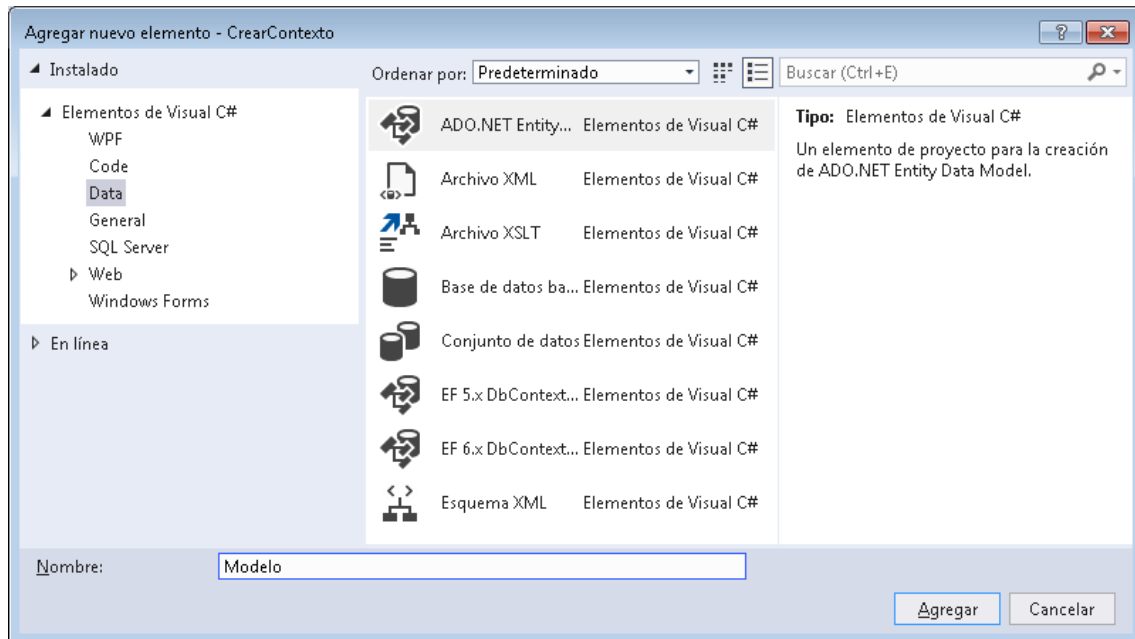


Añadir un contexto de datos a la aplicación

Las aplicaciones modernas se comunican con la base de datos a través de ORMs (en inglés mapeo objeto relacional). Esta herramienta se encarga de transformar por nosotros los datos en crudo de la base de datos en objetos debidamente tipados con los que resulta muy fácil trabajar.

