	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERIA** **ESCUELA DE COMPUTACIÓN**
CICLO 2-2019	**GUÍA DE LABORATORIO N.º 7**
	Nombre de la practica: Conexión a Base de Datos. **Lugar de ejecución:** Laboratorio de Informática **Materia:** Modelamiento y Diseño de Base de datos

I. Objetivos

1. Crear una interfaz gráfica con Visual C#.NET de manera básica
2. Utilizar las sentencias de DML de base de datos en Visual C#.NET

II. Introducción Teórica

En la mayoría de aplicaciones incluidas en SQL Server, no todo el desarrollo se realiza en el propio servidor. Ésa es la esencia de la informática cliente servidor; el trabajo se distribuye entre un servidor central y clientes distribuidos.

Para poder ver y modificar datos del servidor desde una aplicación cliente, se utiliza una biblioteca de acceso a datos.

A lo largo de los años, Microsoft ha presentado diversas bibliotecas de acceso a datos del cliente que pueden usar los datos de SQL Server y la última tecnología que Microsoft ha desarrollado es ADO.NET.

Distintas versiones de ADO.NET

Se llama ADO.NET a todas las clases, interfaces, enumeradores y delegados que se encuentran dentro de los espacios de nombres System.Data y System.xml del .NET Framework de Microsoft. Por cada versión del .NET Framework existe una versión de ADO.NET; dichas versiones van obteniendo mejoras desarrolladas por Microsoft, muchas de ellas por pedido de la comunidad mundial de desarrolladores. Las versiones liberadas hasta este momento son las siguientes:

Versión	Lanzamiento	Herramienta
.NET Framework 1.1	2002	Visual Studio.NET
.NET Framework 2.0	2003	Visual Studio.NET 2005
.NET Framework 4.0	2010	Visual Studio.NET 2010

Conexión

A simple vista, las conexiones pueden parecer uno de los objetos más sencillos de utilizar en ADO.NET. Sin embargo, tomar las decisiones correctas al momento de su uso puede ser la diferencia entre una aplicación de altas prestaciones y otra de bajas. Existe una serie de premisas fundamentales que servirán para favorecer un rendimiento óptimo, las cuales se verán a lo largo de la guía de laboratorio.

En primer lugar, el objeto Connection es un .NET Data Provider, y nos permite establecer la comunicación física entre nuestra aplicación y la base de datos. El .NET Framework posee la interfaz IDBConnection, la cual es implementada por las clases sqlConnection, oledbConnection y OracleConnection en sus versiones 1.x (1.0 y 1.1), y en el caso de su versión 2.0, es implementada por la clase DBConnection, que sirve como la base de los .NET Data Providers.

Cadena de conexión

Las conexiones establecen todos sus parámetros a través de la propiedad ConnectionString, la cual debe ser establecida de un modo correcto previo a su apertura. Esta propiedad es, en realidad, un conjunto de propiedades concatenados en una sola cadena de texto, en el formato "Propiedad=Valor;".

Muchas de estas propiedades que conforman la cadena son obligatorias, y otras poseen valores por defecto si no se establecen.

Existen algunas propiedades bastantes específicas de cada proveedor de datos, y otras que pueden resultar sinónimos. A continuación, veremos en la siguiente tabla las propiedades más comúnmente utilizadas en la cadena de conexión.

CLAVE	VALOR
Data Source	Es la fuente de datos. En el caso de SQL Server es la dirección IP del servidor al cual nos intentamos conectar.
User ID	En el caso de no utilizar la seguridad integrada, deberá indicarse el nombre de usuario reconocido y mantenido por la base de datos.
Password	Igual que el caso anterior. La clave debe de permanecer en forma de texto plano, por lo cual suele resultar poco seguro una cadena de conexión de este tipo. De todos modos, existen casos en donde no puede prescindirse de este tipo de seguridad, con lo cual queda la opción de encriptar la cadena de conexión por completo para obtener un esquema de seguridad optimo.

Comandos

Ya hemos visto que las conexiones nos brindan el vínculo entre las bases de datos y las aplicaciones, pero para poder establecer y ejecutar instrucciones necesitamos de los comandos, los cuales llamaremos indistintamente como xxxCommand. Una de las propiedades fundamentales de los comandos es CommandText que, como veremos más adelante, puede tener sentencias SQL o nombres de objetos de las

bases de datos. Otra propiedad fundamental es Connection, a la cual le asignaremos una instancia de tipo xxxConnection. Las clases xxxCommand tienen constructores sobrecargados que nos permiten declarar, instancias e inicializar nuestros objetos en una sola línea de forma alternativa.

