Herramientas de Programación 1

Tema Nº11-B:

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-B:**

ACTUALIZAR EL MODELO DESDE LA BASE DE DATOS EN ENTITY FRAMEWORK

A menudo cuando ya nuestra aplicación esta en producción (que ya está siendo utilizado por los usuarios en operaciones reales) se hace necesario agregar u optimizar funcionalidades, esto conlleva en muchos casos a modificar la estructura de la BD. Es en estos casos cuando también debemos actualizar el Modelo de EF de tal forma que reflejen la estructura y comportamientos de la BD.

**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 Actualizar el Modelo.

**Objetivos**

* Agregar al Modelo los nuevos objetos de la BD.

**Introducción:**

* Vamos a trabajar sobre el mismo proyecto de la semana 13.
* Durante la actividad de esta semana vamos a crear una nueva tabla y cuatro procedimientos almacenados en la BD\_LATAM, posterior a ello vamos agregar estos nuevos objetos al Modelo creado por EF.

Paso 1. Creamos una nueva tabla en la BD y cuatro procedimientos almacenados que nos permitan hacer el CRUD de la nueva tabla.

1. Ingresamos a SSMS.
2. Ejecutamos el siguiente script:

USE [BD\_LATAM]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Vendedor] Script Date: 10/09/2019 09:34:09 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Vendedor](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Nombres] [nvarchar](50) NULL,

[ApellidoPaterno] [nvarchar](50) NULL,

[ApellidoMaterno] [nvarchar](50) NULL,

[Dni] [nvarchar](10) NULL,

[Salario] [decimal](10, 2) NULL,

[Activo] [bit] NULL,

CONSTRAINT [PK\_Vendedor] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Pasaje] Script Date: 10/09/2019 09:35:55 \*\*\*\*\*\*/

ALTER TABLE [dbo].[Pasaje] ADD [IdVendedor] INT NULL;

GO

ALTER TABLE [dbo].[Pasaje] WITH CHECK ADD CONSTRAINT [FK\_Pasaje\_Vendedor] FOREIGN KEY([IdVendedor])

REFERENCES [dbo].[Vendedor] ([Id])

GO

ALTER TABLE [dbo].[Pasaje] CHECK CONSTRAINT [FK\_Pasaje\_Vendedor]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Vendedor] Script Date: 10/09/2019 09:42:12 \*\*\*\*\*\*/

SET IDENTITY\_INSERT [dbo].[Vendedor] ON

GO

INSERT [dbo].[Vendedor] ([Id], [Nombres], [ApellidoPaterno], [ApellidoMaterno], [Dni], [Salario], [Activo]) VALUES (1, N'SILVIA', N'GAMARRA', N'QU', N'22557711', CAST(1800.00 AS Decimal(10, 2)), 1)

GO

INSERT [dbo].[Vendedor] ([Id], [Nombres], [ApellidoPaterno], [ApellidoMaterno], [Dni], [Salario], [Activo]) VALUES (2, N'DANIELA', N'VILCHEZ', N'ZEGARRA', N'0201577', CAST(3255.55 AS Decimal(10, 2)), 1)

GO

INSERT [dbo].[Vendedor] ([Id], [Nombres], [ApellidoPaterno], [ApellidoMaterno], [Dni], [Salario], [Activo]) VALUES (3, N'ROMINA', N'ARIAS', N'PINTO', N'36367788', CAST(1920.00 AS Decimal(10, 2)), 1)

GO

SET IDENTITY\_INSERT [dbo].[Vendedor] OFF

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPLATAM\_SEL\_VENDEDOR] Script Date: 10/09/2019 11:15:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[SPLATAM\_SEL\_VENDEDOR]

@iVendedorId int = null,

@iNombre nvarchar(50) = null

AS

BEGIN

SELECT Id

,Nombres

,ApellidoPaterno

,ApellidoMaterno

,Dni

,Salario

,Activo

FROM Vendedor

WHERE Id = IsNull(@iVendedorId, Id)

AND (@iNombre IS NULL OR Nombres + ApellidoPaterno + ApellidoMaterno LIKE '%'+ @iNombre +'%')

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPLATAM\_INS\_VENDEDOR] Script Date: 10/09/2019 13:53:58 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROC [dbo].[SPLATAM\_INS\_VENDEDOR]

@iNombres nvarchar(50) = NULL,

@iApellidoPaterno nvarchar(50) = NULL,

@iApellidoMaterno nvarchar(50) = NULL,

@iDni nvarchar(10) = NULL,

@iSalario decimal(10, 2) = NULL,

@iActivo bit = NULL,

@oIdVendedor int output

AS

SET NOCOUNT ON

INSERT INTO [dbo].[Vendedor]

([Nombres]

,[ApellidoPaterno]

,[ApellidoMaterno]

,[Dni]

,[Salario]

,[Activo])

VALUES (@iNombres

,@iApellidoPaterno

,@iApellidoMaterno

,@iDni

,@iSalario

,@iActivo)

SELECT @oIdVendedor = SCOPE\_IDENTITY()

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPLATAM\_UPD\_VENDEDOR] Script Date: 10/09/2019 15:58:26 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROC [dbo].[SPLATAM\_UPD\_VENDEDOR]

@iIdVendedor int = NULL,

@iNombres nvarchar(50) = NULL,

@iApellidoPaterno nvarchar(50) = NULL,

@iApellidoMaterno nvarchar(50) = NULL,

@iDni nvarchar(10) = NULL,

@iSalario decimal(10, 2) = NULL,

@iActivo bit = NULL

AS

SET NOCOUNT ON

UPDATE [dbo].[Vendedor]