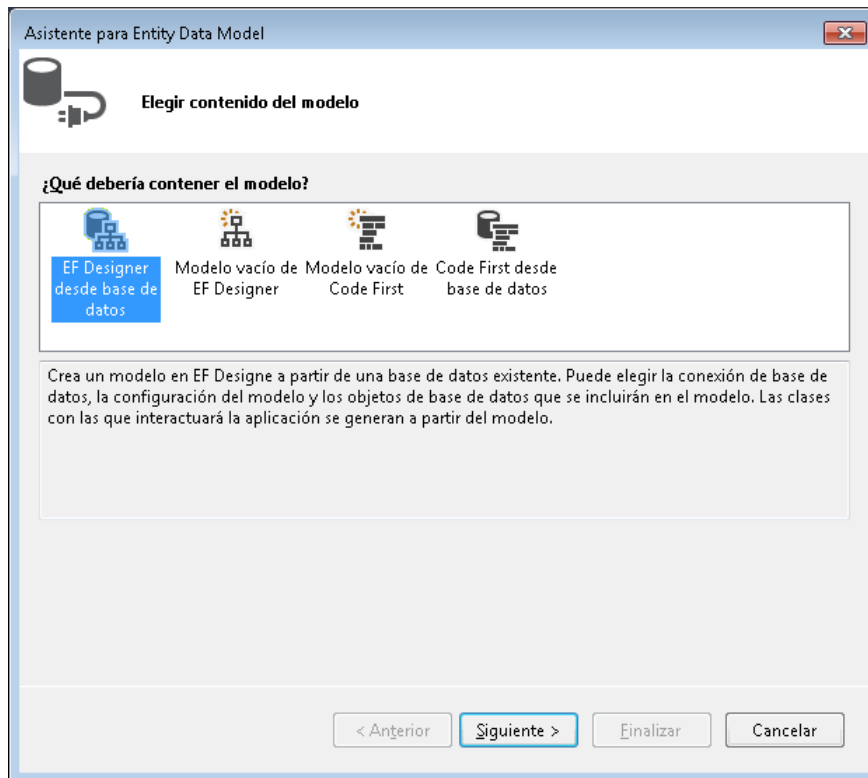
En primer lugar debemos añadir un objeto a nuestra aplicación con la relación entre el modelo de datos y el modelo de objetos. Buscamos en la categoría Data la plantilla “ADO.NET Entity Data Model” y le damos un nombre apropiado, en este caso “Modelo”



Al añadir la plantilla automáticamente se iniciará un asistente, la primera pantalla que nos encontraremos nos permitirá seleccionar la forma en la que vamos a trabajar con la base de datos. Tenemos 4 modos de trabajo.

- **EF Designer desde base de datos**, a partir de una base de datos existente se crean las clases y el contexto.
- Modelo vacío de EF Designer, podemos dibujar la base de datos desde la plantilla y generar scripts sql para sincronizar la base de datos.
- Modelo vacío de Code First, la base de datos se actualiza automáticamente para adaptarse a nuestras clases.
- Code First desde base de datos, generamos clases a partir de la base de datos a partir de lo cual si actualizamos las clases la base de datos se adaptará automáticamente a nuestro código.

Nosotros vamos a emplear siempre el primer tipo “EF Designer desde base de datos” ya que en un entorno profesional los datos siempre son lo importante. Code first es muy cómodo, pero solo si no somos programadores experimentados ya que cualquier cambio en una clase BORRA LA BASE DE DATOS Y LA VUELVE A CREAR eliminando los datos.



Lo siguiente es conectarnos a una base de datos, rellenamos el nombre del servidor, en nuestro caso “(localdb)\mssqllocaldb”. Ya que estamos desarrollando en local podemos indicar que nos autentiquemos con nuestra cuenta de Windows, opción “Autenticación de Windows”. Si fuera un servidor compartido deberíamos indicar el nombre de usuario y la contraseña.

Si los dos campos anteriores son correctos el desplegable para seleccionar la base de datos se rellenará automáticamente, seleccionamos la base de datos **Neumáticos** y aceptamos.

Propiedades de la conexión

Especifique la información para conectarse al origen de datos seleccionado o haga clic en "Cambiar" para elegir otro origen o proveedor de datos.

Origen de datos:
 Microsoft SQL Server (SqlClient) Cambiar...

Nombre del servidor:
 (localdb)\mssqllocaldb Actualizar

Conexión con el servidor
 Autenticación: Autenticación de Windows

Nombre de usuario:

Contraseña:

☐ Guardar mi contraseña

Establecer conexión con una base de datos

☒ Seleccionar o escribir el nombre de la base de datos:
 Neumaticos

☐ Adjuntar un archivo de base de datos:
 Examinar...

Nombre lógico:

Avanzadas...


Probar conexión Aceptar Cancelar

Una vez seleccionada la base de datos se nos pregunta si deseamos guardar la cadena de conexión en nuestra aplicación, que es lo normal, en la parte superior del formulario seleccionamos alguna de las conexiones existentes, si no existiera una conexión a nuestra base de datos podríamos crearla con el botón de “Nueva conexión”.

