

Fundamentos CSharp

Contenido

Variables	1
Clases y objetos	2
Arreglos y Listas.....	3
Interfaces	5
BBDD	8
Update, delete & insert.....	10
Serializacion.....	12
Servicios Web HTTP	13
Solicitudes PUT, GET, DELETE.....	14
Genericidad.....	16
Linq.....	16
Linq versión 2	17
Tutorial Conexión.....	18

Variables

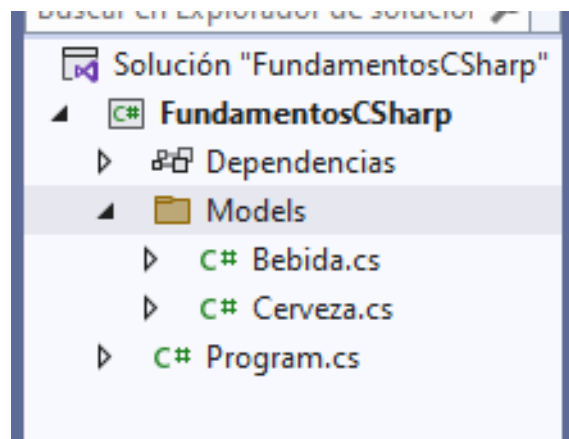
```
1. using System;
2.
3. namespace FundamentosCSharp
4. {
5.     internal class Program
6.     {
7.         static void Main(string[] args)
8.         {
9.             byte numero = 255; // numeros hasta 255
10.            sbyte numero2 = 127; // negativos positivos hasta 127
11.            int numero3 = 1; // negativos/positivos
12.            uint numero4 = 2; // solo positivos
13.            long numero5 = 189;
14.            float numero6 = 189.1f; // 4 bytes
15.            double numero7 = 189.1d; // 8 bytes
16.            decimal numero8 = 189.1m; // 16 bytes
17.            char caracter = 'a'; // caracteres
18.            string cadena = "hola"; // cadena caracteres
19.            DateTime date = DateTime.Now; // Objeto _ fecha
20.            bool siOno = true; // booleano
21.            //int numero9 = new int();
22.            int? numero9 = null; // lo haces nullable
23.
24.            var nombre = "German"; // no es igual a JavaScript
25.            //nombre = 1; // Da error, este var no es como
                JavaScript, este nombre va a ser para siempre un String
26.
27.            var limite = 50; // var de tipo int
```

```

28.
29.         var persona = new { nombre = "German", apellido =
    "Iglesias Ramos" }; // objeto anónimo
30.
31.         Console.WriteLine(numero9.ToString());
32.         Console.WriteLine(persona.nombre);
33.
34.     }
35. }
36. }

```

Clases y objetos



Bebida.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FundamentosCSharp.Models // namespace : grupo
{
    internal class Bebida // internal (desde el mismo dll)
    {
        public string Nombre { get; set; } // public o private o protected
        public int Cantidad { get; set; }

        public Bebida(string Nombre, int Cantidad)
        {
            this.Nombre = Nombre;
            this.Cantidad = Cantidad;
        }

        public void Beberse(int CuantoBebio)
        {
            this.Cantidad -= CuantoBebio;
        }
    }
}

```

Cerveza.cs

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;
/// HERENCIA
namespace FundamentosCSharp.Models
{
    internal class Cerveza: Bebida
    {
        /// <summary>
        /// Constructor base -> hereda del padre
        /// </summary>
        public Cerveza() : base("Cerveza", 500)
        {
        }

        /// <summary>
        /// Constructor con parametros opcionales (Nota: al final siempre)
        /// </summary>
        /// <param name="Cantidad"></param>
        /// <param name="Nombre"></param>
        public Cerveza(int Cantidad=1, string Nombre="Cerveza") :
base(Nombre, Cantidad)
        {
        }
    }
}

```

Program.cs

```

using FundamentosCSharp.Models;
using System;

namespace FundamentosCSharp
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Bebida bebida = new Bebida("Coca Cola", 1000);
            bebida.Berberse(500);
            Console.WriteLine(bebida.Cantidad);

            Cerveza cerveza = new Cerveza(); // creacion de objetos
            cerveza.Berberse(200);
            Console.WriteLine(cerveza.Cantidad);

            Cerveza mahou = new Cerveza(500);
            mahou.Berberse(10);
            Console.WriteLine(mahou.Cantidad);
        }
    }
}

```

Arreglos y Listas

```

using FundamentosCSharp.Models;
using System;
using System.Collections.Generic;