Primero vamos a examinar algunas propiedades útiles de la clase sqlCommand que incluye la siguiente tabla:

Propiedad	Descripción
CommandText	Esta propiedad de sólo lectura nos permite establecer o recuperar la instrucción T-SQL o el nombre del procedimiento almacenado.
CommandTimeOut	Esta propiedad de lectura y escritura obtiene o establece la cantidad de segundos que debe esperar un intento de ejecución de comando. El comando se frustra cuando pasa este tiempo de espera y se emite una excepción. El valor predeterminado es de 30 segundos.
CommandType	Esta propiedad de lectura y escritura indica la forma en que se debe interpretar la propiedad CommandText. Los valores posibles son *StoreProcedure, TableDirect y Text.*
Connection	Esta propiedad de escritura y lectura obtiene o establece el objeto sqlConnection que debe utilizar este objeto de comando.

Ahora vamos a examinar los diversos métodos **Execute** que se pueden llamar en un objeto **Command.**

Propiedad	Descripción
ExecuteNonQuery	Este método ejecuta el comando y devuelve la cantidad de filas afectadas.
ExecuteReader	Este método ejecuta el comando y devuelve un objeto de la clase sqlDataReader. El lector de datos es un cursor de sólo lectura u sólo hacia delante.
ExecuteRow	Este método ejecuta el comando y devuelve la primera columna de la primera fila en forma de objeto genérico. El resto de filas y columnas se ignoran.
ExecuteXmlReader	Este método ejecuta el comando y devuelve un objeto de la clase XmlReader. Este método nos permite utilizar un comando que devuelve el conjunto de resultados en forma de documento XML.

SQL Native Client OLE DB Provider

El proveedor OLE DB de Microsoft SQL Native Client proporciona una interfaz OLE DB a las bases de datos de Microsoft SQL Server 2008. Este proveedor permite que las consultas distribuidas de SQL Server puedan consultar datos de instancias remotas de SQL Server.

III. Materiales y equipo

- Computadora con SQL Server 2012, 2014, 2016 etc. y Visual Studio C#
- Guía Número 7
- Bases de ejemplo.

IV. Procedimiento

Ejercicio 1. Creación de la base de datos y tabla

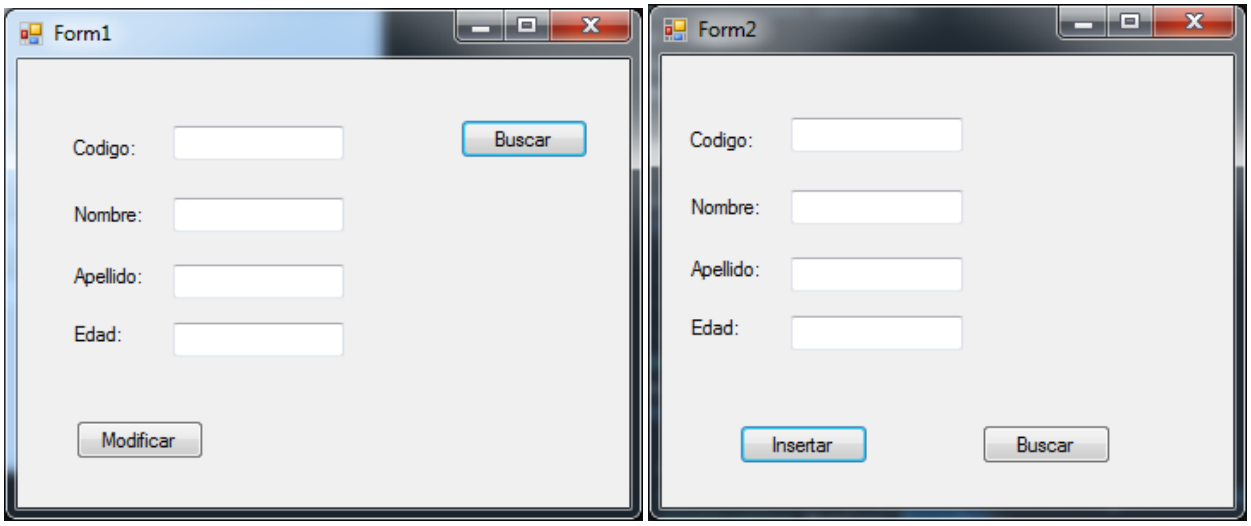
Ingrese a SQL Server y ejecute la siguiente consulta:

```
CREATE DATABASE BD_Sucarnet
GO
```

```
USE BD_Sucarnet
GO
CREATE TABLE Participantes
(Codigo varchar(25) primary key,
Nombres varchar(25),
Apellidos varchar(25),
Edad int)
GO
insert into Participantes values(1,'Alba','Castro',26)
insert into Participantes values(2,'Mario','Abarca',25)
insert into Participantes values(3,'Alberto','Granados',30)
insert into Participantes values(4,'Itzel','Martinez',28)
GO
```

