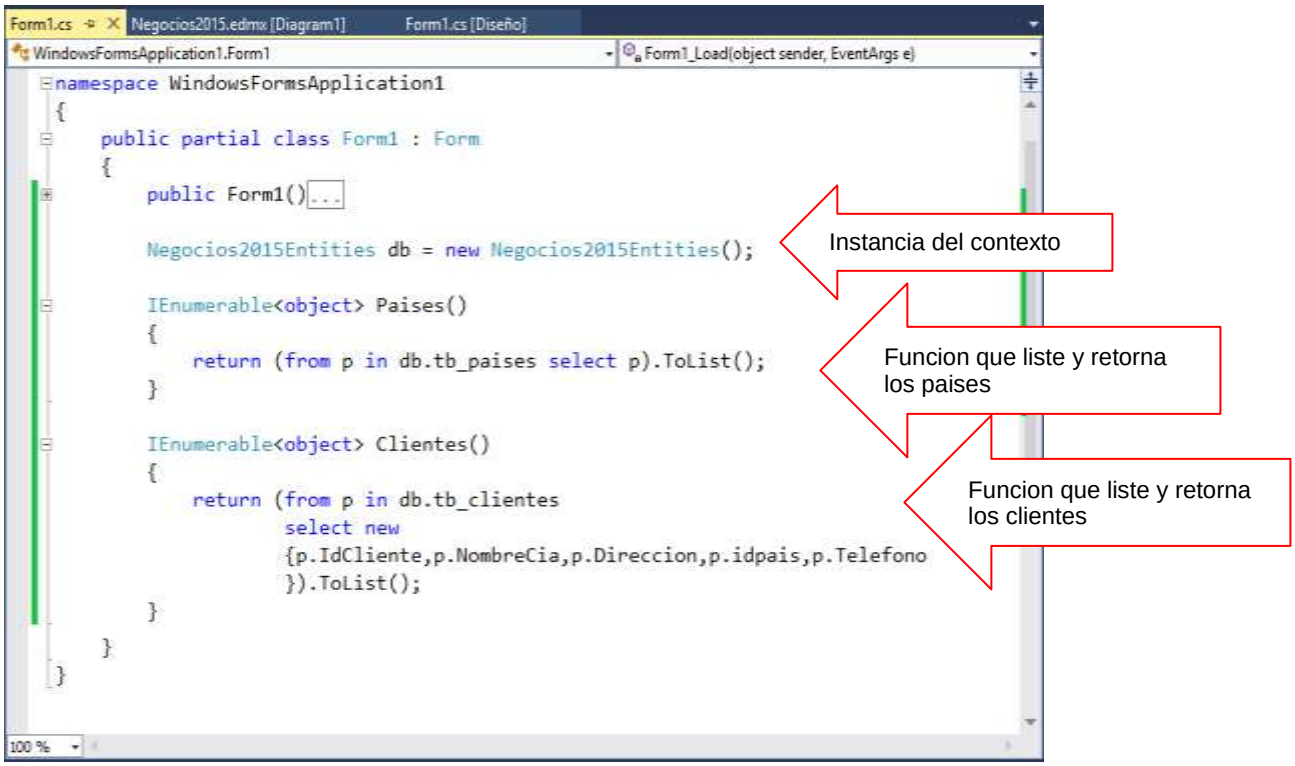
DISEÑO DEL FORMULARIO.

A continuación diseñe el formulario

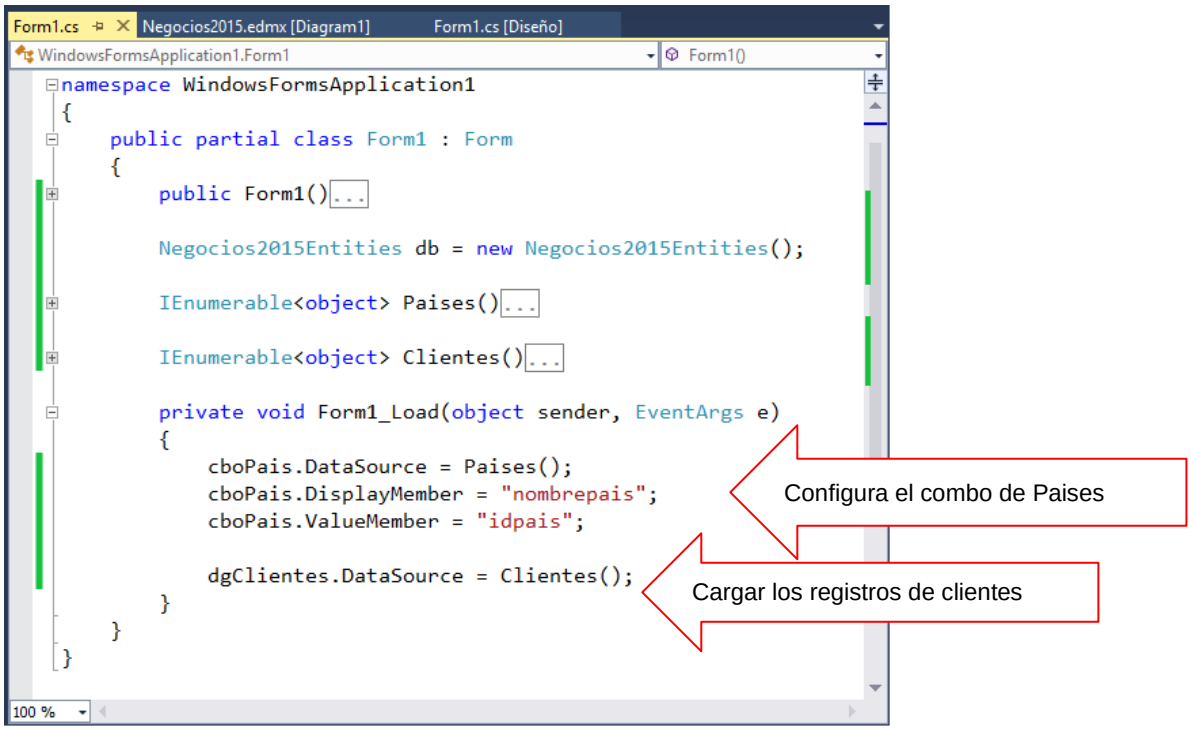


PROGRAMACION.

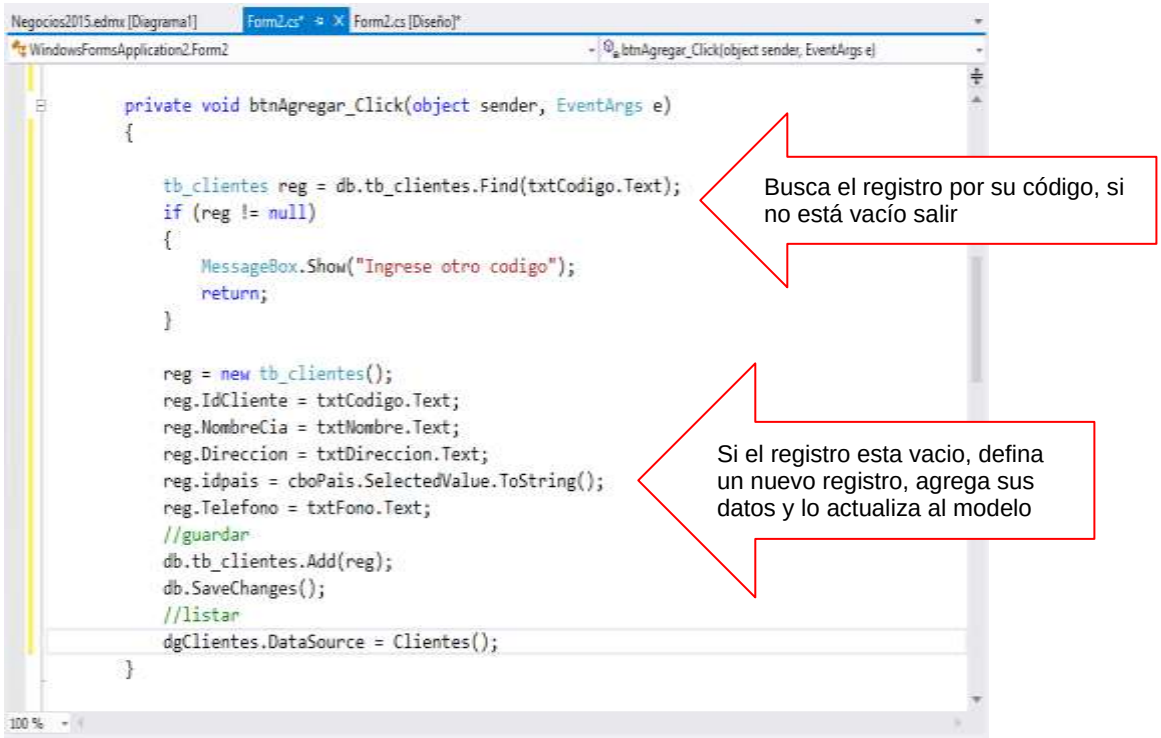
Instanciar el contexto a nivel Formulario y defina la función Paises donde retorna los paises



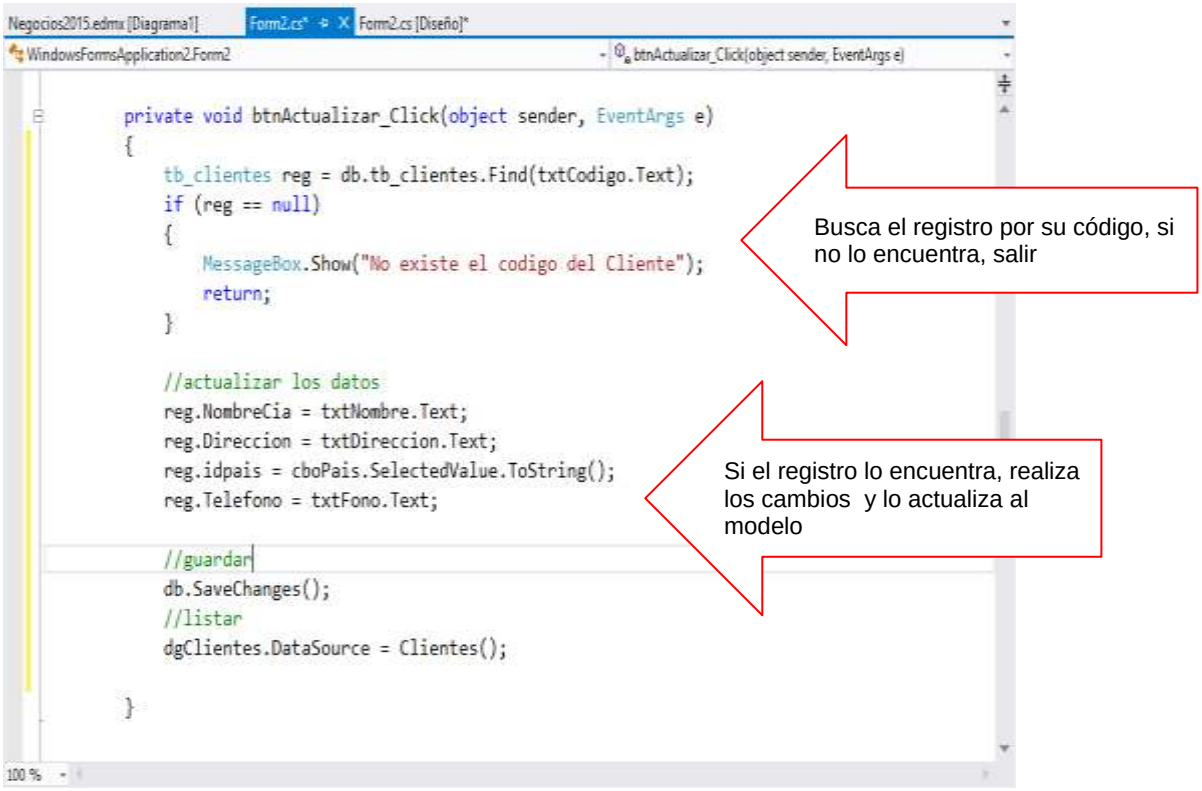
Programa el evento Load del Formulario, donde configura el combo de Pais y el control DataGridView para listar los clientes



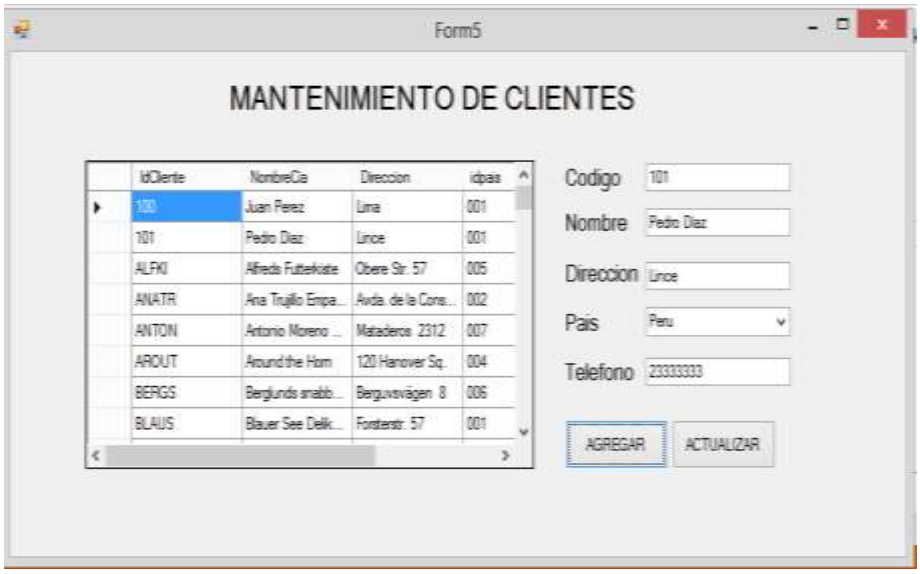
Programa el botón AGREGAR, donde verifique que el código no se repita, si es único, agrega el registro al modelo y actualiza los cambios



Programa el botón ACTUALIZAR, donde verifique que el código existe, si existe el registro, actualiza los cambios en el modelo



Presiona la tecla F5, y ejecuta las operaciones.



Resumen

- El Entity Framework es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos. Los arquitectos y programadores de aplicaciones orientadas a datos se han enfrentado a la necesidad de lograr dos objetivos muy diferentes. Deben modelar las entidades, las relaciones y la lógica de los problemas empresariales que resuelven, y también deben trabajar con los motores de datos que se usan para almacenar y recuperar los datos.
- Entity Framework permite a los desarrolladores trabajar con datos en forma de objetos y propiedades específicos del dominio, como clientes y direcciones de cliente, sin tener que preocuparse por las tablas y columnas de la base de datos subyacente donde se almacenan estos datos.
- El Entity Data Model (EDM) es la pieza central de Entity Framework. Especifica el esquema de diseño, que se usa para generar las clases programables que usa el código de la aplicación intercambiar datos dentro de nuestra organización sería mucho más eficiente si estos datos no ocuparan mucho espacio ya que de ese modo al ser transferidos sobre nuestra red no la sobrecargarían y el transporte sería muchísimo más rápido.
- El Entity Data Model (EDM) es un modelo de entidad relación. El EDM define los datos en un formato neutro que no está restringido por la estructura de los lenguajes de programación o las bases de datos relacionales. Los esquemas EDM se usan para especificar los detalles de las entidades y las relaciones, y para implementarlos como estructuras de datos
- Entity SQL con el proveedor EntityClient es el proveedor EntityClient define las clases y mapea los datos para interactuar con modelos de entidades de datos. EntityClient transforma las operaciones de las entidades hacia operaciones directas a las base de datos.
- La clase EntityConnection permite conectarse con un determinado origen de datos mediante una cadena de conexión en la que proporciona la información de autenticación necesaria
- El proveedor EntityClient posee una clase que permite ejecutar consultas sobre el modelo de entidad de datos: EntityCommand
- El proveedor EntityClient posee una clase que permite recuperar los datos de un origen de datos EntityDataReader. Un EntityDataReader no tiene un constructor público. Solo se puede obtener a través de una sobrecarga del método EntityCommand.ExecuteReader.
- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

http://msdn.microsoft.com/es-es/library/bb399572(v=vs.110).aspx

http://geeks.ms/blogs/jorge/archive/2012/02/10/publicado-entity-framework-4-3.aspx

http://msdn.microsoft.com/es-es/library/bb386876(v=vs.110).aspx

UNIDAD

5

INTRODUCCION AL DESARROLLO WEB

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno realiza consultas y actualización de datos a través del lenguaje de consulta integrado utilizando la plataforma ORM en un entorno de una aplicación Windows y en una aplicación Modelo Vista Controlador (MVC).

Temario

Tema 11: Introducción a ASP.NET MVC (6 horas)

1. Introducción a ASP.NET MVC
1.1. Arquitectura de la Web
1.2. Patrón MVC (Modelo Vista Controlador)
1.3. ASP.NET MVC 5.0
1.4. Scaffolding y Razon
1.5. Web Form y MVC

ACTIVIDADES PROPUESTAS

- Los alumnos conocer el modelo vista controlador
- Los alumnos realizan operaciones de consulta utilizando el patrón MVC

11. INTRODUCCION A ASP.NET MVC

11.1 Arquitectura de la Web

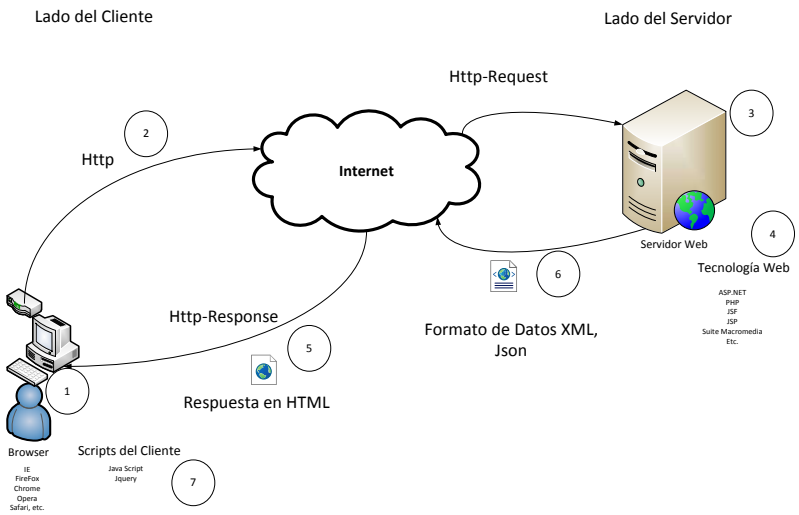
La World Wide Web (“WWW” o simplemente la "Web") es un medio global de información cuyos usuarios pueden leer y escribir a través de computadoras conectadas a Internet. El término es a menudo usado erróneamente como un sinónimo para la Internet misma, pero la Web es un servicio que opera sobre la Internet, como también lo hace el correo electrónico.

En 1980, Tim Berners-Lee, un contratista independiente en la Organización Europea para la Investigación Nuclear (CERN por sus siglas en ingles), Suiza, desarrollo ENQUIRE, como una base de datos personal de gente y modelos de software, pero también como una forma de interactuar con el hipertexto; cada nueva página de información en ENQUIRE debía estar enlazada a una página existente

En la navidad de 1990, Berners-Lee había desarrollado todas las herramientas necesarias para trabajar la Web: el Protocolo de transferencia de hipertexto, el Lenguaje de Marcado de Hipertexto, el primer navegador web (llamado WorldWideWeb, que fue también un editor de páginas web), el primer servidor de aplicaciones HTTP, el primer servidor web (http://info.cern.ch) y las primeras páginas web que describían el proyecto mismo.
El 6 de agosto de 1991,9 Berners-Lee publicó un breve resumen del proyecto de la World Wide Web en el grupo de noticias alt.hypertext.10 Esta fecha también marca el inicio de la Web como un servicio públicamente disponible en Internet.

La Web hoy en día cuenta con una serie de componentes adicionales que interactúan entre sí para poder hacer realidad su existencia y correcto funcionamiento.

La siguiente figura muestra los principales componentes de su arquitectura:



1 Cliente Web o Browser: es quien da inicio a la interacción de la web a través de un programa especial llamado Navegador o Browser y solicitando una dirección URL, actualmente en el mercado los principales navegadores son Internet Explorer, Firefox, Chrome, Opera, Safari, etc.

2 Protocolo de comunicación Http: este protocolo hace posible la comunicación desde la maquina cliente hasta la maquina servidora donde se encuentra la aplicaciones web, se vale del ruteo a través del router, direcciones IP, servidores DNS, etc.

3 Servidor Web: son máquinas especiales que alojan las aplicaciones Web construidas, en el mercado existe una diversidad de servidores Web tales como Apache, IIS, TomCat, Http-IBM, etc.

4 Tecnología Web: son las herramientas que nos ayudan a crear las aplicaciones Web, las herramientas más conocidas son PHP, ASP.NET, JSP, JSF, DreamWeaver, etc.

5 Respuesta HTML: cuando el servidor web recepciona la solicitud de página y procesa el resultado, siempre hacia el usuario final envía un documento de formato especial llamado HTML, este lenguaje HTML es interpretado por los Browser para formatear y representar la página final.

6 Formato de datos: en algunos escenarios también se envía data hacia el lado del cliente (Web Browser), dos de los formatos más utilizados para el envío de data son XML y Json

7 Scripts del cliente: finalmente, si se quiere dar mayor interacción en el Browser, se ejecutan programas clientes conocidos como códigos script, dos de las opciones más utilizadas para la ejecución de script son JavaScript y Jquery.

11.2 Patrón MVC (MODELO VISTA CONTROLADOR)

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

Modelo
- Contiene el núcleo de la funcionalidad (dominio) de la aplicación.
- Encapsula el estado de la aplicación.
- No sabe nada / independiente del Controlador y la Vista.

Vista
- Es la presentación del Modelo.
- Puede acceder al Modelo pero nunca cambiar su estado.
- Puede ser notificada cuando hay un cambio de estado en el Modelo.

Controlador
- Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente

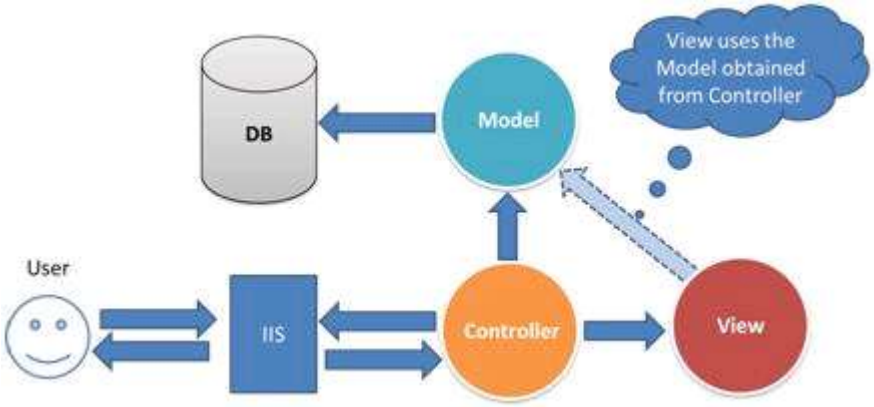
Es importante mencionar que el patrón MVC no es exclusivo para el diseño Web, en sus inicios fue muy utilizado para el desarrollo de interfaces graficas de usuario

(GUI), por otro lado tampoco es una implementación propietaria de alguna empresa tecnológica, sea Microsoft, Oracle o IBM.

MVC está implementando por muchas herramientas tales como:

- Ruby
- Java
- Perl
- PHP
- Python
- .NET

La siguiente figura muestra la idea grafica del patrón MVC para el entorno de la Web.



11.3 ASP.NET MVC 5.0

Como se comentó en el apartado anterior, el patrón MVC es implementado por muchas herramientas tecnológicas, Microsoft ha implementa el patrón MVC en su tecnología de ASP.NET, para el desarrollo de aplicaciones web.

ASP.NET MVC es un poderoso framework para la construcción de sitios Web basándose en los estándares de internet actuales tales como HTML 5, jquery, CSS 3, etc.

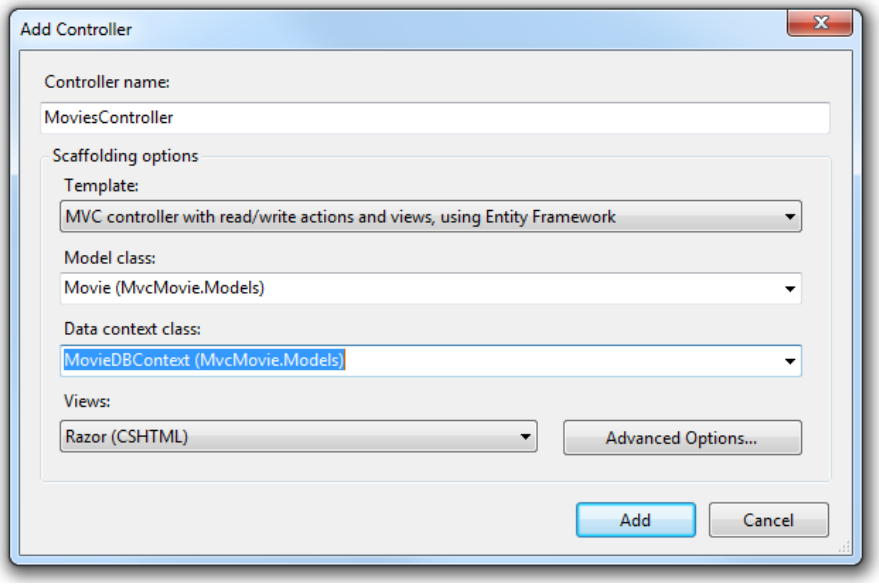
En el momento de crear este manual la versión actual es la MVC 5.0 que presenta las siguientes nuevas características:

- Soporte para la creación de aplicaciones para Facebook.
- Soporte para proveedores de autenticación a través del OAuth Providers.
- Plantillas por default renovadas, con un estilo mejorado.
- Mejoras en el soporte para el patrón Inversion Of Control e integración con Unity
- Mejoras en el ASP.NET Web Api, para dar soporte a las implementaciones basadas en RESTful
- Validaciones en lado del modelo
- Uso de controladores Asíncronos
- Soporte para el desarrollo de aplicaciones Web Móvil, totalmente compatible con los navegadores de los modernos SmartPhone (Windows Phone, Apple y Android), etc.

11.4 Scaffolding y Razor

La palabra **Scaffold** está en inglés y en español significa "Andamio", pero en programación el scaffolding es un método para construir aplicaciones basadas en bases de datos, esta técnica está soportada por algunos frameworks del tipo MVC en el cuál el programador escribe una especificación que describe cómo debe ser usada la base de datos. Luego el compilador utiliza esa especificación para generar el código que la aplicación usará para crear, leer, actualizar y eliminar registros de la base de datos, esto es conocido como CRUD (create, read, update, delete).El Scaffolding fue popularizado por el framework Ruby on Rails y ahora es utilizado por otros frameworks tales como CakePHP, Symfony, ASP.NET MVC, etc.
El valor agregado de ASP.NET MVC es su extremada sencillez al momento de hacer uso del Scaffold, generando aplicaciones de mantenimiento en tiempo record.

La siguiente imagen muestra la parte donde se va a generar todo el código a través del Scaffolding.



Por otro lado el equipo de ASP.NET vio la necesidad de contar con un motor de renderizado que sea simple, útil y que de una mayor interacción entre el modelo y la vista.
El resultado de esta necesidad es **Razor**.

