Herramientas de Programación 1

Tema Nº11-A:

EF y LINQ.

Indicador de logro Nº11:Desarrolla aplicaciones de escritorio con acceso a base de datos utilizando EF y LINQ

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº11-A:**

CONTROL: DATAGRIDVIEW1

El control DataGridView proporciona una forma eficaz y flexible de mostrar datos en formato de tabla. Puede usar el control DataGridView para mostrar vistas de solo lectura de una pequeña cantidad de datos, o puede ampliarlo para mostrar vistas editables de conjuntos de datos muy grandes.

Puede ampliar el control DataGridView de varias maneras para construir comportamientos personalizados en sus aplicaciones. Por ejemplo, puede especificar mediante programación sus propios algoritmos de ordenación y puede crear sus propios tipos de celdas.

Puede personalizar fácilmente la apariencia del control DataGridView eligiendo entre varias propiedades. Se pueden usar muchos tipos de almacenes de datos como origen de datos, o el control DataGridView puede funcionar sin ningún origen de datos enlazado a él.

Propiedades

Las propiedades más comunes de un DataGridView son:

|  |  |
| --- | --- |
| Propiedad | Descripción |
| AllowUserToAddRows | Obtiene o establece un valor que indica si se muestra al usuario la opción de agregar filas. |
| AllowUserToDeleteRows | Obtiene o establece un valor que indica si el usuario puede eliminar filas del control. |
| MultiSelect | Obtiene o establece un valor que indica si el usuario puede seleccionar a la vez varias celdas, filas o columnas del control. |
| Name | Obtiene o establece el nombre del control. |
| ReadOnly | Obtiene o establece un valor que indica si el usuario puede editar las celdas del control. |

Eventos

Los eventos más comunes de un DataGridView son:

|  |  |
| --- | --- |
| Evento | Descripción |
| CellClick | Se produce cuando se hace clic en cualquier parte de una celda. |
| CellDoubleClick | Tiene lugar cuando el usuario hace doble clic en cualquier parte de una celda. |

**Ejemplos:**

**RECURSOS**

1. Hardware

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

1. Software

* Visual Studio Professional 2017 / 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 11:**

11.1 CRUD de la tabla Pasajero.

**Objetivos**

* Crear el Modelo a partir de la base de datos.
* Realizar el mantenimiento de la tabla Pasajero.

**Introducción:**

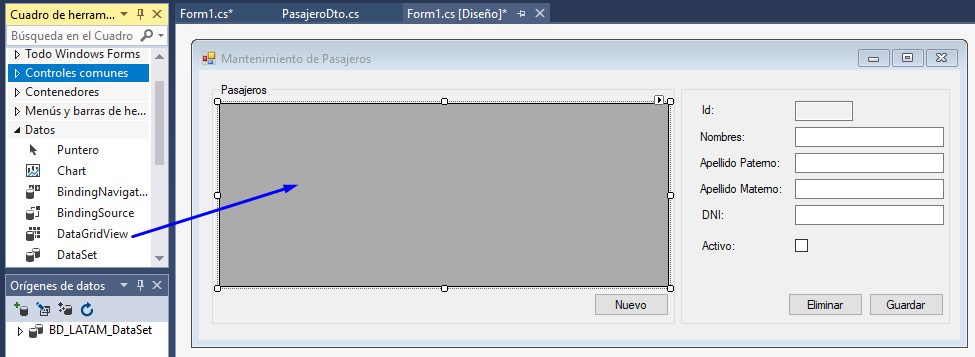
* Durante esta actividad, vamos a realizar el mantenimiento (CRUD: Create, Read, Update, Delete) de la tabla Pasajero de la base de datos BD\_LATAM utilizando los formularios de Windows.

Paso 1. Preparar el entorno.

1. Creamos el proyecto Semana13.
2. Agregamos un origen de datos a la base de datos BD\_LATAM, de la misma forma que lo hicimos en la semana 11.
3. Creamos el Modelo a partir de la BD\_LATAM, de la misma forma como se hizo en la semana 12.

Paso 2. Diseñar el formulario.