SET [Nombres] = @iNombres

,[ApellidoPaterno] = @iApellidoPaterno

,[ApellidoMaterno] = @iApellidoMaterno

,[Dni] = @iDni

,[Salario] = @iSalario

,[Activo] = @iActivo

WHERE [Id] = @iIdVendedor

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPLATAM\_DEL\_VENDEDOR] Script Date: 10/09/2019 16:03:07 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROC [dbo].[SPLATAM\_DEL\_VENDEDOR]

@iIdVendedor int

AS

SET NOCOUNT ON

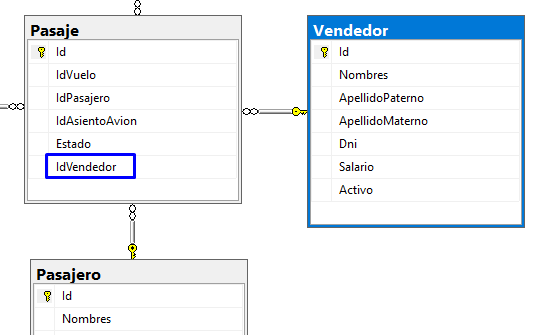
DELETE

FROM [dbo].[Vendedor]

WHERE [Id] = @iIdVendedor

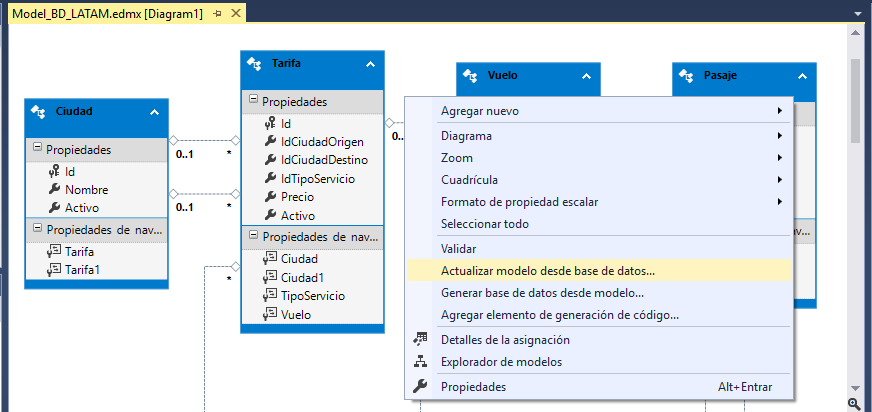
GO

1. Validamos los cambios en la BD\_LATAM

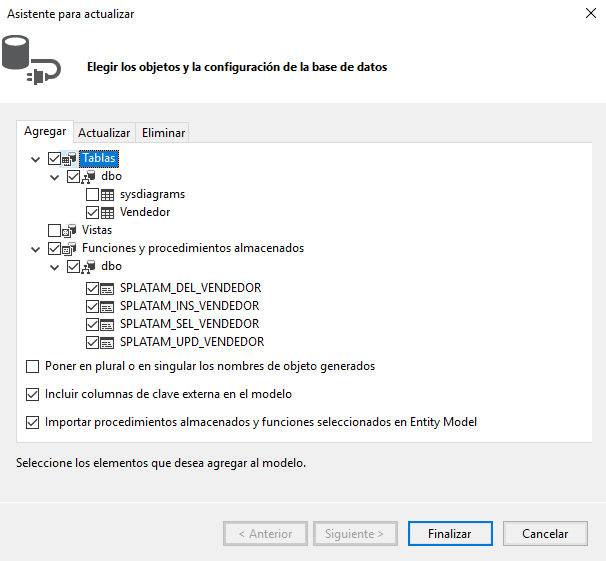


Paso 2. Actualizamos el Modelo.

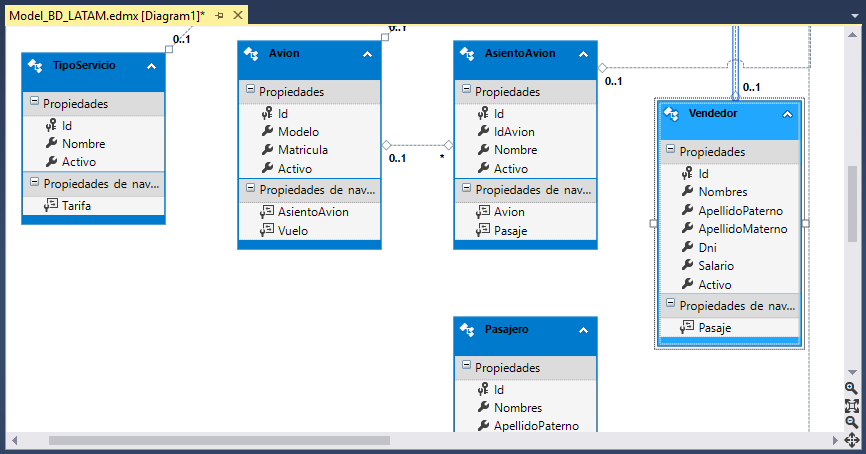
1. Abrimos el proyecto de la Semana13.
2. Nos ubicamos en el Modelo, hacemos click derecho y seleccionamos la opción “Actualizar modelo desde base de datos…”



