CONSUMIR PROCEDIMIENTOS ALMACENADOS DESDE ADO.NET

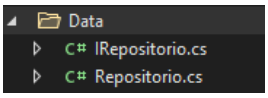
1. Creamos un modelo para interpretar lo que viene de la base de datos

```
namespace EmpresaAPI.Models
{
    1 referencia
    public class Ciudad
    {
        0 referencias
        public int Id { get; set; }
        0 referencias
        public string NombreCiudad { get; set; }
    }
}
```

```
namespace EmpresaAPI.Models
{
    1 referencia
    public class TipoIdent
    {
        0 referencias
        public int Id { get; set; }
        0 referencias
        public string TipoIdentificacion { get; set; }
    }
}
```

```
namespace EmpresaAPI.Models
{
    0 referencias
    public class Empleado
    {
        0 referencias
        public int Id { get; set; }
        0 referencias
        public string Nombre { get;}
        0 referencias
        public TipoIdent refTipoIdent { get; set; }
        0 referencias
        public string Cedula { get; }
        0 referencias
        public Ciudad refCiudad { get; set; }

    }
}
```

2. Vamos a crear una carpeta Data y dentro creamos una clase para la lógica de interacción con la base de datos



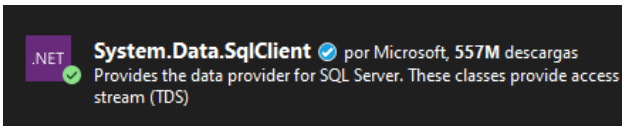
3. Vamos a crear los métodos para la interfaz IRepositorio que va a recibir cualquier clase al definir el “where T: class”

```
namespace EmpresaAPI.Data
{
    0 referencias
    public interface IRepositorio<T> where T : class
    {
        0 referencias
        Task<List<T>> Lista();
        0 referencias
        Task<bool> Guardar(T modelo);
        0 referencias
        Task<bool> Editar(T modelo);
        0 referencias
        Task<bool> Eliminar(int id);
    }
}
```

4. Vamos al archivo appsettings.json y agregamos la conexión.

```
"ConnectionStrings": {
  "conexionSQL": "Data Source=(localdb)\\MSSQLLocalDB; Initial Catalog=Empresa; Integrated Security=true" }
```

5. Instalar el paquete System.Data.SqlClient



6. Dentro del archivo program.cs vamos a implementar los repositorios para poder usarlos dentro de toda la aplicación.

```
using EmpresaAPI.Models;
using EmpresaAPI.Data;


var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddScoped<IRepositorio<Empleado>, EmpleadoRepositorio>();
builder.Services.AddScoped<IRepositorio<Ciudad>, CiudadRepositorio>();
builder.Services.AddScoped<IRepositorio<TipoIdent>, TipoIdentRepositorio>();
```

NOTA: Este paso lo debemos hacer después de implementar todos los repositorios

7. Creamos el Repositorio de ciudades que hereda de IRepositorio

```
using EmpresaAPI.Models;
using EmpresaAPI.Data;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;

namespace EmpresaAPI.Data
{
    1 referencia
    public class CiudadRepositorio : IRepositorio<Ciudad>
    {
        private readonly string _conexion = String.Empty;

        0 referencias
        public CiudadRepositorio(IConfiguration configuration)
        {
            _conexion = configuration.GetConnectionString("conexionSQL");
        }
        1 referencia
        public async Task<List<Ciudad>> Lista()
        {
            List<Ciudad> ciudades = new List<Ciudad>();
            using (var conexion = new SqlConnection(_conexion))
            {
                conexion.Open();
                SqlCommand cmd = new SqlCommand("ConsultarCiudades", conexion);
                cmd.CommandType = CommandType.StoredProcedure;

                using (var dr = await cmd.ExecuteReaderAsync())
                {
                    while (await dr.ReadAsync())
                    {
                        ciudades.Add(new Ciudad
                        {
                            Id = (int)dr["Id"],
                            NombreCiudad = dr["NombreCiudad"].ToString()
                        });
                    }
                }
            }
            return ciudades;
        }
```