1. Diseñamos el formulario de acuerdo a:



|  |  |  |
| --- | --- | --- |
| Control | Propiedad | Valor |
| Form1 | (Name) | frmPasajeros |
|  | Text | Mantenimiento de Pasajeros |
| DataGridView1 | (Name) | dgvPasajeros |
| TextBox1 | (Name) | txtId |
|  | ReadOnly | True |
| TextBox2 | (Name) | txtNombres |
| TextBox3 | (Name) | txtApellidoPaterno |
| TextBox4 | (Name) | txtApellidoMaterno |
| TextBox5 | (Name) | txtDni |
| CheckBox1 | (Name) | chkActivo |
| Button1 | (Name) | btnNuevo |
|  | Text | Nuevo |
| Button2 | (Name) | btnEliminar |
|  | Text | Eliminar |
| Button3 | (Name) | btnGuardar |
|  | Text | Guardar |

Paso 3. Mostrar Datos de la tabla Pasajero de la BD.

1. 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

orderby n.Nombres

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:

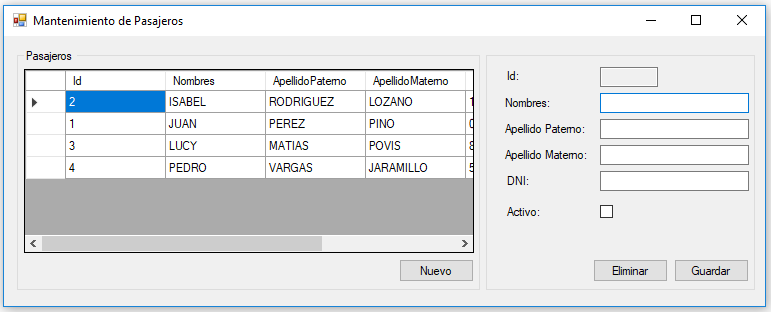
private void frmPasajeros\_Load(object sender, EventArgs e)

{

dgvPasajeros.DataSource = ObtenerPasajeros();

}

1. Ejecutamos la aplicación:



Paso 4. Agregar un registro a la tabla Pasajero de la BD.

1. Primero vamos a crear un método que limpie los controles que contienen los datos del pasajero

private void LimpiarFormulario()

{

txtId.Text = "";

txtNombres.Text = "";

txtApellidoPaterno.Text = "";

txtApellidoMaterno.Text = "";

txtDni.Text = "";

chkActivo.Checked = false;

txtNombres.Focus();

}

1. Agregamos código al evento Click del botón Nuevo de acuerdo a:

private void btnNuevo\_Click(object sender, EventArgs e)

{

LimpiarFormulario();

}

1. Agregamos un método que nos permita validar que todos los datos del formulario han sido ingresados

private bool FormularioValido()

{

bool valido = false;

string nombres = txtNombres.Text.Trim();

string apellidoPaterno = txtApellidoPaterno.Text.Trim();

string apellidoMaterno = txtApellidoMaterno.Text.Trim();

string dni = txtDni.Text.Trim();

if (nombres.Length > 0 && apellidoPaterno.Length > 0 && apellidoMaterno.Length > 0 && dni.Length > 0)

valido = true;

return valido;

}

1. Agregamos 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. Agregamos un método que permita obtener el Id del pasajero recientemente creado:

public PasajeroDto ObtenerUltimoPasajero()

{

try

{

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

{

var consulta = (from n in bd.Pasajero

orderby n.Id descending

select new PasajeroDto

{

Id = n.Id,

Nombres = n.Nombres,

ApellidoPaterno = n.ApellidoPaterno,

ApellidoMaterno = n.ApellidoMaterno,

Dni = n.Dni,

Activo = n.Activo

}).ToList();

PasajeroDto pasajeroDto = consulta.First();

return pasajeroDto;

}

}

catch (Exception)

{

throw;

}

}

1. Agregamos código al evento Click del botón Guardar:

private void btnGuardar\_Click(object sender, EventArgs e)

{

if (FormularioValido())