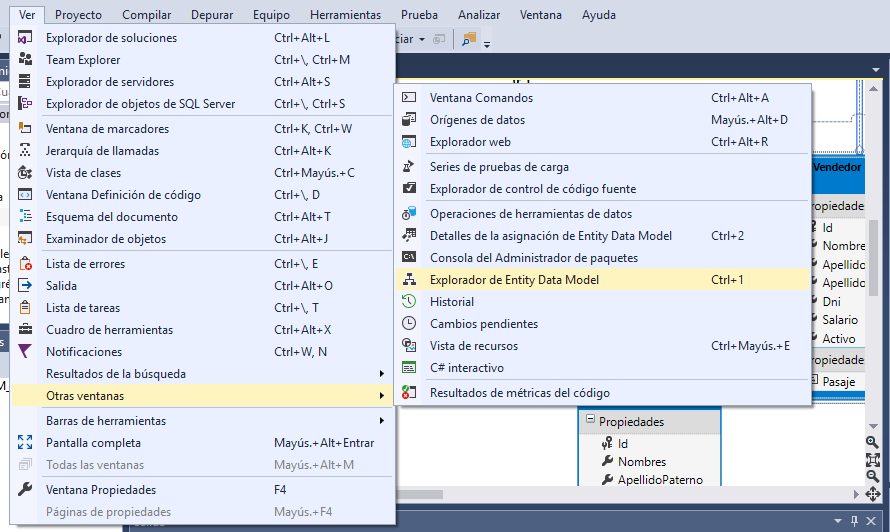
1. Seleccionamos los nuevos objetos y presionamos en el botón “Finalizar”

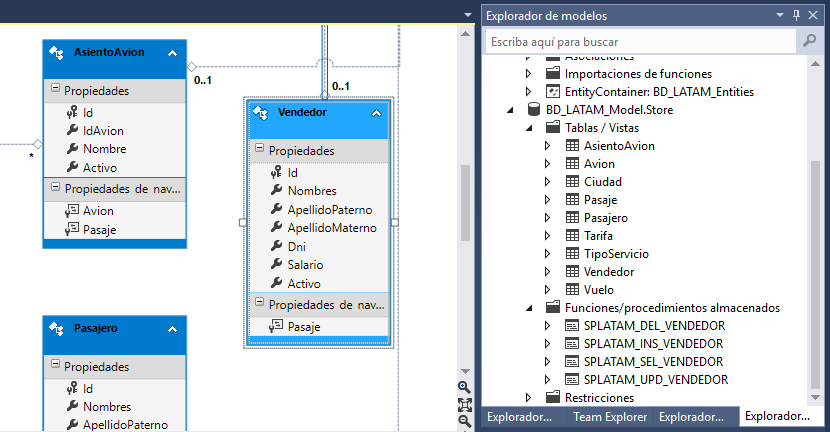


1. Una vez actualizado el Modelo ya podemos ver la nueva tabla

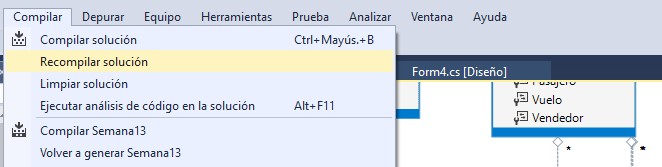


1. Para visualizar los procedimientos almacenados importados al Modelo abrimos el explorador de Modelos seleccionando la opción:

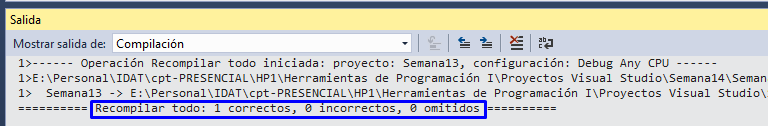




1. Es recomendable, cada vez que actualizamos el Modelo, recompilar la aplicación, para ello seleccionamos la opción:



1. Verificamos que no se muestra errores al recompilar la aplicación:



11.2 CRUD de la tabla Vendedor.

Objetivos

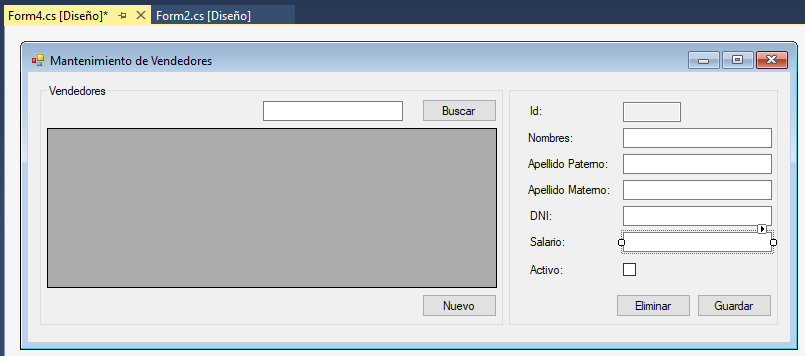
* Realizar el mantenimiento de la tabla Vendedor utilizando procedimientos almacenados con EF.

Introducción:

* Durante esta actividad, vamos a realizar el mantenimiento (CRUD: Create, Read, Update, Delete) de la nueva tabla Vendedor de la base de datos BD\_LATAM utilizando procedimientos almacenados con EF.

Paso 1. Diseñar el formulario.