En la parte inferior marcamos la opción **“Guardar configuración de conexión en App.Config como:”** y le damos un nombre, debemos tener en cuenta que por defecto el nombre de la cadena de conexión es también el nombre del contexto de acceso a la base de datos. Por eso, lo vamos a llamar “NeumaticosBD”.

NOTA: App.config es un fichero xml que contiene las opciones de configuración de nuestra aplicación y que se añade al directorio raíz de nuestro proyecto, si más adelante quisiéramos cambiar el servidor en el que se encuentra nuestra base de datos podríamos editarlo fácilmente.

Asistente para Entity Data Model

 Elegir la conexión de datos

¿Qué conexión de datos debe usar la aplicación para conectarse a la base de datos?

portatil\localdb#43378c64.Numaticos.dbo Nueva conexión...

Esta cadena de conexión parece contener datos confidenciales (por ejemplo, una contraseña) que son necesarios para conectarse con la base de datos. Almacenar datos confidenciales en la cadena de conexión puede suponer un riesgo para la seguridad. ¿Desea incluir estos datos en la cadena de conexión?

☐ No, excluir datos confidenciales de la cadena de conexión. Los estableceré en el código de mi aplicación.

☐ Sí, incluir datos confidenciales en la cadena de conexión.

Cadena de conexión:

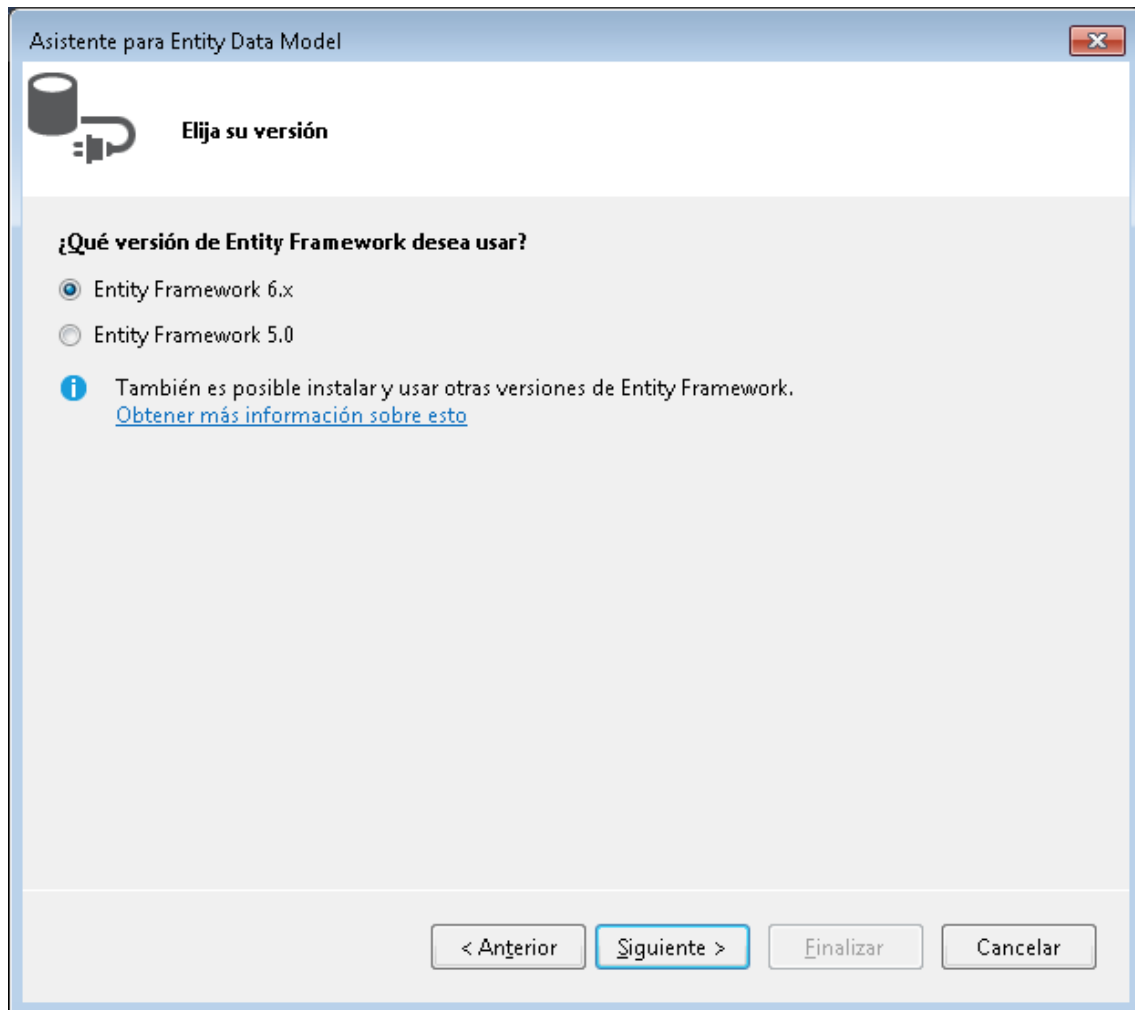
```
metadata=res://*/Modelo.csdl|res://*/Modelo.ssdl|
res://*/Modelo.msl;provider=System.Data.SqlClient;provider connection string="data source=
(localdb)\mssqllocaldb;initial catalog=Neumaticos;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Guardar configuración de conexión en App.Config como:

NeumaticosBD

< Anterior Siguiente > Finalizar Cancelar

La siguiente ventana es opcional, y nos aparecerá en el caso de que todavía no tengamos cargado el paquete de nuget de entity framework. Elegimos la última versión y aceptamos, si quisiéramos otro ORM tendríamos que cargarlo nosotros previamente.




Añadimos a nuestro modelo los elementos de la base de datos que necesitamos, al añadirlos se crearan las clases que los representan junto con sus propiedades. En este caso seleccionamos todas las tablas.

Otra opción interesante y que vamos a activar es “Poner en plural o en singular los nombres de los objetos generados”, de esta forma las clases que se creen tendrán un nombre en singular, por ejemplo, para la tabla “Clientes” su clase correspondiente será “Cliente”.

Por último, el nombre del espacio de nombres, **resulta muy cómodo que nuestras clases o modelos de la base de datos tengan un espacio de nombre acorde con el del proyecto ya que así no tendremos que estar continuamente importando el espacio de nombres**, en este caso como el proyecto de demostración se denomina “CrearContexto” nosotros le vamos a asignar el mismo nombre a las clases de la base de datos.

