Herramientas de Programación 1

Tema Nº10:

Entity Framework (EF) - LINQ

Indicador de logro Nº10:Aplica LINQ como lenguaje de acceso y resuelve la gestión de información a través de EF, logrando gestionar los datos de forma transparente.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº1:**

ENTITY FRAMEWORK - LINQ TO SQL

Implementaciones De Entity Framework

Existen tres formas de implementar Entity Framework en nuestras aplicaciones:

Database First:

Como su nombre sugiere, este método funciona con una base de datos ya existente. Esto puede ser porque la aplicación que estamos desarrollando utiliza una base de datos que ya estaba siendo utilizada por otra aplicación, o bien porque el equipo de desarrollo prefiere primero crear la base de datos junto con sus tablas, y posteriormente incorporarlas en la aplicación. Una vez que tengamos nuestras tablas creadas, las incorporamos de forma gráfica a nuestra aplicación.

Este métodos es la que usaremos en el desarrollo de nuestras aplicaciones.

Model First:

A diferencia de la alternativa anterior, este método permite crear la base de datos desde Visual Studio. Para ello contamos con un diseñador, en el cual podemos (de forma gráfica) agregar las entidades, sus propiedades, relaciones y otros elementos necesarios. Esta alternativa es muy recomendable para aquellas personas que necesiten crear una base de datos nueva, pero de una manera práctica y donde podremos personalizar todos (o gran parte) de aquellos aspectos necesarios en las tablas asociadas a las entidades. Podemos ver su implementación como el camino inverso a Database First. Generalmente esta es la alternativa recomendada para diseñadores o analistas.

Code First:

Este método, al igual que el anterior, nos permite crear la base de datos a partir de las entidades que definamos en nuestra aplicación. La diferencia con Model First radica en que aquí las entidades serán clases definidas a través de código. Esta alternativa es normalmente más apropiada para desarrolladores, los cuales prefieren definir las entidades, sus propiedades, relaciones y características propias que tendrán en la base de datos a través de código C#, ya que requiere mayor trabajo y conocimiento específico de cómo implementar estas características.

LINQ

LINQ (Language Integrated Query) son un conjunto herramientas de Microsoft para realizar todo tipo de consultas a distintas fuentes de datos: objetos, xmls, bases de datos, etc... Para ello, usa un tipo de funciones propias, que unifica las operaciones más comunes en todos los entornos, con esto, se consigue un mismo lenguaje para todo tipo de tareas con datos.

LINQ TO SQL

Es una tecnología de LINQ que le permite a los desarrolladores de software de gestión la posibilidad de abstraerse de la capa de acceso a datos de sus aplicaciones, así como definir consultas a la base de datos en lenguaje LINQ.

Ahora vamos a ver un sencillo ejemplo, con una consulta linq en c# para objetos (LINQ to objetcs):

**Ejemplos:**

var lista = from c in coleccion

where c.propiedad1 == true

select c;

Como podemos ver, accedemos a una colección y filtramos todos los elementos que cumplan que la propiedad 1 sea verdadera. Del resultado de esta consulta, podemos sacar un listado de elementos con ToList(), el número de ellos con la funcion count(), extraer los datos a un arreglo, etc.

**Ejemplos:**

**RECURSOS**

1. Hardware

* Una computadora con 4 GB e RAM y procesador de 2.0 GHz.

1. Software

* Visual Studio Professional 2015 / 2019
* SQL Server 2016 / 2019

En los ejemplos se utilizó la versión Express 2016, puede usar las versiones comerciales o la Developer.

* SQL Server Management Studio (SSMS)

Es la herramienta cliente más usada para gestionar las BD de SQL Server.