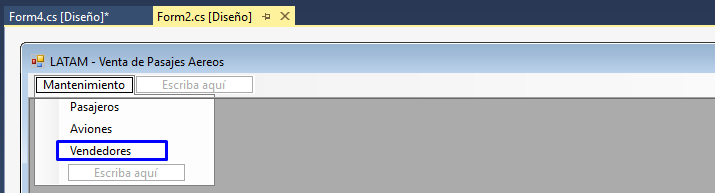
1. Diseñamos el formulario de acuerdo a:



|  |  |  |
| --- | --- | --- |
| Control | Propiedad | Valor |
| Form4 | (Name) | frmPasajeros |
|  | Text | Mantenimiento de Pasajeros |
| DataGridView1 | (Name) | dgvVendedores |
| TextBox1 | (Name) | txtId |
|  | ReadOnly | True |
| TextBox2 | (Name) | txtNombres |
| TextBox3 | (Name) | txtApellidoPaterno |
| TextBox4 | (Name) | txtApellidoMaterno |
| TextBox5 | (Name) | txtDni |
| TextBox6 | (Name) | txtSalario |
| CheckBox1 | (Name) | chkActivo |
| Button1 | (Name) | btnNuevo |
|  | Text | Nuevo |
| Button2 | (Name) | btnEliminar |
|  | Text | Eliminar |
| Button3 | (Name) | btnGuardar |
|  | Text | Guardar |
| Button4 | (Name) | btnBuscar |
|  | Text | Buscar |
| TextBox7 | (Name) | txtBuscar |

Paso 2. Ejecutar el nuevo formulario desde la opción de menú: Mantenimiento / Vendedores.

1. En diseño agregar la opción de menú: Mantenimiento / Vendedores



1. Agregar código al evento Click de la opción de menú de acuerdo a:

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

{

if (miFormularioDeVendedores == null)

{

miFormularioDeVendedores = new frmVendedores();

miFormularioDeVendedores.MdiParent = this;

miFormularioDeVendedores.FormClosed += new FormClosedEventHandler(CerrarFormularioDeVendedores);

miFormularioDeVendedores.Show();

}

else

{

miFormularioDeVendedores.Activate();

}

}

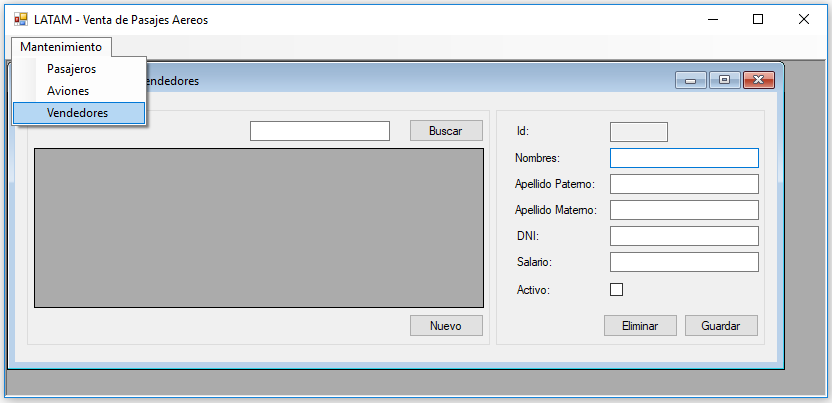
void CerrarFormularioDeVendedores(object sender, FormClosedEventArgs e)

{

miFormularioDeVendedores = null;

}

1. Ejecutar la aplicación y mostrar el nuevo formulario



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

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<SPLATAM\_SEL\_VENDEDOR\_Result> ObtenerVendedores(string pNombre)//

{

try

{

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

{

var consulta = bd.SPLATAM\_SEL\_VENDEDOR(null, pNombre).ToList();

return consulta;

}

}

catch (Exception)

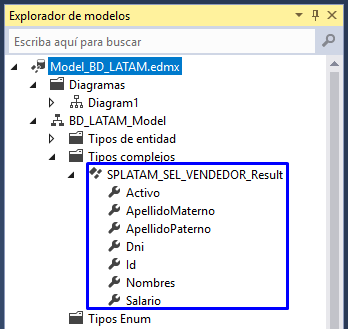
{

throw;

}

}

Debemos notar que esta función nos devuelve una lista de elementos de tipo SPLATAM\_SEL\_VENDEDOR\_Result. Cuando se genera el Modelo, por cada procedimiento almacenado que devuelve registros, EF genera una clase con los mismos campos que devuelve el procedimiento almacenado, esto lo podemos verificar revisando en el Explorador de modelos:



Esta función recibe un parámetro, es el que nos va a permitir realizar la búsqueda de los Vendedores. Si queremos que nos devuelva todos los vendedores de la BD debemos enviarle una cadena de texto vacía.

1. Agregamos código al evento Load del formulario:

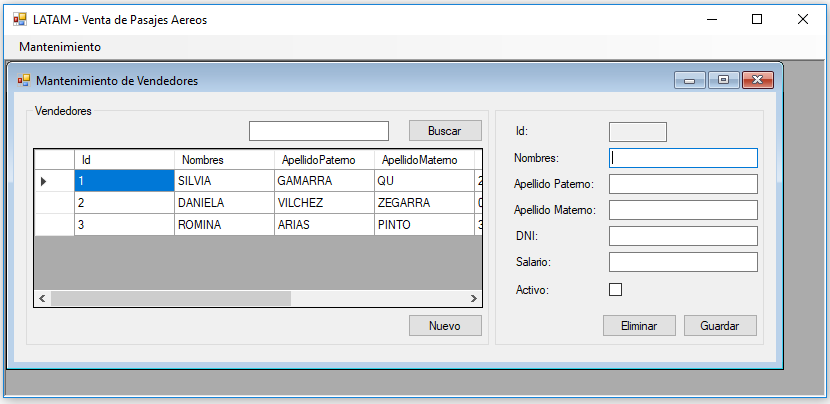
private void frmVendedores\_Load(object sender, EventArgs e)//

{

dgvVendedores.DataSource = ObtenerVendedores("");

}

1. Ejecutamos la aplicación:



Paso 4. Agregar un filtro que permita hacer búsquedas en la lista de Vendedores.