Asistente para Entity Data Model

 Elegir los objetos y la configuración de la base de datos

¿Qué objetos de la base de datos desea incluir en su modelo?

- ☒ Tablas
 - ☒ dbo
 - ☒ Clientes
 - ☒ DetalleFacturas
 - ☒ Facturas
 - ☒ Productos
- ☐ Vistas
- ☐ Funciones y procedimientos almacenados

☒ Poner en plural o en singular los nombres de objeto generados

☒ Incluir columnas de clave externa en el modelo

☐ Importar procedimientos almacenados y funciones seleccionados en Entity Model

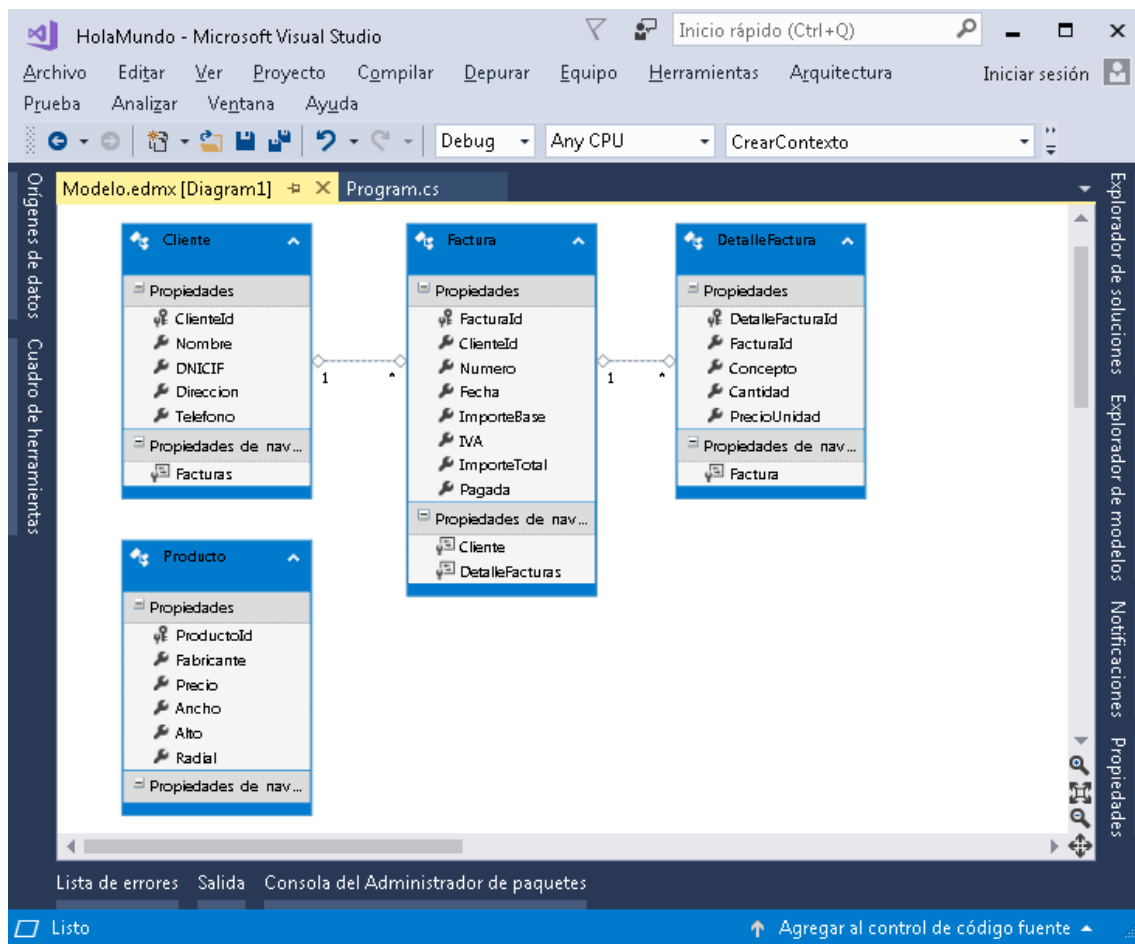
Espacio de nombres del modelo:

CrearContexto

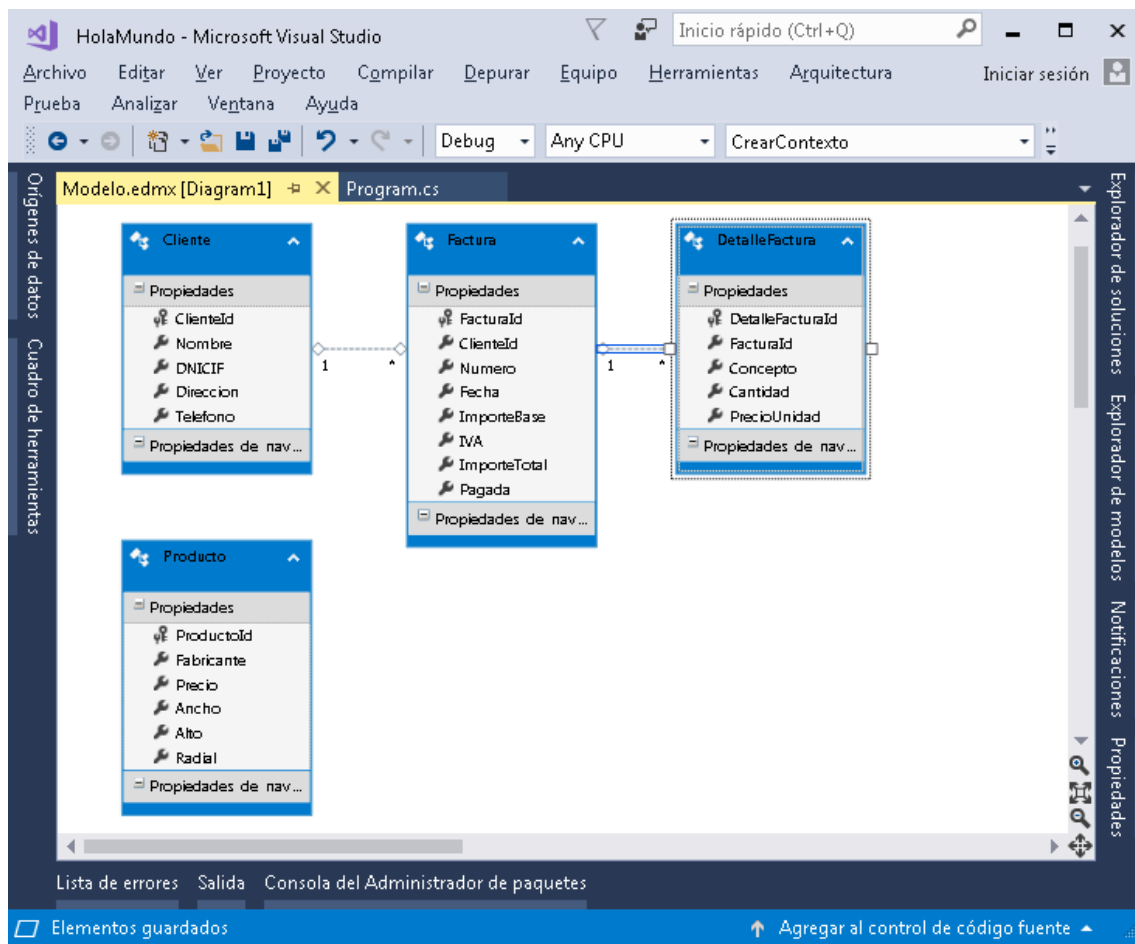
< Anterior Siguiente > Finalizar Cancelar

Una vez finalizado el asistente cada vez que abramos el fichero con el modelo se abrirá un asistente para que podamos dibujar nuestros objetos. Fíjate como se han mantenido las relaciones entre tablas, los tipos de las propiedades y las claves.

Aparece un concepto nuevo que es el de las **“propiedades de navegación”**, si te fijas **por cada relación se crean en las clases objetos que hacen referencia a los extremos de la relación**. Por ejemplo en factura tenemos una propiedad Cliente y otra propiedad DetalleFacturas con la lista de detalles de la factura.