{

Pasajero pasajero = new Pasajero();

pasajero.Id = 0;

pasajero.Nombres = txtNombres.Text.Trim();

pasajero.ApellidoPaterno = txtApellidoPaterno.Text.Trim();

pasajero.ApellidoMaterno = txtApellidoMaterno.Text.Trim();

pasajero.Dni = txtDni.Text.Trim();

pasajero.Activo = chkActivo.Checked;

//Modificar Pasajero

if (txtId.Text.Trim().Length > 0)

{

//Código para modificar un Pasajero

}

//Nuevo Pasajero

else

{

InsertarPasajero(pasajero);

txtId.Text = ObtenerUltimoPasajero().Id.ToString();

}

MessageBox.Show("Datos guardados satisfactoriamente", "Ok", MessageBoxButtons.OK, MessageBoxIcon.Information);

dgvPasajeros.DataSource = ObtenerPasajeros();

}

else

{

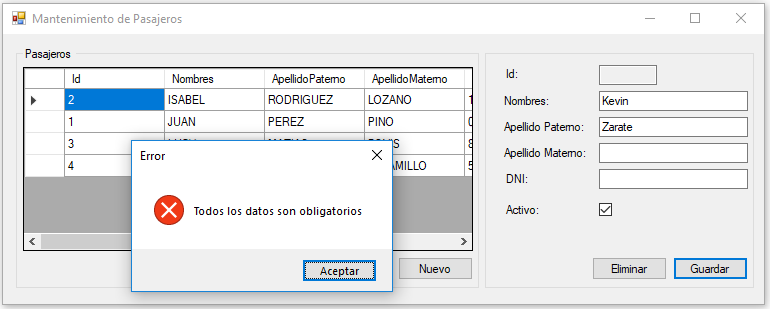
MessageBox.Show("Todos los datos son obligatorios", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

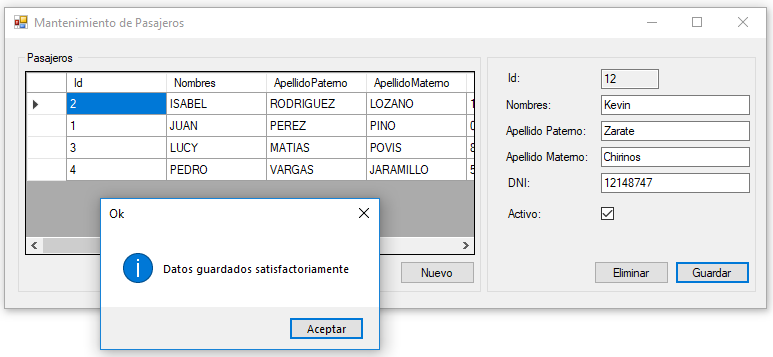
txtNombres.Focus();

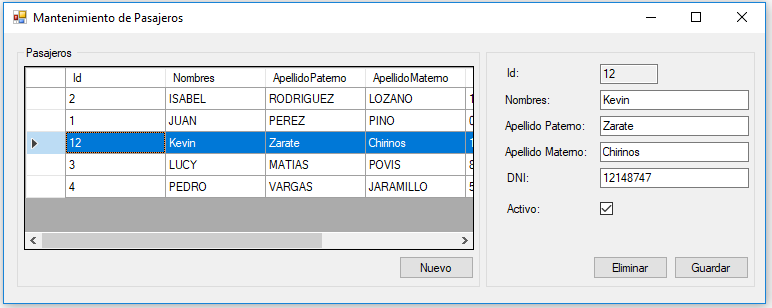
}

}

1. Ejecutamos la aplicación y verificamos la funcionalidad de insertar un nuevo pasajero.







Paso 5. Modificar los datos de un registro en la tabla Pasajero de la BD.