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

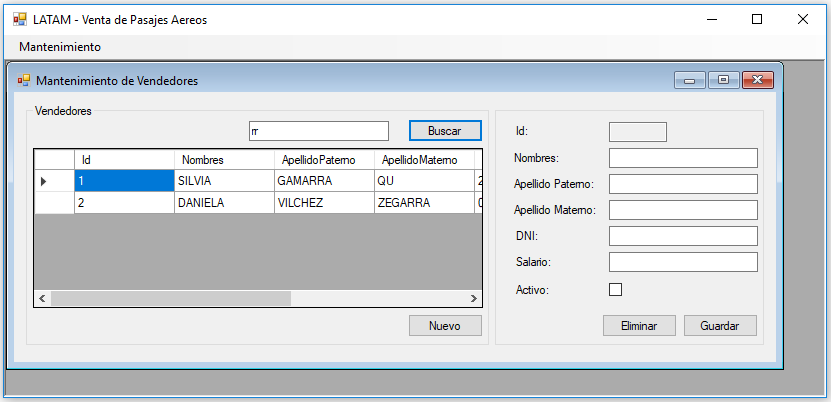
private void btnBuscar\_Click(object sender, EventArgs e)

{

dgvVendedores.DataSource = ObtenerVendedores(txtBuscar.Text.Trim());

}

1. Ejecutamos la aplicación y validamos la búsqueda:



Paso 5. Agregar un registro a la tabla Vendedor de la BD.

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

private void LimpiarFormulario()

{

txtId.Text = "";

txtNombres.Text = "";

txtApellidoPaterno.Text = "";

txtApellidoMaterno.Text = "";

txtDni.Text = "";

txtSalario.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 dos métodos que nos permita validar los datos del formulario

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();

string salario = txtSalario.Text.Trim();

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

dni.Length > 0 && salario.Length > 0)

{

if (EsNumero(salario))

{

valido = true;

}

else

{

MessageBox.Show("El Salario debe ser un número.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtSalario.Text = "";

txtSalario.Focus();

}

}

else

{

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

txtNombres.Focus();

}

return valido;

}

private bool EsNumero(string pNumero)

{

bool esNumero = true;

try

{

decimal numero = Convert.ToDecimal(pNumero);

return esNumero;

}

catch (Exception)

{

esNumero = false;

return esNumero;

}

}

1. Agregamos un método que nos permita insertar un nuevo vendedor a la tabla Vendedor de la BD

public int InsertarVendedor(string pNombres, string pApellidoPaterno, string pApellidoMaterno,

string pDni, decimal pSalario, bool pActivo)

{

try

{

System.Data.Entity.Core.Objects.ObjectParameter idVendedorNuevo = new System.Data.Entity.Core.Objects.ObjectParameter("oIdVendedor", typeof(int));

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

{

bd.SPLATAM\_INS\_VENDEDOR(pNombres, pApellidoPaterno, pApellidoMaterno,

pDni, pSalario, pActivo, idVendedorNuevo);

}

return Convert.ToInt32(idVendedorNuevo.Value.ToString());

}

catch (Exception e)

{

throw;

//MessageBox.Show(e.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

Debemos notar que esta función, además de insertar un nuevo Vendedor a la BD, nos devuelve el Id del vendedor recientemente ingresado.

Recordar que el procedimiento almacenado este id es un parámetro de salida.

1. Agregamos código al evento Click del botón Guardar:

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

{

if (FormularioValido())

{

int id = 0;

string nombres = txtNombres.Text.Trim();

string apellidoPaterno = txtApellidoPaterno.Text.Trim();

string apellidoMaterno = txtApellidoMaterno.Text.Trim();

string dni = txtDni.Text.Trim();

string salario = txtSalario.Text.Trim();

bool activo = chkActivo.Checked;

//Modificar Vendedor

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

{

//Código para modificar un Vendedor

}

//Nuevo Vendedor

else

{

id = InsertarVendedor(nombres, apellidoPaterno, apellidoMaterno,

dni, Convert.ToDecimal(salario), activo);

txtId.Text = id.ToString();

}

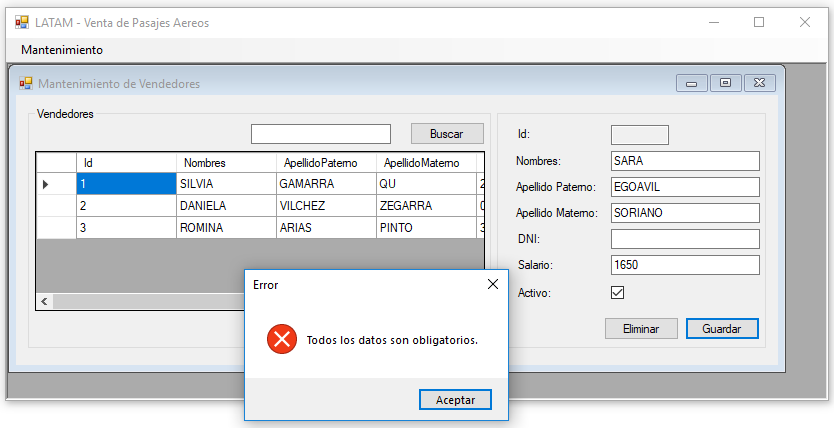
MessageBox.Show("Datos guardados satisfactoriamente", "Ok", MessageBoxButtons.OK, MessageBoxIcon.Information);