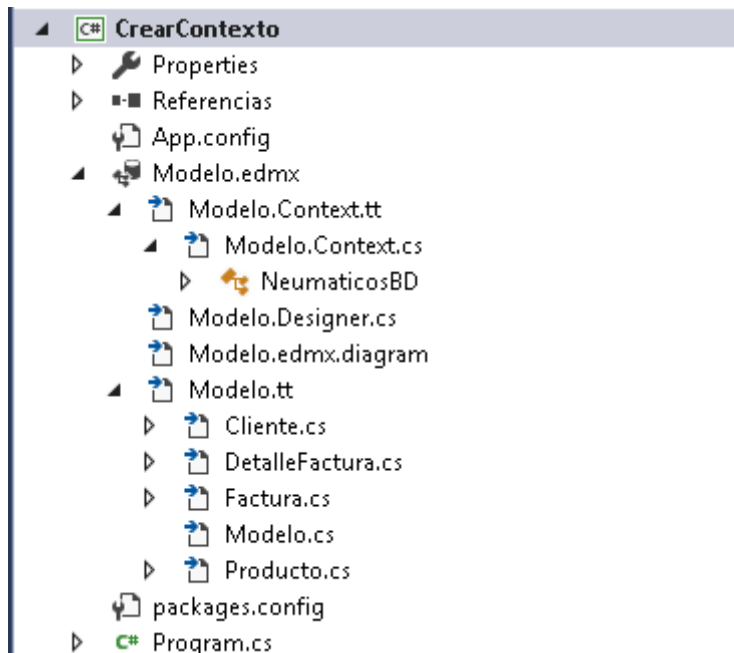


Por desgracia las propiedades de navegación dan más problemas que facilidades en las operaciones de actualización por lo que no se suelen usar nunca. Deberás actualizar el diseñador y eliminar las propiedades de navegación quedando el modelo de la siguiente manera.



NOTA: IMPORTANTE, cada vez que modifiques algo en el diseñador y pulses el botón de guardar automáticamente se sobrescribirán todas las clases asociadas, por lo que no suele ser buena idea meter código a mano en las clases generadas automáticamente.

Si expandes en el explorador el modelo puedes ver los ficheros con las clases generados automáticamente, "NeumaticosBD" es el contexto o clase a través de la cual podremos comunicarnos con la base de datos. Es muy recomendable que abras este fichero y localices las propiedades o colecciones asociadas a las tablas de la base de datos. A partir de estas propiedades trabajar con la base de datos es tan fácil como recorrer un array.



Consultas a través del contexto

El siguiente código muestra las operaciones habituales sobre la base de datos, búsqueda de múltiples elementos según una condición, búsqueda de un elemento según una condición, listar todos los elementos de una tabla, añadir un nuevo elemento y confirmar los cambios.