ASP.NET Web Pages-Razor proporciona una sintaxis de programación simple para escribir código en páginas web donde el código basado en servidor se incrusta en el formato HTML de las páginas web. El código de Razor se ejecuta en el servidor antes de que la página se envíe al explorador. Este código de servidor puede crear dinámicamente contenido de cliente, es decir, puede generar formato HTML u otro contenido sobre la marcha y, a continuación, enviarlo al explorador junto con cualquier código HTML estático que contenga la página

Finalmente Razor no es un nuevo lenguaje de programación, por el contrario se basa en sintaxis de C# y VB, teniendo como principal objetivo reutilizar el conocimiento de los programadores de .NET.

En la siguiente figura se muestra un ejemplo de la sintaxis Razor.

11.5 WEB FORM Y MVC

ASP.NET ofrece dos alternativas para el desarrollo de aplicaciones Web, la tecnología ASP.NET MVC y la tecnología de ASP.NET Web Form.
Ambas opciones presentan una serie de ventajas y desventajas que se muestran en los siguientes cuadros:

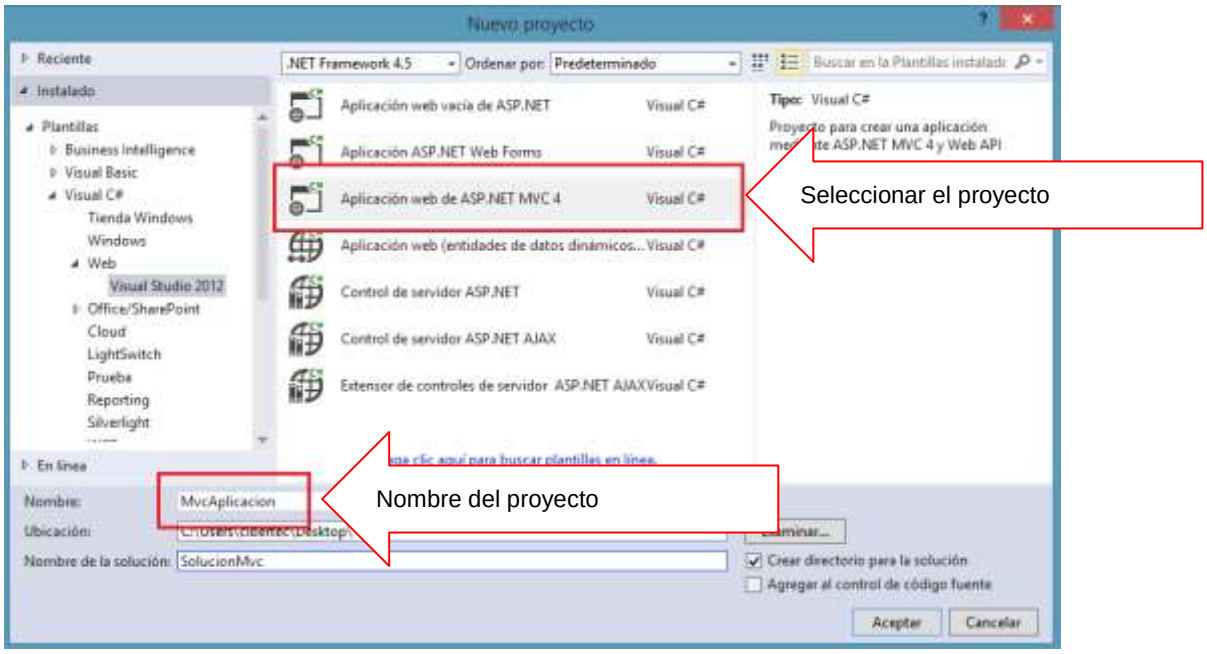
ASP.NET Web Form	
Ventajas	**Desventajas**
Tecnología madura y estable con soporte por miles de controles y herramientas de terceros	Problemas para dar soporte al paradigma de separación de conceptos (Soc)
Modelo de programación orientado a eventos, haciéndose muy similar al desarrollo de aplicaciones Windows.	Problemas para orientarlo a un proyecto de Testing, TDD (Desarrollo orientado al Test)
Soporte al manejo de estados.	Problemas de rendimiento cuando se hace uso excesivo del manejo de estados.
Se necesita un conocimiento básico de HTML y javascript para construir interfaces Web	Menos control del HMTL generado
Excelentes mecanismos de seguridad construidos de manera automática.	Menos soporte para el desarrollo con múltiples equipos (desarrollo paralelo)

ASP.NET MVC	
Ventajas	**Desventajas**
Ruteo URL y mejor soporte para los motores de búsqueda	Mayor esfuerzo en el aprendizaje del framework.
Excelente soporte para el desarrollo de tipo TDD (Desarrollo orientado al Test)	
Mejora en la separación de conceptos ya que usa el Modelo, la Vista y el controlador	
Fácil manejo en desarrollo con múltiples equipos.	
Total control en el HTML para las vistas	
MVC es un framework extensible y es un proyecto Open Source.	

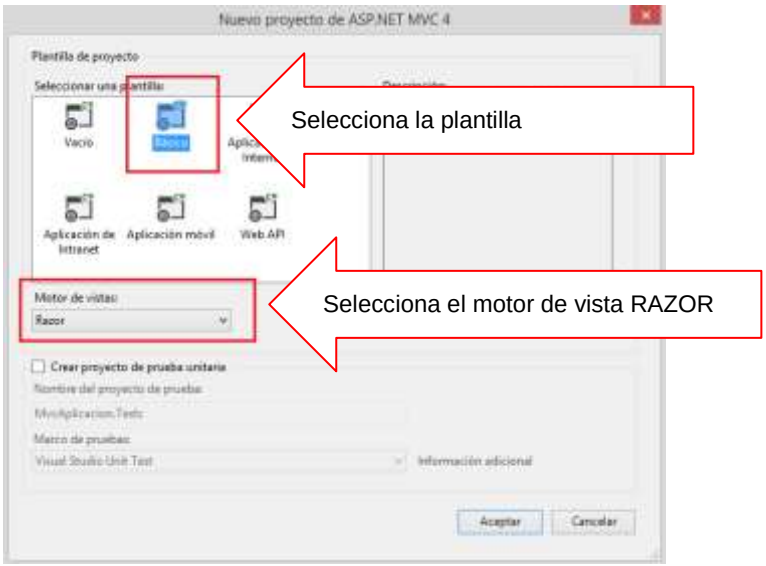
Laboratorio 11.1
Crear un proyecto básico en MVC

Implementa una aplicación Web MVC que nos de la bienvenida.

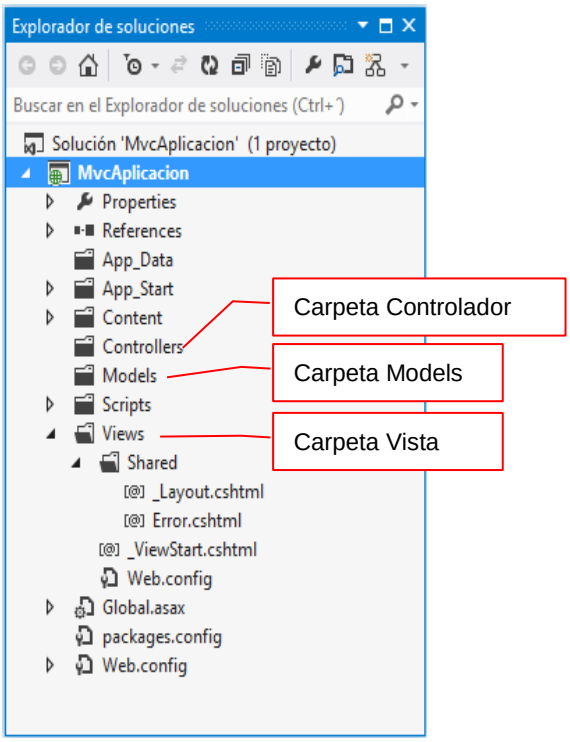
1. Creando el proyecto, selecciona el proyecto Aplicación web de ASP.NET MVC4,tal como se muesta. Asigne el nombre al proyecto



A continuación selecciona la plantilla del proyecto “Basico” y el motor de Vista “Razor”, tal como se muestra, luego presiona el boton ACEPTAR

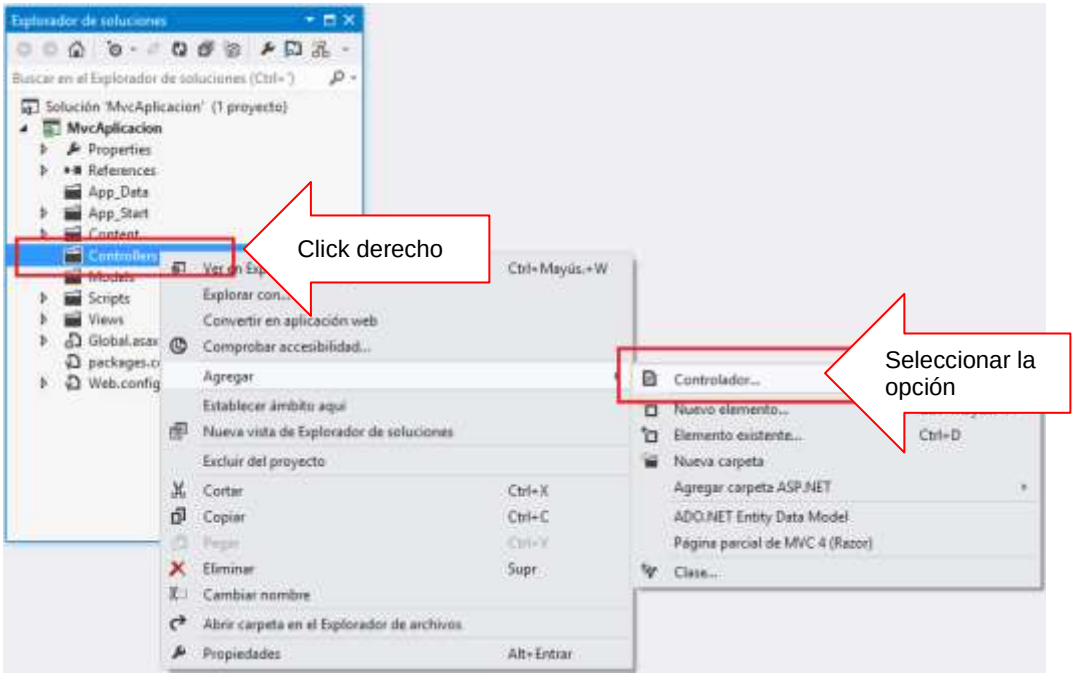


Ventana del explorador, donde se visualiza las carpetas del Modelo.

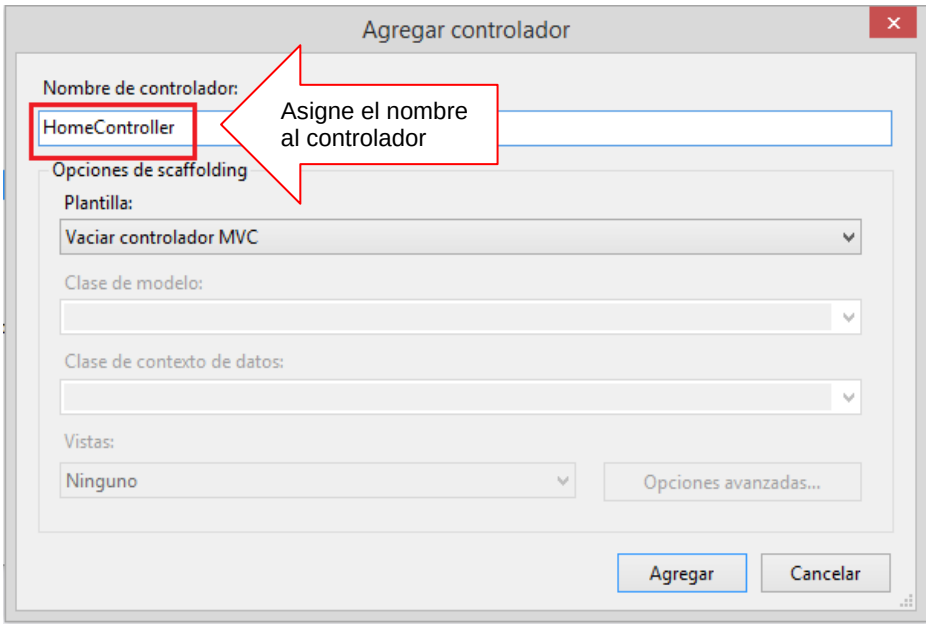


<u>Agregando un controlador</u>

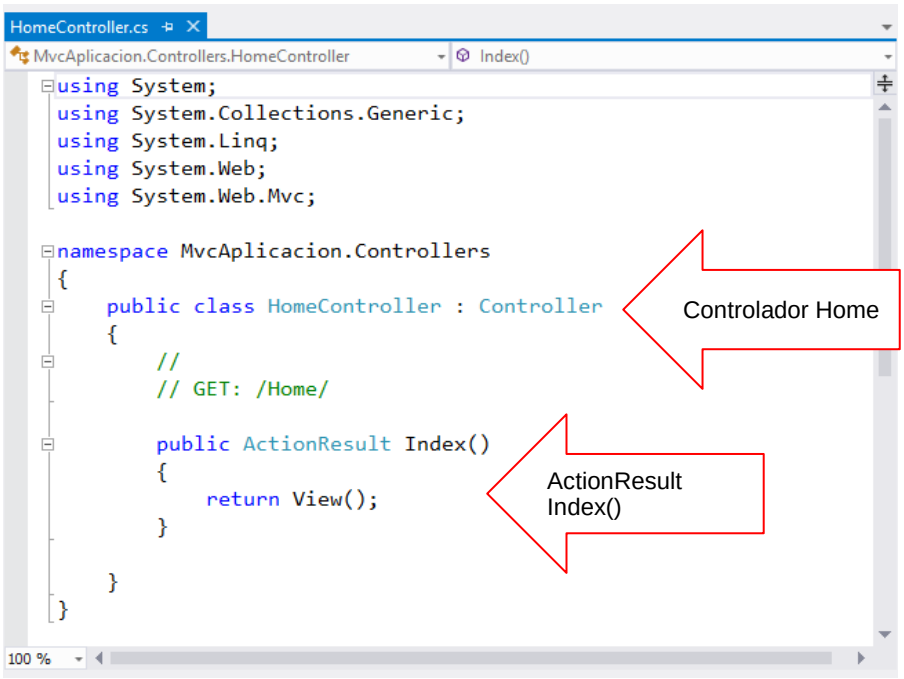
Desde la carpeta controlador, selecciona desde AGREGAR, la opción Controlador, tal como se muestra



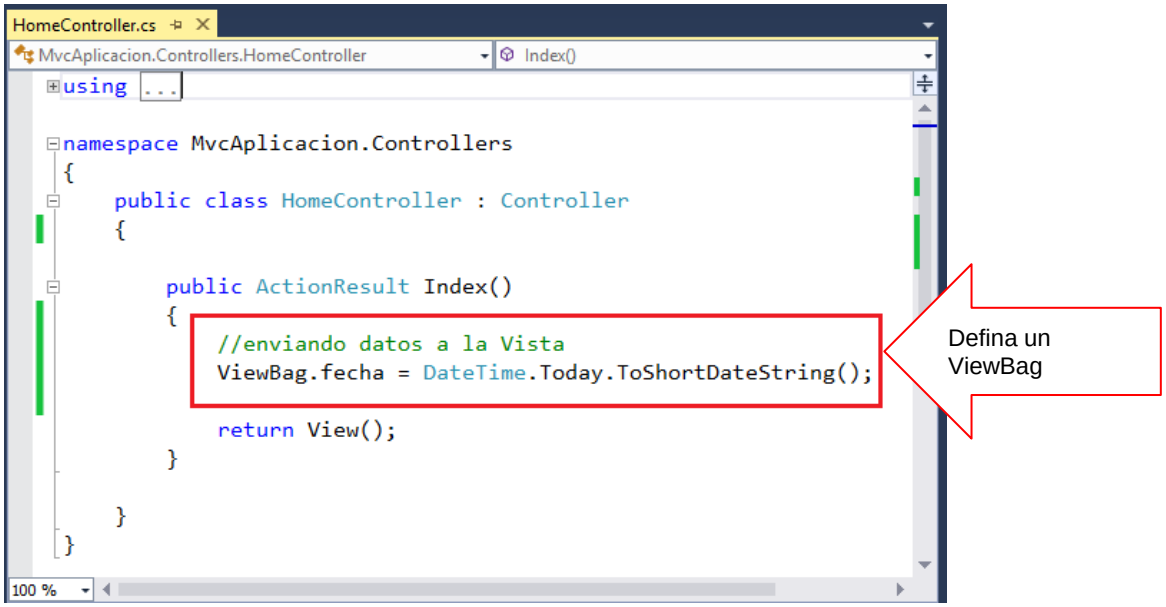
Asigne el nombre al controlador: HomeController, presiona el boton Agregar



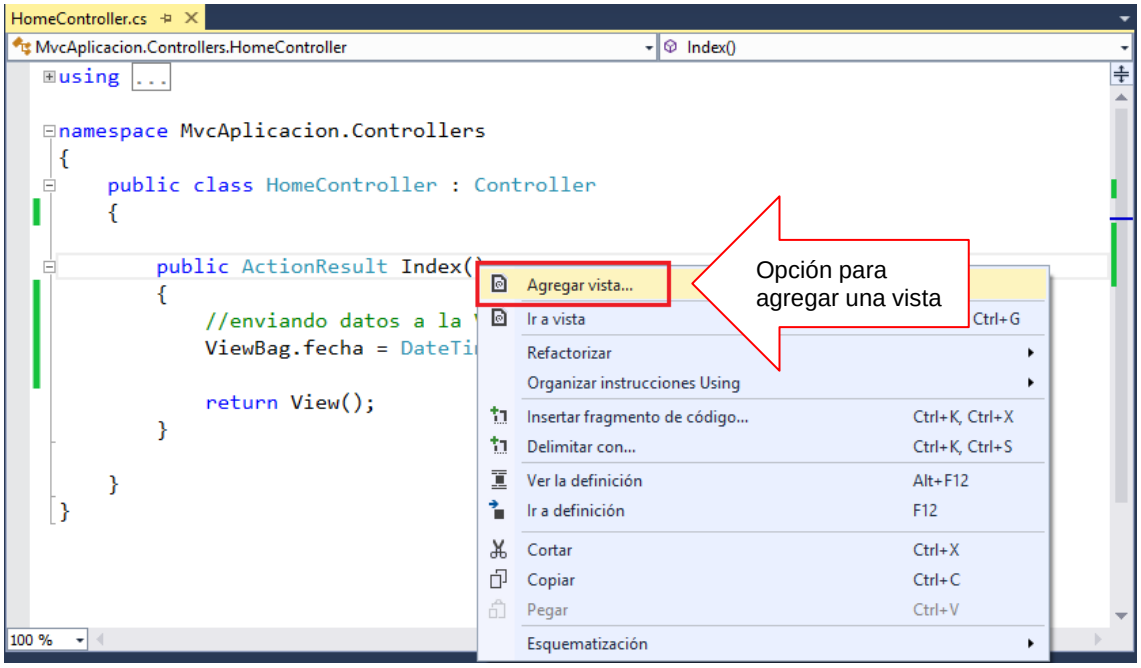
A crearse el controlador, se define el ActionResult Index()



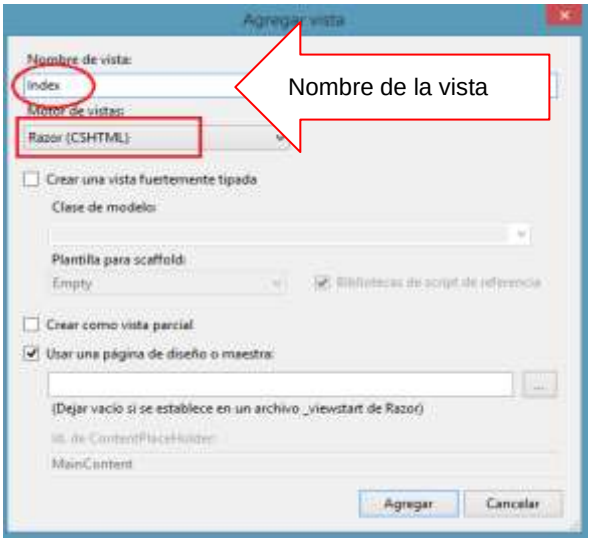
En el ActionResult Index, defina un ViewBag (diccionario) el cual le recibe el valor de la fecha del sistema.



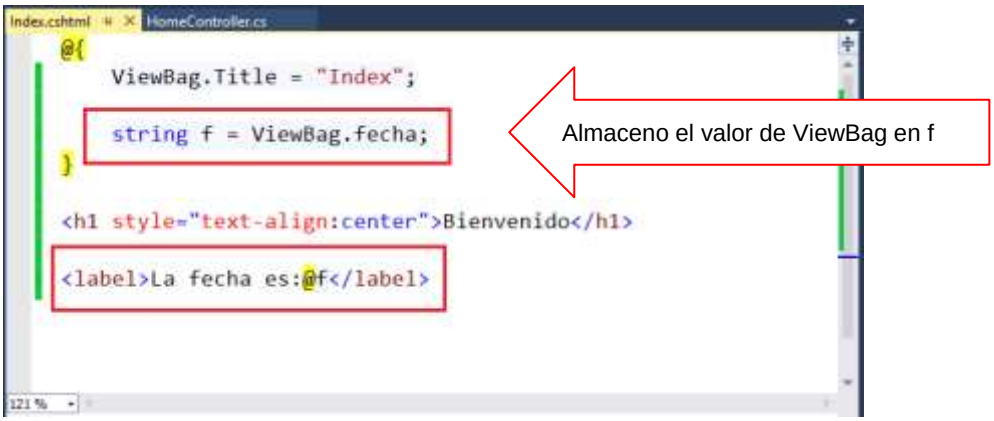
A continuación agregamos una vista. Desde el Index(), hacer un click derecho y selecciona la opción **Agregar Vista…,** tal como se muestra



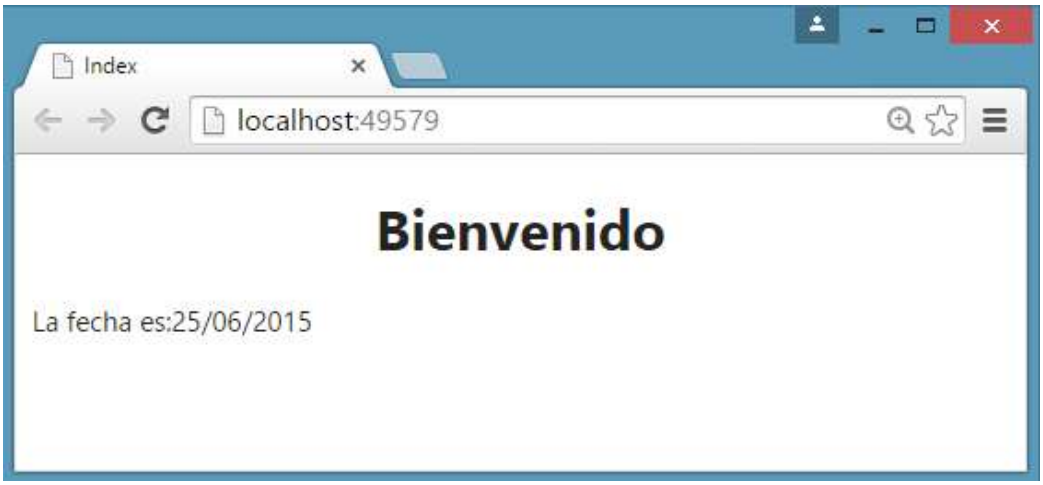
En la ventana Agregar Vista, se muestra el nombre de la vista, presiona el botón AGREGAR



En la vista, recibimos el valor del ViewBag a través de una variable llamada f, luego la visualizamos, a través de un label, en la vista: @f



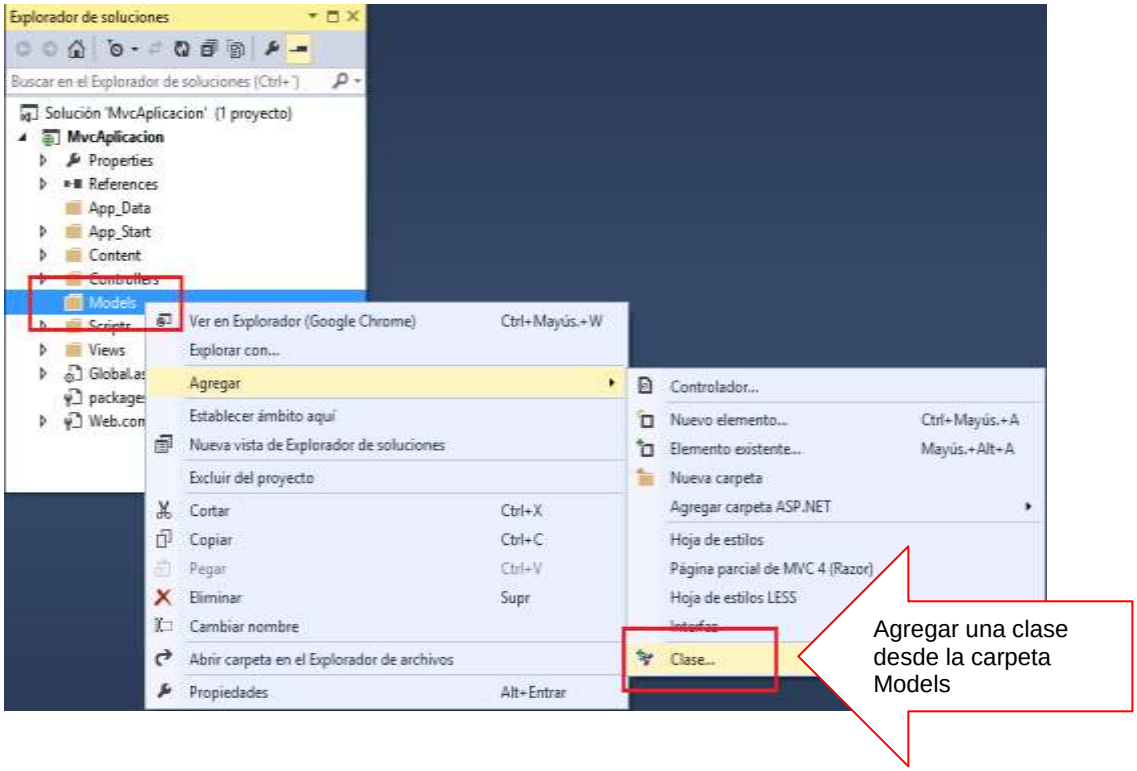
Para ejecutar el proyecto, presiona la tecla F5, donde se visualiza la Vista Index y su contenido, tal como se muestra.