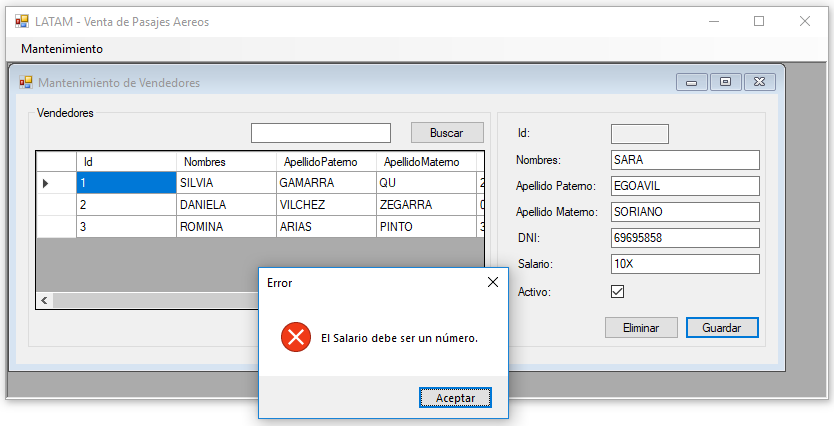
dgvVendedores.DataSource = ObtenerVendedores("");

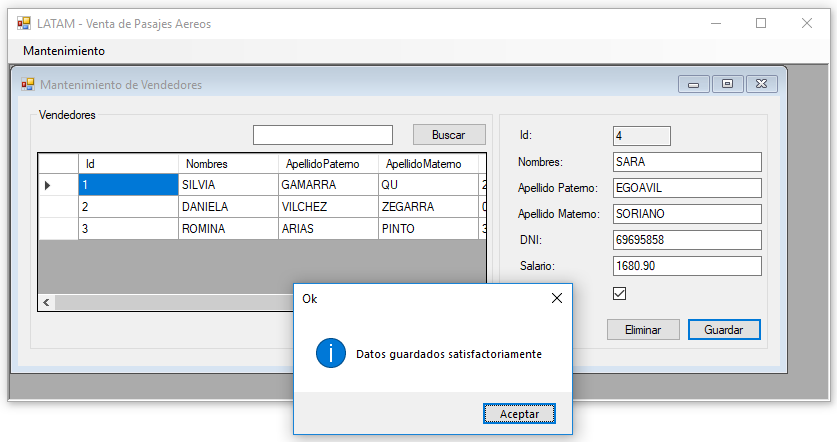
}

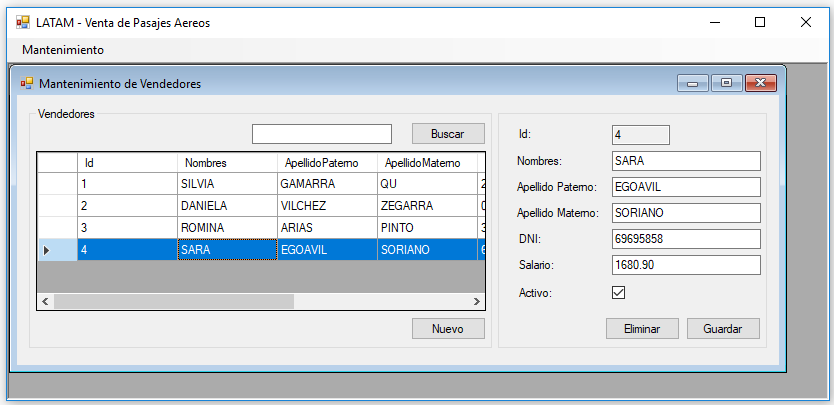
}

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









Paso 6. Modificar los datos de un registro en la tabla Vendedor de la BD.

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

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

{

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

{

dgvVendedores.CurrentRow.Selected = true;

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

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

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

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

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

txtSalario.Text = dgvVendedores.Rows[e.RowIndex].Cells["Salario"].FormattedValue.ToString();

chkActivo.Checked = false;

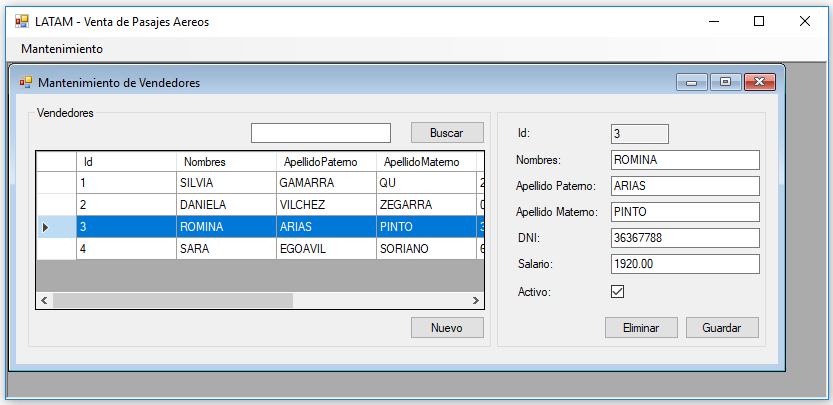
if (dgvVendedores.Rows[e.RowIndex].Cells["Activo"].FormattedValue.ToString() == "True")