```
static void Main(string[] args)
{
    //instanciamos el contexto
    NeumaticosBD ctx = new NeumaticosBD();

    //ejemplos de consultas
    //where para devolver multiples objetos o una lista
    ctx.Clientes.Where(x => x.Nombre == "PEPE");
    //firstordefault para devolver 1 elemento, null si no encuentra
    ctx.Clientes.FirstOrDefault(x => x.ClienteId == 10);
    ctx.Clientes.ToList();

    //como crear un nuevo elemento
    Producto nuevo = new Producto() {
        Alto = 55, Ancho = 225, Radial = 18,
        Precio = 110, Fabricante = "Michelin"
    };
    ctx.Productos.Add(nuevo);

    //guarda los cambios pendiente, crear, borrar, actualizar
    ctx.SaveChanges();
}
```

En el caso de los informes puede ser interesante realizar consultas manuales por lo que el siguiente código muestra cómo realizar una consulta de forma manual, ten en cuenta que trabajamos siempre con objetos por lo que nuestra consulta necesitará ajustarse a un tipo de dato que tenga los mismos campos que los de los de la consulta, esta asociación se realiza a través del nombre por lo que si los escribimos mal ese campo aparecerá con valor nulo.

```
private static void ConsultaPersonalizada()
{
    NeumaticosBD ctx = new NeumaticosBD();
    string consultaSql = "Select fabricante, count(*) as modelos from
productos group by fabricante";

    List<TipoRespuestaSQL> resultado =
ctx.Database.SqlQuery<TipoRespuestaSQL>(consultaSql, new object[0]).ToList();
}

private class TipoRespuestaSQL
{
    public string Fabricante { get; set; }
    public int Modelos { get; set; }
}
```