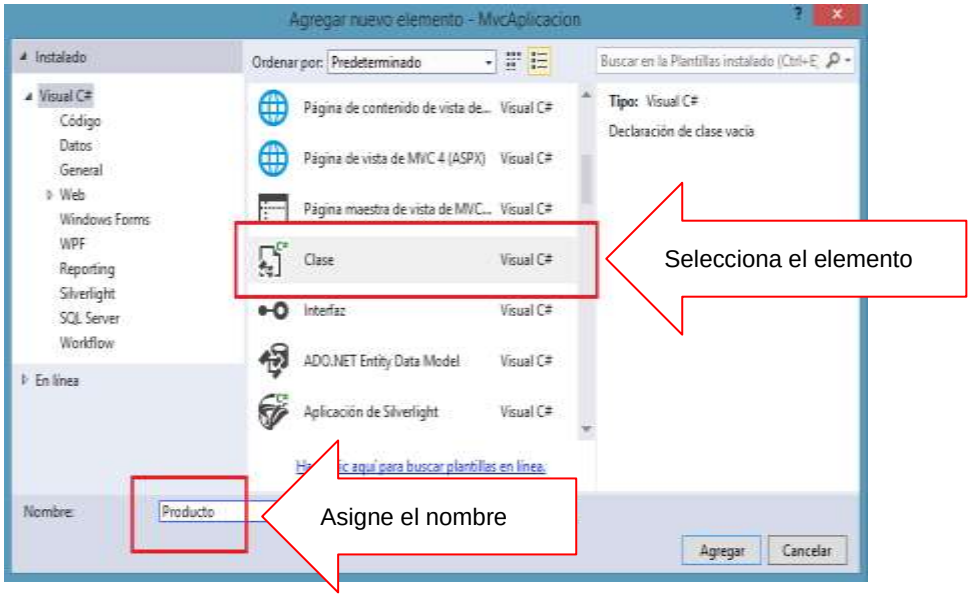


Laboratorio 11.2
Trabajando con parámetros en MVC

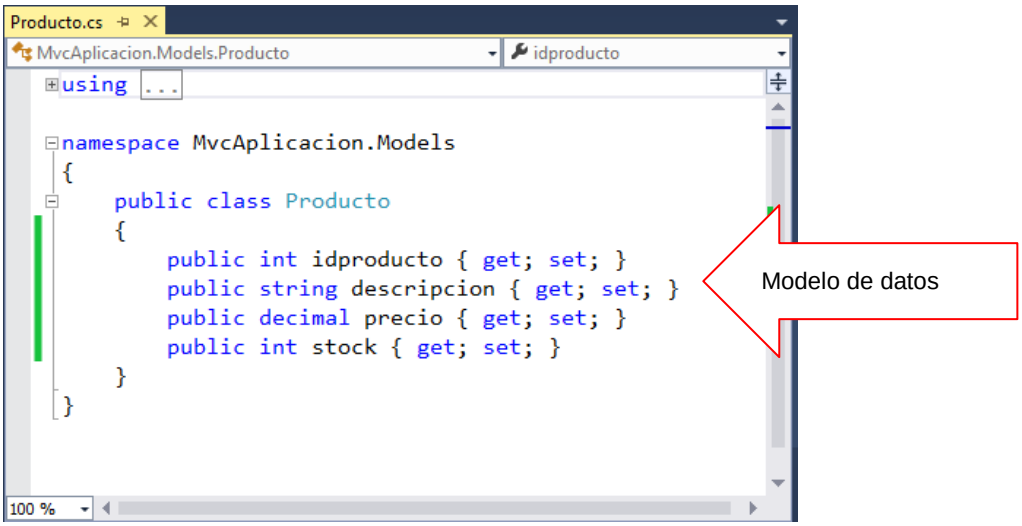
Implementa una aplicación donde permita visualizar los datos ingresados desde un controlador a una vista

En la carpeta Models, agregamos una clase llamada Producto, tal como se muestra.

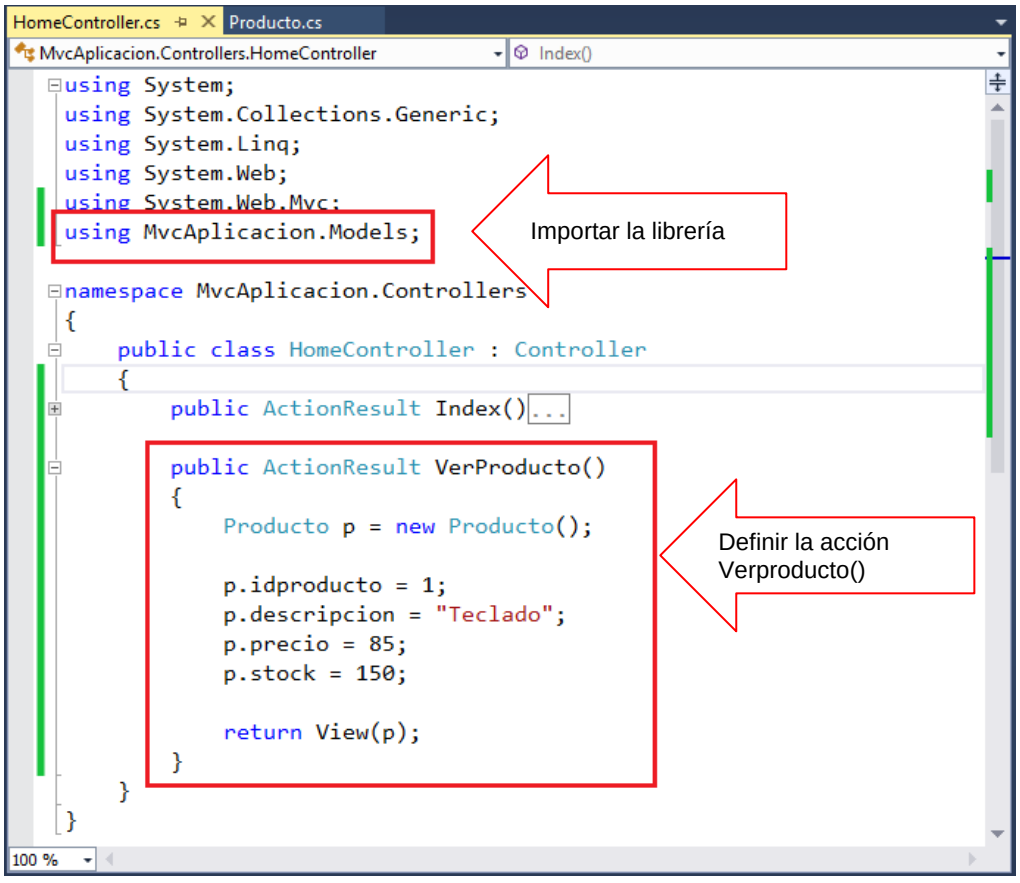




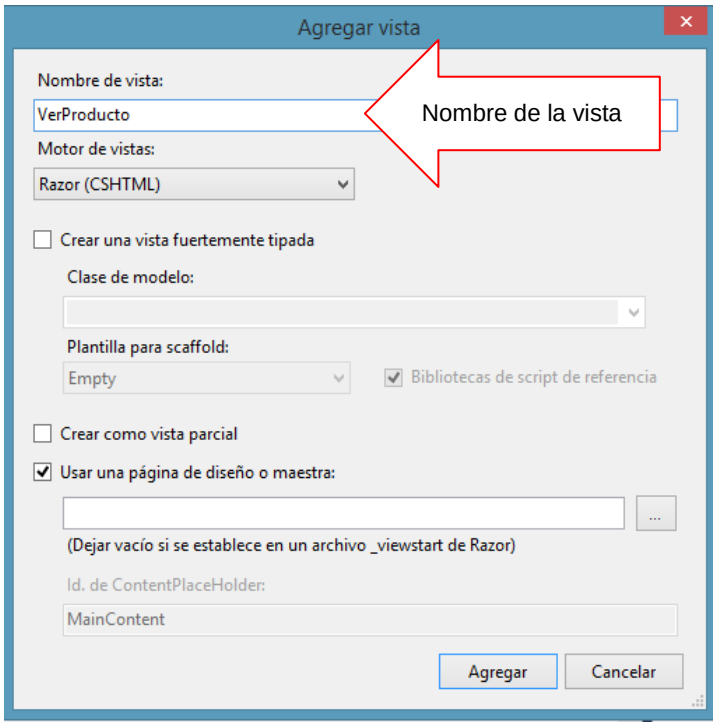
Defina la estructura de datos de la clase Producto, tal como se muestra.



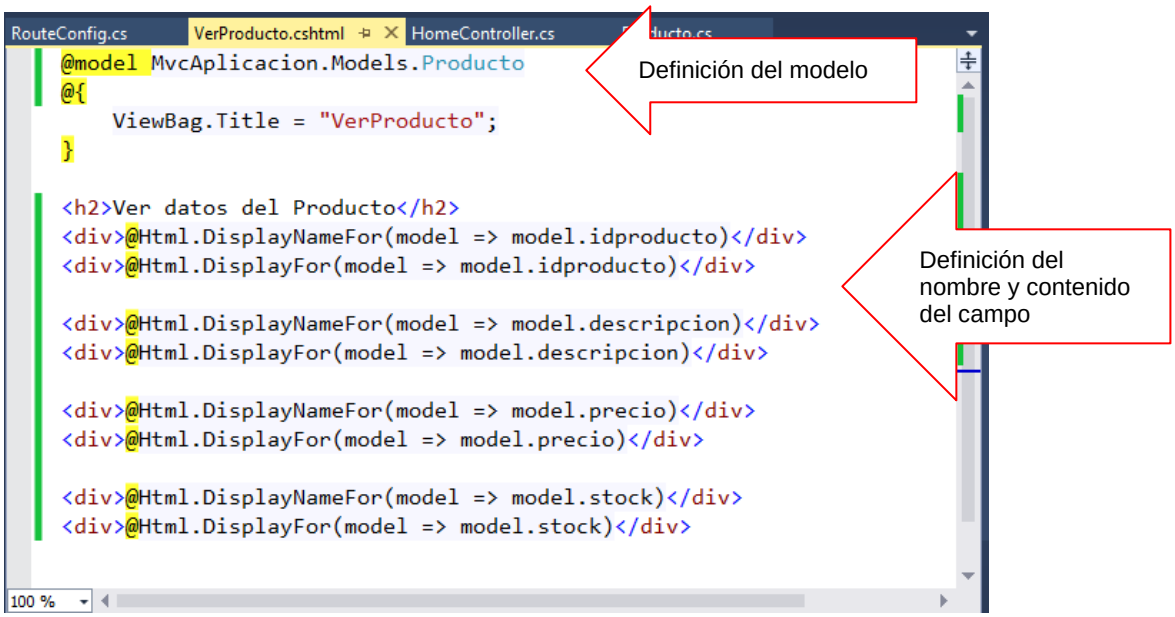
En el controlador Home, primero hacemos una referencia hacia la carpeta Models y definimos una acción llamada verProducto.



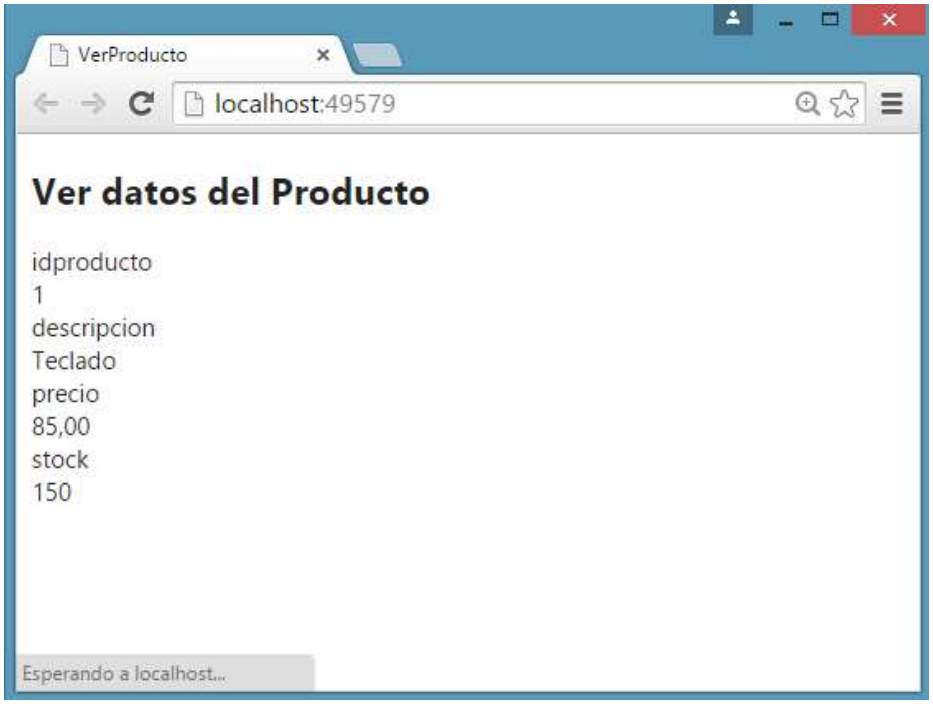
Hacer click derecho al ActionResult y seleccionar la opción Agregar Vista. Al visualizar la ventana aparece el nombre de la vista y su motor Razor, presiona el botón AGREGAR



En la vista, defina el modelo de datos @model. Definido el modelo definimos el nombre del campo **DisplayNameFor** y contenido del campo **DisplayFor** tal como se muestra



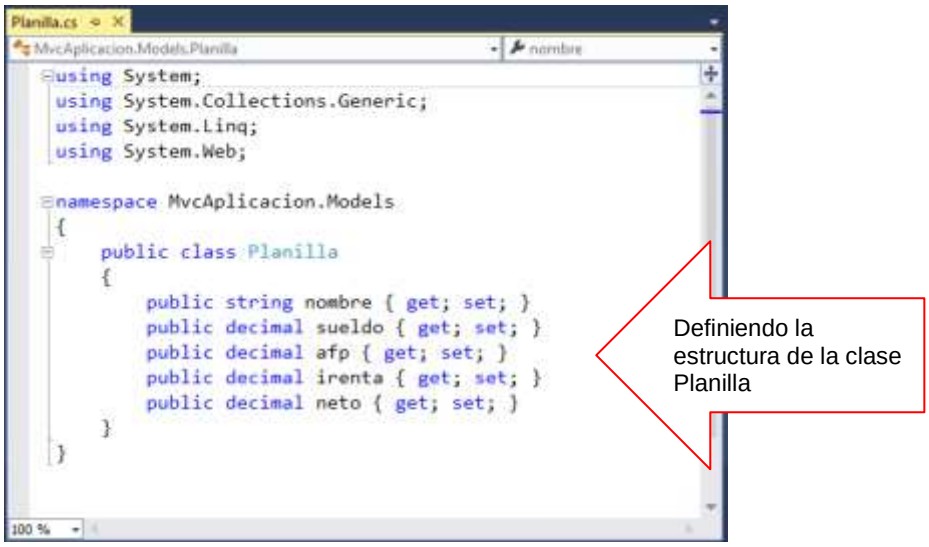
En en Route.config, cambiamos la acción por VerProducto y presiona F5 para visualizar la pagina



Laboratorio 11.3
Trabajando con Formularios

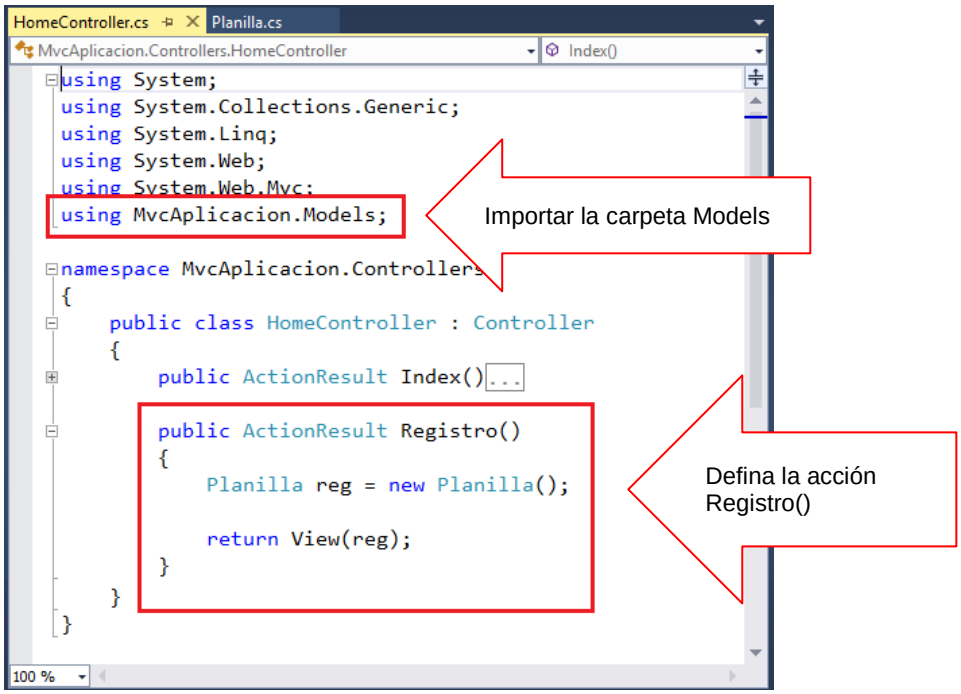
Definicion del modelo

En la carpeta Models agrega una clase llamada Planilla, defina la estructura de la clase.

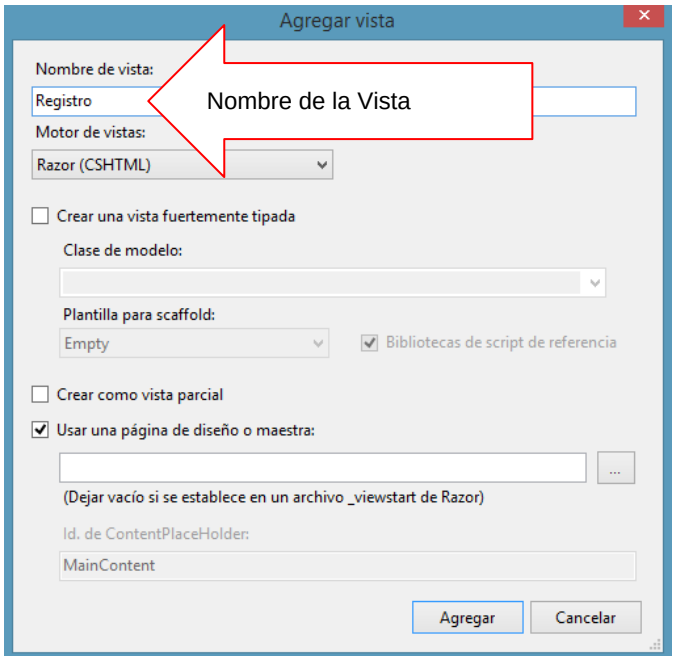


Trabajando con el Controlador

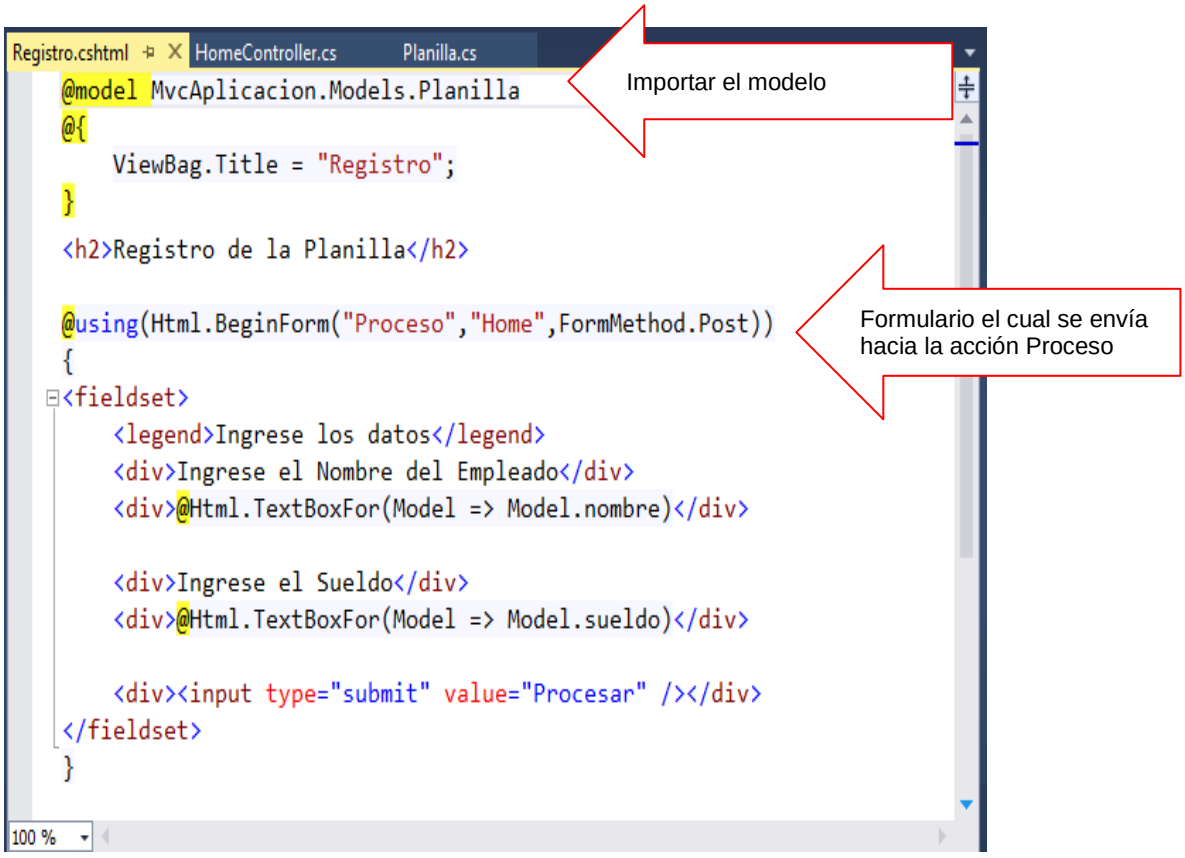
En el controlador Home, agregar la acción Registro, el cual envía la estructura de Planilla a la vista



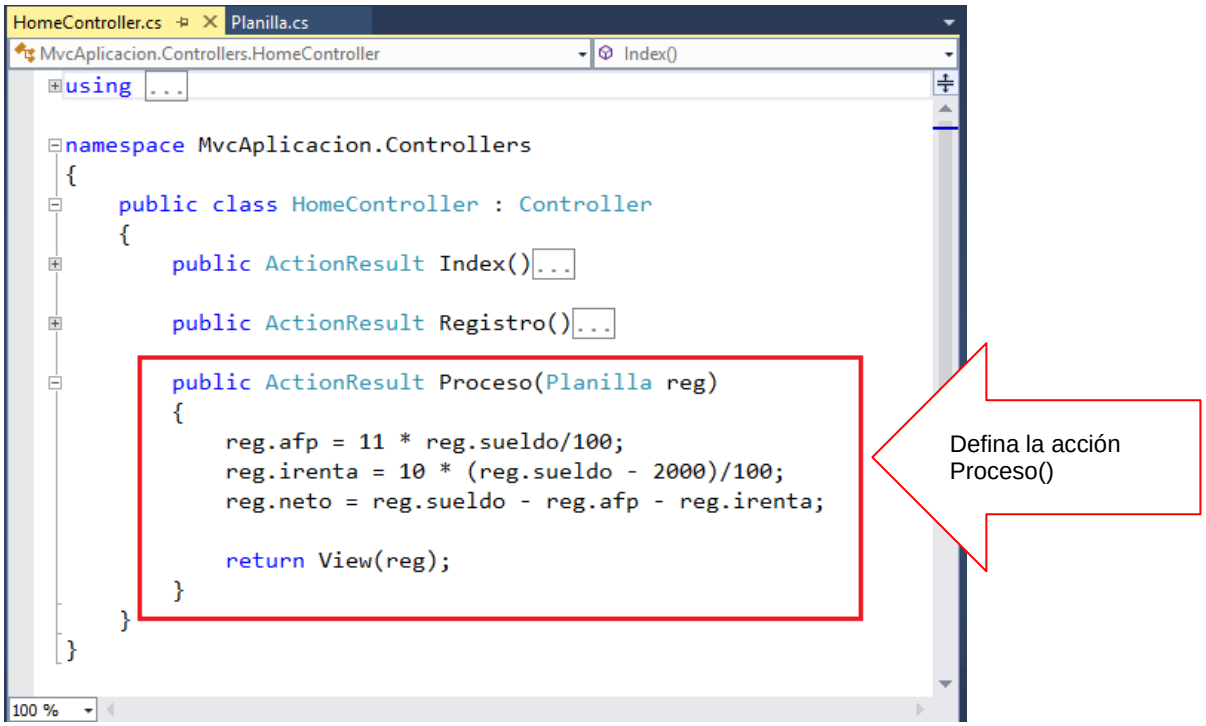
Agregar la vista a la acción Registro



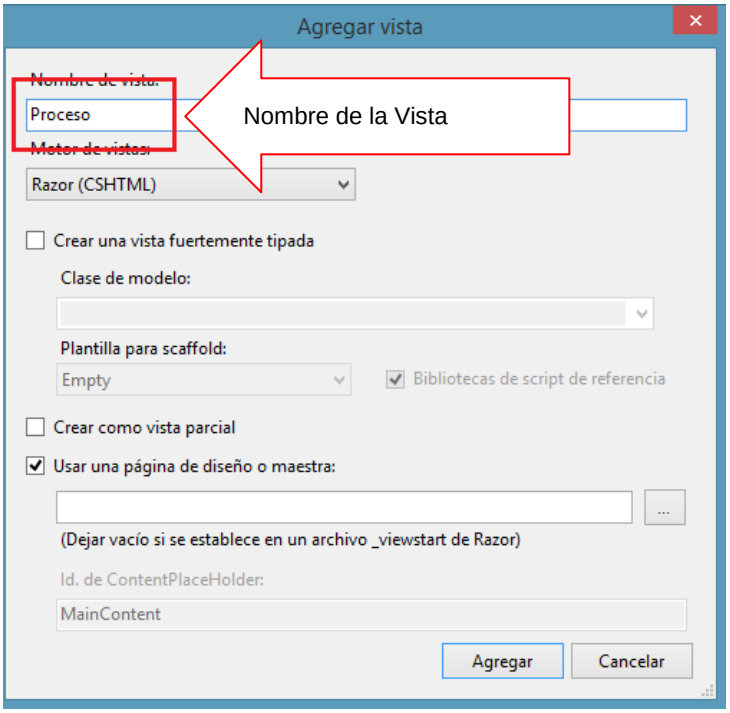
Defina la vista Registro, tal como se muestra