1. Antes de modificar los datos de un pasajero debemos seleccionarlo del DataGridView y mostrar sus datos en el formulario, para ello agregamos código al evento CellClick del DataGridView:

private void dgvPasajeros\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if(e.RowIndex > -1 && e.ColumnIndex > -1 && dgvPasajeros.Rows[e.RowIndex].Cells[e.ColumnIndex].Value != null)

{

dgvPasajeros.CurrentRow.Selected = true;

txtId.Text = dgvPasajeros.Rows[e.RowIndex].Cells["Id"].FormattedValue.ToString();

txtNombres.Text = dgvPasajeros.Rows[e.RowIndex].Cells["Nombres"].FormattedValue.ToString();

txtApellidoPaterno.Text = dgvPasajeros.Rows[e.RowIndex].Cells["ApellidoPaterno"].FormattedValue.ToString();

txtApellidoMaterno.Text = dgvPasajeros.Rows[e.RowIndex].Cells["ApellidoMaterno"].FormattedValue.ToString();

txtDni.Text = dgvPasajeros.Rows[e.RowIndex].Cells["Dni"].FormattedValue.ToString();

chkActivo.Checked = false;

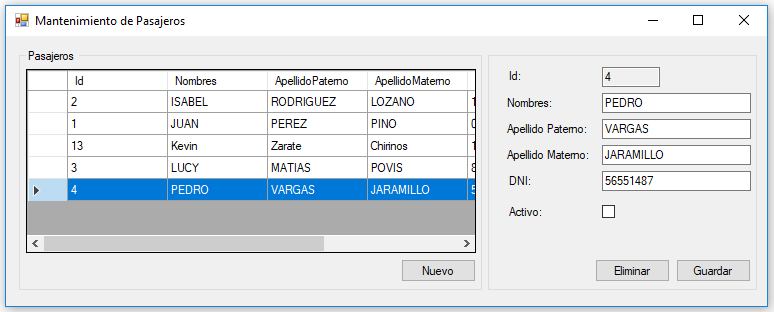
if (dgvPasajeros.Rows[e.RowIndex].Cells["Activo"].FormattedValue.ToString() == "True")

chkActivo.Checked = true;

}

}

1. Ejecutamos la aplicación y validamos la funcionalidad de seleccionar en el DataGridView:



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 Click del Boton Guardar de acuerdo a:

private void btnGuardar\_Click(object sender, EventArgs e)//

{

if (FormularioValido())

{

Pasajero pasajero = new Pasajero();

pasajero.Id = 0;

pasajero.Nombres = txtNombres.Text.Trim();

pasajero.ApellidoPaterno = txtApellidoPaterno.Text.Trim();

pasajero.ApellidoMaterno = txtApellidoMaterno.Text.Trim();

pasajero.Dni = txtDni.Text.Trim();

pasajero.Activo = chkActivo.Checked;

//Modificar Pasajero

if (txtId.Text.Trim().Length > 0)

{

pasajero.Id = Convert.ToInt32(txtId.Text);

ActualizarPasajero(pasajero);

}

//Nuevo Pasajero

else

{

InsertarPasajero(pasajero);

txtId.Text = ObtenerUltimoPasajero().Id.ToString();

}

MessageBox.Show("Datos guardados satisfactoriamente", "Ok", MessageBoxButtons.OK, MessageBoxIcon.Information);

dgvPasajeros.DataSource = ObtenerPasajeros();

}

else

{

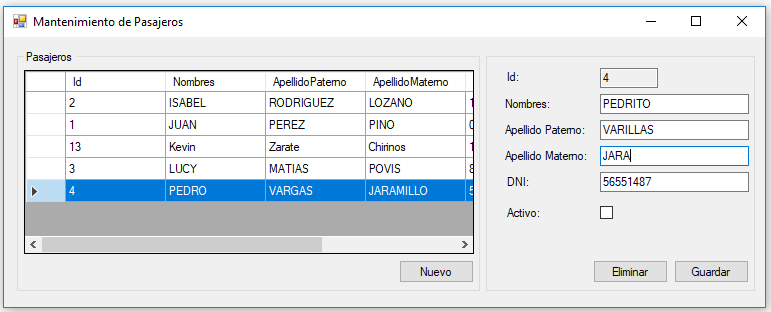
MessageBox.Show("Todos los datos son obligatorios", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