Ejercicio 2. Creación del proyecto en Visual C#

1. Creen un nuevo proyecto en Visual C# con el nombre de Guia7
2. Crear el siguiente los siguientes formularios:



3. En los elementos del Form1 hacer los siguientes cambios:

Elemento	Propiedades	
	Name	Visible
textBox1	textcod1	true
textBox2	textnom1	False
textBox3	textapel1	False
textBox4	textedad1	False
Button1	buscar1	true
Button2	modificar1	false

4. En los elementos del Form2 hacer los siguientes cambios:

Elemento	Propiedades	
	Name	Visible
textBox1	textcod2	true
textBox2	textnom2	true
textBox3	textape12	true
textBox4	textedad2	true
Button1	insertar2	true
Button2	buscar2	true

5. Ubícate en el Explorador de soluciones y dando clic derecho sobre el proyecto Guia7, selecciona la opción agregar y luego selecciona nuevo elemento.
6. En cuadro de dialogo nuevo elemento selecciona Clase y colócale el nombre de **conexion.cs**
7. Digita las siguientes líneas de código como se muestra a continuación:

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

namespace Guia7
{
    0 references
    class conexion
    {
        //Parámetros para la cadena conexión
        public string servidor, usuario, clave, db;
        public string cadena;
        // función que tendrá la cadena de conexión
        0 references
        public void conec()
        {
            servidor = "NombredelServidorSQL";
            db = "BD_Sucarnet";
            usuario = "sa";
            clave = "123456";
            cadena = "server=" + servidor + ";uid=" + usuario + ";pwd=" + clave + ";database=" + db;
        }
    }
}
```

Agregar esta librería a la clase

Agregar el nombre del servidor como se conectó a SQL Server

8. En el código de la aplicación (hacer clic derecho sobre el Form2 y seleccionar la opción Ver código) digitar lo siguiente:

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Guia7
{
    2 references
    public partial class Form2 : Form
    {
        private SqlConnection conn;
        private SqlCommand insert1;
        private string sCn;
        0 references
        public Form2()
        {
            InitializeComponent();
            //usando la clase conexión
            // creo un nuevo objeto de tipo Conexión y lo asigno a cn
            conexion cn = new conexion();
            //acceso a la función conec de la clase conexión
            cn.conec();
            //agrego la variable scn a la cadena conexión
            sCn = cn.cadena;
            //creo la conexión pensándole como argumento la cadena
            conn = new SqlConnection(sCn);
            //abro la conexión
            conn.Open();
        }
    }
}
```

Agregar esta librería a la clase

9. Damos doble clic sobre el botón Insertar del Form2 y digitamos lo siguiente:

```
try
{
    // creo la variable que contendrá la consulta sql de inserción
    string inserparticipante;
    inserparticipante = "INSERT INTO Participantes(Codigo,Nombres,Apellidos,Edad)";
    inserparticipante += "VALUES(@carnet,@nombre,@apellido,@edad)";
    insert1 = new SqlCommand(inserparticipante, conn);
    insert1.Parameters.Add(new SqlParameter("@carnet", SqlDbType.VarChar));
    insert1.Parameters["@carnet"].Value = textcod2.Text;
    insert1.Parameters.Add(new SqlParameter("@nombre", SqlDbType.VarChar));
    insert1.Parameters["@nombre"].Value = textnom2.Text;
    insert1.Parameters.Add(new SqlParameter("@apellido", SqlDbType.VarChar));
    insert1.Parameters["@apellido"].Value = textapel2.Text;
    insert1.Parameters.Add(new SqlParameter("@edad", SqlDbType.Int));
    insert1.Parameters["@edad"].Value = textedad2.Text;
```

```
    insert1.ExecuteNonQuery();
    //Limpiamos los textBox
    textcod2.Text = "";
    textnom2.Text = "";
    textapel2.Text = "";
    textedad2.Text = "";
    MessageBox.Show("Registro agregado");
    conn.Close();
}
catch
{
    MessageBox.Show("Error");
}
```

10. Damos doble clic sobre el botón Buscar del Form2 y digitamos lo siguiente:

```
Form1 formu1 = new Form1();// instanciamos un objeto de tipo
formu1.Show();// mostramos el Form1
this.Hide();// ocultamos el Form2
```

11. Ahora modificaremos el Form1, dar clic derecho sobre este y seleccionar la opción ver código y agregue las librerías que se utiliza para realizar las conexiones a SQL Server, antes de namespace Guia7

```
using System.Data.OleDb;
using System.Data.SqlClient;

namespace Guia7
```