**PROCEDIMIENTO**

**Laboratorio 10:**

10.1 Entity Framework: DataBase First.

**Objetivos**

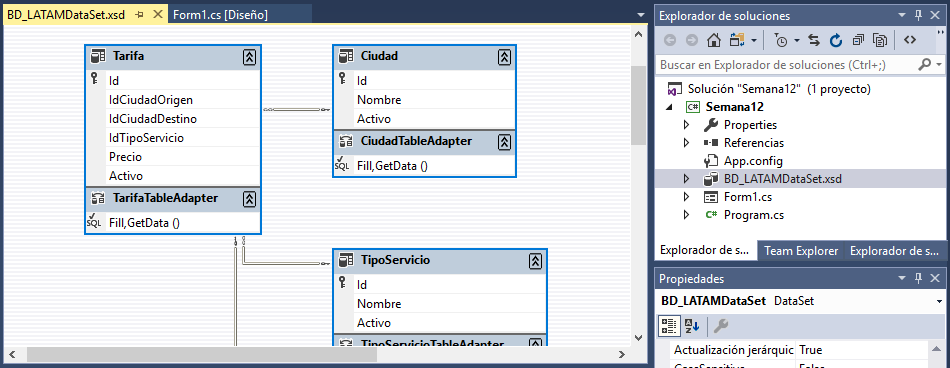
* Crear el Modelo a partir de la base de datos.

**Introducción:**

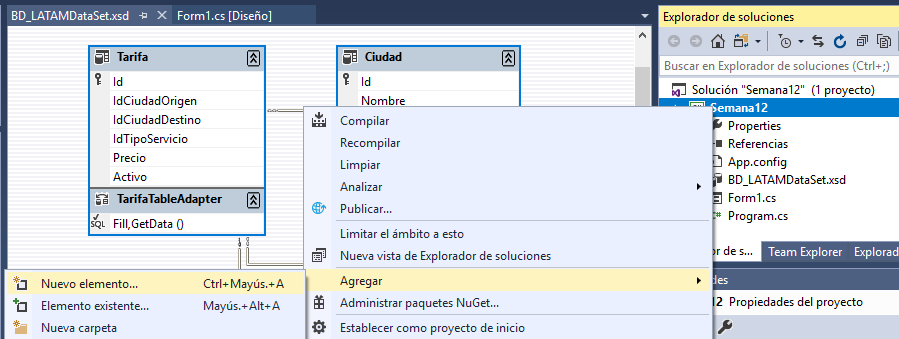
* Durante esta actividad, vamos a generar el Modelo a partir de la base de datos BD\_LATAM, para ello haremos uso del origen de datos creado en la semana 9 y el método Data Base First.

Paso 1. Generar el Modelo a partir de la base de datos BD\_LATAM.

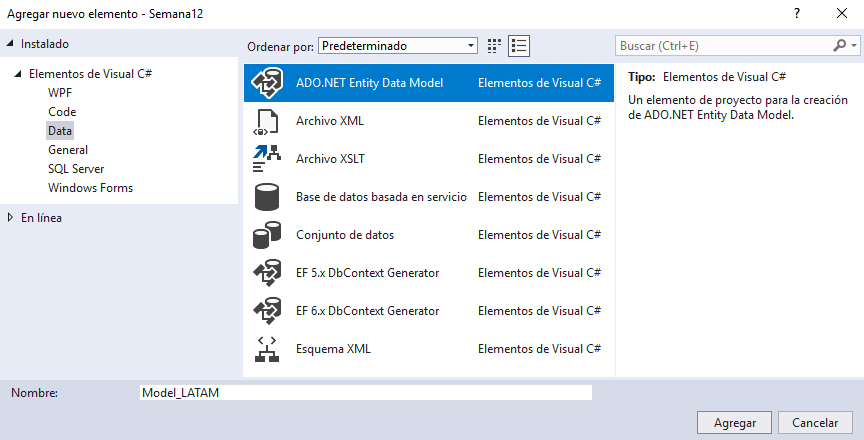
1. Creamos el proyecto Semana12.
2. Agregamos un origen de datos a la base de datos BD\_LATAM, de la misma forma que lo hicimos en la semana 11.