```

```

namespace FundamentosCSharp
{
    internal class Program
    {
        static void Main(string[] args)
        {
            // array -> se le da una dimension, mas rapido que una lista,
            // orientado a una coleccion
            int[] numeros = new int[10] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 }; //
            // coleccion de numero

            int numero = numeros[0]; // acceder al primer elemento

            for (int i = 0; i < numeros.Length; i++) // recorrer 'for'
            {
                Console.WriteLine("iteracion: " + i + " - " + numeros[i]);
            }

            foreach (var n in numeros) // recorrer 'foreach'
            {
                Console.WriteLine(n);
            }

            Console.WriteLine(" ++++++ ");

            // lista -> suele implementar una interfaz, dinamica (longitud
            // varia)
            List<int> lista = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
            };
            lista.Add(1);
            lista.Add(2);
            lista.Add(3);
            lista.Remove(2);

            foreach (var nu in lista)
            {
                Console.WriteLine("elemento: " + nu);
            }

            Console.WriteLine(" ++++++ ");

            Console.ReadLine();

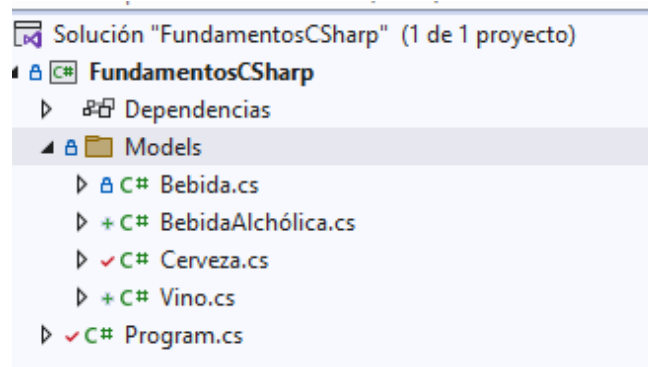
            List<Cerveza> cervezas = new List<Cerveza>() { new Cerveza(10,
            "Cerveza premium") };
            cervezas.Add(new Cerveza(500));
            Cerveza estrella = new Cerveza(200, "Cerveza de trigo");
            cervezas.Add(estrella);

            foreach (var cer in cervezas)
            {
                Console.WriteLine("cerveza: " + cer.Nombre);
            }
            // cola
            Queue<int> cola = new Queue<int>();
            // pila

        }
    }
}

```

Interfaces



Bebida.cs

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

namespace FundamentosCSharp.Models // namespace : grupo
{
    internal class Bebida // internal (desde el mismo dll)
    {
        public string Nombre { get; set; } // public o private o protected
        public int Cantidad { get; set; }

        public Bebida(string Nombre, int Cantidad)
        {
            this.Nombre = Nombre;
            this.Cantidad = Cantidad;
        }
        public void Beberse(int CuantoBebio)
        {
            this.Cantidad -= CuantoBebio;
        }
    }
}
```

IBebidaAlchólica.cs

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

namespace FundamentosCSharp.Models
{
    interface IBebidaAlchólica
    {
        public int Alcohol { get; set; } // CSharp puedes poner atributos

        public void MaxRecomendado(); // metodo
    }
}
```

Vino.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/// HERENCIA
namespace FundamentosCSharp.Models
{
    internal class Vino : Bebida, IBebidaAlchólica
    {
        public int Alcohol { get; set; }
        /// <summary>
        /// Constructor base -> hereda del padre
        /// </summary>
        public Vino() : base("Vino", 500)
        {
        }
        /// <summary>
        /// Constructor con parametros opcionales (Nota: al final siempre)
        /// </summary>
        /// <param name="Cantidad"></param>
        /// <param name="Nombre"></param>
        public Vino(int Cantidad = 1, string Nombre = "Vino") : base(Nombre,
Cantidad)
        {
        }

        public void MaxRecomendado()
        {
            Console.WriteLine("El max permido de un vino es 10");
        }
    }
}
```

Cerveza.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/// HERENCIA
namespace FundamentosCSharp.Models
{
    internal class Cerveza : Bebida, IBebidaAlchólica
    {
        public int Alcohol { get; set; }
        /// <summary>
        /// Constructor base -> hereda del padre
        /// </summary>
        public Cerveza() : base("Cerveza", 500)
        {
        }
        /// <summary>
        /// Constructor con parametros opcionales (Nota: al final siempre)
        /// </summary>
        /// <param name="Cantidad"></param>
```

```

        /// <param name="Nombre"></param>
        public Cerveza(int Cantidad = 1, string Nombre = "Cerveza") :
base(Nombre, Cantidad)
        {

        }

        public void MaxRecomendado()
        {
            Console.WriteLine("El max permido de una cerveza es 10");
        }
    }
}