12. Declara las siguientes variables antes del constructor del Form1

```
public partial class Form1 : Form
{
    //Defino una variable de tipo Connection
    private SqlConnection conn1;
    //Defino una variable de tipo DataAdapter
    private SqlDataAdapter da1;
    //Defino una variable de tipo DataReader
    private SqlDataReader dr1;
    //define una variable que contendrá la cadena de conexión
    private string sCn1;
    //instacio un variable OleDbConection
    OleDbConnection cnn = new OleDbConnection();
    2 references
    public Form1()
    {
        InitializeComponent();
    }
}
```

13. Escriba el siguiente código después del InitializeComponent() en el Form1:

```
public Form1()
{
    InitializeComponent();
    //línea de conexión con el servido de base de datos SQL por OLEDB
    cnn.ConnectionString =
    @"PROVIDER=SQLOLEDB;Server=NombredelServidorSQL;Database=BD_Sucarnet;Uid=sa;Pwd=123456";
    // ocultamos el botón modificar
    modificar1.Visible = false;
    //conexión por medio de SQLCLIENT
    conexion cn1 = new
    conexion();
    cn1.conec();
    sCn1 = cn1.cadena;
    conn1 = new SqlConnection(sCn1);
    conn1.Open();
}
```

Agregar el nombre del servidor como se conectó a SQL Server

14. De clic en el botón buscar del Form1 y digite el siguiente código:

```
//Mostramos los textbox ocultos
textedad1.Visible = true;
textnom1.Visible = true;
textapel1.Visible = true;
modificar1.Visible = true;
//variable que tendrá la consulta.
string seleccion;
seleccion = "Select *From Participantes where Codigo= '" + textcod1.Text + "'";
da1 = new SqlDataAdapter(seleccion, conn1);
SqlParameter prm = new SqlParameter("Codigo", SqlDbType.VarChar);
prm.Value = textcod1.Text;
da1.SelectCommand.Parameters.Add(prm); dr1 = da1.SelectCommand.ExecuteReader();
while (dr1.Read())
{
    textnom1.Text = dr1["Nombres"].ToString().Trim();
    textapel1.Text = dr1["Apellidos"].ToString().Trim();
    textedad1.Text = dr1["Edad"].ToString().Trim();
}
if (dr1 != null)
{
    MessageBox.Show("Datos Encontrados");
    dr1.Close();
}
```

15. En el botón **Modificar** del Form1 ingresa el siguiente código:

```
string actualizar;
actualizar = "update Participantes set ";
actualizar += " Nombres= '" + textnom1.Text + "', Apellidos= '" +
textapel1.Text;
```

```
actualizar += "', Edad=" + textedad1.Text + " where Codigo= '" + textcod1.Text + "'";
OleDbCommand datos = new OleDbCommand(actualizar, cnn);
cnn.Open();
//mandando sql a base de datos
datos.ExecuteNonQuery();
cnn.Close();
MessageBox.Show("REGISTRO ACTUALIZADO");
Reset();
```

16. Declara una **función Reset** para limpiar las cajas de texto del formulario (TextBox) deberá crearla afuera del método del botón Modificar:

```
private void Reset()
{
    textcod1.Text = "";
    textnom1.Text = "";
    textapel1.Text = "";
    textedad1.Text = "";
    textedad1.Visible = false;
    textnom1.Visible = false;
    textapel1.Visible = false;
    modificar1.Visible = false;
    Form2 formu2 = new Form2();
    //Mostramos el Form2
    formu2.Show();
    // ocultamos el Form1
    this.Hide();
}
```

17. Por último, abre el archivo **Program.cs** del proyecto y modifica la siguiente línea para que cargue automáticamente el Form2.

```
Application.Run(new Form2());
```

V. Ejercicio Complementario

Configure el botón eliminar que se pueda realizar el mantenimiento.

VI. Análisis de resultados

Crear los mantenimientos (actualización, eliminación, inserción y búsqueda de registros) para las siguientes tablas:

S09-PC26.tarea - d....tarea - dbo.Alumno*

Column Name	Data Type	Allow Nulls
CodigoAlumno	char(8)	☐
PrimerNombre	varchar(20)	☑
SegundoNombre	varchar(20)	☐
PrimerApellido	varchar(20)	☑
SegundoApellido	varchar(20)	☐
Edad	int	☑
Direccion	varchar(100)	☑

S09-PC26.tarea - dbo.Materia

Column Name	Data Type	Allow Nulls
CodigoMateria	char(5)	☐
NombreMateria	varchar(25)	☑
UV	int	☑
Prerequisitos	varchar(150)	☑

VII. Bibliografía

Francisco Charte Ojeda, SQL Server 2008. Madrid, España: ANAYA, 2009 1era edición