8. Creamos el Repositorio de tipoIdent que hereda de IRepositorio

```
using EmpresaAPI.Models;
using EmpresaAPI.Data;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;

namespace EmpresaAPI.Data
{
    1 referencia
    public class TipoIdentRepositorio : IRepositorio<TipoIdent>
    {
        private readonly string _conexion = String.Empty;
        0 referencias
        public TipoIdentRepositorio(IConfiguration configuration)
        {
            _conexion = configuration.GetConnectionString("conexionSQL");
        }

        1 referencia
        public async Task<List<TipoIdent>> Lista()
        {
            List<TipoIdent> tiposIdent = new List<TipoIdent>();
            using (var conexion = new SqlConnection(_conexion))
            {
                conexion.Open();
                SqlCommand cmd = new SqlCommand("ConsultarTiposIdent", conexion);
                cmd.CommandType = CommandType.StoredProcedure;

                using (var dr = await cmd.ExecuteReaderAsync())
                {
                    while (await dr.ReadAsync())
                    {
                        tiposIdent.Add(new TipoIdent
                        {
                            Id = (int)dr["Id"],
                            TipoIdentificacion = dr["TipoIdentificacion"].ToString()
                        });
                    }
                }
            }
            return tiposIdent;
        }
```

9. Creamos el Repositorio de empleado que hereda de IRepositorio

```
namespace EmpresaAPI.Data
{
    1 referencia
    public class EmpleadoRepositorio : IRepositorio<Empleado>
    {
        private readonly string _conexion = String.Empty;
        0 referencias
        public EmpleadoRepositorio(IConfiguration configuration)
        {
            _conexion = configuration.GetConnectionString("conexionSQL");
        }
        1 referencia
        public async Task<List<Empleado>> Lista()
        {
            List<Empleado> empleados = new List<Empleado>();
            using (var conexion = new SqlConnection(_conexion))
            {
                conexion.Open();
                SqlCommand cmd = new SqlCommand("SeleccionarTodo", conexion);
                cmd.CommandType = CommandType.StoredProcedure;
                using (var dr = await cmd.ExecuteReaderAsync())
                {
                    while (await dr.ReadAsync())
                    {
                        empleados.Add(new Empleado
                        {
                            Id = (int)dr["Id"],
                            Nombre = dr["Nombre"].ToString(),
                            refTipoIdent = new TipoIdent()
                            {
                                Id = (int)dr["Id"],
                                TipoIdentificacion = dr["TipoIdentificacion"].ToString()
                            },
                            Cedula = dr["Cedula"].ToString(),
                            refCiudad = new Ciudad()
                            {
                                Id = (int)dr["Id"],
                                NombreCiudad = dr["NombreCiudad"].ToString()
                            }
                        });
                    }
                }
            }
            return empleados;
        }
        public async Task<bool> Editar(Empleado modelo)
        {
            using (var conexion = new SqlConnection(_conexion))
            {
                conexion.Open();
                SqlCommand cmd = new SqlCommand("ActualizarEmpleado", conexion);
                cmd.Parameters.AddWithValue("Id", modelo.Id);
                cmd.Parameters.AddWithValue("Nombre", modelo.Nombre);
                cmd.Parameters.AddWithValue("IdTipoIdent", modelo.refTipoIdent.Id);
                cmd.Parameters.AddWithValue("Cedula", modelo.Cedula);
                cmd.Parameters.AddWithValue("IdCiudad", modelo.refCiudad.Id);
                cmd.CommandType = CommandType.StoredProcedure;

                int filasAfectadas = await cmd.ExecuteNonQueryAsync();
                if (filasAfectadas > 0)
                    return true;
                else
                    return false;
            }
        }

        1 referencia
        public async Task<bool> Guardar(Empleado modelo)
        {
            using (var conexion = new SqlConnection(_conexion))
            {
                conexion.Open();
                SqlCommand cmd = new SqlCommand("InsertarEmpleado", conexion);
                cmd.Parameters.AddWithValue("Nombre", modelo.Nombre);
                cmd.Parameters.AddWithValue("IdTipoIdent", modelo.refTipoIdent.Id);
                cmd.Parameters.AddWithValue("Cedula", modelo.Cedula);
                cmd.Parameters.AddWithValue("IdCiudad", modelo.refCiudad.Id);
                cmd.CommandType = CommandType.StoredProcedure;

                int filasAfectadas = await cmd.ExecuteNonQueryAsync();
                if(filasAfectadas>0)
                    return true;
                else
                    return false;
            }
        }
        public async Task<bool> Eliminar(int id)
        {
            using (var conexion = new SqlConnection(_conexion))
            {
                conexion.Open();
                SqlCommand cmd = new SqlCommand("EliminarEmpleado", conexion);
                cmd.Parameters.AddWithValue("Id", id);
                cmd.CommandType = CommandType.StoredProcedure;

                int filasAfectadas = await cmd.ExecuteNonQueryAsync();
                if (filasAfectadas > 0)
                    return true;
                else
                    return false;
            }
        }

    }
}
```