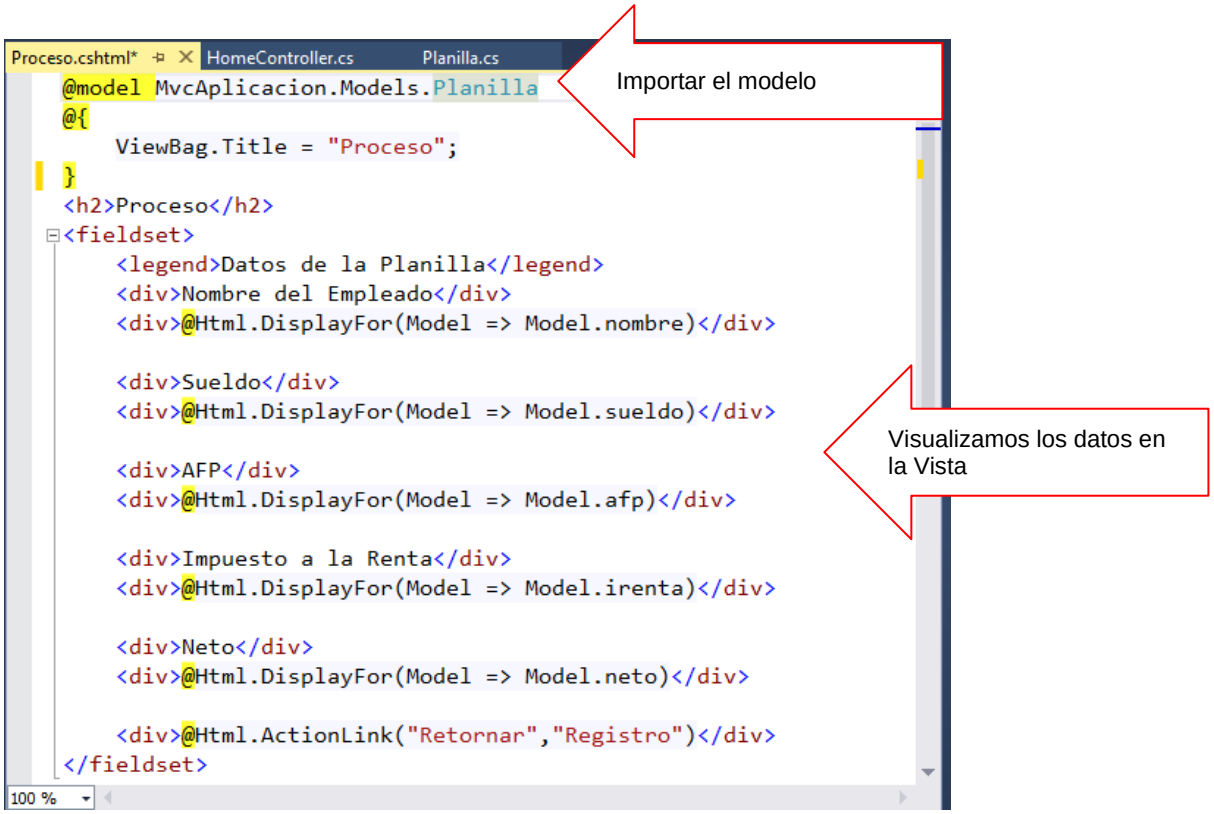
En el controlador Home, agregar la acción Proceso, el cual envía el objeto “reg” con los cálculos realizados



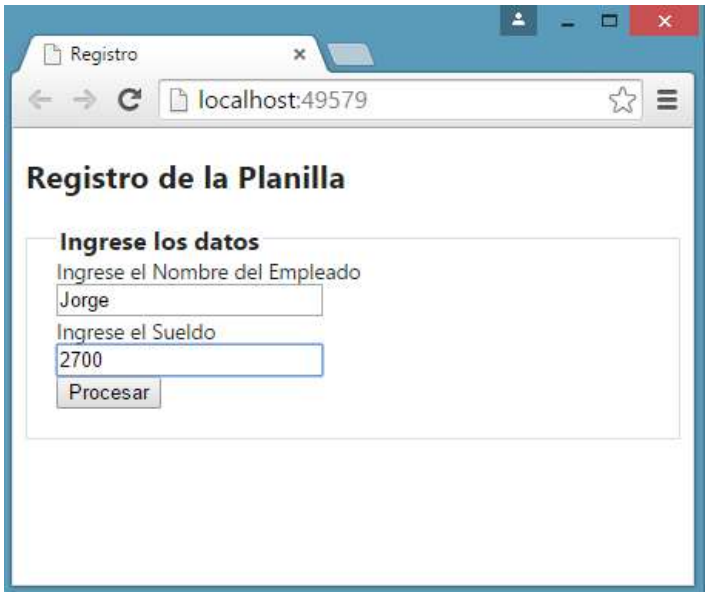
Agregar una vista a la acción Proceso()



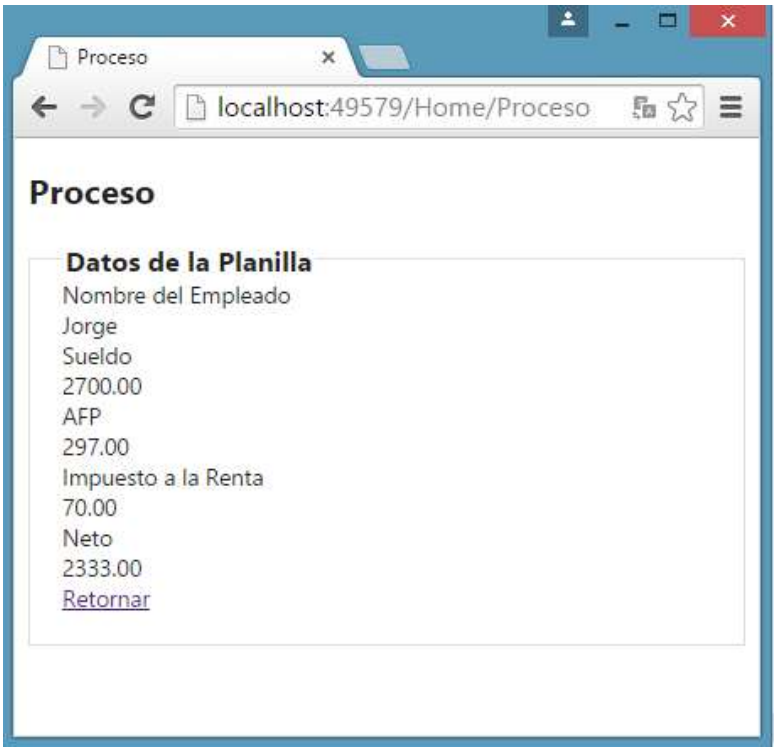
Defina la vista Proceso, donde visualiza los datos de Planilla.



Ejecuta el proyecto, ingresa los datos en el registro de Planilla, al presionar el botón Procesar…



Visualizamos los datos restantes de la Planilla, tal como se muestra



Resumen

- El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.
- Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.
- La palabra Scaffold está en inglés y en español significa "Andamio", pero en programación el scaffolding es un método para construir aplicaciones basadas en bases de datos, esta técnica está soportada por algunos frameworks del tipo MVC en el cuál el programador escribe una especificación que describe cómo debe ser usada la base de datos.
- ASP.NET Web Pages-Razor proporciona una sintaxis de programación simple para escribir código en páginas web donde el código basado en servidor se incrusta en el formato HTML de las páginas web. El código de Razor se ejecuta en el servidor antes de que la página se envíe al explorador.
- Como se comentó en el apartado anterior, el patrón MVC es implementado por muchas herramientas tecnológicas, Microsoft ha implementa el patrón MVC en su tecnología de ASP.NET, para el desarrollo de aplicaciones web. ASP.NET MVC es un poderoso framework para la construcción de sitios Web basándose en los estándares de internet actuales tales como HTML 5, jquery, CSS 3, etc.
- Este código de servidor puede crear dinámicamente contenido de cliente, es decir, puede generar formato HTML u otro contenido sobre la marcha y, a continuación, enviarlo al explorador junto con cualquier código HTML estático que contenga la página
- Finalmente Razor no es un nuevo lenguaje de programación, por el contrario se basa en sintaxis de C# y VB, teniendo como principal objetivo reutilizar el conocimiento de los programadores de .NET.
- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

http://www.asp.net/mvc/tutorials/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application

http://www.variablenotfound.com/2011/05/sintaxis-razor-con-vbnet.html

http://learn.geraldguido.com/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application/

http://msdn.microsoft.com/en-us/library/bb918115.aspx

UNIDAD

5

INTRODUCCION AL DESARROLLO WEB

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno realiza consultas y actualización de datos a través del lenguaje de consulta integrado utilizando la plataforma ORM en un entorno de una aplicación Windows y en una aplicación Modelo Vista Controlador (MVC).

Temario

Tema 12: Mantenimiento de datos con MVC (8 horas)

1. Mantenimientos de datos
1.1. Introducción a la clase Entity FrameWork
1.2. La clase DBContext
1.3. Anotaciones y Validaciones
1.4. Mantenimiento y consulta de un modelo de clases

ACTIVIDADES PROPUESTAS

- Los alumnos realiza operaciones de consulta y actualización de datos en un modelo de datos y MVC
- Los alumnos realizan operaciones de consulta utilizando el patrón MVC

12. MANTENIMIENTO DE DATOS CON MVC

12.1Introducción a la clase Entity Framework

ADO.NET Entity Framework admite aplicaciones y servicios centrados en datos, y proporciona una plataforma para la programación con datos que eleva el nivel de abstracción del nivel lógico relacional al nivel conceptual.

Los desarrolladores que utilizan datos en sus aplicaciones, tienen generalmente dos problemas distintos:

- Modelar las entidades, relaciones y lógica de la capa de negocio
- Trabajar con los motores de datos

Entity Framework permite trabajar con los datos en forma de objetos específicos del dominio (clientes, facturas, empleados, etc.) sin tener que preocuparse por las tablas o columnas en las que los datos están almacenados. Genera un nivel de abstracción más elevado a la hora de trabajar con datos, y nos permite tener aplicaciones con menos código.
Entity Framework es un ORM: herramientas que nos permiten almacenar objetos del dominio en una base de datos, sin tener que utilizar mucha programación.

Los ORM están compuestos de:
- Objetos de clases del dominio
- Objetos de la base de datos relacional
- Información sobre cómo se mapean los objetos del dominio a los objetos de la base de datos

Los ORM nos ayudan a mantener el diseño de la base de datos separado del diseño del dominio, lo cual hace que nuestras aplicaciones sean más simples de mantener y escalar. También automatizan las operaciones CRUD (Create, Read, Update, Delete - Crear, Leer, Actualizar, Borrar) para que no tengamos que escribirlas por nosotros.

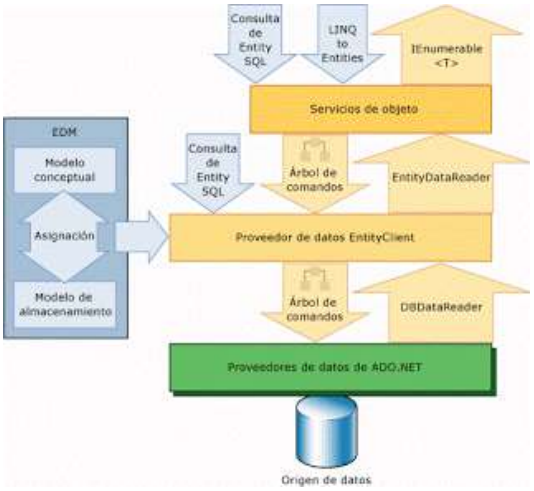


Figura: 1
Referencia: http://naizona.blogspot.com/2009/03/introduccion-adonet-entity-framework.html

El Entity Framework incluye el proveedor de datos de EntityClient. Este proveedor administra las conexiones, traduce las consultas de entidad en

consultas específicas del origen de datos y devuelve un lector de datos que Servicios de objeto usa para materializar los datos de la entidad en los objetos. Cuando no se requiere la materialización de los objetos, el proveedor de EntityClient también se puede utilizar como un proveedor de datos ADO.NET estándar habilitando las aplicaciones para ejecutar las consultas de Entity SQL y usar el lector de datos de solo lectura devuelto.

Los componentes principales son:

- Un modelo de Datos de Entidades (EDM)
- Un lenguaje que describe el esquema conceptual (CSDL).
- Un lenguaje de mapeado de esquema lógico a conceptual (MSL)
- Un lenguaje que describe el esquema lógico (SSDL).
- Un motor de mapeado completo que traduce del nivel conceptual al lógico (relacional).
- Un lenguaje de consultas denominado eSQL, similar a SQL, pero con soporte para objetos/entidades.
- Tres modos de acceso a datos a traves de la capa de entidades conceptuales.
- Un proveedor de acceso a datos denominado EntityClient.
- Un modelo de objetos denominado Object Services.
- Soporte para consultas vía LINQ (LINQ to Entity).

12.2 Clase DBContext

Una instancia de DbContext representa una combinación de los modelos de unidades de trabajo y repositorio, de modo que pueda emplearse para realizaciones operaciones de consulta una base de datos y agrupar los cambios que, seguidamente, se volverán a escribir en el almacenamiento como una unidad. DbContext es conceptualmente similar a ObjectContext.

DbContext se usa normalmente con un tipo derivado que contiene propiedades de DbSet<TEntity> para las entidades raíz del modelo. Estos conjuntos se inicializan automáticamente cuando se crea la instancia de la clase derivada. Este comportamiento se puede modificar si se aplica el atributo SuppressDbSetInitializationAttribute a toda la clase de contexto derivado o a propiedades individuales de la clase. Entity Data Model que respalda el contexto puede especificarse de varias maneras. Cuando se usa el enfoque Code First, las propiedades de DbSet<TEntity> en el contexto derivado se emplean para crear un modelo por convención.

El método protegido OnModelCreating se puede reemplazar para retocar este modelo. Se puede obtener más control sobre el modelo usado para el enfoque Model First creando explícitamente un DbCompiledModel a partir de un DbModelBuilder y pasando este modelo a uno de los constructores de DbContext. Cuando se usa el enfoque Database First o Model First, se puede crear el Entity Data Model con el Entity Designer (o manualmente creando un archivo EDMX) y, después, este modelo se puede especificar mediante una cadena de conexión de entidad o un objeto EntityConnection. La conexión a la base de datos (incluido el nombre de la base de datos) se puede especificar de varias maneras. Si se llama al constructor DbContext sin parámetros desde un contexto derivado, se usa el nombre del contexto derivado para buscar una cadena de conexión en el archivo app.config o web.config. Si no se encuentra

ninguna cadena de conexión, el nombre se pasa al DefaultConnectionFactory registrado en la clase Database.

Figura: 2
Referencia: http://www.entityframeworktutorial.net/EntityFramework4.3/dbcontext-vs-objectcontext.aspx

DbContext es responsable de las siguientes actividades:

- **EntitySet**: DbContext contiene conjunto de entidades (DbSet <TEntity>) para todas las entidades que se asignan a las tablas de base de datos.
- **Consulta**: DbContext convierte LINQ-to-Entidades consultas para consulta SQL y enviarlo a la base de datos.
- **Cambio de seguimiento**: Se realiza un seguimiento de los cambios que se produjeron en las entidades después de que ha sido la consulta de la base de datos.
- **La persistencia de datos**: Se realiza también la inserción, actualización y supresión de la base de datos, en base a lo que dice la entidad.
- **El almacenamiento en caché**: DbContext hace caché de primer nivel por defecto. Almacena las entidades que han sido recuperados durante el tiempo de vida de una clase de contexto.
- **Manejo de relaciones**: DbContext también gestiona relación utilizando CSDL, MSL y SSDL en DB-Primera o enfoque o el uso de la API de fluidez en Código-Primera aproximación Modelo-Primera.
- **Materialización de Objetos**: DbContext convierte los datos en bruto en la tabla de objetos de entidad.

12.3 Anotaciones y Validaciones

Code First de Entity Framework le permite usar sus propias clases de dominio para representar el modelo en el que se basa Entity Framework para realizar las consultas, el seguimiento de los cambios y las funciones de actualización.

Code First usa un modelo de programación conocido como convención sobre la configuración. Lo que significa esto es que Code First supondrá primero que las clases siguen las convenciones que EF usa. En ese caso, EF podrá obtener los detalles que necesita para realizar su trabajo. Sin embargo, si las clases no siguen dichas convenciones, tiene la capacidad de agregar configuraciones a las clases para proporcionar a EF la información que necesita.

Code First le ofrece dos maneras de agregar estas configuraciones a las clases. Una es usar atributos simples denominados DataAnnotations y la otra es emplear la API fluida de Code First, que le proporciona una manera de describir las configuraciones obligatorias, en el código.

Las validaciones y anotaciones se centrará en el uso de DataAnnotations (en el espacio de nombres System.ComponentModel.DataAnnotations) para configurar las clases, haciendo hincapié en las configuraciones que suelen requerirse.

Diversas aplicaciones .NET, como ASP.NET MVC, también entienden DataAnnotations y esto les permite usar las mismas anotaciones para las validaciones en el lado cliente.

Key: Entity Framework se basa en cada entidad que tiene un valor de clave que se usa para seguir las entidades. Una de las convenciones de las que depende Code First es el modo en que supone que la propiedad sea la clave de cada una de clases Code First. Esa convención implica buscar una propiedad denominada “Id” u otra que combine el nombre de clase y el “Id” (identificador), como “BlogId”. La propiedad se asignará a una columna de clave principal en la base de datos. Si Code First no encuentra una propiedad que coincida con esta convención, se producirá una excepción debido al requisito de Entity Framework respecto a que debe tener una propiedad de clave. Puede usar la anotación de clave para especificar qué propiedad debe usarse como EntityKey.

Required: La anotación Required indica a EF que una propiedad determinada es necesaria.

MaxLength y **MinLength**: Estas notaciones evalúan las longitudes de un elemento.

DataAnnotations no solo le permite describir la validación del lado cliente y servidor en las clases de Code First, sino que también le permite mejorar e incluso corregir las suposiciones que Code First realizará sobre las clases basándose en sus convenciones. Con DataAnnotations no solo puede controlar la generación del esquema de la base de datos sino que también puede asignar las clases de Code First a una base de datos existente.

Aunque son muy flexibles, tenga en cuenta que DataAnnotations solo proporciona los cambios de configuración más necesarios que puede realizar en las clases de Code First. Para configurar las clases para algunos de los casos extremos, debe buscar un mecanismo alternativo de configuración, la API fluida de Code First.

12.4 Mantenimiento y consulta de un modelo de clases

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

ADO.NET Entity Framework es un Framework ORM para la plataforma .NET.

Antes de trabajar en la operación CRUD (Create, Read, Update, Delete), es importante entender el ciclo de vida de entidad y cómo está siendo manejado por el EntityFramework.

Durante la vida de una entidad, cada entidad tiene un estado de entidad sobre la base de la operación realizada sobre el mismo a través del contexto (DbContext). El estado de entidad es una enumeración de tipo System.Data.Entity.EntityState que incluye los siguientes valores:

- Añadido
- Suprimido
- Modificado
- Sin alterar
- Separado

El contexto no sólo contiene la referencia a todos los objetos recuperados de la base de datos sino que también mantiene los estados de entidad y mantiene las modificaciones realizadas en las propiedades de la entidad. Esta función se conoce como el seguimiento de cambios.

El cambio de estado de la entidad Sin modificar el estado de modificación es el único estado que ha manejado automáticamente por el contexto. Todos los demás cambios deben hacerse de forma explícita el uso de métodos apropiados de DbContext y DbSet.

La siguiente figura muestra cómo la operación realizada en la entidad cambia sus "estados que, a su vez, afecta la operación de base de datos.

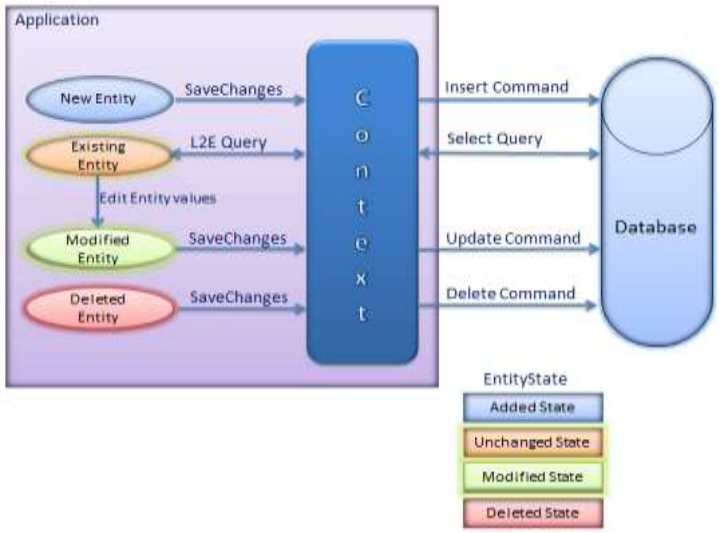


Figura 3
Referencia: http://www.entityframeworktutorial.net/entity-lifecycle.aspx

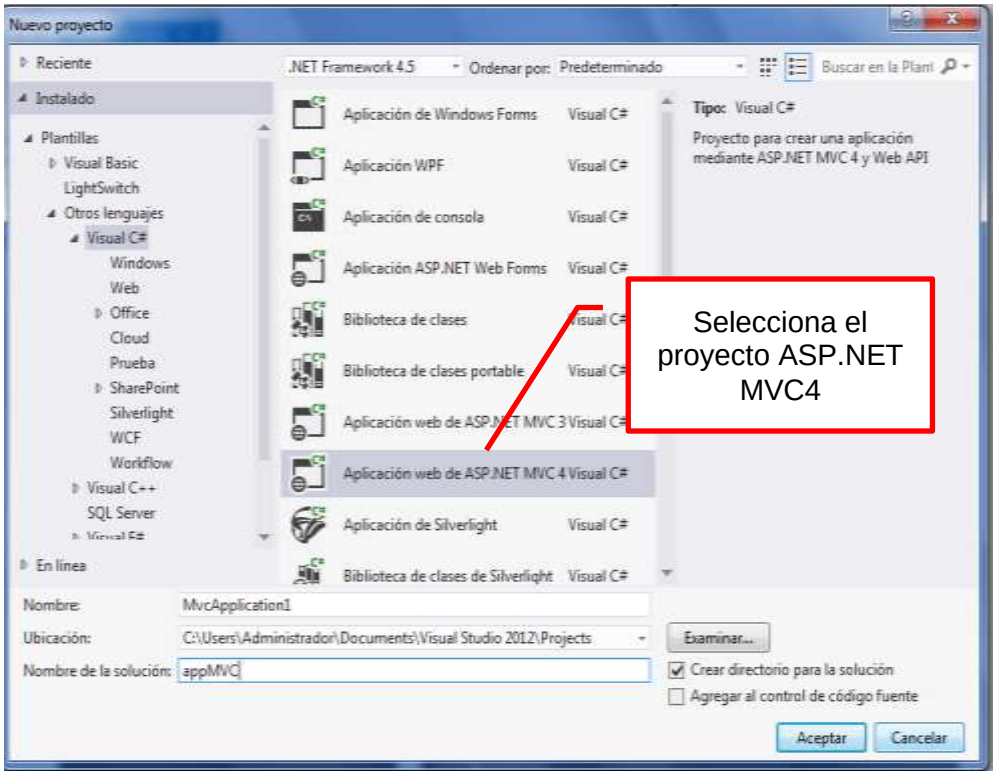
Como se puede ver en la figura anterior, la nueva entidad en el contexto ha añadido estado de entidad. Así que el contexto ejecutará comandos de inserción a la base de datos. De la misma forma, al recuperar una entidad existente mediante consultas L2e, tendrá estado sin cambios, esto se debe a que usted acaba de recuperar una entidad y no ha realizado ninguna operación en él todavía. Al modificar los valores de entidad existente, cambia su estado a Modificado que a su vez ejecutar comando de actualización en SaveChanges.

Entidad borrado de contexto se habrá eliminado el estado que a su vez ejecutar el comando delete para la base de datos.

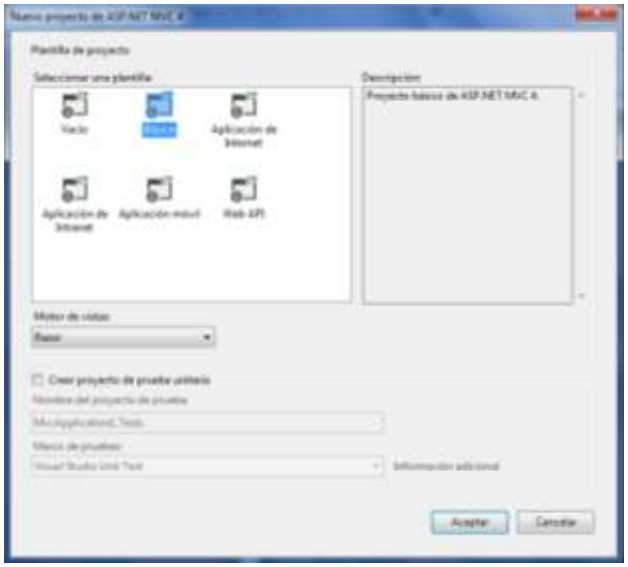
Así, de esta manera, las operaciones realizadas en entidades cambia de estado. Contexto construye y ejecuta los comandos de bases de datos basadas en el estado de una entidad.

Laboratorio 12.1
Trabajando con una aplicación MVC4

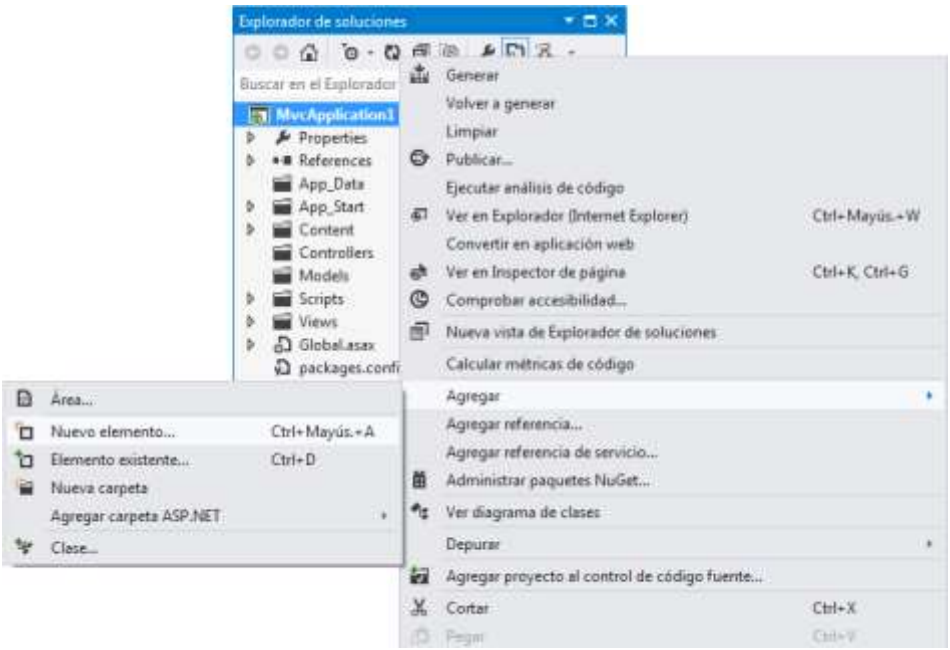
1. Selecciona el proyecto ASP.NET MVC 4



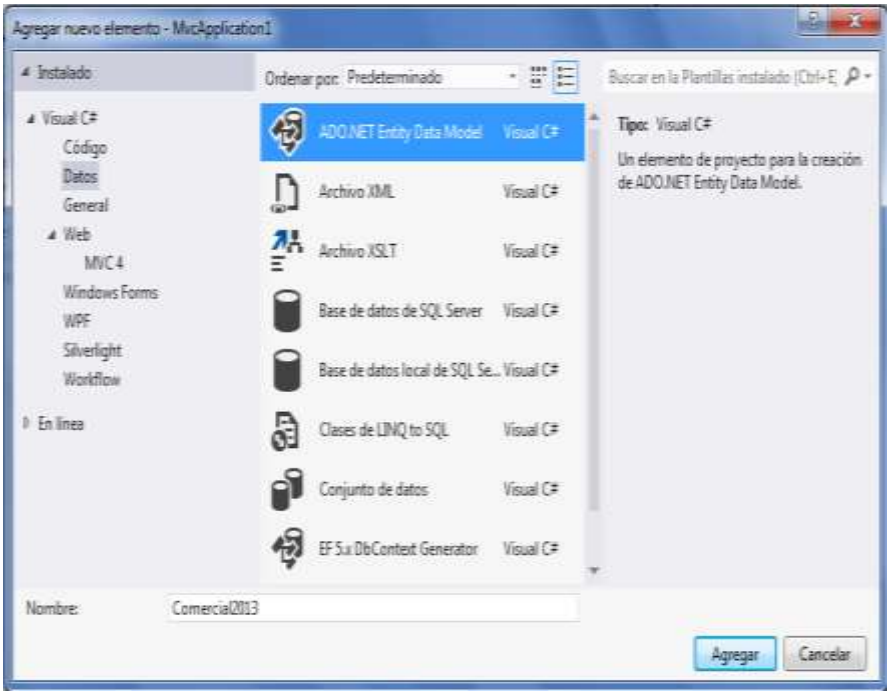
2. A continuación selecciona la plantilla de proyecto **Básico** y el Motor de Vista RAZOR tal como se muestra



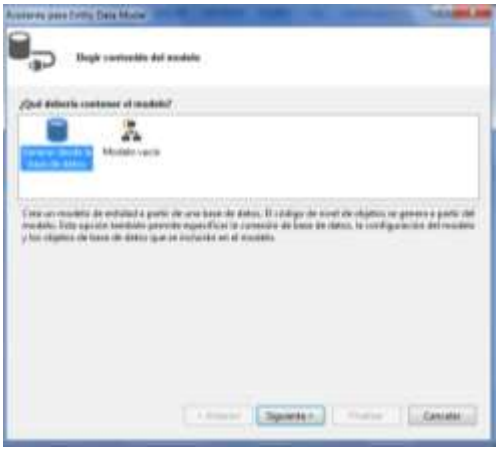
3. A continuación, vamos a agregar un Modelo de Datos Entity, para ello, desde tu aplicación selecciona la opción AGREGAR → Nuevo Elemento.