```

Program.cs

```

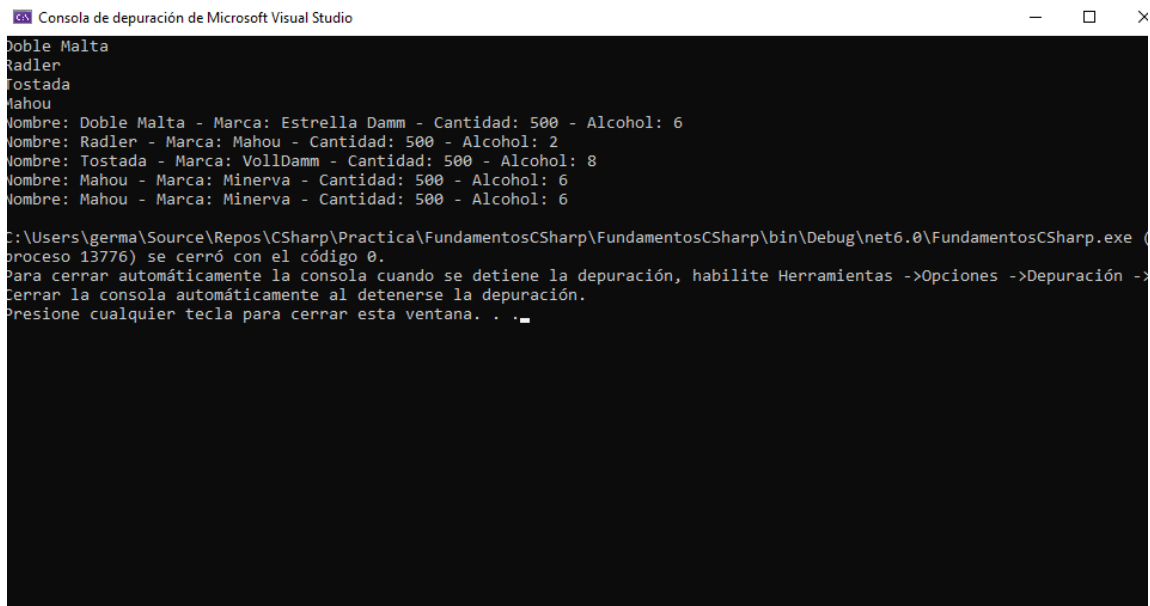
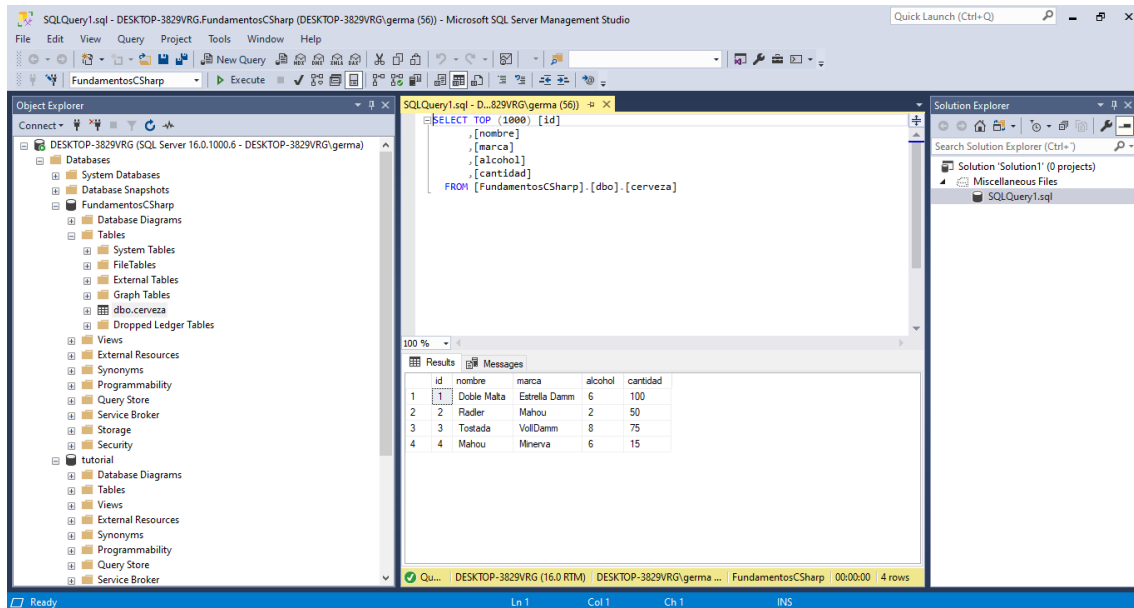
using FundamentosCSharp.Models;
using System;
using System.Collections.Generic;

namespace FundamentosCSharp
{
    internal class Program
    {
        static void Main(string[] args)
        {
            var bebidaAlchólica = new Cerveza(100); // creamos una cerveza
            //var bebidaAlchólica = new Vino(100); // creamos un vino
            MostrarRecomendacion(bebidaAlchólica);
            List<string> lista = new List<string>();
        }

        static void MostrarRecomendacion(IBebidaAlchólica bebida)
        {
            bebida.MaxRecomendado();
        }
    }
}

