GUIA N° 5 - DESARROLLO DE SOFTWARE A TRES CAPAS

OBJETIVO:

- ✓ Desarrollar una aplicación de software a tres capas (Presentación – Lógica – Persistencia) donde se establezca una conexión con una base de datos usando lenguaje C# y ADO.NET

II. CREACIÓN DE UN PROYECTO .NET TRES CAPAS

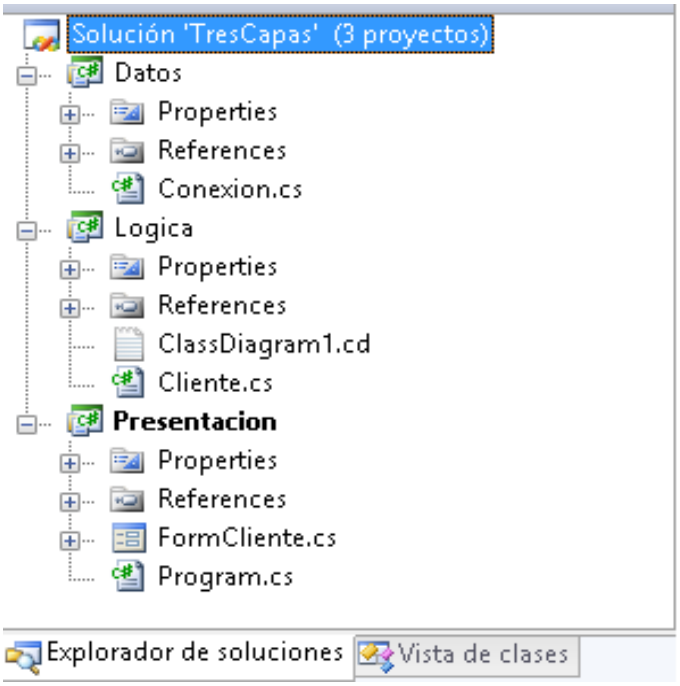
1. En la opción Nuevo proyecto de Visual Studio seleccione la opción: *Otros tipos de proyecto – Solución En Blanco,* establezca como nombre de solución “TresCapas”

2. En el explorador de soluciones haga click derecho sobre el proyecto “TresCapas”, escoja la opción agregar nuevo proyecto, agregue una nueva aplicación para Windows (C#) y nómbrela como “Presentacion”

3. En el explorador de soluciones haga click derecho sobre el proyecto “TresCapas”, escoja la opción agregar nuevo proyecto, agregue una nueva Biblioteca de Clases (C#) y nómbrela como “Logica”.

4. En el explorador de soluciones haga click derecho sobre el proyecto “TresCapas”, escoja la opción agregar nuevo proyecto, agregue una nueva Biblioteca de Clases (C#) y nómbrela como “Datos”.

5. Finalmente el proyecto debe verse similar a como se muestra a continuación:



I. CAPA DE PERSISTENCIA (“BASE DE DATOS“)

Utilizando el motor de bases de datos SQL Server cree una base de datos, colóquele como nombre “Directorio” y en ella cree una tabla con el nombre *“clientes”* con la siguiente distribución de campos:

	Nombre campo	Tipo de dato	Tamaño del Campo
Primary Key	Identificacion	Número	Entero largo
	Nombre	Texto	150
	Apellido	Texto	150
	Fijo	Texto	50
	Celular	Texto	50

A continuación se muestra un script SQL que permiten crear una tabla con las características antes mencionadas en el motor de bases de datos SQL Server:

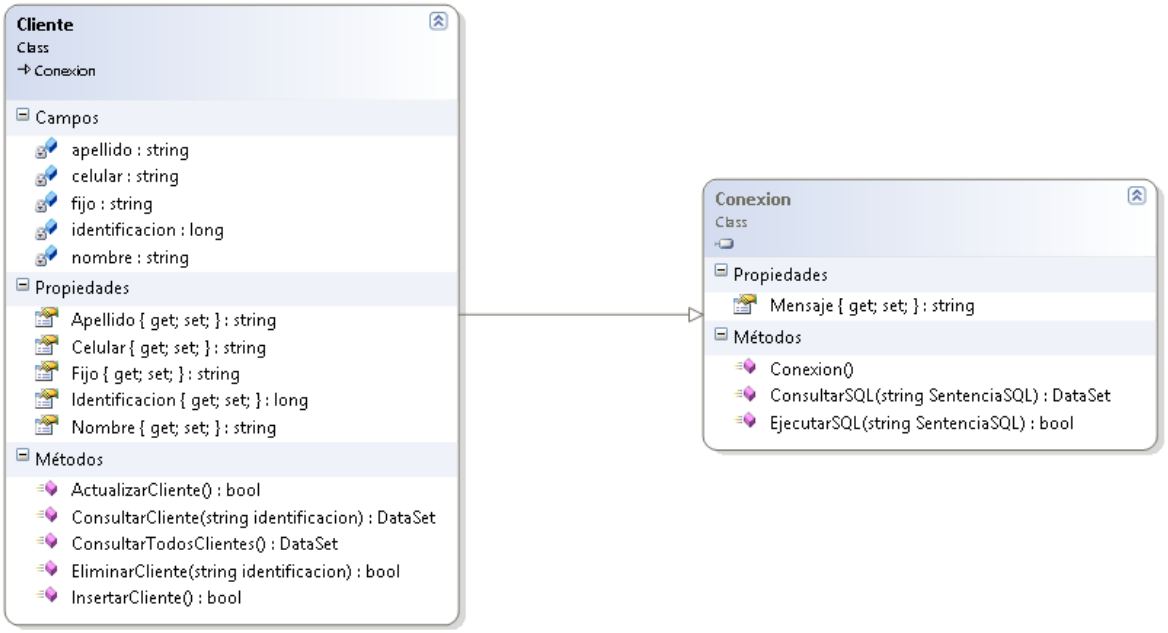
Script SQL (SQL Server)

```
CREATE TABLE Clientes (
  Identificacion Bigint PRIMARY KEY,
  Nombre VARCHAR(150),
  Apellido VARCHAR(150),
  Fijo VARCHAR(50),
  Celular VARCHAR(50));
```

III. CAPA DE LOGICA DEL NEGOCIO (“LÓGICA DE LA APLICACIÓN”):

En esta capa se ubican los componentes que permiten la conexión con el motor de base de datos. El Framework de .Net ofrece una serie de clases predefinidas que hacen parte del componente llamando “ADO.Net”. Este conjunto de clases permite realizar las tareas de conexión y ejecución de sentencias SQL sobre una determinada base de datos.

La capa de lógica de este ejercicio consta de dos clases definidas así:



La clase "Conexion" que en este ejercicio está ubicada en la capa de "Datos" como su nombre lo indica permite el acceso a la base datos por medio de los parámetros de conexión correspondientes, parámetros de conexión que deben ser coherentes con el motor de base de datos seleccionado.

En el ejercicio propuesto en esta guía se debe utilizar el motor SQL Server el cual requiere un determinado formato de cadena de conexión tal y como se muestra a continuación:

Motor de Base de Datos	Formato de Cadena de Conexión
SQL Server 2005/2008/2012/2014 (Autenticación Windows)	`Data Source = Nombre/IP_DelServidorSQl; Initial Catalog = NombreBaseDeDatos; Integrated Security=True`
SQL Server 2005/2008/2012/2014 (Autenticación SQL Server)	`server= Nombre/IP_DelServidorSQl; database = NombreBaseDeDatos; uid = sa; pwd = passwordDeusuario"`

1. CODIGO CLASE CONEXIÓN:

Desde el explorador de soluciones de Visual Studio .Net, en la parte de la solución llamada "Datos" agregue una nueva Clase y llámela "Conexion". Ubique la clase para escribir dentro las instrucciones según se muestran a continuación:

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

namespace Datos
{
    public class Conexion
    {

        private string mensaje;
        private SqlConnection conn;


        public string Mensaje
        {   get { return mensaje; }
            set { mensaje = value; }
        }


        public Conexion()
         {

            //Cadena de Conexion Para SQL Server:
           String cadenaconexion =
          "Data Source= Nombre/IP_ServidorSQL;Initial Catalog = NombreBaseDeDatos;
           Integrated Security=True"

           conn = new SqlConnection(cadenaconexion);
         }
```

```
 public DataSet ConsultarSQL(String SentenciaSQL)
    {
     try
     {
      conn.Open();
      SqlDataAdapter objRes = new SqlDataAdapter(SentenciaSQL, conn);
      DataSet datos = new DataSet();
      objRes.Fill(datos, "DatosConsultados");
      mensaje = "La consulta de datos fue Exitosa";
      return datos;
     }
     catch (Exception MiExc)
     {
        DataSet sindatos=new DataSet();
        mensaje = "ERROR: " + MiExc.Message;
        return sindatos;          }

    finally
    {
        conn.Close();
    }
 }

public bool EjecutarSQL(String SentenciaSQL)
   {

      try
        {
            conn.Open();
            SqlCommand miComando = new SqlCommand();
            miComando.Connection = conn;
            miComando.CommandText = SentenciaSQL;
            miComando.ExecuteNonQuery();

            mensaje = "Proceso Ejecutado con Exito";
            return true ;
        }
        catch (Exception e)
        {

            mensaje = "Se presento el siguiente error " + e.Message;
            return false;
        }
        finally
        {
            conn.Close();
        }
    }
  }
}
```

2. CODIGO CLASE CLIENTE:

1. ***IMPORTANTE:*** Desde el explorador de soluciones en la parte de la solución llamada “Logica” y dentro del submenú llamado “References” escoja la opción agregar referencia. En la ventana Agregar Referencia escoja la pestaña “Proyectos”, seleccione “Datos” en nombre de proyecto y finalmente presione click en el botón aceptar.

2. Desde el explorador de soluciones en la parte de la solución llamada “Logica” agregue una nueva Clase y llámela “Cliente”.

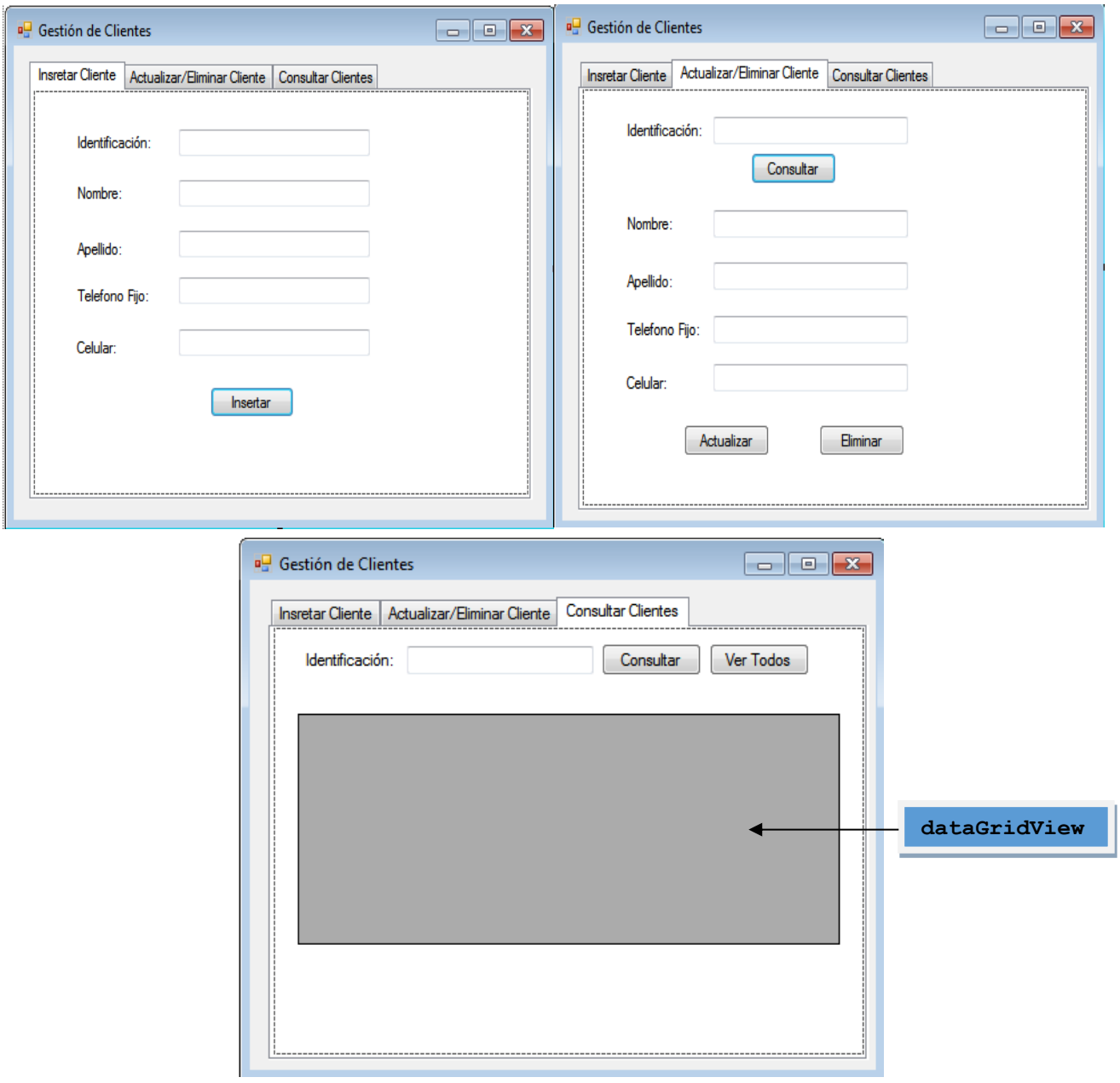
3. Ubique la clase cliente para escribir dentro las instrucciones según se muestran a continuación:

```
using System.Data;

using Datos;

namespace Logica
{

   public class Cliente : Conexion // Herencia desde la clase conexion
    {
        private long identificacion;
        private string nombre;
        private string apellido;
        private string fijo;
        private string celular;

        public long Identificacion
        {
            get { return identificacion;} set{ identificacion = value;}
        }

        public string Nombre
        {
            get { return nombre;} set {nombre = value;}
        }

        public string Apellido
        {
            get { return apellido;} set {apellido = value;}
        }

        public string Fijo
        {
            get { return fijo;} set {fijo = value;}
        }

        public string Celular
        {
            get { return celular;} set {celular = value;}
        }
```

```
public bool InsertarCliente()
  {
      string cadenaSQLInsertar = "INSERT INTO Clientes (identificacion,
      Nombre, Apellido, Fijo, Celular ) VALUES
      (" + this.identificacion + ",'" + this.nombre + "','" +
      this.apellido + "','" + this.fijo + "','" + this.celular +"')";

      bool respuestaSQL =  EjecutarSQL(cadenaSQLInsertar);
      return respuestaSQL;
  }


public DataSet ConsultarCliente(string identificacionxconsultar)
  {
      string cadenaSQLConsultar = "SELECT * FROM Clientes WHERE
      identificacion = " + identificacionxconsultar + "";

      DataSet ConsultaResultante = ConsultarSQL(cadenaSQLConsultar);
      return ConsultaResultante;
  }


public DataSet ConsultarTodosClientes()
  {
      string cadenaSQLConsultar = "SELECT * FROM Clientes";

      DataSet ConsultaResultante = ConsultarSQL(cadenaSQLConsultar);
      return ConsultaResultante;
  }


public bool ActualizarCliente()
  {
      string cadenaSQLActualizar = "UPDATE Clientes SET Nombre = '" +
      this.nombre + "', Apellido = '"+ this.apellido + "',Fijo='"+
      this.fijo + "',Celular='"+ this.celular + "' WHERE (identificacion=
      " + this.identificacion + ")";

       bool respuestaSQL = EjecutarSQL(cadenaSQLActualizar);
       return respuestaSQL;
   }


public bool EliminarCliente(string identificacionxconsultar)
   {
      string cadenaSQLEliminar = "DELETE FROM Clientes WHERE
      identificacion = " + identificacionxconsultar + "";