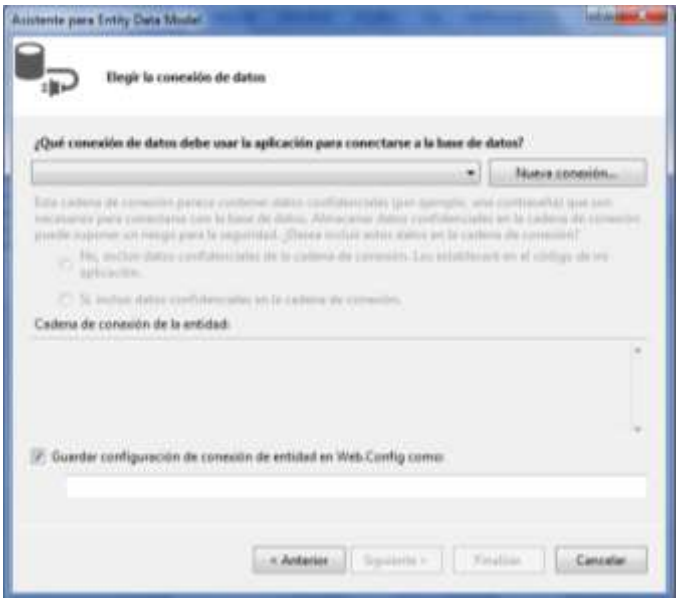
4. En la opción AGREGAR NUEVO ELEMENTO, selecciona la opción ADO.NET Entity Data Model, y agregar un nombre: Comercial2013, presiona el botón AGREGAR



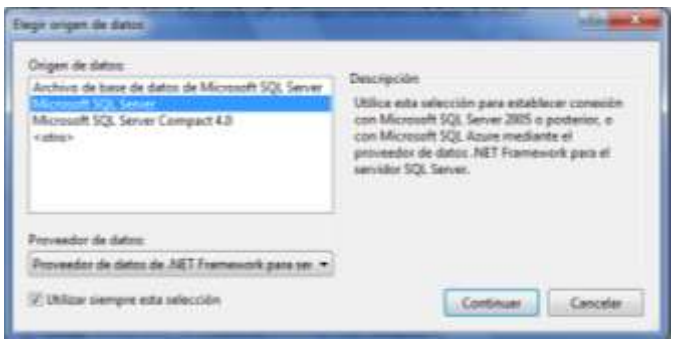
5. En la ventana asistente, elije la opción Generar desde la base de datos, presiona el botón Siguiente.



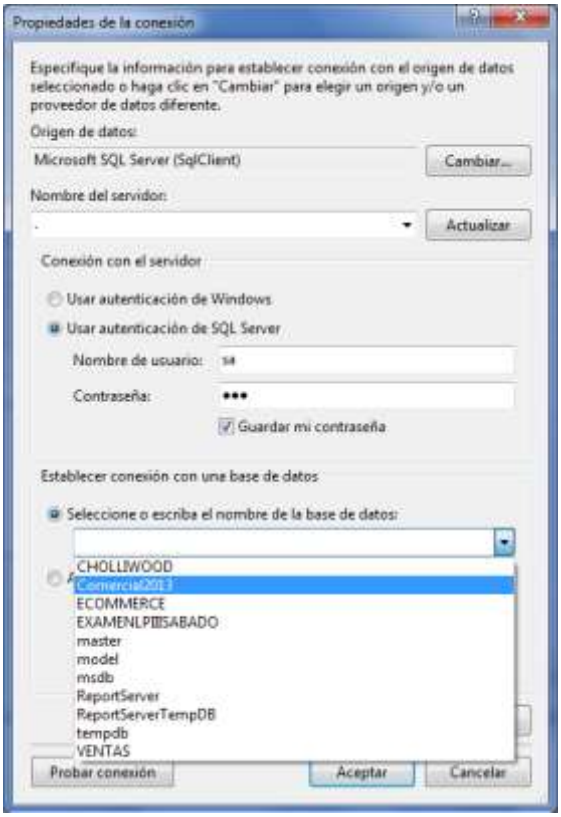
6. A continuación definir la conexión a la base de datos: Comercial2013, para ello debe ir a la opción **Nueva Conexión..**



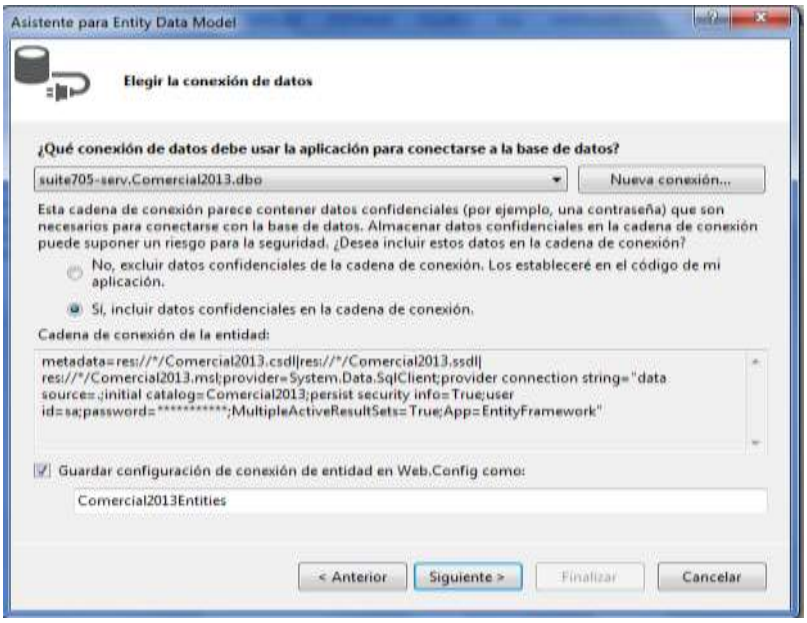
7. Elija el origen de datos: Microsoft SQL Server, tal como se muestra, presiona el botón CONTINUAR



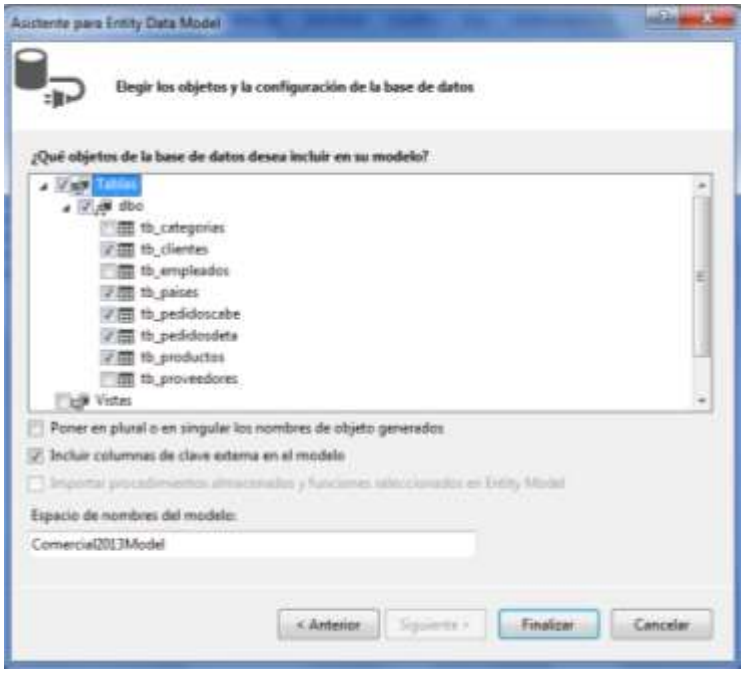
8. Defina las propiedades de conexión a la base de datos, donde selecciona el nombre del servidor: indicas punto si el servidor local; seleccionar la conexión con el servidor: Usar autenticación de SQL Server, ingrese el nombre del usuario y su clave. Si esta correcto selecciona la base de datos y presiona el botón ACEPTAR



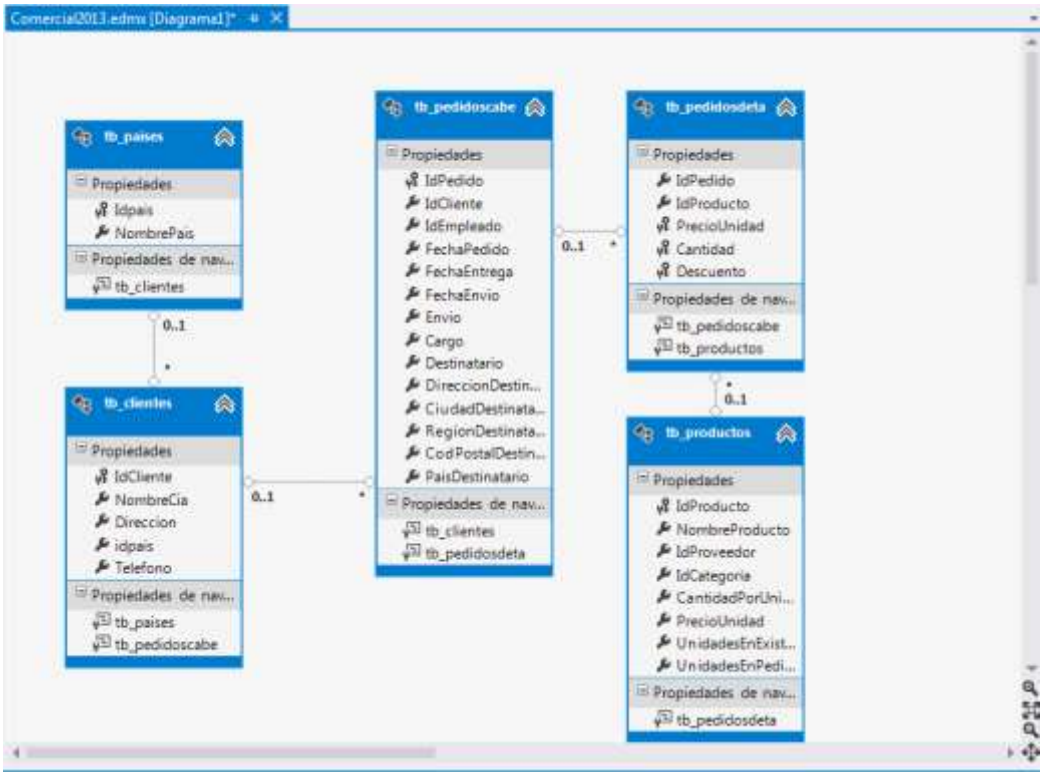
9. Creada la conexión, selecciona la opción Si, incluir datos confidenciales en la cadena de conexión. Presiona el botón Siguiente.



10. A continuación, selecciona las tablas del origen de datos, tal como se muestra y luego presiona el botón Finalizar



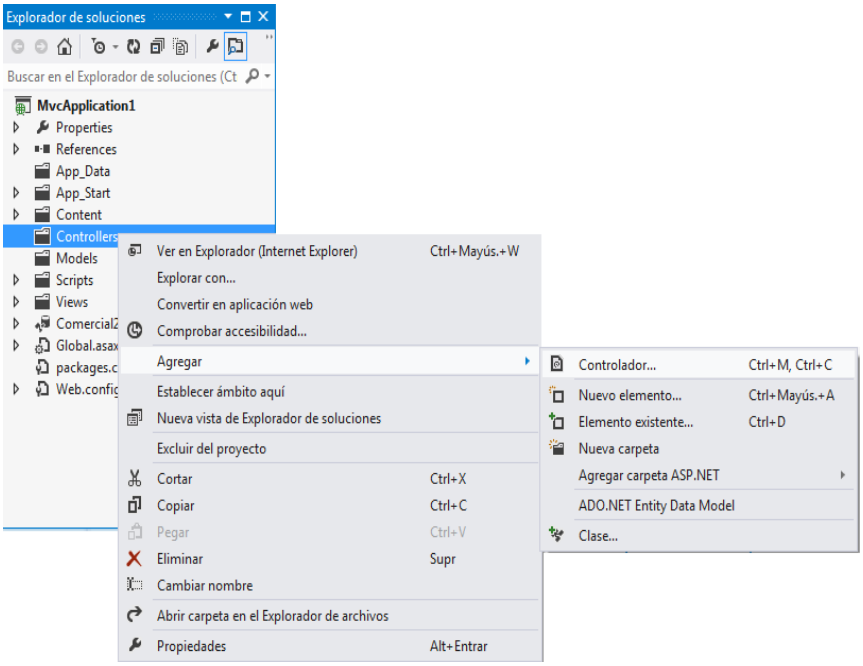
11. Al finalizar se muestra el Modelo de Contexto



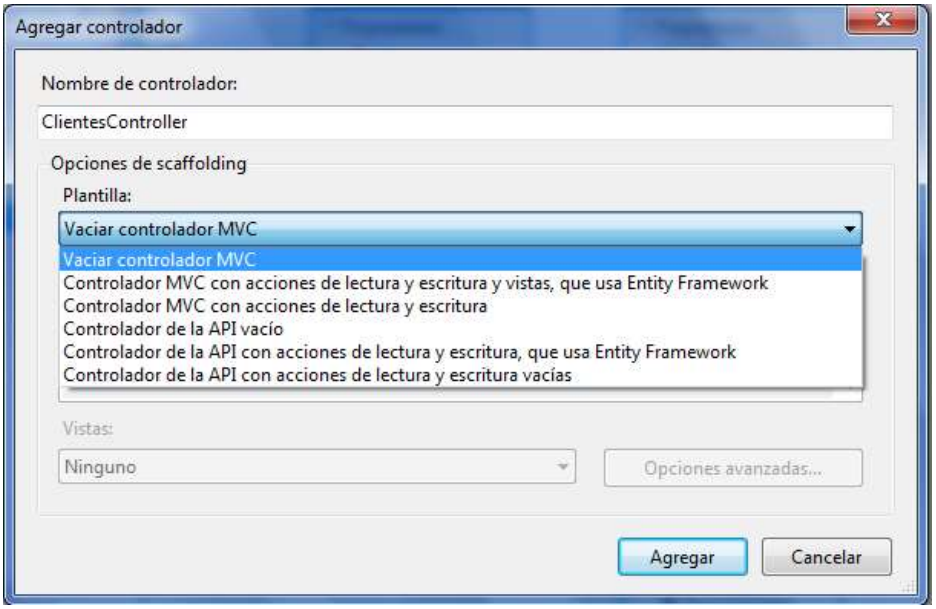
Laboratorio 12.2: CONSULTA Y LISTADOS

Implementa un Controlador que permita listar los datos de los Clientes

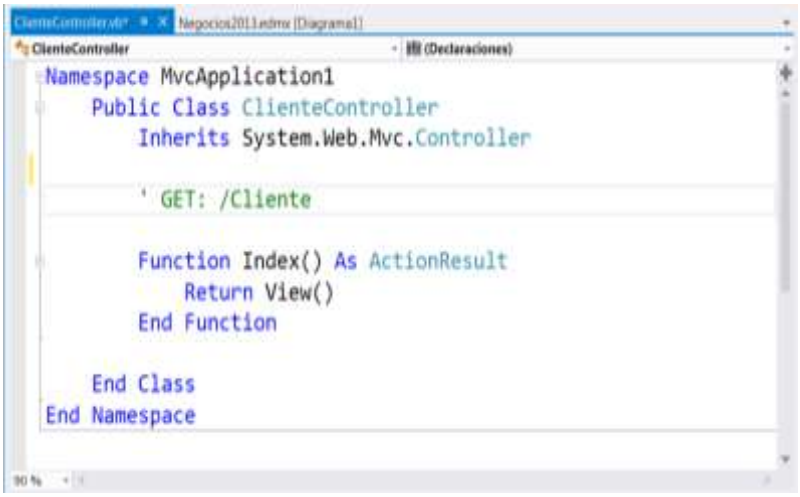
1. Agregar un Controlador al proyecto, desde la carpeta Controller AGREGAR → controlador tal como se muestra.



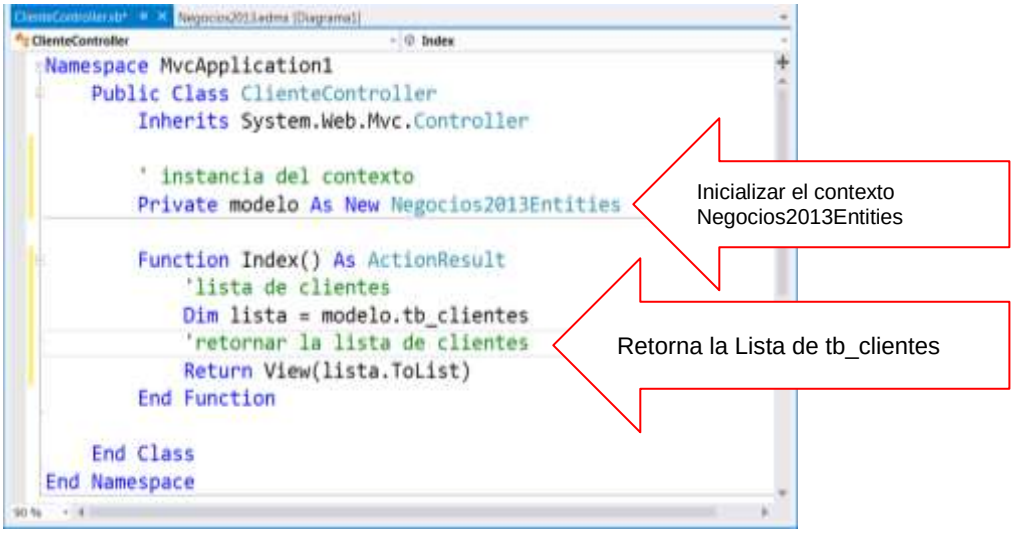
2. Defina el nombre: ClientesController y selecciona una plantilla: Vaciar controlador MVC, tal como se muestra, presiona el botón AGREGAR

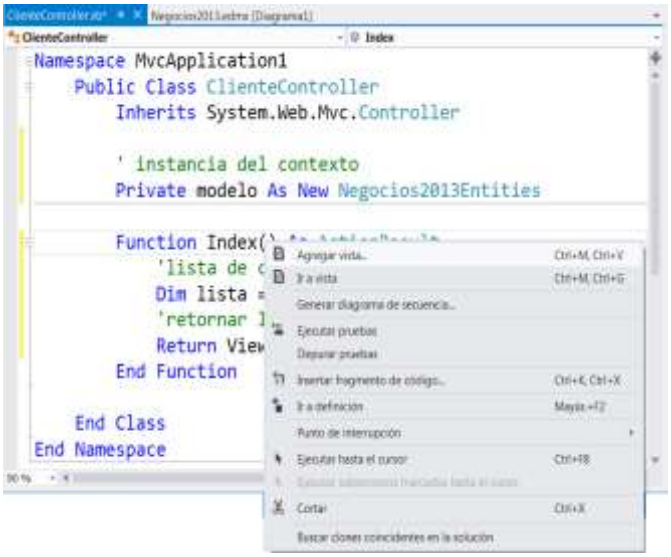


3. En el controlador Clientes, defina el proceso para el listado de los clientes, tal como se muestra:



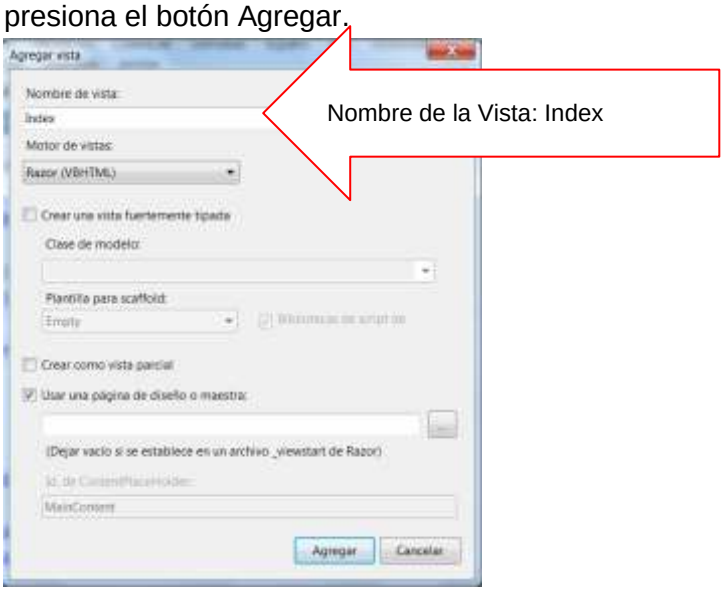
4. Programa la clase: defina la instancia Comercial2013Entities, en el método ActionResult retorna la lista de clientes.



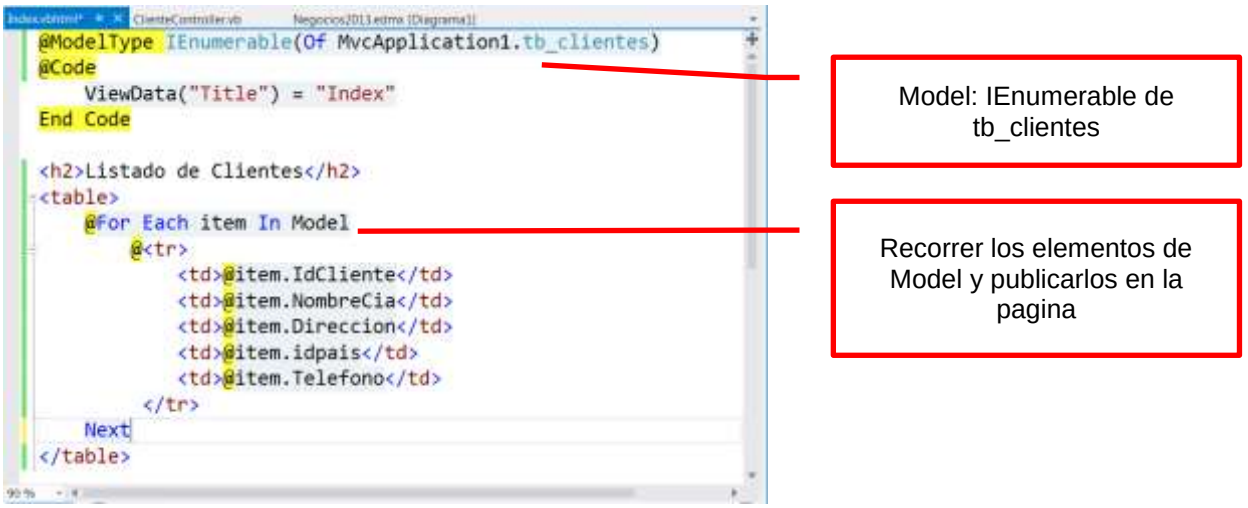


5. A continuación, agrega una Vista al Controlador, Click derecho a Index, selecciona la opción **Agregar vista**

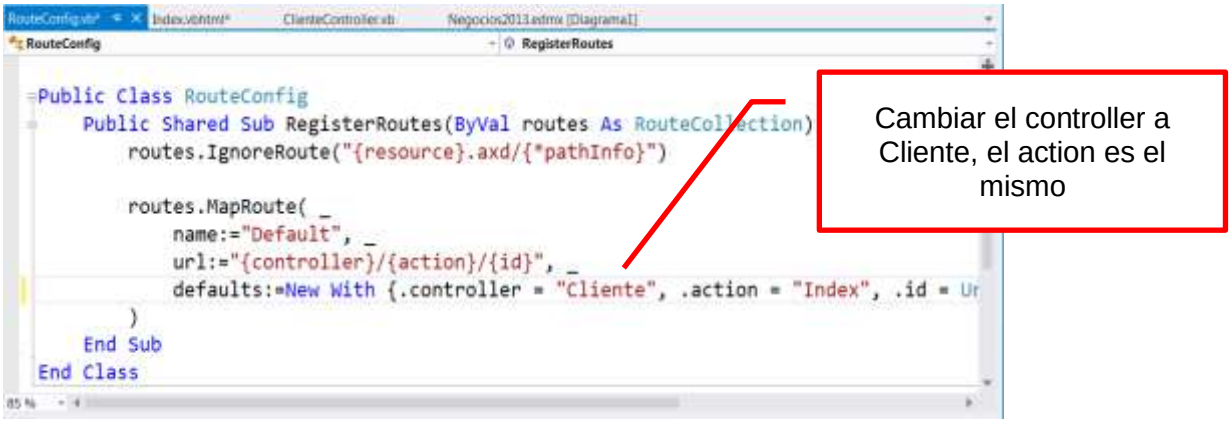
6. En la ventana Agregar Vista, se muestra el nombre de la Vista, no modificar, y presiona el botón Agregar.