```

BBDD



CervezaBD.cs

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

namespace FundamentosCSharp.Models
{
    class CervezaDB
    {
    }
```



```

        private string connectionString = "Data Source=localhost; Initial
Catalog=FundamentosCSharp; User=sa; Password=123456;"; //servidor
        // nombre BBD
        //
        // puerto por defecto

        public List<Cerveza> Get()
        {
            List<Cerveza> cervezas = new List<Cerveza>();

            string query = "select nombre, marca, alcohol, cantidad" +
                "from cerveza";

            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                SqlCommand command = new SqlCommand(query, connection);
                connection.Open();
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    string nombre = reader.GetString(0);
                    int cantidad = reader.GetInt32(3);
                    Cerveza cerveza = new Cerveza();
                    cerveza.Alcohol = reader.GetInt32(2);
                    cerveza.Marca = reader.GetString(1);
                    cervezas.Add(cerveza);
                }

                connection.Close();
            }

            return cervezas;
        }
    }
}

```

Program.cs

```

using FundamentosCSharp.Models;
using System;
using System.Collections.Generic;

namespace FundamentosCSharp
{
    internal class Program
    {
        static void Main(string[] args)
        {
            CervezaDB cervezaBD = new CervezaDB();
            var cervezas = cervezaBD.Get();

            foreach (var item in cervezas )
            {
                Console.WriteLine(item.Nombre);
            }
        }
    }
}

```

Update, delete & insert

Program.cs

```
using FundamentosCSharp.Models;
using System;
using System.Collections.Generic;

namespace FundamentosCSharp
{
    internal class Program
    {
        static void Main(string[] args)
        {
            CervezaDB cervezaBD = new CervezaDB();

            // insertamos nuevas cervezas
            {
                Cerveza cerveza = new Cerveza(15, "Mahou");
                cerveza.Marca = "Minerva";
                cerveza.Alcohol = 6;
                cervezaBD.Add(cerveza);
            }

            // editamos una cerveza
            {
                Cerveza cerveza = new Cerveza(15, "Mahou");
                cerveza.Marca = "Minerva";
                cerveza.Alcohol = 5;
                cervezaBD.Edit(cerveza, 5);
            }

            // borrar
            {
                cervezaBD.Delete(5);
            }

            // obtener todas las cervezas
            var cervezas = cervezaBD.Get();

            foreach (var item in cervezas)
            {
                Console.WriteLine(item.Nombre);
            }
        }
    }
}
```

CervezaBD.cs

```
using Microsoft.Data.SqlClient;
using System;
using System.Collections.Generic;

namespace FundamentosCSharp.Models
{
    class CervezaDB
    {
        private string connectionString = "Data Source=DESKTOP-3829VRG;Initial Catalog=FundamentosCSharp;Integrated Security=True";
    }
}
```

```

public List<Cerveza> Get()
{
    List<Cerveza> cervezas = new List<Cerveza>();
    string query = "SELECT nombre, marca, alcohol, cantidad FROM
cerveza";

    using (SqlConnection connection = new
SqlConnection(connectionString))
    {
        SqlCommand command = new SqlCommand(query, connection);
        connection.Open();
        SqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            Cerveza cerveza = new Cerveza();
            cerveza.Nombre = reader.GetString(0);
            cerveza.Marca = reader.GetString(1);
            cerveza.Alcohol = reader.GetInt32(2);
            // Asigna cantidad si es necesario
            cervezas.Add(cerveza);
        }
        connection.Close();
    }
    return cervezas;
}