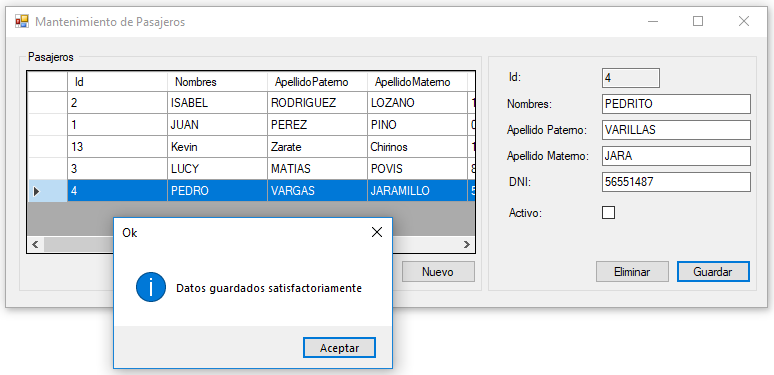
txtNombres.Focus();

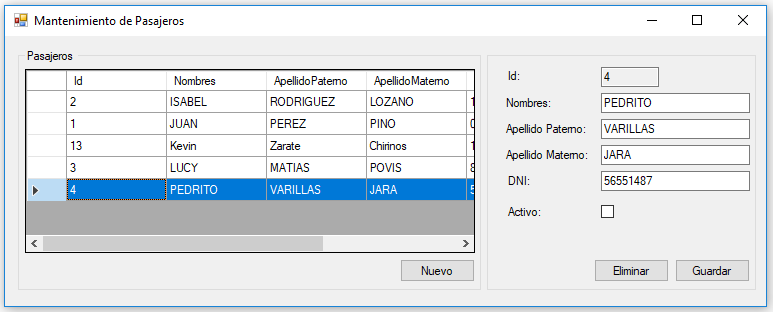
}

}

1. Ejecutamos la aplicación y verificamos que se modifiquen los datos de un pasajero correctamente.







Paso 6. 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. Agregamos código al evento Click el botón Eliminar de acuerdo a:

private void btnEliminar\_Click(object sender, EventArgs e)

{

if(txtId.Text.Trim().Length > 0)

{

DialogResult respuestaAdvertencia = DialogResult.OK;

respuestaAdvertencia = MessageBox.Show("¿Está se guro de eliminar el pasajero?", "Eliminar", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (respuestaAdvertencia == DialogResult.Yes)

{

Pasajero pasajero = new Pasajero();

pasajero.Id = Convert.ToInt32(txtId.Text);

EliminarPasajero(pasajero);

dgvPasajeros.DataSource = ObtenerPasajeros();

LimpiarFormulario();

MessageBox.Show("Registro eliminado satisfactoriamente", "Ok", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else

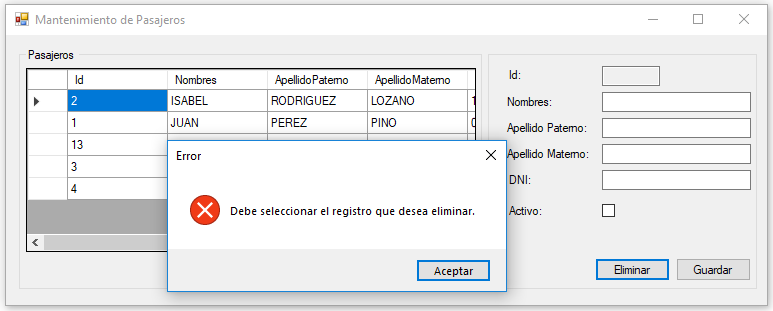
{

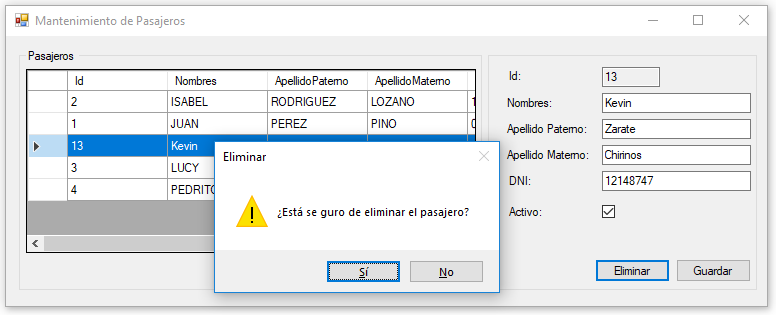
MessageBox.Show("Debe seleccionar el registro que desea eliminar.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