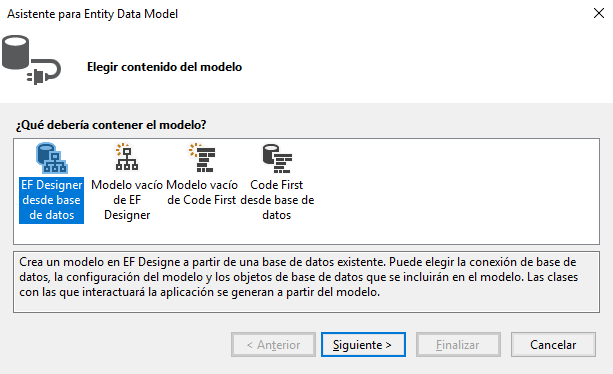
1. Vamos a crear el Modelo a partir de la BD\_LATAM, para ello primero agregamos al proyecto un nuevo elemento mediante la siguiente opción:



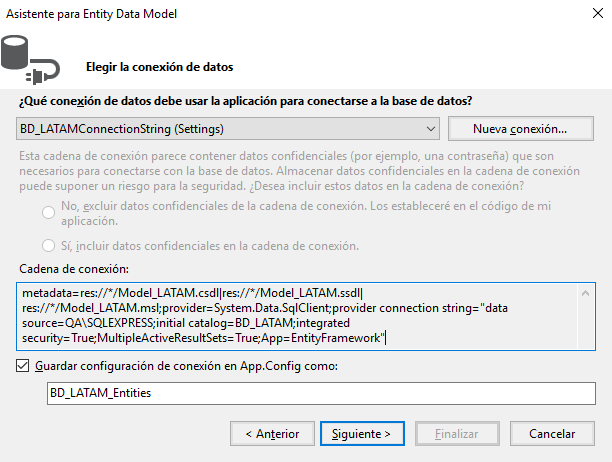
1. En el panel izquierdo seleccionamos la opción “Data”. En el panel central seleccionamos la opción “ADO.NET Entity Data Model”. Y al modelo le ponemos el nombre “Model\_LATAM”.



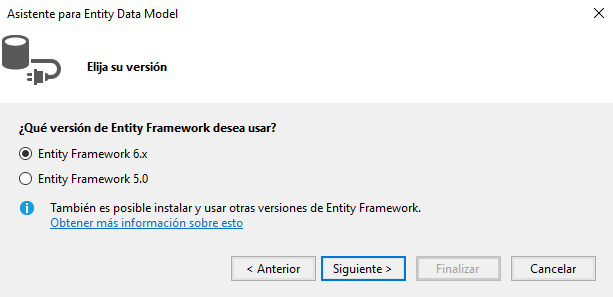
1. Visualizamos los distintos métodos para implementar Entity Framework, seleccionamos “EF Designer desde base de datos” y presionamos en el botón “Siguiente >”



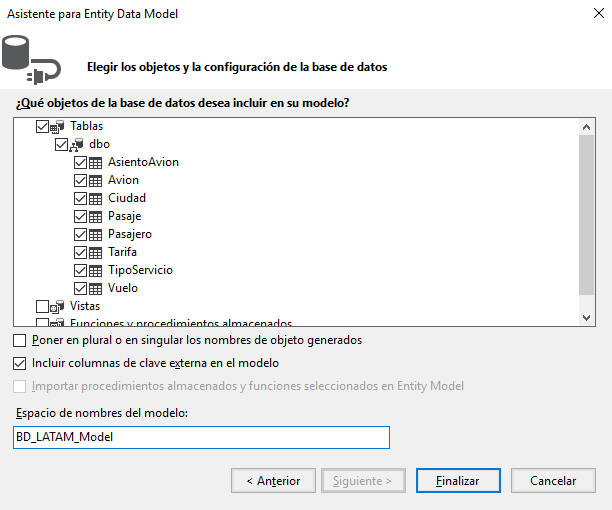
1. Seleccionamos la conexión que creamos en el paso b. y presionamos en el botón “Siguiente >”