//
public void Add(Cerveza cerveza)
{
    string query = "INSERT into cerveza(nombre, marca, alcohol,
cantidad) " +
        "values(@nombre, @marca, @alcohol, @cantidad)";
    using (var connection = new SqlConnection(connectionString))
    {
        var command = new SqlCommand(query, connection);
        command.Parameters.AddWithValue("@nombre", cerveza.Nombre);
        command.Parameters.AddWithValue("@marca", cerveza.Marca);
        command.Parameters.AddWithValue("@alcohol",
cerveza.Alcohol);
        command.Parameters.AddWithValue("@cantidad",
cerveza.Cantidad);

        connection.Open();
        command.ExecuteNonQuery();

        connection.Close();
    }
}

public void Edit(Cerveza cerveza, int id)
{
    string query = "UPDATE cerveza set nombre=@nombre," +
        "marca=@marca, alcohol=@alcohol, cantidad=@cantidad " +
        "where id=@id";
    using (var connection = new SqlConnection(connectionString))
    {
        var command = new SqlCommand(query, connection);
        command.Parameters.AddWithValue("@nombre", cerveza.Nombre);
        command.Parameters.AddWithValue("@marca", cerveza.Marca);
        command.Parameters.AddWithValue("@alcohol",
cerveza.Alcohol);

```

```

        command.Parameters.AddWithValue("@cantidad",
cerveza.Cantidad);
        command.Parameters.AddWithValue("@id", id);

        connection.Open();
        command.ExecuteNonQuery();

        connection.Close();
    }
}

public void Delete(int id)
{
    string query = "DELETE from cerveza " +
        "where id=@id";
    using (var connection = new SqlConnection(connectionString))
    {
        var command = new SqlCommand(query, connection);
        command.Parameters.AddWithValue("@id", id);

        connection.Open();
        command.ExecuteNonQuery();

        connection.Close();
    }
}
}
}

```

Serializacion

Program.cs

```

using FundamentosCSharp.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Text.Json;
using System.Text.Json.Serialization;

namespace FundamentosCSharp
{
    internal class Program
    {
        static void Main(string[] args)
        {
            //SERIALIZACION

            //Cerveza cerveza = new Cerveza(10, "Cerveza");
            //string miJson = JsonSerializer.Serialize(cerveza);
            //File.WriteAllText("objeto.txt", miJson);

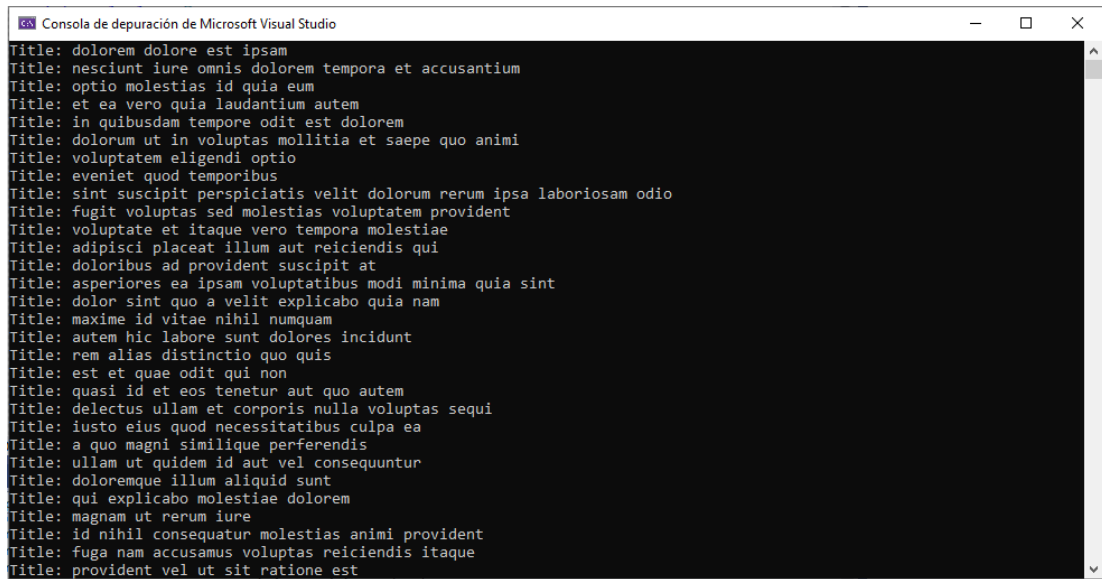
            // DESERIALIZACION
            string miJson = File.ReadAllText("objeto.txt");
            Cerveza cerveza = JsonSerializer.Deserialize<Cerveza>(miJson);

            //" { Cantidad: 10, Nombre: "Cerveza", cosas: []}" // ObjetoJSON
        }
    }
}

```

Servicios Web HTTP

Json placeholder → <https://jsonplaceholder.typicode.com/posts>



Program.cs