      bool respuestaSQL = EjecutarSQL(cadenaSQLEliminar);
      return respuestaSQL;
   }
  }
}
```

IV. CAPA DE PRESENTACIÓN ("INTERFAZ GRAFICA DE USUARIO")

La capa de presentación del ejercicio consta de un formulario con tres pestañas distribuidas así:



Nota: Las imágenes corresponden al mismo formulario. Se debe usar un "tabcontrol" que es el control del framework de .Net que permite el uso de pestañas en un formulario tipo Windows

CODIGO PARA LA CAPA DE PRESENTRACIÓN:

1. ***IMPORTANTE:*** Desde el explorador de soluciones en la parte de la solución llamada "Presentacion" y dentro del submenú llamado "References" escoja la opción agregar referencia. En la ventana "Agregar Referencia escoja la pestaña "Proyectos", seleccione "Logica" y "Datos" en nombre de proyecto y finalmente presione click en el botón aceptar.

2. En el Formulario "FormCliente" ubique cada bloque de código (Controles y Eventos) para escribir las instrucciones según se muestra a continuación:

```
using Logica;

namespace Presentacion
{
    public partial class FormCliente : Form
    {
        public FormCliente()
        {
            InitializeComponent();
        }

        // Codigo Boton Guardar de la pestaña 1
        private void button1_Click(object sender, EventArgs e)
        {
            Cliente ObjCliente = new Cliente();
            try
            {
                ObjCliente.Identificacion = long.Parse(textBox1.Text);
                ObjCliente.Nombre = textBox2.Text;
                ObjCliente.Apellido = textBox3.Text;
                ObjCliente.Fijo = textBox4.Text;
                ObjCliente.Celular = textBox5.Text;

                bool respuestaSQL = ObjCliente.InsertarCliente();

                if (respuestaSQL == true)
                {
                  MessageBox.Show("Los datos del nuevo cliente fueron
                  insertados correctamente");
                   textBox1.Text = ""; textBox2.Text = ""; textBox3.Text =
                  ""; textBox4.Text = ""; textBox5.Text = "";
                }
                else
                {
                    MessageBox.Show(ObjCliente.Mensaje);
                }
            }
            catch (Exception Ex)
            {
                  MessageBox.Show("Error!: " + Ex.Message + " " +
                  ObjCliente.Mensaje);
            }
        }
```

```
// Codigo Boton Consultar de la pestaña 2
private void button2_Click(object sender, EventArgs e)
    {

     Cliente ObjCliente = new Cliente();

     try
       {
        DataSet DatosCliente = ObjCliente.ConsultarCliente(textBox6.Text);
        DataTable DatosConsultados = DatosCliente.Tables["DatosConsultados"];

        int numregistros = DatosConsultados.Rows.Count;

        if (numregistros == 0)
         {
             MessageBox.Show("No existe en la Base de Datos Cliente con
             esta identificación");
         }
        else
         {
          textBox7.Text = DatosConsultados.Rows[0]["Nombre"].ToString();
          textBox8.Text = DatosConsultados.Rows[0]["Apellido"].ToString();
          textBox9.Text = DatosConsultados.Rows[0]["Fijo"].ToString();
          textBox10.Text = DatosConsultados.Rows[0]["Celular"].ToString();
         }
      }
      catch (Exception Ex)
        {
             MessageBox.Show("Error!: " + Ex.Message + " " +
             ObjCliente.Mensaje);
        }
  }

// Codigo Boton Actualizar de la pestaña 2
private void button3_Click(object sender, EventArgs e)
    {
         Cliente ObjCliente = new Cliente();

         try
           {
                ObjCliente.Identificacion = int.Parse(textBox6.Text);
                ObjCliente.Nombre = textBox7.Text;
                ObjCliente.Apellido = textBox8.Text;
                ObjCliente.Fijo = textBox9.Text;
                ObjCliente.Celular = textBox10.Text;

                bool respuestaSQL = ObjCliente.ActualizarCliente();

                if (respuestaSQL == true)
                 {
                  MessageBox.Show("Los datos de este cliente fueron
                  actualizados correctamente");
```

```
                    textBox6.Text = ""; textBox7.Text = ""; textBox8.Text =
                    ""; textBox9.Text = ""; textBox10.Text = "";
                   }
                  else
                   {
                      MessageBox.Show(ObjCliente.Mensaje);
                   }
              }
              catch (Exception Ex)
              {
                    MessageBox.Show("Error!: " + Ex.Message + " " +
                    ObjCliente.Mensaje);
              }
    }