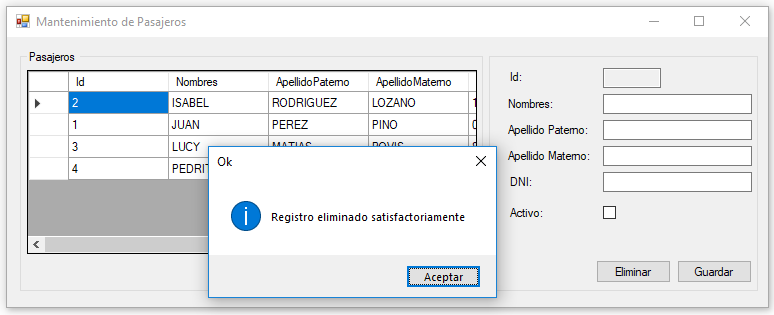
}

}

1. Ejecutamos la aplicación y verificamos que se eliminen pasajeros correctamente.







**Actividad:**

Revisa y analiza los siguientes y desarrolla las actividades propuestas en la plataforma virtual.

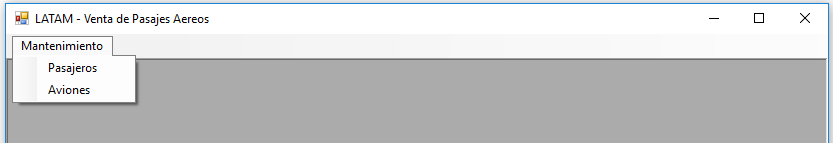
* Control: DataGridView

<https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.listbox?view=netframework-4.8>

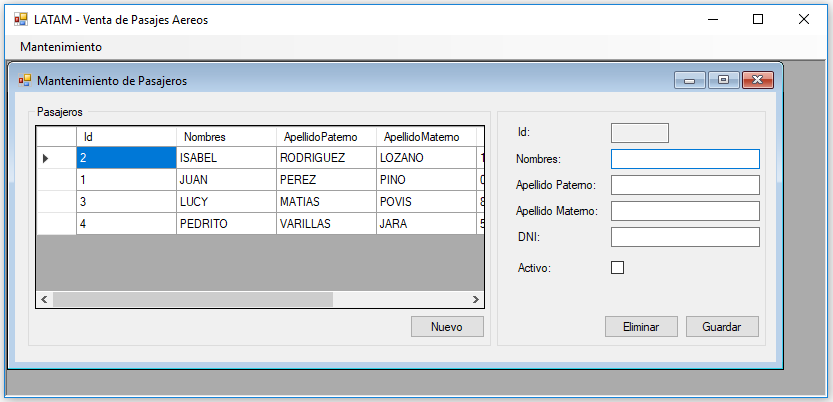
<https://docs.microsoft.com/es-es/dotnet/framework/winforms/controls/datagridview-control-windows-forms>

1. Agregar a la aplicación un formulario MDI que cumpla lo siguiente:

* El formulario MDI debe ser el primero en mostrar cuando se inicia la aplicación.
* Agregar un menú al formulario MDI
* Agregar al menú la opción “Mantenimiento” y dentro de esta opción dos submenús: “Pasajeros” y “Aviones”.



* Si se presiona la opción “Pasajeros” debe mostrar el formulario del mantenimiento de Pasajeros que hemos desarrollado



1. Agregar el formulario que permita el CRUD de la tabla avión, de la misma forma que se hizo con el mantenimiento de Pasajeros.

* Si se presiona la opción “Aviones” del menú debe mostrar el formulario del mantenimiento de Aviones.