10. Para consumir estos métodos vamos a la clase controller e implementamos las interfaces, añadimos using hacia el data y los modelos.

```
using EmpresaAPI.Models;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;
using EmpresaAPI.Data;

namespace EmpresaAPI.Controllers
{
    3 referencias
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private readonly IRepositorio<Ciudad> ciudadRepositorio;
        private readonly IRepositorio<TipoIdent> tipoIdentRepositorio;
        private readonly IRepositorio<Empleado> empleadoRepositorio;

        0 referencias
        public HomeController(ILogger<HomeController> logger,
            IRepositorio<Ciudad> ciudadRepositorio,
            IRepositorio<TipoIdent> tipoIdentRepositorio,
            IRepositorio<Empleado> empleadoRepositorio)
        {
            _logger = logger;
            this.ciudadRepositorio = ciudadRepositorio;
            this.tipoIdentRepositorio = tipoIdentRepositorio;
            this.empleadoRepositorio = empleadoRepositorio;
        }
```

11. A continuación vamos a hacer una prueba de consumir los métodos lista() de todos los repositorios

```
public async Task<IActionResult> Index()
{
    List<Empleado> empleados = await empleadoRepositorio.Lista();
    return View();
}
```

```
public async Task<IActionResult> Index()
{
    List<TipoIdent> TiposIdent = await tipoIdentRepositorio.Lista();
    return View();
}
```

```
public async Task<IActionResult> Index()
{
    List<Ciudad> ciudades = await ciudadRepositorio.Lista();
    return View();
}
```

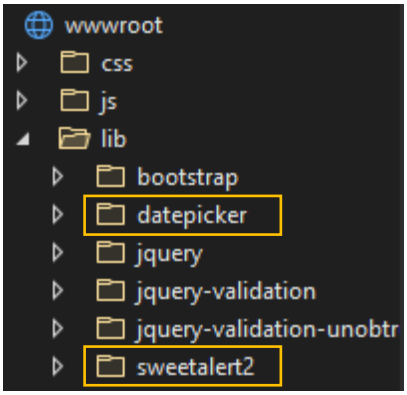
12. Ahora vamos a crear los métodos en el controlador para pasarlos a la vista.
    - Métodos Get para mostrar los datos.

```
[HttpGet]
0 referencias
public async Task<IActionResult> listaCiudades()
{
    List<Ciudad> ciudades = await ciudadRepositorio.Lista();
    return StatusCode(StatusCodes.Status200OK, ciudades);
}

[HttpGet]
0 referencias
public async Task<IActionResult> listaTiposIdent()
{
    List<TipoIdent> tiposIdent = await tipoIdentRepositorio.Lista();
    return StatusCode(StatusCodes.Status200OK, tiposIdent);
}

[HttpGet]
0 referencias
public async Task<IActionResult> listaEmpleados()
{
    List<Empleado> empleados= await empleadoRepositorio.Lista();
    return StatusCode(StatusCodes.Status200OK, empleados);
}
```