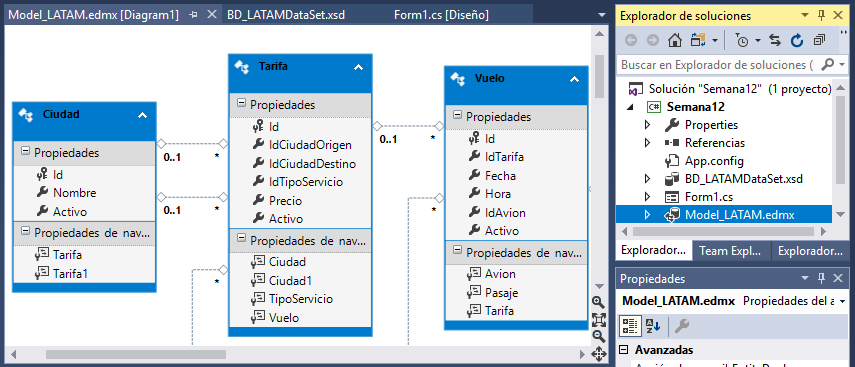
1. Seleccionamos la versión más reciente de Entity Framework y presionamos en el botón “Siguiente >”



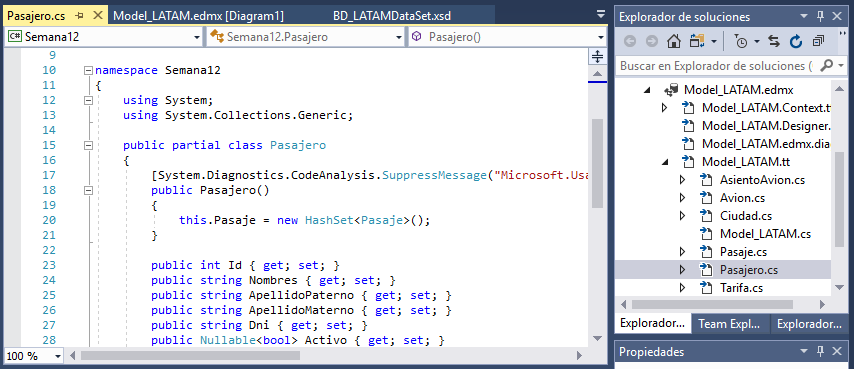
1. Seleccionamos todas las tablas y las opciones que se muestran, luego presionamos en el botón “Finalizar”



1. Una vez creado el Modelo ya lo podemos visualizar (Model\_LATAM.edmx):



1. Dentro del Modelo podemos también visualizar el código generado, esto es, las clases creadas para cada tabla de la BD.



10.2 LINQ to SQL.

Objetivos

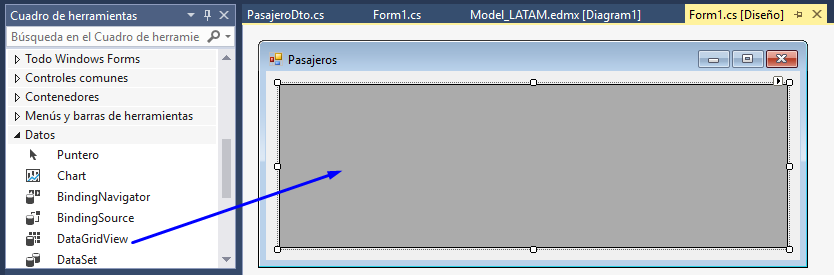
* Mostrar información de la BS en un formulario.
* Utilizar LINQ to SQLpara gestionar la información de una BD.

Introducción:

* Durante esta actividad, vamos a dar una introducción al mantenimiento de datos (CRUD) de una tabla de la BD utilizando LINQ to SQL.

Paso 1. Mostrar Datos de la tabla Pasajero de la BD.