```
using FundamentosCSharp.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Text.Json;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace FundamentosCSharp
{
    internal class Program
    {
        static async Task Main(string[] args)
        {
            string url = "https://jsonplaceholder.typicode.com/posts";
            HttpClient client = new HttpClient();

            // Usa await para obtener el resultado de la tarea
            var httpResponse = await client.GetAsync(url);

            // Accede a la propiedad Content y verifica si el estado fue
            exitoso
            if (httpResponse.IsSuccessStatusCode)
            {
                var content = await
                httpResponse.Content.ReadAsStringAsync();

                // Deserializar el contenido JSON a una lista de objetos
                Post
                List<Post>? posts =
                JsonSerializer.Deserialize<List<Post>>(content);

                foreach (var post in posts ?? new List<Post>())
            }
        }
    }
}
```

```

        {
            Console.WriteLine($"Title: {post.title}");
        }
    }
    else
    {
        Console.WriteLine("Error en la solicitud HTTP.");
    }
}

//Console.WriteLine("mordi mi nugget");

//await res; // termina o espera . Proceso hasta que no acabe no sigue
//Console.WriteLine();
}

```

Posts.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FundamentosCSharp.Models
{
    public class Post
    {
        public int userId { get; set; }
        public int id { get; set; }
        public string title { get; set; }
        public string body { get; set; }
    }
}

```

Solicitudes PUT, GET, DELETE

Json placeholder → <https://jsonplaceholder.typicode.com/posts>

Routes

All HTTP methods are supported. You can use http or https for your requests.

GET	/posts
GET	/posts/1
GET	/posts/1/comments
GET	/comments?postId=1
POST	/posts
PUT	/posts/1
PATCH	/posts/1
DELETE	/posts/1

Note: see [guide](#) for usage examples.

```
using FundamentosCSharp.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Text.Json;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace FundamentosCSharp
{
    internal class Program
    {
        static async Task Main(string[] args)
        {
            //string url = "https://jsonplaceholder.typicode.com/posts";
            POST //string url = "https://jsonplaceholder.typicode.com/posts/99";
            PUT //string url = "https://jsonplaceholder.typicode.com/posts/99";
            string url = "https://jsonplaceholder.typicode.com/posts/99";
            //DELETE
            var client = new HttpClient();

            Post post = new()
            {
                userId = 50,
                body = "Hola como estan",
                title = "titulo de saludo"
            };

            var data = JsonSerializer.Serialize<Post>(post);
            //HttpContent content = new StringContent(data,
            System.Text.Encoding.UTF8, "application/json");
```

```

        //POST
        //var httpResponse = await client.PostAsync(url, content);
        //PUT
        //var httpResponse = await client.PutAsync(url, content);
        //DELETE
        var httpResponse = await client.DeleteAsync(url);

        if (httpResponse.IsSuccessStatusCode)
        {
            var result = await httpResponse.Content.ReadAsStringAsync();

            //var postResult = JsonSerializer.Deserialize<Post>(result);
        }
    }
}

```

Genericidad

Linq

Program.cs

```

using FundamentosCSharp.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.Json;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace FundamentosCSharp
{
    internal class Program
    {
        static async Task Main(string[] args)
        {
            List<int> numeros = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
            var numero7 = numeros.Where(d => d == 7).FirstOrDefault();
            var numerosOrdenados = numeros.OrderBy(n => n);
            var numerosAgregados = numeros.Sum(d => d);
            var promedio = numeros.Average(d => d);
            foreach (var i in numerosOrdenados)
            {
                Console.WriteLine(i);
            }
            Console.WriteLine(numero7);
            Console.WriteLine(numerosAgregados);

            Console.WriteLine(promedio);

            List<Cerveza> cervezas = new List<Cerveza>()
            {
                new Cerveza() { Alcohol = 7, Cantidad = 10, Nombre =
"Estrella", Marca = "Lala" },

```



```

        new Cerveza() { Alcohol = 8, Cantidad = 11, Nombre =
"Mahou", Marca = "Aguila" },
        new Cerveza() { Alcohol = 68, Cantidad = 20, Nombre =
"Damm", Marca = "Fuensanta" },
    };
    Console.ReadLine();
    var ordenarMarca = cervezas.OrderBy(c => c.Nombre);
    foreach(var i in ordenarMarca)
    {
        Console.WriteLine(i.Nombre);
    }
}
}
}