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 vendedor en la tabla Vendedor de la BD

public void ActualizarVendedor(int pId, string pNombres, string pApellidoPaterno,

string pApellidoMaterno, string pDni, decimal pSalario, bool pActivo)

{

try

{

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

{

bd.SPLATAM\_UPD\_VENDEDOR(pId, pNombres, pApellidoPaterno, pApellidoMaterno,

pDni, pSalario, pActivo);

}

}

catch (Exception)

{

throw;

}

}

1. Modificamos el código del evento Click del botón Guardar de acuerdo a:

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

{

if (FormularioValido())

{

int id = 0;

string nombres = txtNombres.Text.Trim();

string apellidoPaterno = txtApellidoPaterno.Text.Trim();

string apellidoMaterno = txtApellidoMaterno.Text.Trim();

string dni = txtDni.Text.Trim();

string salario = txtSalario.Text.Trim();

bool activo = chkActivo.Checked;

//Modificar Vendedor

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

{

id = Convert.ToInt32(txtId.Text);

ActualizarVendedor(id, nombres, apellidoPaterno, apellidoMaterno,

dni, Convert.ToDecimal(salario), activo);

}

//Nuevo Vendedor

else

{

id = InsertarVendedor(nombres, apellidoPaterno, apellidoMaterno,

dni, Convert.ToDecimal(salario), activo);

txtId.Text = id.ToString();

}

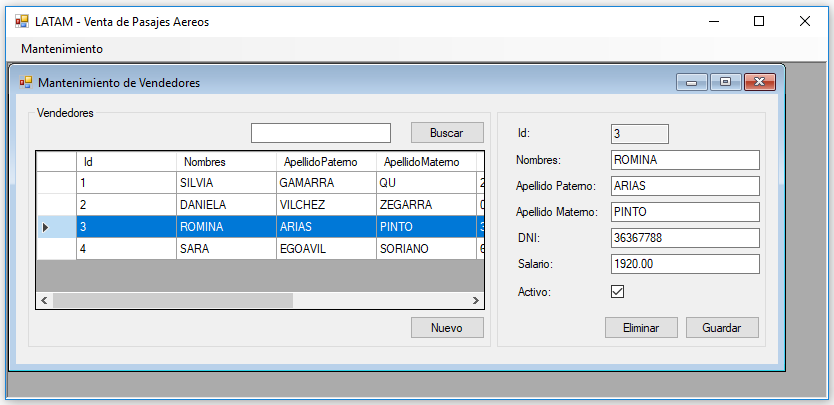
MessageBox.Show("Datos guardados satisfactoriamente", "Ok", MessageBoxButtons.OK, MessageBoxIcon.Information);

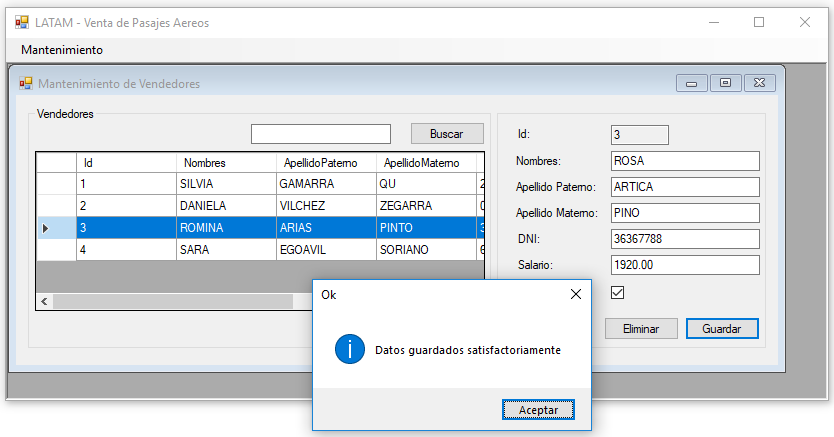
dgvVendedores.DataSource = ObtenerVendedores("");

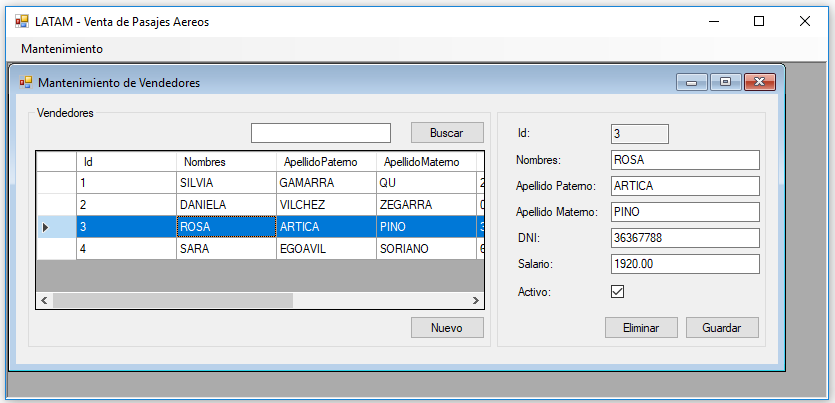
}

}

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







Paso 7. Eliminar un registro en la tabla Vendedor de la BD.