1. Diseñamos el formulario de acuerdo a:



|  |  |  |
| --- | --- | --- |
| Control | Propiedad | Valor |
| Form1 | (Name) | frmPasajeros |
|  | Text | Pasajeros |
| DataGridView1 | (Name) | dgvPasajeros |

Creamos la clase PasajeroDto, de acuerdo a:

public class PasajeroDto

{

public int Id { get; set; }

public string Nombres { get; set; }

public string ApellidoPaterno { get; set; }

public string ApellidoMaterno { get; set; }

public string Dni { get; set; }

public Nullable<bool> Activo { get; set; }

}

1. Dentro de la clase del formulario creamos un método que nos va a devolver los pasajeros que están en la BD\_LATAM.

public List<PasajeroDto> ObtenerPasajeros()

{

try

{

using (BD\_LATAM\_Entities bd = new BD\_LATAM\_Entities())

{

var consulta = (from n in bd.Pasajero

select new PasajeroDto

{

Id = n.Id,

Nombres = n.Nombres,

ApellidoPaterno = n.ApellidoPaterno,

ApellidoMaterno = n.ApellidoMaterno,

Dni = n.Dni,

Activo = n.Activo

}).ToList();

return consulta;

}

}

catch (Exception)

{

throw;

}

}

1. Agregamos código al evento Load del formulario:

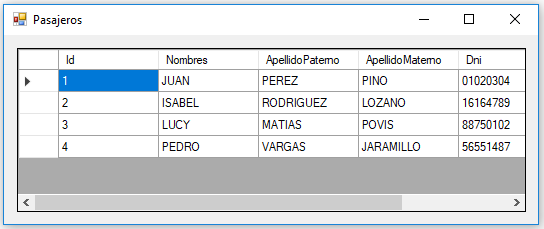
private void frmPasajeros\_Load(object sender, EventArgs e)

{

dgvPasajeros.DataSource = ObtenerPasajeros();

}

1. Ejecutamos la aplicación:



Paso 2. Agregar un registro a la tabla Pasajero de la BD.

1. En la clase del formulario creamos un método que nos permita insertar un nuevo pasajero a la tabla Pasajero de la BD

public void InsertarPasajero(Pasajero pPasajero)

{

try

{

using (BD\_LATAM\_Entities bd = new BD\_LATAM\_Entities())

{

bd.Pasajero.Add(pPasajero);

bd.SaveChanges();

}

}

catch (Exception)

{

throw;

}

}

1. Modificamos el código del evento Load del formulario de acuerdo a:

private void frmPasajeros\_Load(object sender, EventArgs e)

{

Pasajero pasajero = new Pasajero();

pasajero.Id = 0;

pasajero.Nombres = "Mario";

pasajero.ApellidoPaterno = "Zambrano";

pasajero.ApellidoMaterno = "Tello";

pasajero.Dni = "99887766";

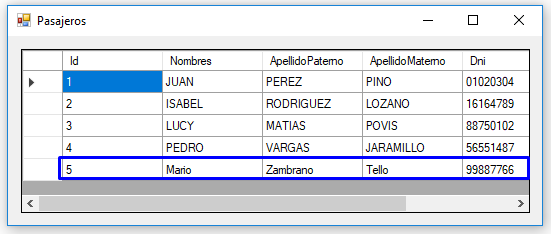
pasajero.Activo = true;

InsertarPasajero(pasajero);

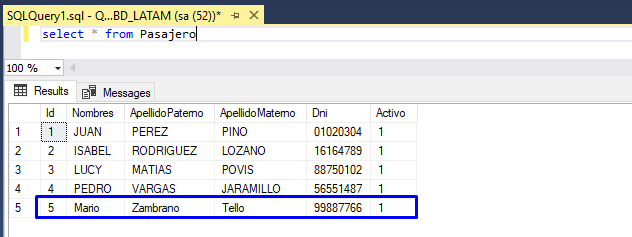
dgvPasajeros.DataSource = ObtenerPasajeros();

}

1. Ejecutamos la aplicación y verificamos que el registro ha sido insertado correctamente.



1. Verificamos en la BD:



Paso 3. Modificar los datos de un registro en la tabla Pasajero de la BD.