// Codigo Boton Eliminar de la pestaña 2
private void button4_Click(object sender, EventArgs e)
    {
       Cliente ObjCliente = new Cliente();

       try
        {
        bool respuestaSQL = ObjCliente.EliminarCliente(textBox6.Text);

        if (respuestaSQL == true)
          {
            MessageBox.Show("Los datos de este cliente fueron
            Eliminados correctamente");
            textBox6.Text = ""; textBox7.Text = ""; textBox8.Text = "";
            textBox9.Text = ""; textBox10.Text = "";
          }
        else
          {
             MessageBox.Show(ObjCliente.Mensaje);
          }
        }

       catch (Exception Ex)
         {
            MessageBox.Show("Error!: " + Ex.Message + " " +
            ObjCliente.Mensaje);
         }
    }

// Codigo Boton Consultar de la pestaña 3
private void button5_Click(object sender, EventArgs e)
  {
   Cliente ObjCliente = new Cliente();

   try
   {
    DataSet DatosCliente = ObjCliente.ConsultarCliente(textBox11.Text);
    DataTable DatosConsultados = DatosCliente.Tables["DatosConsultados"];
```

```
     int numregistros = DatosConsultados.Rows.Count;

     if (numregistros == 0)
        {
             MessageBox.Show("No existe en la Base de Datos Cliente con
             esta identificación");
        }
      else
       {
             dataGridView1.DataSource = DatosConsultados;
       }
      }

    catch (Exception Ex)
      {
      MessageBox.Show("Error!: " + Ex.Message + " " + ObjCliente.Mensaje);
      }
   }

// Codigo Boton Ver Todos de la pestaña 3
private void button6_Click(object sender, EventArgs e)
   {

    Cliente ObjCliente = new Cliente();

    try
    {
     DataSet DatosCliente = ObjCliente.ConsultarTodosClientes();
     DataTable DatosConsultados = DatosCliente.Tables["DatosConsultados"];

     int numregistros = DatosConsultados.Rows.Count;

     if (numregistros == 0)
        {
          MessageBox.Show("No hay ningun Cliente en la Base de Datos");
        }
     else
        {
          dataGridView1.DataSource = DatosConsultados;
        }
      }
       catch (Exception Ex)
         {
             MessageBox.Show("Error!: " + Ex.Message + " " +
             ObjCliente.Mensaje);
         }
       }
     }
}
```

TALLER:

Una institución educativa requiere una aplicación a tres capas para administrar el préstamo de libros en su Biblioteca Escolar. La aplicación a desarrollar debe satisfacer los siguientes requerimientos:

1. Permitir ingresar los datos correspondientes de los estudiantes, los cuales deben ser: Identificación, Nombres, Apellidos, Teléfono, Celular, Fecha de nacimiento, correo electrónico, jornada, Curso, Dirección, Barrio.

2. Permitir Ingresar los datos de cada uno de sus libros (Código, Nombre, Autor**,** Descripción, fecha De publicación, Estado, Identificación (del estudiante cuando esté prestado))

3. Permitir modificar o eliminar la información almacenada de cualquier Estudiante o Libro.

4. Debe tener un módulo de consultas donde se tengan las siguientes opciones de búsqueda

- ✓ Estudiantes: Por Nombre, Por Apellido, Por Jornada, Por Curso, Por Código de Libro (Para saber quién tiene el libro prestado), Por (Jornada y Curso ->Simultáneos)
- ✓ Libros: Por Código, Por Nombre. Por Autor, Por (Nombre y Autor -> Simultáneos)
- ✓ Cuando Un libro este prestado, en la misma pantalla se deben mostrar los datos del libro y del estudiante al que se le prestó

Recomendaciones Importantes:

- ✓ Debe crear mínimo dos tablas que deben estar relacionadas (Estudiantes - Libros)
- ✓ En la biblioteca de este ejercicio a un estudiante se le pueden prestar varios libros al mismo tiempo, pero un libro solo puede ser prestado a un solo estudiante a la vez.
- ✓ Cuando un libro sea prestado, en la tabla que almacena los datos de cada libro debe reflejarse en su campo estado que está prestado o no y en el campo *"Identificacion"* se debe guardar la identificación del estudiante al que se le prestó el libro.