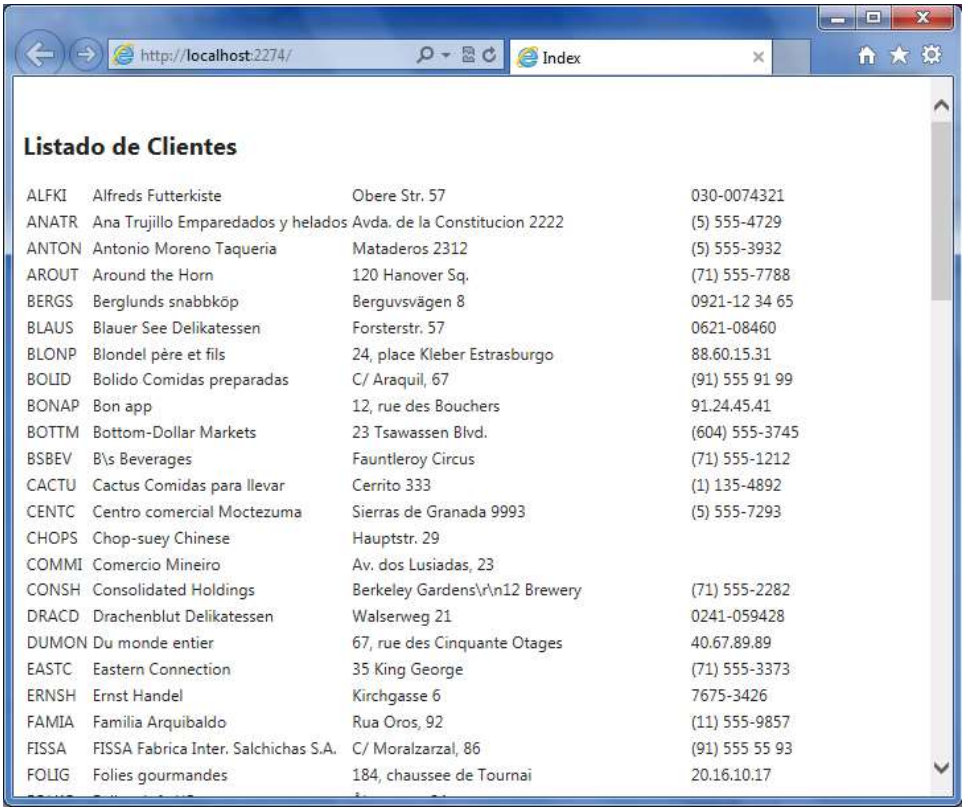
7. En la vista Index, codifique el código tal como se muestra



8. En el archivo RouteConfig, cambiar el controller a Clientes y ejecutar



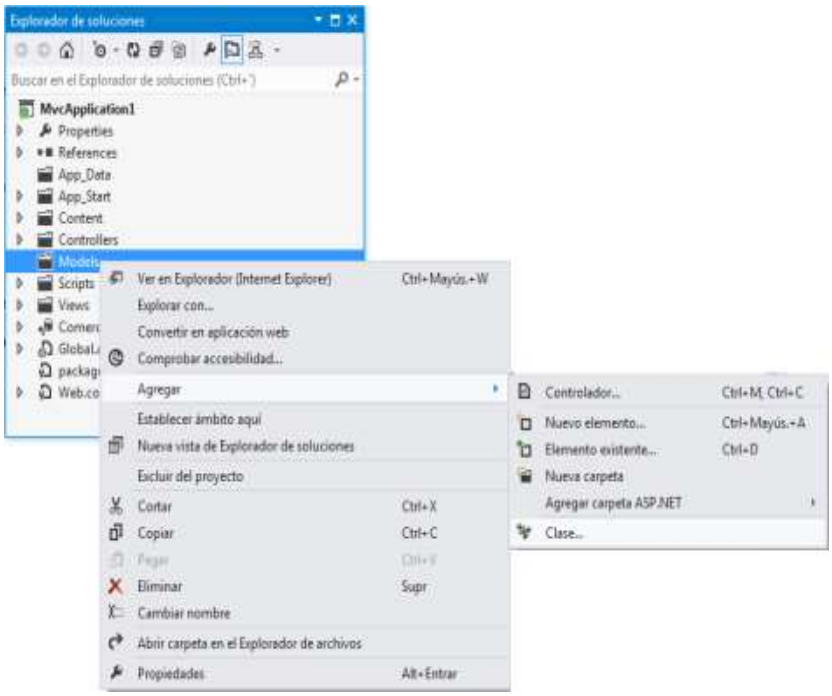
Presiona la tecla F5, para listar los registros de los clientes, tal como se muestra



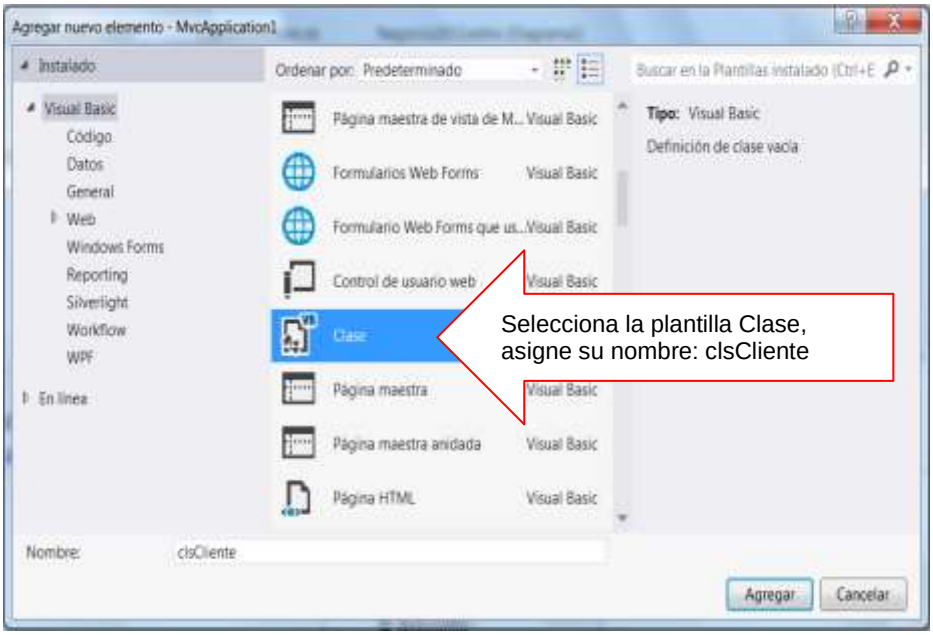
Laboratorio 12.3: CONSULTA Y LISTADOS

Implementa un Controlador que permita listar los datos de los Clientes, e incluya el nombre del país.

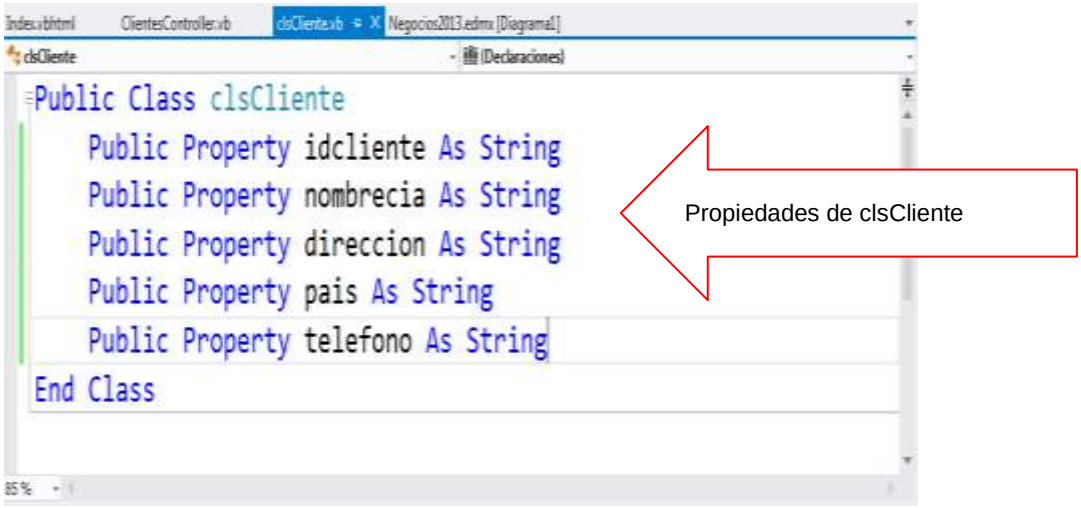
1. Agrega una clase al proyecto: desde la carpeta Models AGREGAR → clase…, tal como se muestra



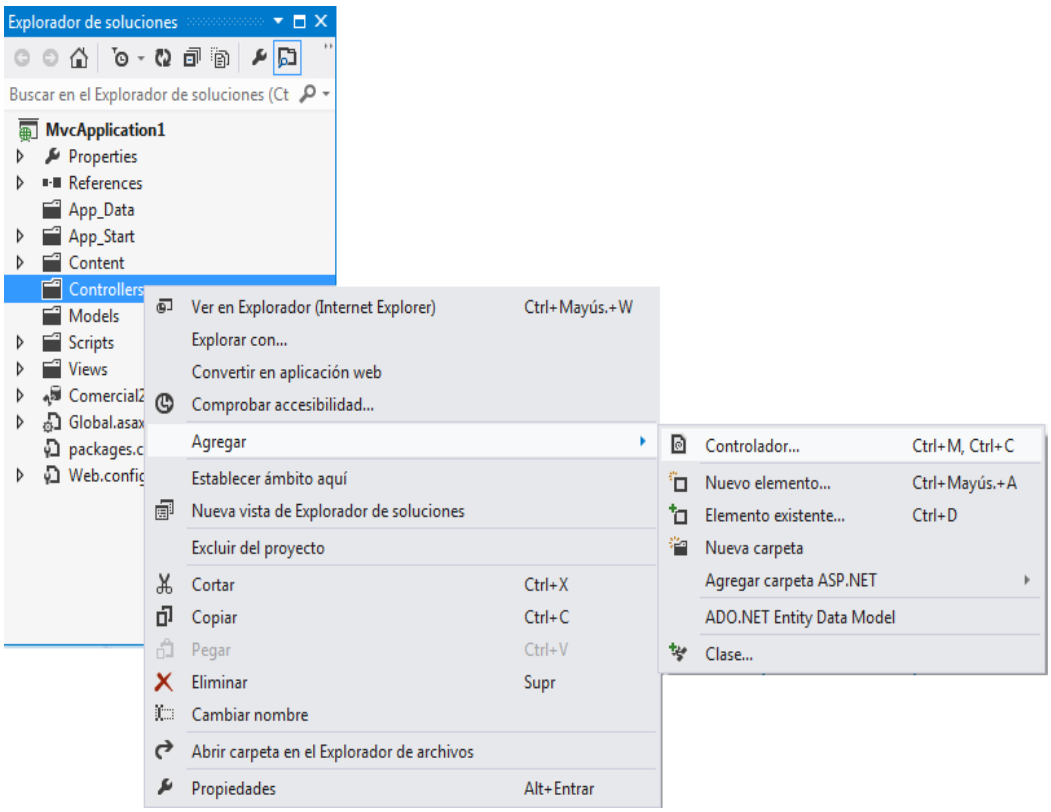
2. Defina el nombre de la clase: clsCliente, presiona el botón AGREGAR



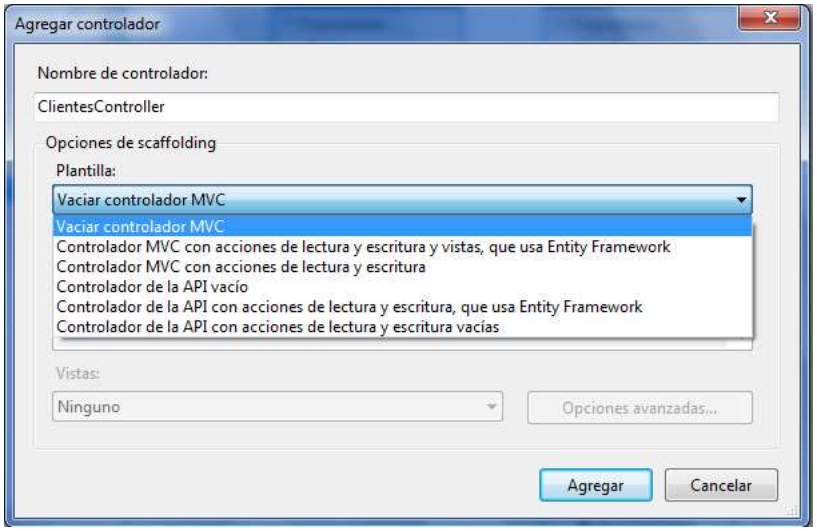
3. Defina las propiedades de la clase clsCliente, tal como se muestra



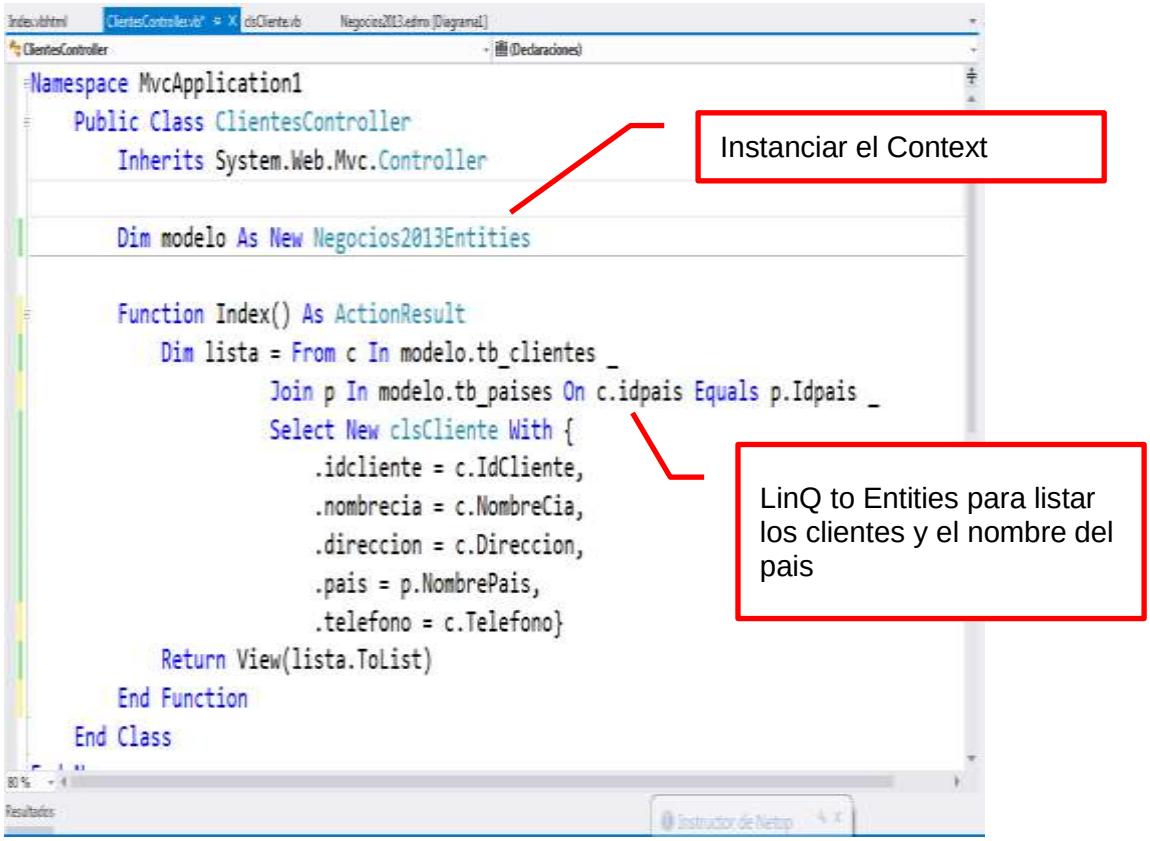
4. Agregar un Controlador al proyecto, desde la carpeta Controller AGREGAR → controlador tal como se muestra.



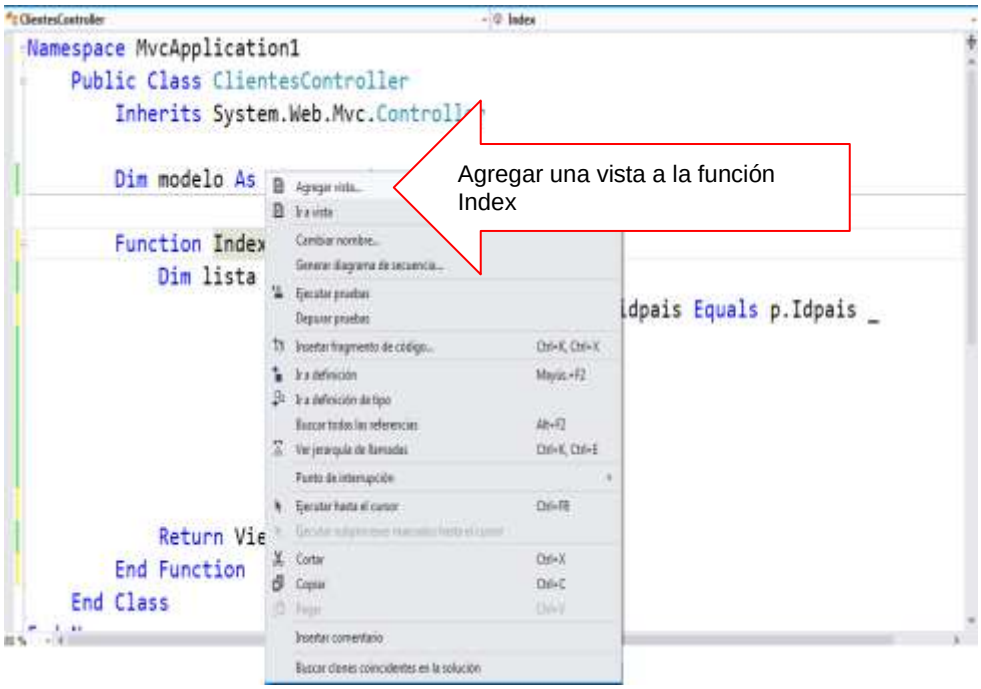
5. Defina el nombre: ClientesController y selecciona una plantilla: Vaciar controlador MVC, tal como se muestra, presiona el botón AGREGAR



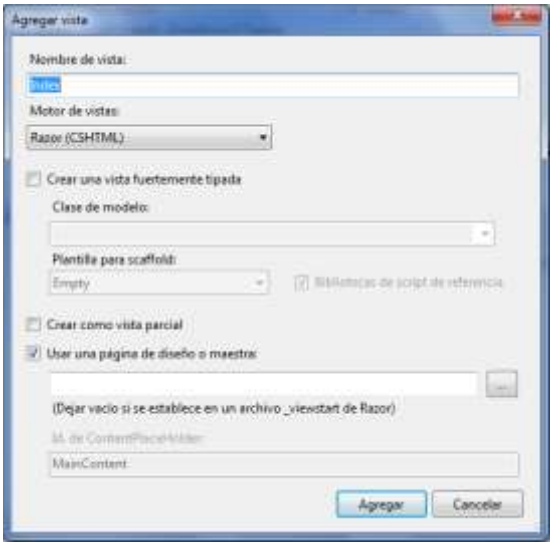
6. En el controlador Clientes, defina el proceso para el listado de los clientes, tal como se muestra:



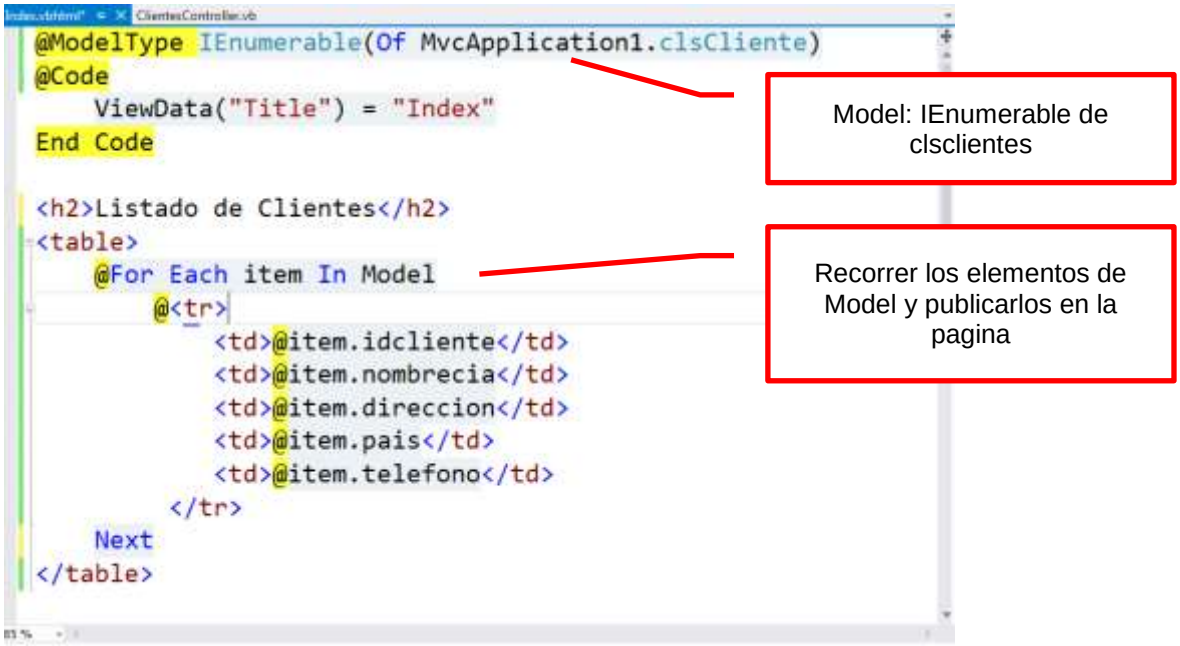
7. A continuación, agrega una Vista al Controlador, Click derecho a Index, selecciona la opción Agregar vista



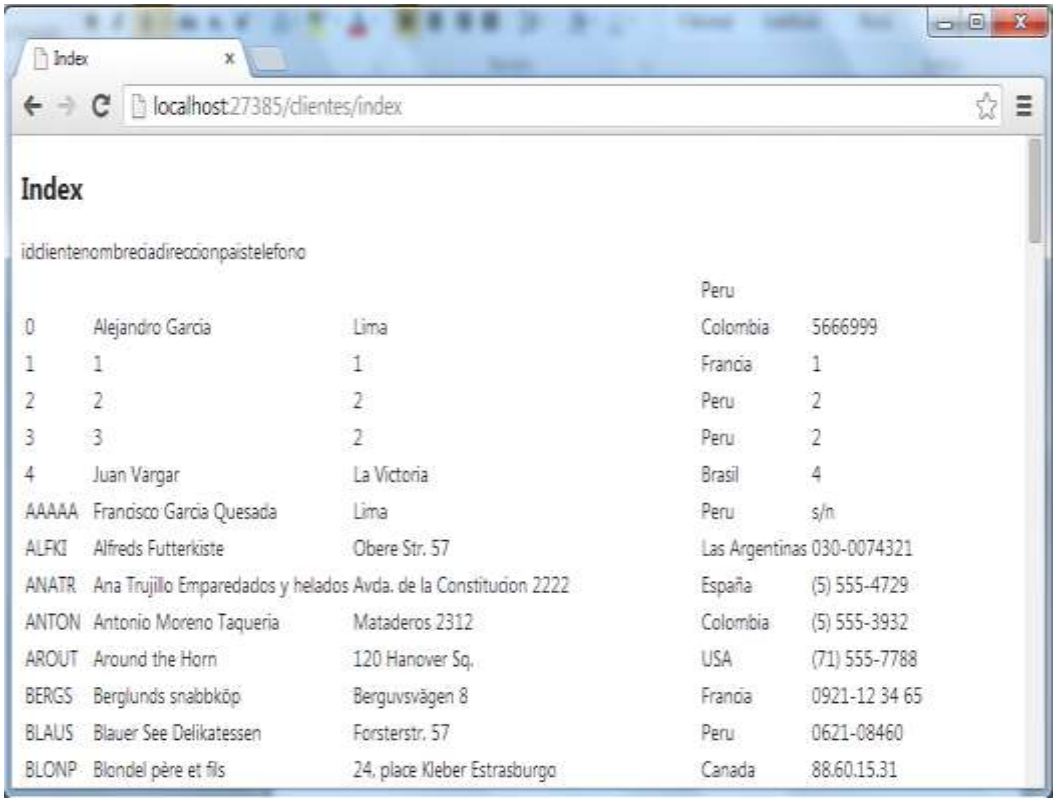
8. En la ventana Agregar Vista, se muestra el nombre de la Vista, no modificar, y presiona el botón Agregar.



9. En la vista Index, defina el código que permita listar los registros de clientes



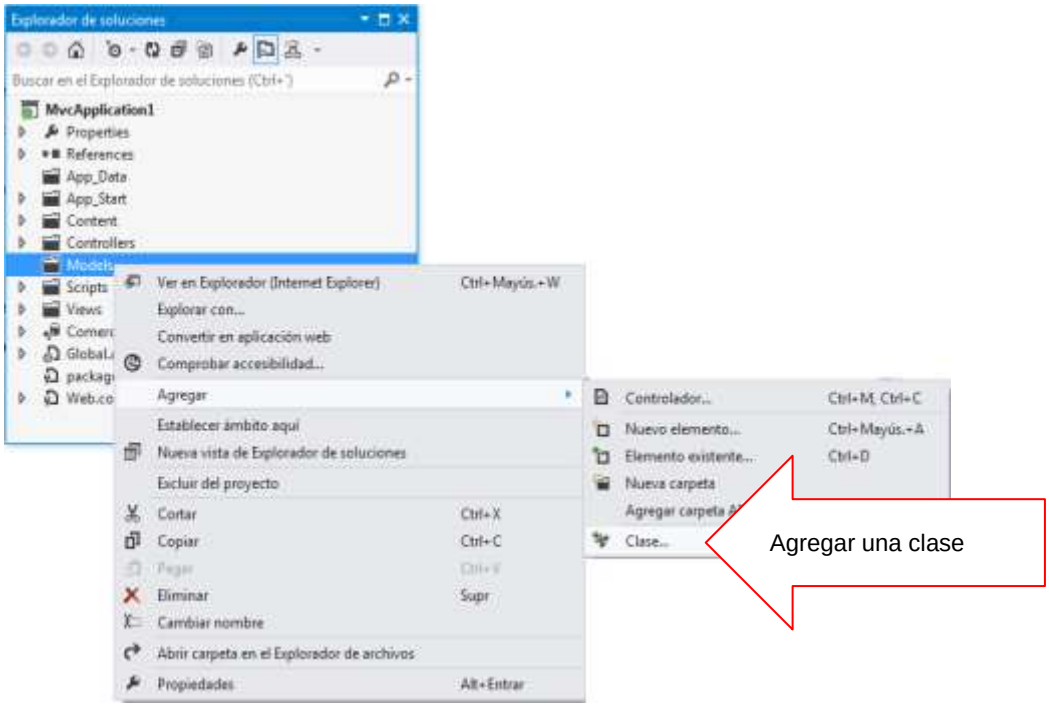
10. Presiona la tecla F5 para ejecuta la Vista



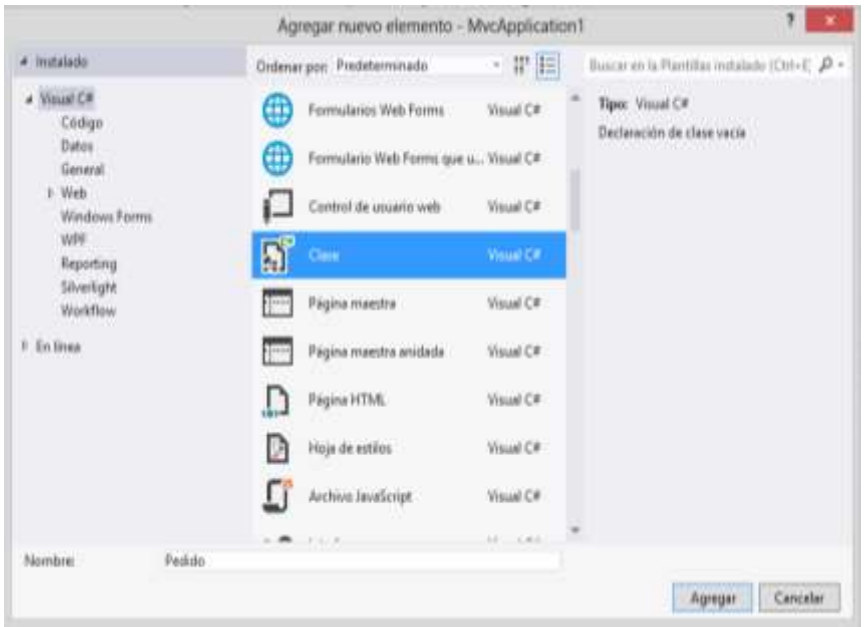
Laboratorio 12.4: FILTROS

Implementa un Modelo que permita listar los datos de los Pedidos, para ello definimos una clase.