1. En la clase del formulario creamos un método que nos permita modificar los datos de un pasajero en la tabla Pasajero de la BD

public void ActualizarPasajero(Pasajero pPasajero)

{

try

{

using (BD\_LATAM\_Entities bd = new BD\_LATAM\_Entities())

{

var pasajeros = from p in bd.Pasajero

where p.Id == pPasajero.Id

select p;

foreach (Pasajero pas in pasajeros)

{

pas.Nombres = pPasajero.Nombres;

pas.ApellidoPaterno = pPasajero.ApellidoPaterno;

pas.ApellidoMaterno = pPasajero.ApellidoMaterno;

pas.Dni = pPasajero.Dni;

pas.Activo = pPasajero.Activo;

}

bd.SaveChanges();

}

}

catch (Exception)

{

throw;

}

}

1. Modificamos el código del evento Load del formulario de acuerdo a:

private void frmPasajeros\_Load(object sender, EventArgs e)

{

Pasajero pasajero = new Pasajero();

pasajero.Id = 5;

pasajero.Nombres = "Martha";

pasajero.ApellidoPaterno = "Zegarra";

pasajero.ApellidoMaterno = "Terrones";

pasajero.Dni = "987654321";

pasajero.Activo = false;

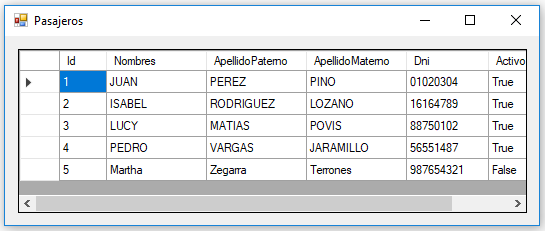
//InsertarPasajero(pasajero);

ActualizarPasajero(pasajero);

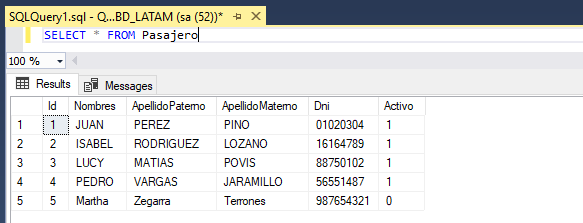
dgvPasajeros.DataSource = ObtenerPasajeros();

}

1. Ejecutamos la aplicación y verificamos que el registro ha sido modificado correctamente.



1. Verificamos en la BD:



Paso 4. Eliminar un registro en la tabla Pasajero de la BD.

1. En la clase del formulario creamos un método que nos permita eliminar un pasajero en la tabla Pasajero de la BD

public void EliminarPasajero(Pasajero pPasajero)

{

try

{

using (BD\_LATAM\_Entities bd = new BD\_LATAM\_Entities())

{

var pasajeros = from p in bd.Pasajero

where p.Id == pPasajero.Id

select p;

foreach (Pasajero pas in pasajeros)

{

bd.Pasajero.Remove(pas);

}

bd.SaveChanges();

}

}

catch (Exception)

{

throw;

}

}

1. Modificamos el código del evento Load del formulario de acuerdo a:

private void frmPasajeros\_Load(object sender, EventArgs e)

{

Pasajero pasajero = new Pasajero();

pasajero.Id = 5;

//pasajero.Nombres = "Martha";

//pasajero.ApellidoPaterno = "Zegarra";

//pasajero.ApellidoMaterno = "Terrones";

//pasajero.Dni = "987654321";

//pasajero.Activo = false;