```

Linq versión 2

```

using FundamentosCSharp.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.Json;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace FundamentosCSharp
{
    internal class Program
    {
        static async Task Main(string[] args)
        {
            List<Cerveza> cervezas = new List<Cerveza>()
            {
                new Cerveza() { Alcohol = 7, Cantidad = 10, Nombre =
"Estrella", Marca = "Lala" },
                new Cerveza() { Alcohol = 8, Cantidad = 11, Nombre =
"Mahou", Marca = "Aguila" },
                new Cerveza() { Alcohol = 68, Cantidad = 20, Nombre =
"Damm", Marca = "Fuensanta" },
                new Cerveza() { Alcohol = 33, Cantidad = 120, Nombre =
"CocaCola", Marca = "Pompeii" },
            };

            List<Bar> bares = new List<Bar>()
            {
                new Bar("El Bar")
                {
                    cervezas = new List<Cerveza>()
                    {
                        new Cerveza() { Alcohol = 7, Cantidad = 10, Nombre =
"Estrella", Marca = "Lala" },
                        new Cerveza() { Alcohol = 8, Cantidad = 11, Nombre =
"Mahou", Marca = "Aguila" },
                        new Cerveza() { Alcohol = 68, Cantidad = 20, Nombre =
"Damm", Marca = "Fuensanta" },
                    }
                }
            }
        }
    }
}

```

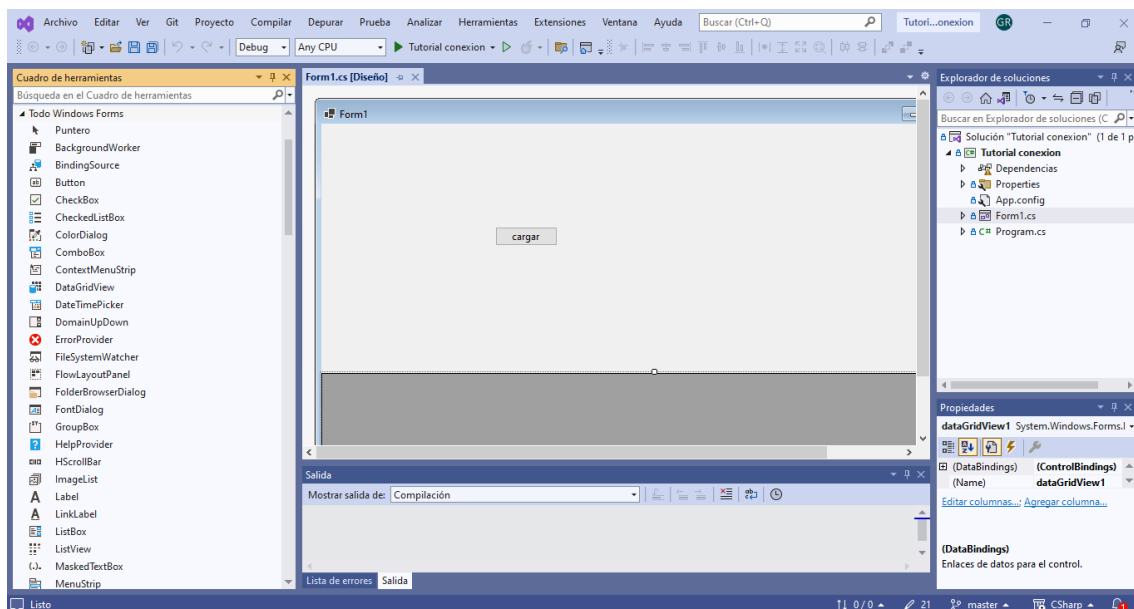
```

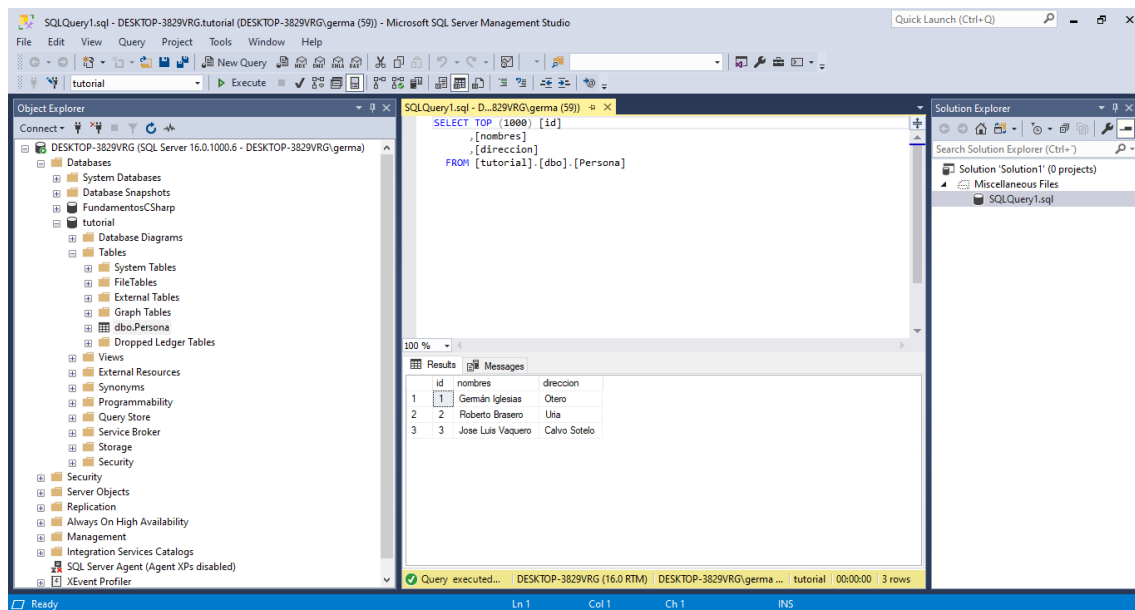
    }

    },
    new Bar("El Bar2")
    { cervezas = new List<Cerveza>()
      {
        new Cerveza() { Alcohol = 7, Cantidad = 10, Nombre = "Estrella", Marca = "Lala" },
        new Cerveza() { Alcohol = 33, Cantidad = 120, Nombre = "CocaCola", Marca = "Pompeii" },
      }
    },
    new Bar("El Bar3")
    { cervezas = new List<Cerveza>()
      {
        new Cerveza() { Alcohol = 44, Cantidad = 22, Nombre = "Vino", Marca = "Pomp" },
        new Cerveza() { Alcohol = 2, Cantidad = 4, Nombre = "Leche", Marca = "Cuca" },
      }
    }
  };
}
}