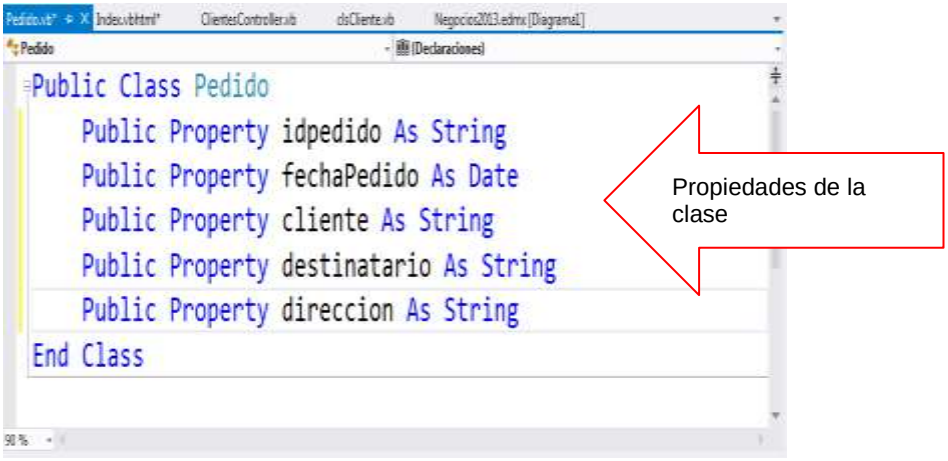
1. Agrega una clase al proyecto: desde la carpeta Models AGREGAR → clase…, tal como se muestra



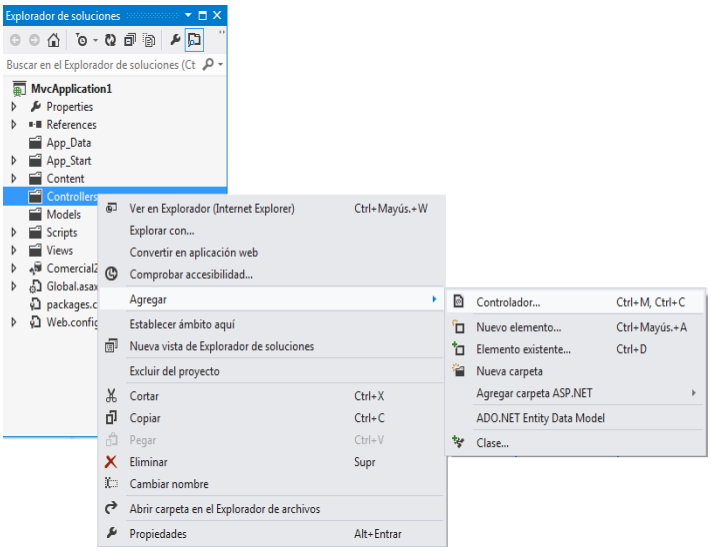
2. Defina el nombre de la clase: Pedido, presiona el botón AGREGAR



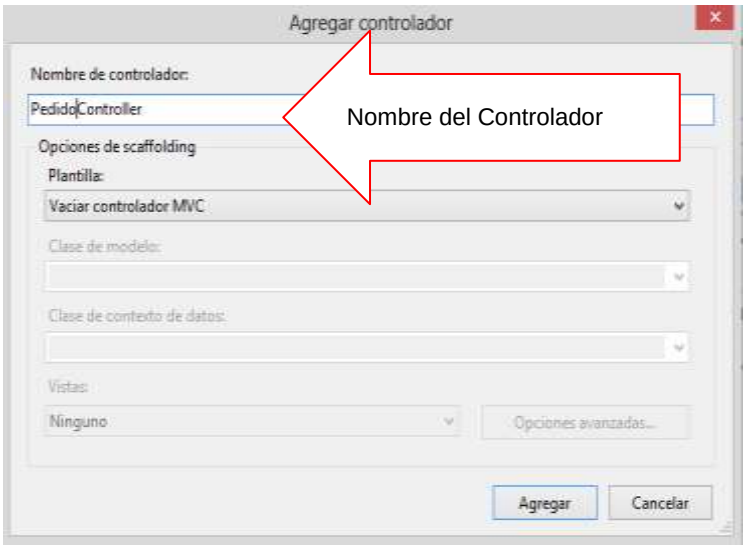
3. Defina las propiedades de la clase Pedido, tal como se muestra



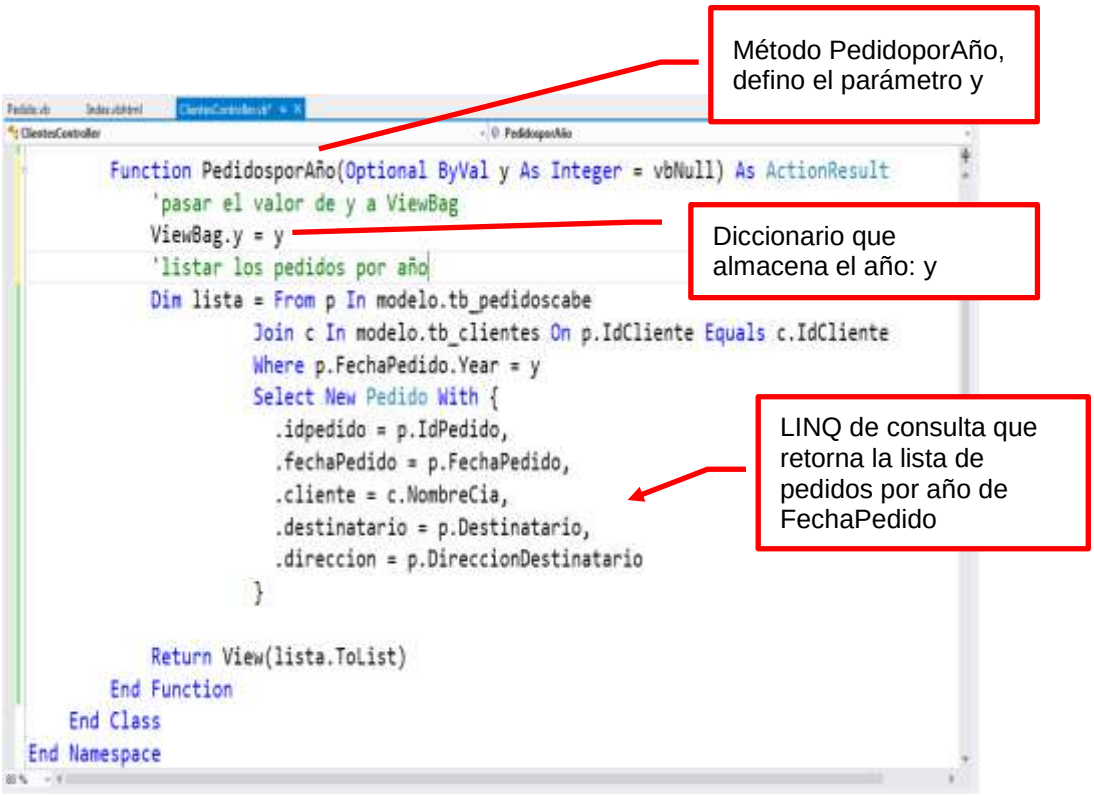
4. Agregar un Controlador al proyecto, desde la carpeta Controller AGREGAR → controlador tal como se muestra.



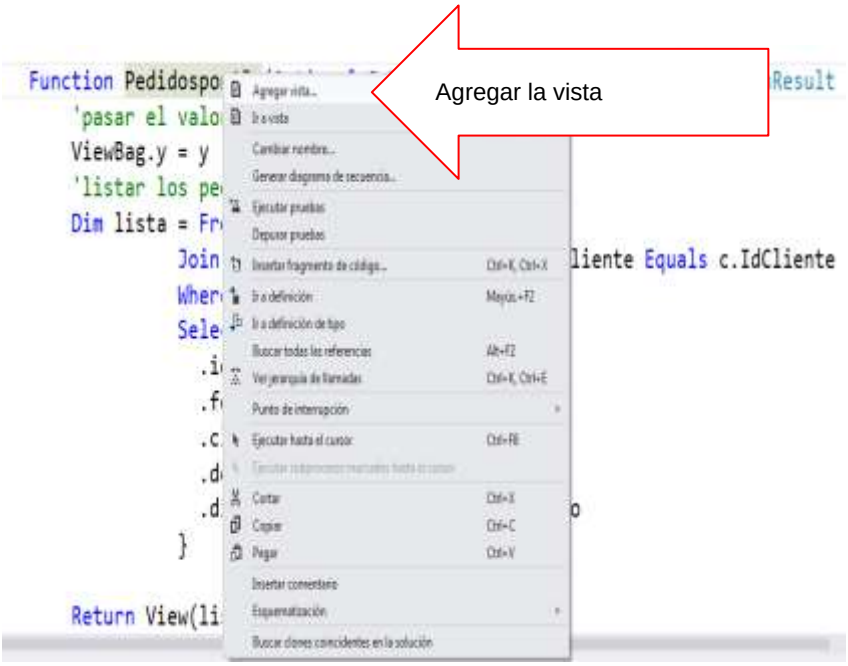
5. Defina el nombre: PedidoController y selecciona una plantilla: Vaciar controlador MVC, tal como se muestra, presiona el botón AGREGAR



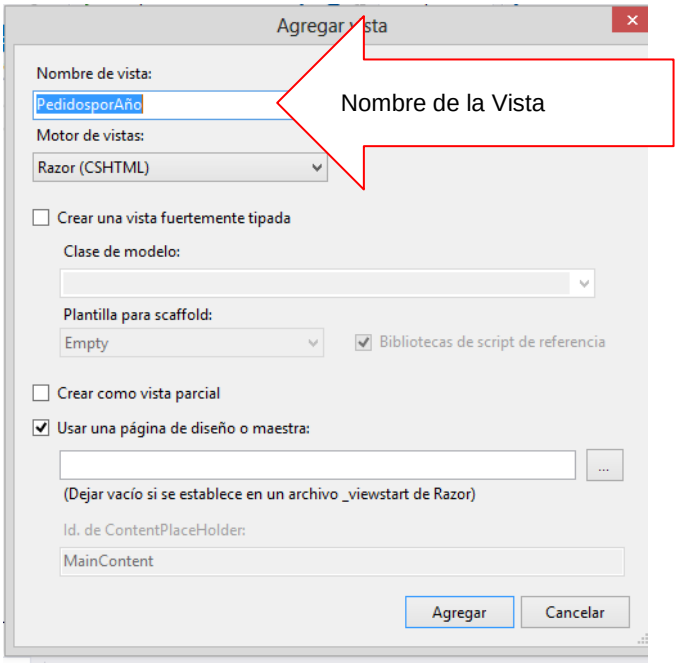
6. Defina el ActionResult para el listado de los Pedidos por un determinado Año de FechaPedido. Definición del proceso de buscar Pedidos por Año de FechaPedido.



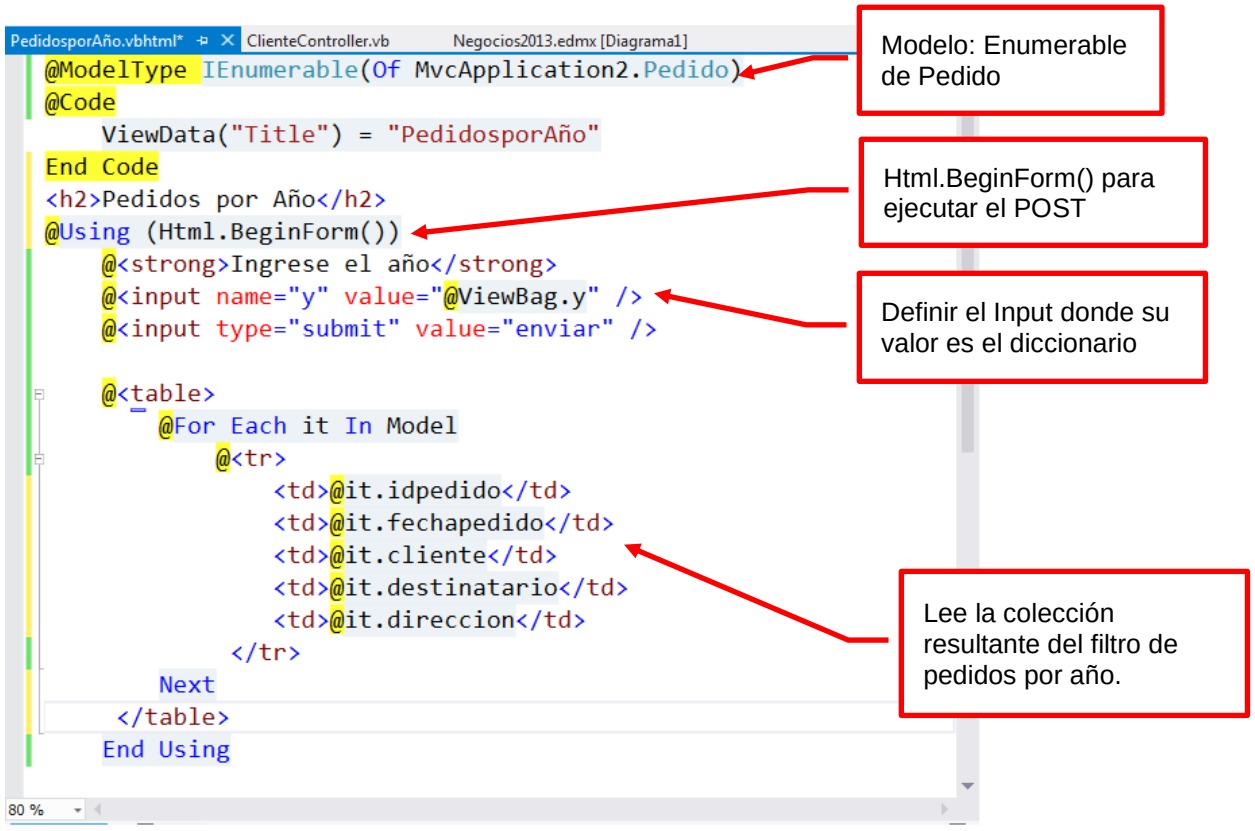
7. A continuación, agrega una Vista al Controlador, Click derecho a PedidoporAño, selecciona la opción Agregar vista



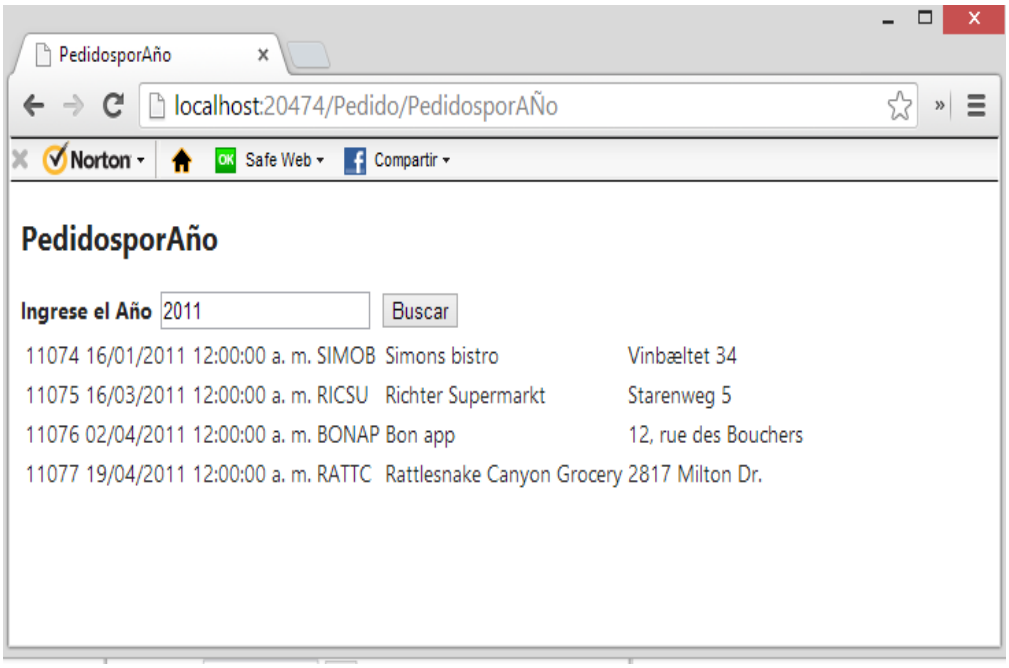
8. En la ventana Agregar Vista, se muestra el nombre de la Vista (PedidoporAño), no modificar, y presiona el botón Agregar.



9. Defina el código en la vista, tal como se muestra



Presiona la tecla Ctrl + F5 para ejecutar, ingrese el año, al presionar el botón Filtrar, se visualizan a los Pedidos por Año.



Resumen

- El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.
- Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.
- La palabra Scaffold está en inglés y en español significa "Andamio", pero en programación el scaffolding es un método para construir aplicaciones basadas en bases de datos, esta técnica está soportada por algunos frameworks del tipo MVC en el cuál el programador escribe una especificación que describe cómo debe ser usada la base de datos.
- ASP.NET Web Pages-Razor proporciona una sintaxis de programación simple para escribir código en páginas web donde el código basado en servidor se incrusta en el formato HTML de las páginas web. El código de Razor se ejecuta en el servidor antes de que la página se envíe al explorador.
- Como se comentó en el apartado anterior, el patrón MVC es implementado por muchas herramientas tecnológicas, Microsoft ha implementa el patrón MVC en su tecnología de ASP.NET, para el desarrollo de aplicaciones web. ASP.NET MVC es un poderoso framework para la construcción de sitios Web basándose en los estándares de internet actuales tales como HTML 5, jquery, CSS 3, etc.
- Este código de servidor puede crear dinámicamente contenido de cliente, es decir, puede generar formato HTML u otro contenido sobre la marcha y, a continuación, enviarlo al explorador junto con cualquier código HTML estático que contenga la página
- Finalmente Razor no es un nuevo lenguaje de programación, por el contrario se basa en sintaxis de C# y VB, teniendo como principal objetivo reutilizar el conocimiento de los programadores de .NET.
- Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

http://www.asp.net/mvc/tutorials/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application

http://www.variablenotfound.com/2011/05/sintaxis-razor-con-vbnet.html

http://learn.geraldguido.com/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application/

http://msdn.microsoft.com/en-us/library/bb918115.aspx

UNIDAD

A

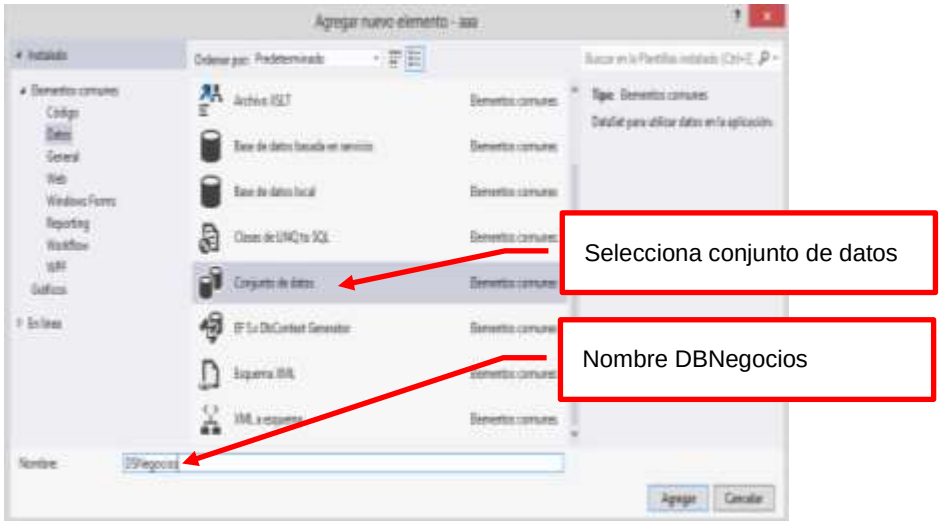
MANEJO DE CRYSTAL REPORT

Crystal Report, para Visual Studio .NET, es la herramienta de elaboración de informes estándar para este Framework. Permite crear contenido interactivo con calidad de presentación en la plataforma .NET, lo que ha supuesto una ventaja fundamental para Crystal Report durante años.

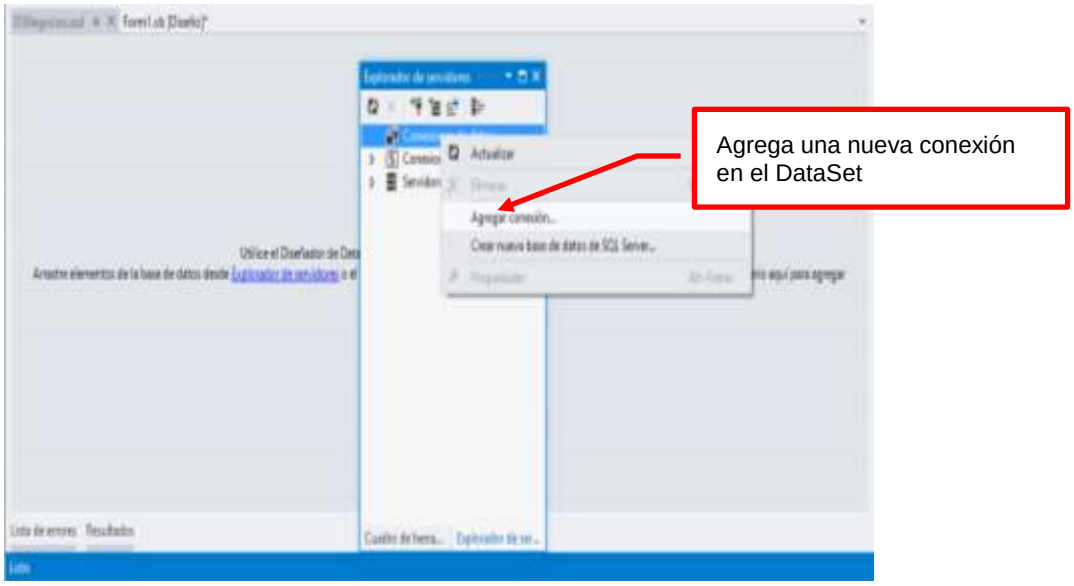
Desarrollo Práctico

Implemente un Reporte que permita listar los pedidos registrados en la base de datos Negocios2013

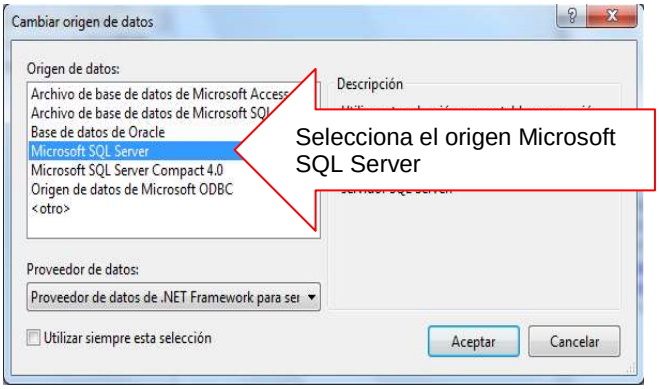
11.5.1 Agregue un DataSet al proyecto. Selecciona el elemento Conjunto de Datos, asigne el nombre DsNegocios, tal como se muestra

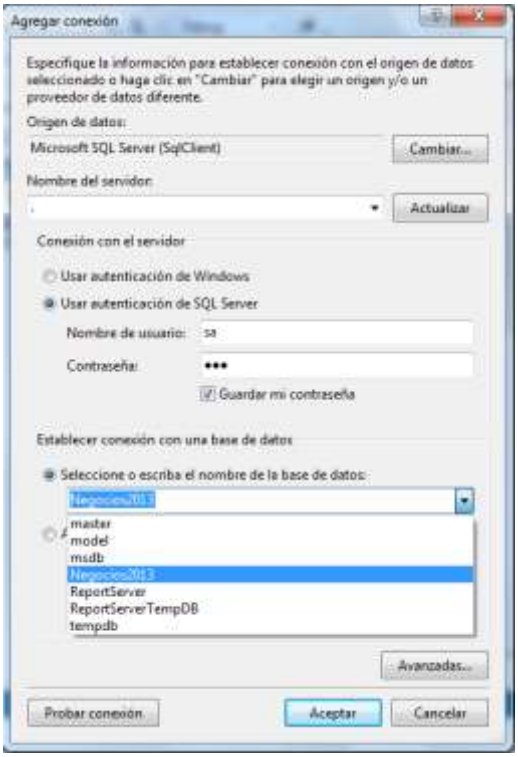


11.5.2 En el DataSet, en el Explorador de servidores, agregue una nueva conexión.



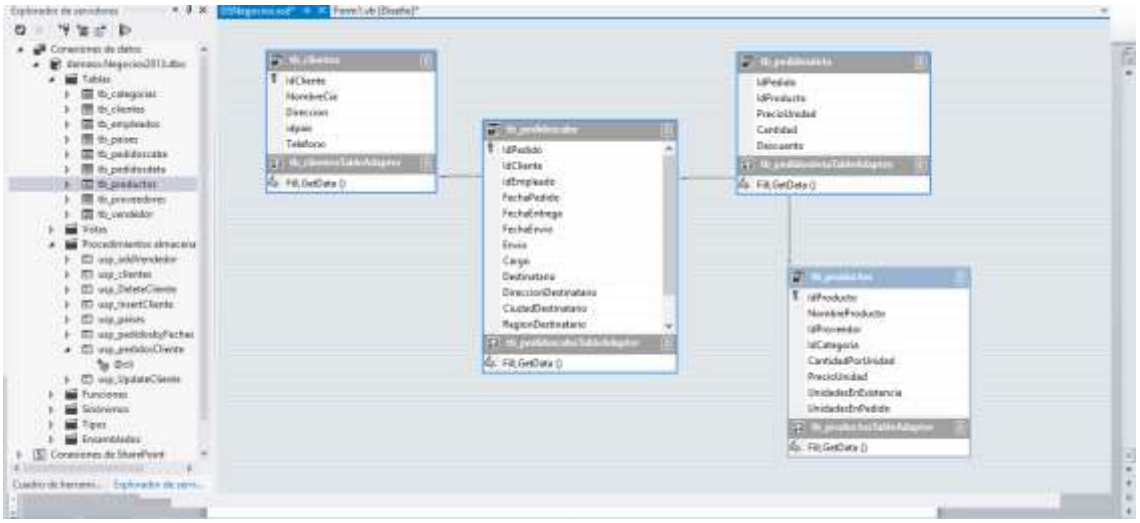
Selecciona el origen de datos: Microsoft SQL Server