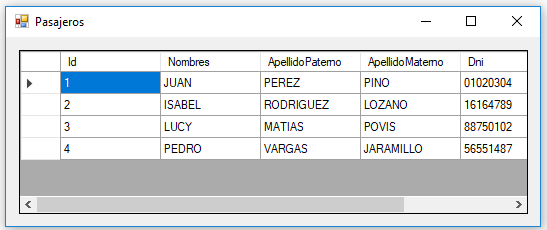
//InsertarPasajero(pasajero);

//ActualizarPasajero(pasajero);

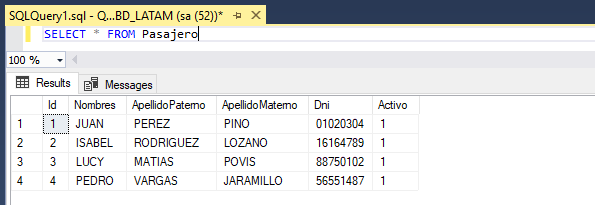
EliminarPasajero(pasajero);

dgvPasajeros.DataSource = ObtenerPasajeros();

1. Ejecutamos la aplicación y verificamos que el registro ha sido eliminado correctamente.



1. Verificamos en la BD:



**Actividad:**

Ingresa a la plataforma virtual. Luego revisa y analiza los siguientes enlaces para desarrollar las actividades propuestas:

* Entity Framework. Métodos de implementación.

<https://www.campusmvp.es/recursos/post/entity-framework-code-first-database-first-y-model-first-en-que-consiste-cada-uno.aspx>

* Entity Framework. Data Base First.

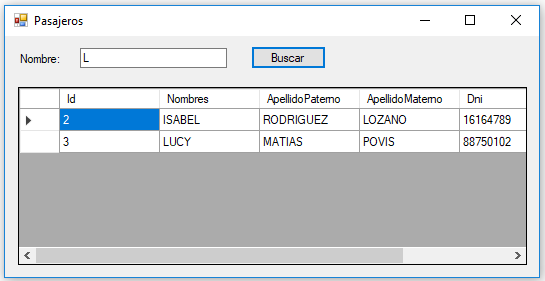
<https://www.youtube.com/watch?v=lC-NfgP-0qo>

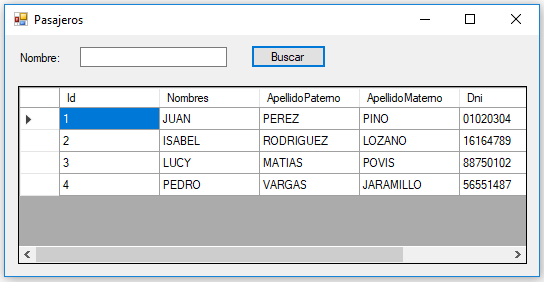
* LINQ TO SQL.

<https://docs.microsoft.com/es-es/dynamics365/customer-engagement/developer/org-service/linq-query-examples>

1. Agregar al formulario la opción de búsqueda.

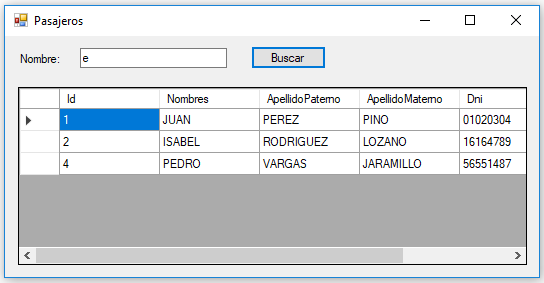
* El texto que se ingrese será el filtro para buscar al pasajero por el campo Nombre, por ejemplo:



* Si no se ingresa texto deberá mostrarse todos los pasajeros de la BD

1. Modificar la opción de búsqueda.

* El texto que se ingrese será el filtro para buscar al pasajero por el campo Nombre, Apellido Paterno o Apellido Materno, por ejemplo:



* Si no se ingresa texto deberá mostrarse todos los pasajeros de la BD.