```

Tutorial Conexión





100 %

Results Messages

	id	nombres	direccion
1	1	Germán Iglesias	Otero
2	2	Roberto Braserio	Una
3	3	Jose Luis Vaquero	Calvo Sotelo

Form1.cs

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

namespace Tutorial_conexion
{
    public partial class Form1 : Form
    {
        SqlConnection conn = new SqlConnection("Data Source=DESKTOP-3829VRG;Initial Catalog=tutorial;Integrated Security=True");
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            cargarDatos();
        }

        private void cargarDatos()
        {
            SqlDataAdapter da = new SqlDataAdapter("select * from persona",
conn);
            DataTable dt = new DataTable();
            da.Fill(dt);
        }
    }
}
```

```

        this.dataGridView1.DataSource = dt;
    }

    private void Form1_Load(object sender, EventArgs e)
    {
    }
}

```

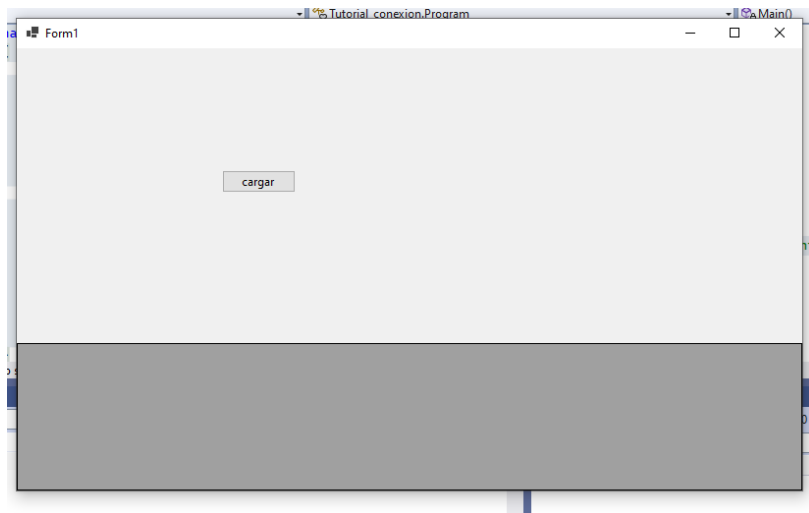
Program.cs

```

namespace Tutorial_conexion
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // To customize application configuration such as set high DPI
            settings or default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new Form1());
        }
    }
}

```

Antes



Después

Form1

cargar

	id	nombres	direccion
▶	1	Germán Iglesias	Otero
	2	Roberto Brasero	Uria
	3	Jose Luis Vaquero	Calvo Sotelo
*			