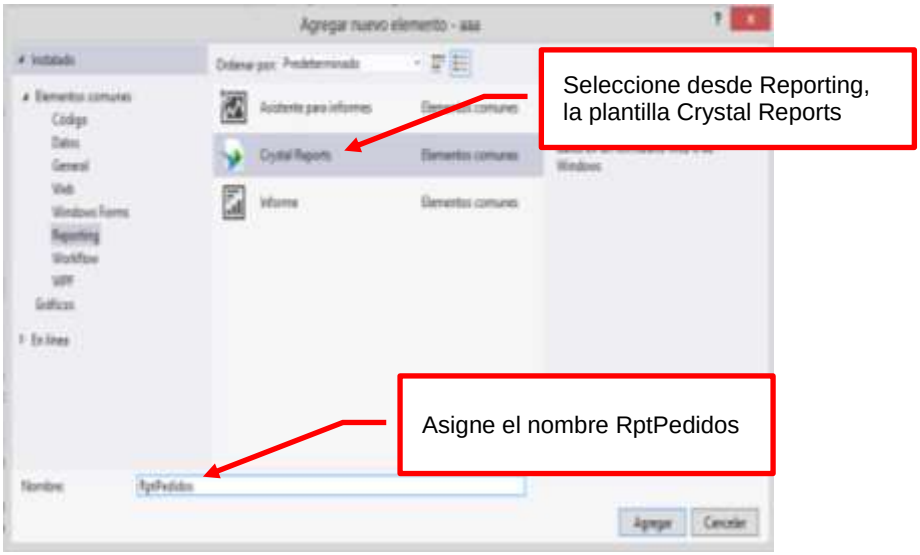


Defina la conexión: nombre del servidor, Autenticacion por SQL Server: usuario y clave; y selecciona la base de datos: Negocios2013. Presiona el botón Aceptar

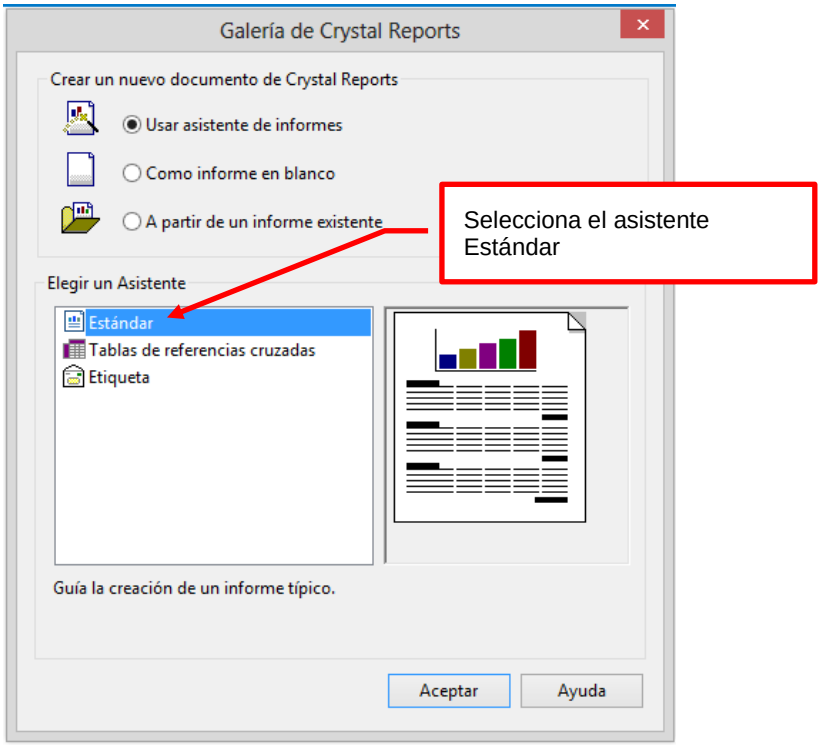
Al finalizar, arrastre las tablas al DataSet, tal como se muestra



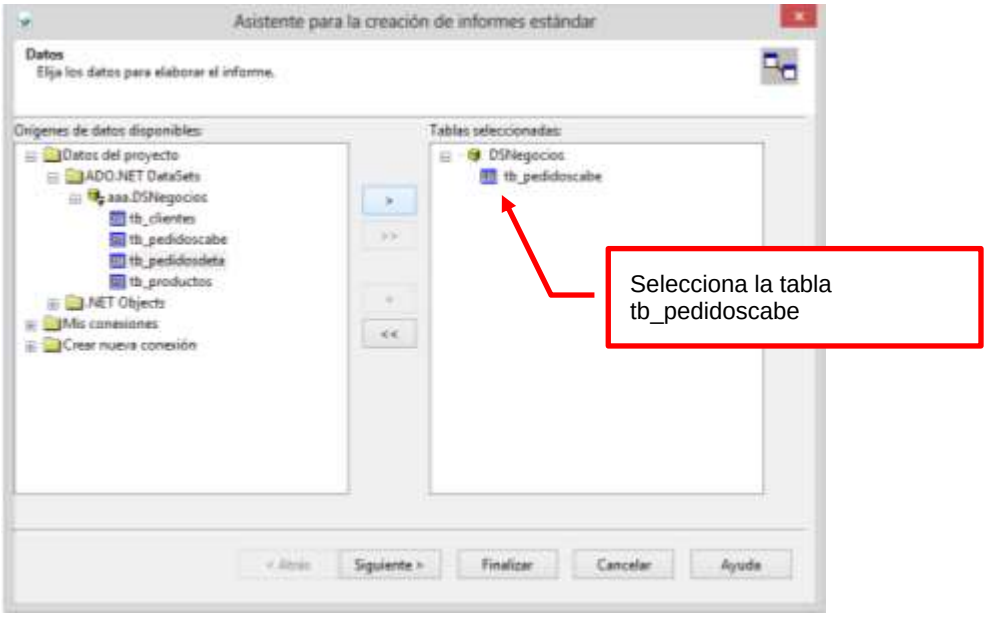
11.5.3 A continuación agregue un archivo Crystal Report para listar los pedidos



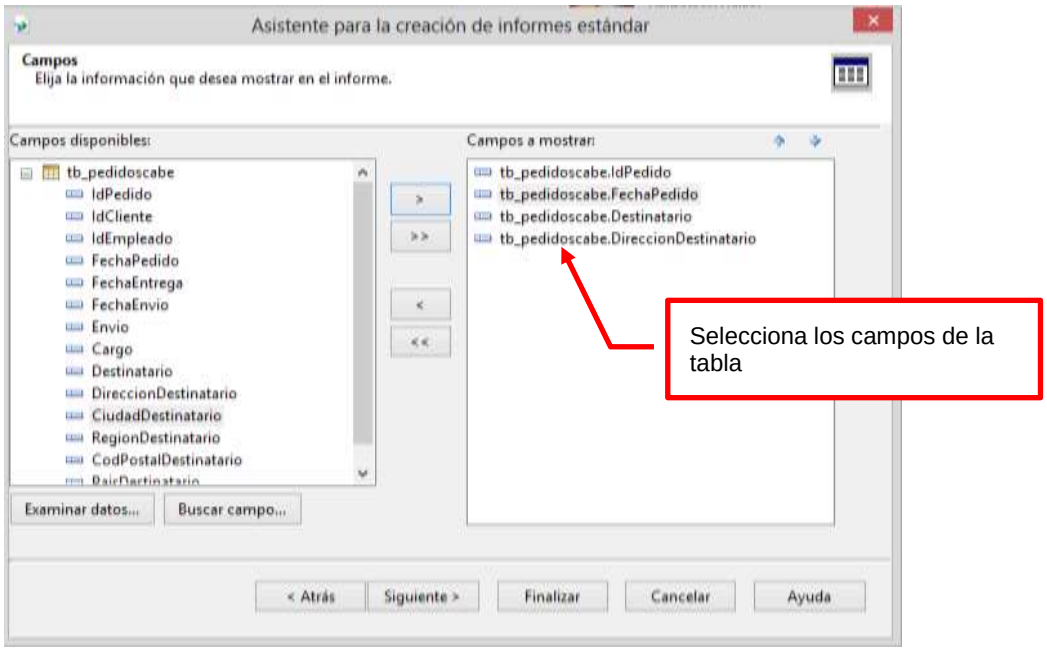
Selecciona desde la opción Galeria, el asistente Estándar, presiona el botón ACEPTAR



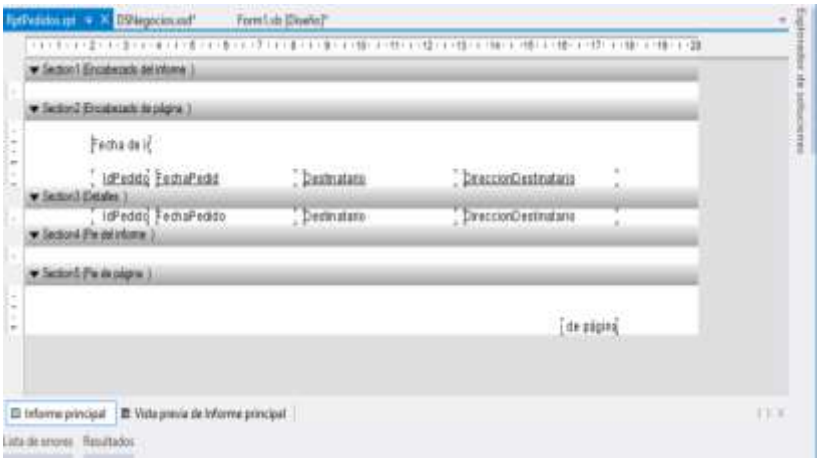
Selecciona desde la carpeta DataSet: DSNegocios, la tabla de trabajo: tb_pedidoscabe, tal comose muestra



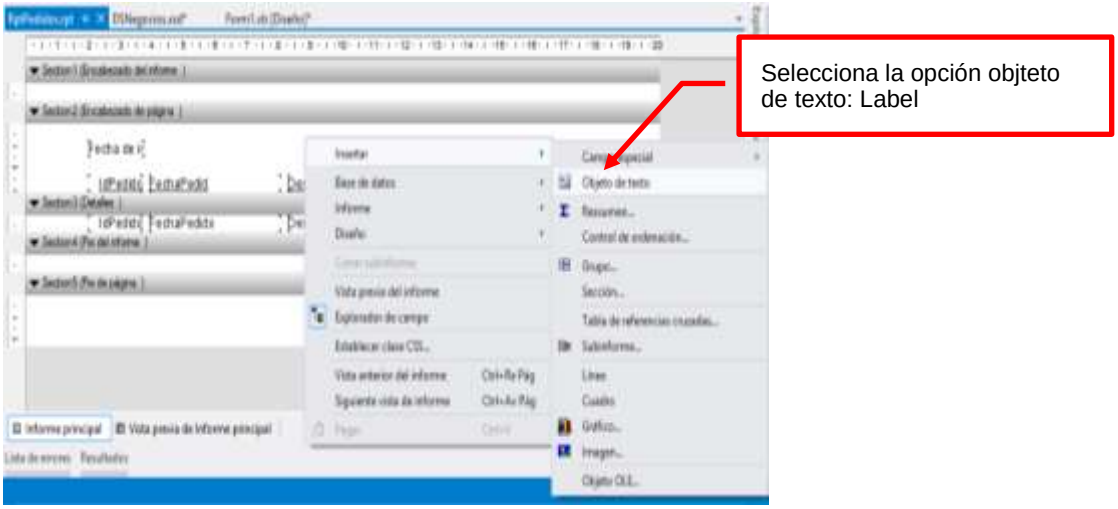
Selecciona los campos de la tabla selecciona, para terminar presiona el botón FINALIZAR



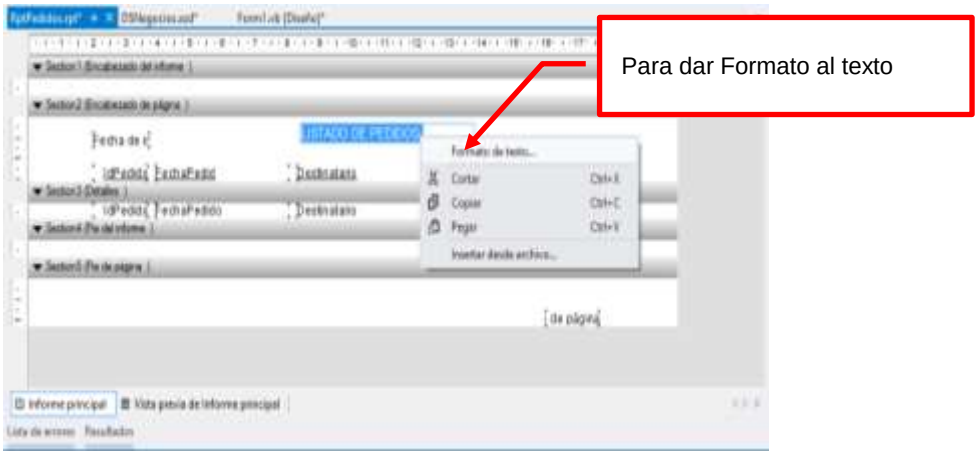
Archivo RptPedidos diseñado por el asistente



Para agregar un Label, hacer un click derecho en un sector: Sector2, seleccione desde la opción INSERTAR, objeto de texto, tal como se muestra

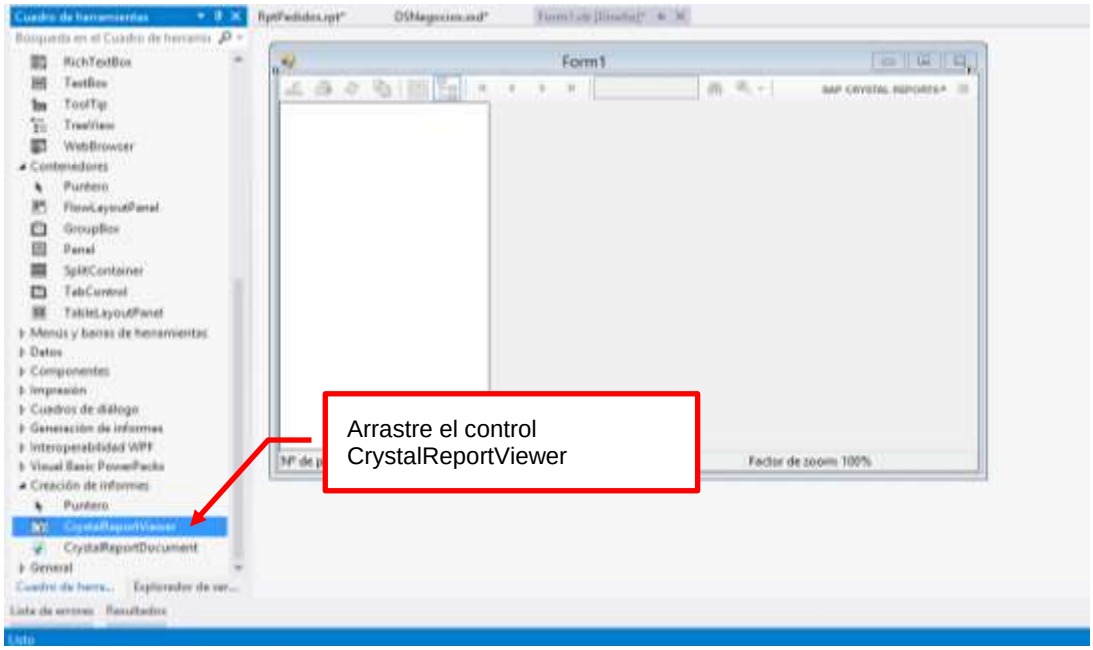


Para dar formato, seleccione el objeto, hacer click derecho y selecciona Formato de Texto

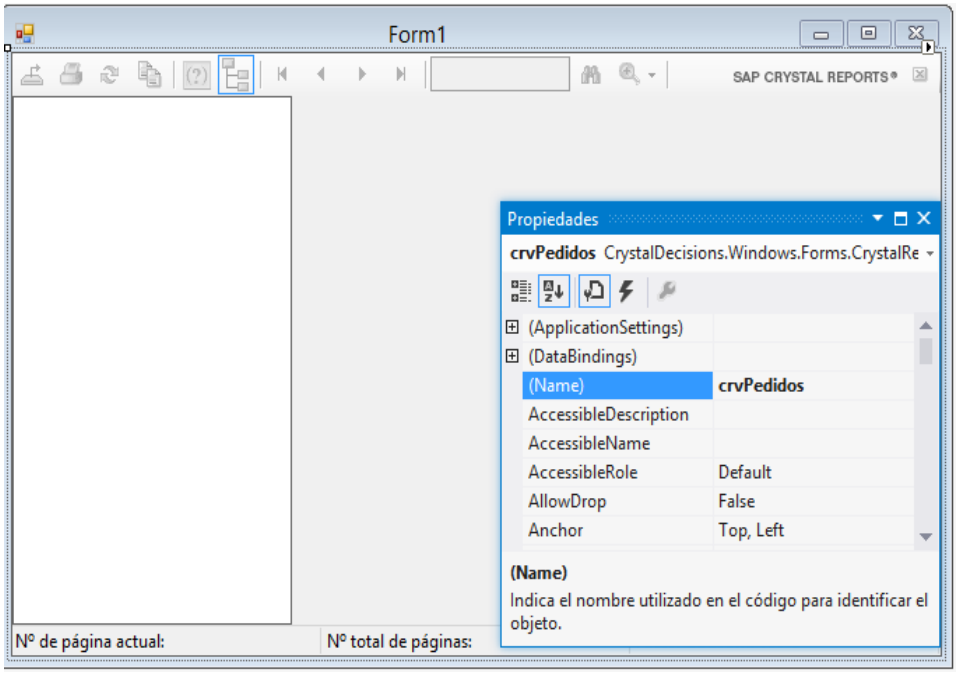


Programacion

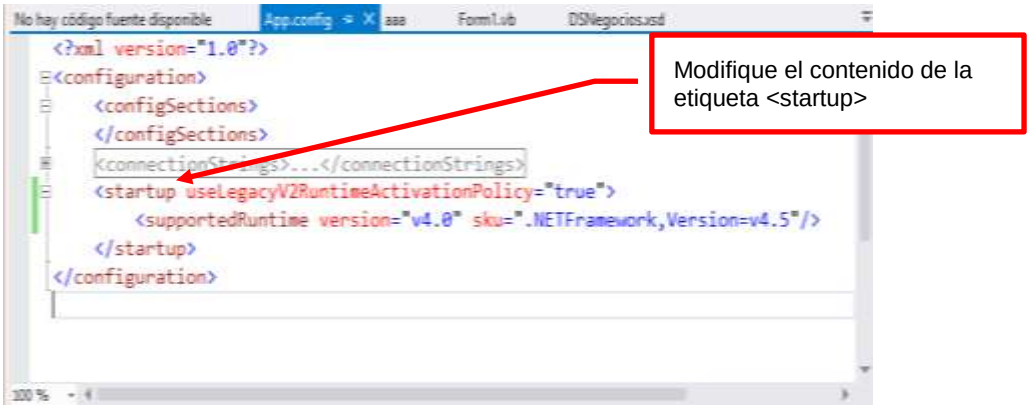
Desde el formulario, agrega el contro CrystalReportViewer, que es el visor del Archivo Crystal Report



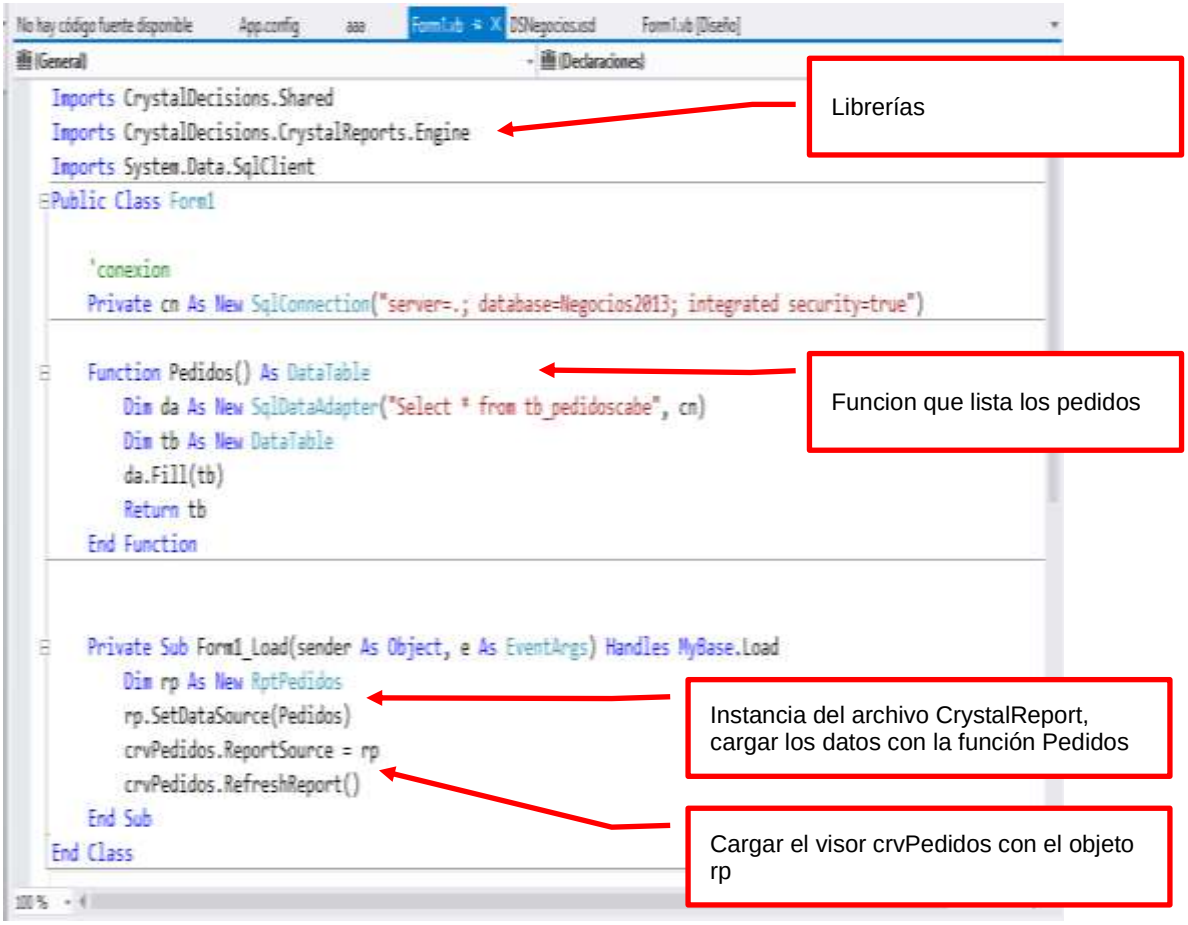
Asigne el nombre al control: crvPedidos, tal como se muestra



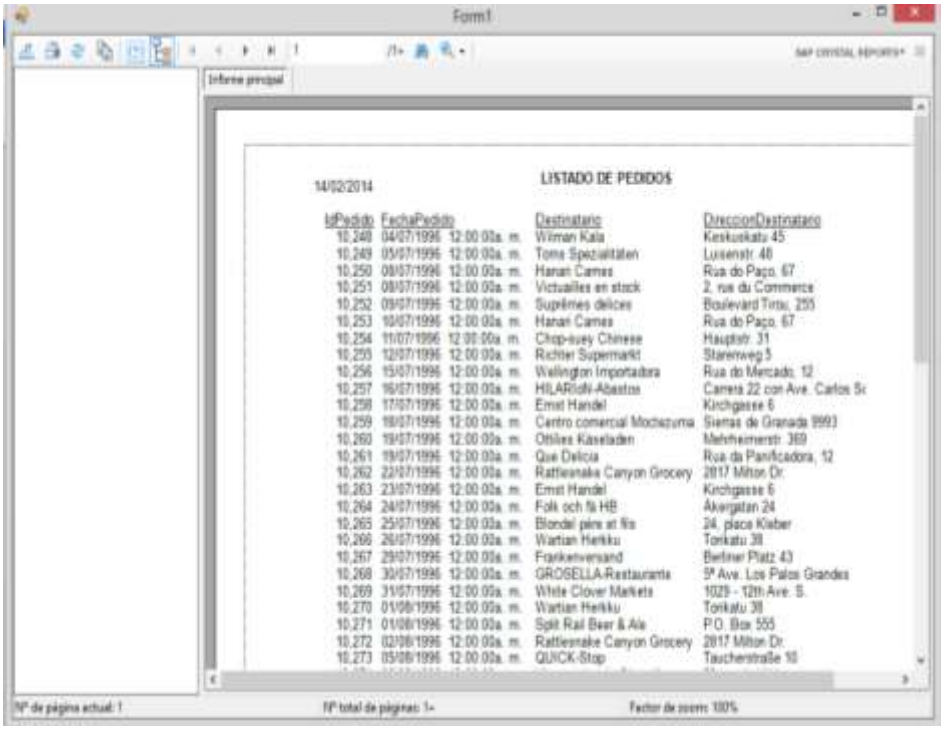
En el archivo app.Config, agrregue el codigo en la etiqueta <startup>



Programa el formulario, el cual lista los datos en el informe, visualizando los datos en el Visor del CrystalReport

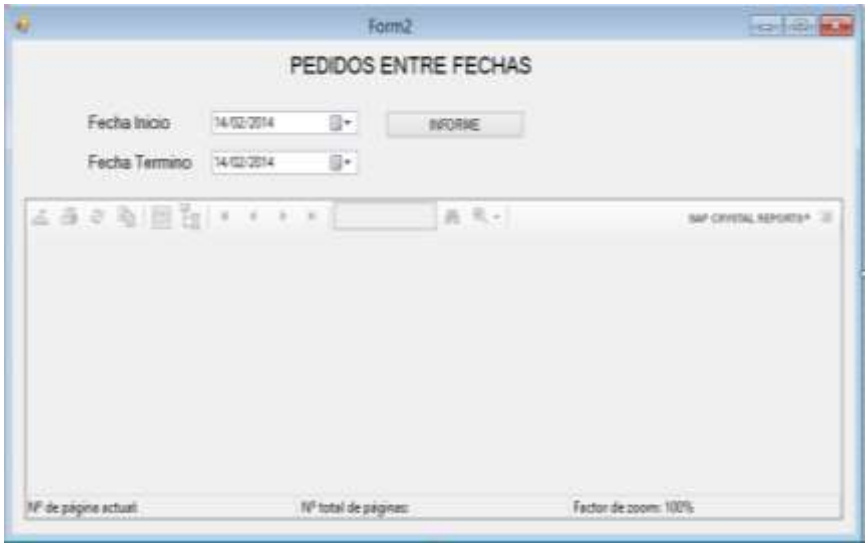


Presiona la tecla F5 para ejecutar la aplicación, visualizando en el Form1 los datos del informe de pedidos



DESARROLLO PRÁCTICO

Implemente un informe donde liste los pedidos entre un rango de dos fechas.



Programación.

Programa el formulario para listar los pedidos entre fechas, tal como se muestra

```
Imports CrystalDecisions.Shared
Imports CrystalDecisions.CrystalReports.Engine
Imports System.Data.SqlClient
Public Class Form2

    'conexion
    Private cn As New SqlConnection("server=.; database=Negocios2013; integrated security=true")

    Function Pedidos(fi As Date, ft As Date) As DataTable
        Dim da As New SqlDataAdapter("Select * from tb_pedidoscabe Where FechaPedido Between @fi and @ft", cn)
        da.SelectCommand.Parameters.AddWithValue("@fi", fi)
        da.SelectCommand.Parameters.AddWithValue("@ft", ft)
        Dim tb As New DataTable
        da.Fill(tb)
        Return tb
    End Function

    Private Sub btnInforme_Click(sender As Object, e As EventArgs) Handles btnInforme.Click
        Dim rp As New RptPedidos
        rp.SetDataSource(Pedidos(dtpInicio.Value, dtpTermino.Value))
        crvPedidos.ReportSource = rp
        crvPedidos.RefreshReport()
    End Sub
End Class
```

Librerías

Funcion que retorne los Pedidos entre dos fechas

Instancia del archivo CrystalReport, cargar los datos con la función Pedidos

Cargar el visor crvPedidos con el objeto rp

Presiona la tecla F5 para ejecutar el formulario. Seleccione las fechas, presiona el botón Informe, donde se visualiza los registros de pedidos entre el rango de fechas