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

public void EliminarVendedor(int pIdVendedor)

{

try

{

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

{

bd.SPLATAM\_DEL\_VENDEDOR(pIdVendedor);

}

}

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 vendedor?", "Eliminar", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (respuestaAdvertencia == DialogResult.Yes)

{

EliminarVendedor(Convert.ToInt32(txtId.Text));

dgvVendedores.DataSource = ObtenerVendedores("");

LimpiarFormulario();

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

}

}

else

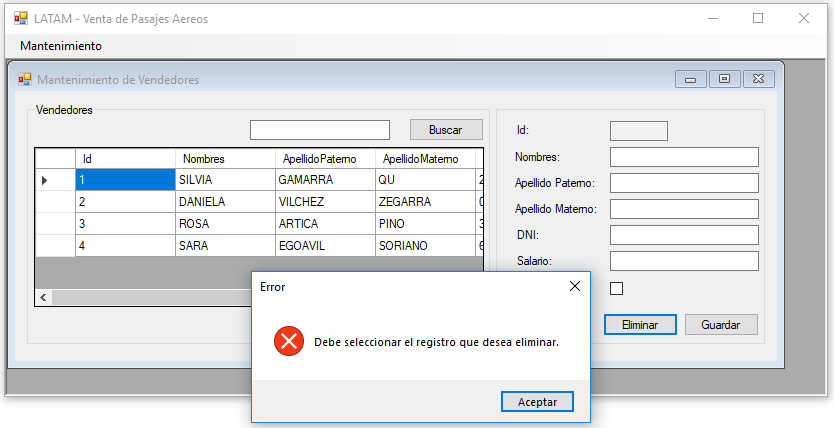
{

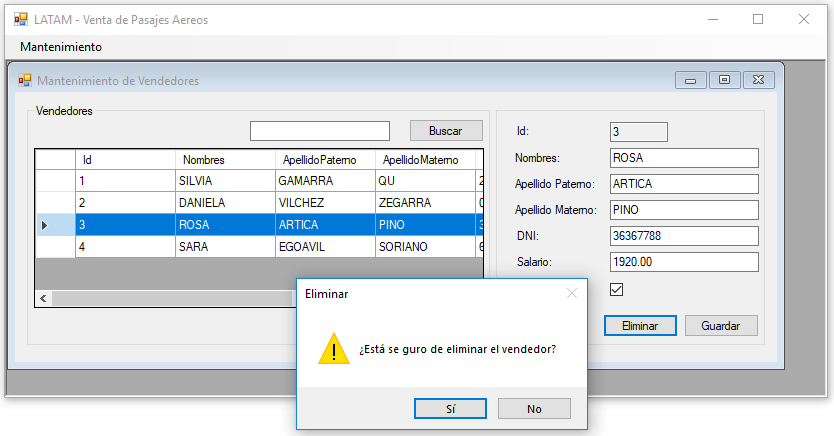
MessageBox.Show("Debe seleccionar el registro que desea eliminar.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

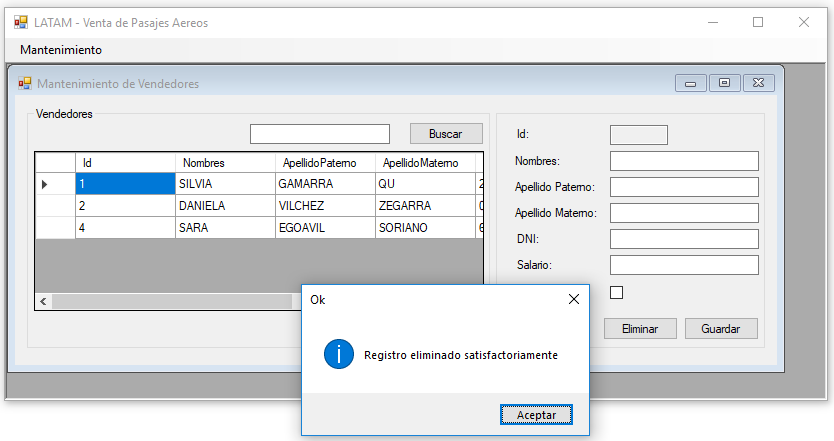
}

}

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







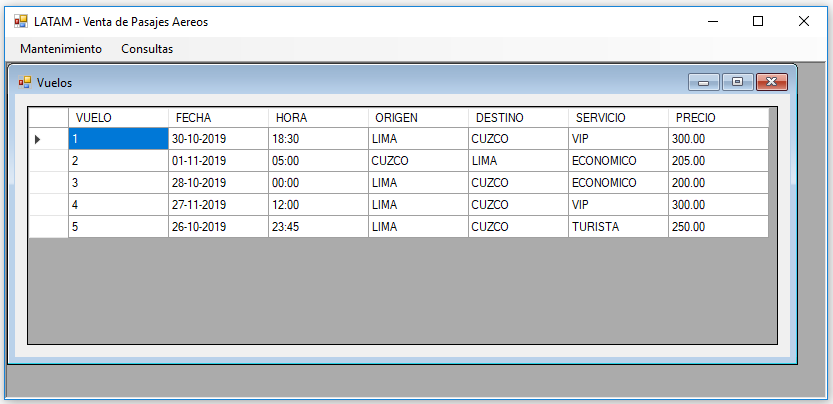
**Actividad:**

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

* Entity Framework. Actualizar el Modelo

<https://www.youtube.com/watch?v=rNROUK5ddFc>

1. Agregar a la aplicación un formulario que permita consultar todos los vuelos que se tiene registrado en la BD\_LATAM de acuerdo al siguiente diseño:



**Requisitos:**

* Crear el procedimiento almacenado que devuelva los datos que se van a mostrar en la consulta con las columnas que se muestran en el diseño.
* Actualizar el Modelo de tal forma que se agregue el nuevo procedimiento almacenado.
* Crear una opción del menú que permita abrir el formulario que muestra la consulta.