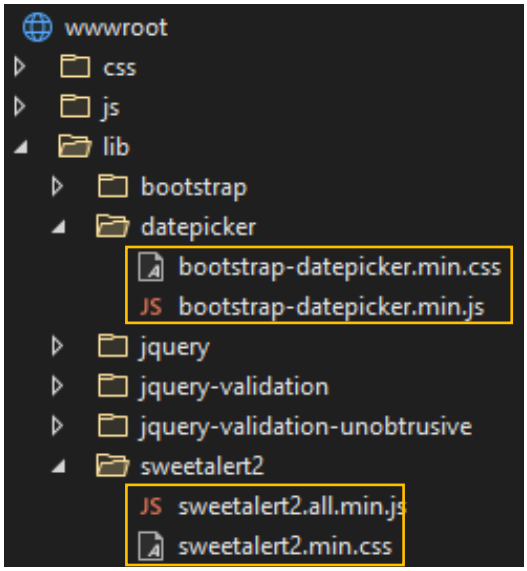
- Método Post, Put y Delete para guardar,actualizar y eliminar un empleado nuevo.

```
[HttpPost]
0 referencias
public async Task<IActionResult> guardarEmpleado([FromBody] Empleado modelo)
{
    bool resultado = await empleadoRepositorio.Guardar(modelo);
    if (resultado)
        return StatusCode(StatusCodes.Status200OK, new { valor = resultado, msg = "ok" });
    else
        return StatusCode(StatusCodes.Status500InternalServerError, new { valor = resultado, msg = "error" });
}

[HttpPut]
0 referencias
public async Task<IActionResult> editarEmpleado([FromBody] Empleado modelo)
{
    bool resultado = await empleadoRepositorio.Editar(modelo);
    if (resultado)
        return StatusCode(StatusCodes.Status200OK, new { valor = resultado, msg = "ok" });
    else
        return StatusCode(StatusCodes.Status500InternalServerError, new { valor = resultado, msg = "error" });
}

[HttpDelete]
0 referencias
public async Task<IActionResult> eliminarEmpleado(int id)
{
    bool resultado = await empleadoRepositorio.Eliminar(id);
    if (resultado)
        return StatusCode(StatusCodes.Status200OK, new { valor = resultado, msg = "ok" });
    else
        return StatusCode(StatusCodes.Status500InternalServerError, new { valor = resultado, msg = "error" });
}
```

13. Ahora vamos a implementar las vistas. Primero agregamos las siguientes librerías para mostrar un calendario y para mostrar mensajes.



14. Vamos a la carpeta de Views->Shared->_Layout.cshtml y agregamos las siguientes referencias arrastrándolos:
    - Estos archivos se arrastran

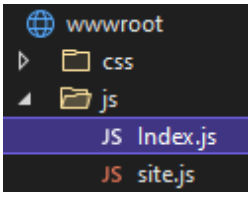


- Al archivo _Layout.cshtml

```
<link rel="stylesheet" href="~/lib/sweetalert2/sweetalert2.min.css" />
<link rel="stylesheet" href="~/lib/datepicker/bootstrap-datepicker.min.css" />
```

```
<script src="~/lib/sweetalert2/sweetalert2.all.min.js"></script>
<script src="~/lib/datepicker/bootstrap-datepicker.min.js"></script>
```

15. Creamos un archivo Index.js JavaScript en wwwroot->js



16. Agregamos el archivo Index.js a la vista Home/Index

```
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">
</div>

@section Scripts{
    <script src="~/js/Index.js"></script>
}
```

17. Creamos el código html de la vista index. Los siguientes códigos van después del y antes del @section.
    - Primero la parte de la tabla

```
@{
    ViewData["Title"] = "Home Page";
}

<div class="row justify-content-center">
    <div class="col-10">

        <div class="alert alert-info" role="alert">
            Crud Empleados
        </div>
        <button type="button" class="btn btn-success btn-sm btn-nuevo">Nuevo Empleado</button>
        <table class="table table-striped" id="tablaEmpleados">
            <thead>
                <tr>
                    <th>Nombre</th>
                    <th>Tipo de Identificacion</th>
                    <th>Cedula</th>
                    <th>Ciudad</th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
</div>
```

- Segundo creamos un modal que es como un popup para editar o guardar la información del empleado.

```html
<!-- Modal -->
<div class="modal fade" id="modalEmpleado" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Modal Empleado</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">

                <div class="mb-3">
                    <label class="form-label">Nombre</label>
                    <input type="text" class="form-control" id="txtNombre" autocomplete="off">
                </div>
                <div class="mb-3">
                    <label class="form-label">Tipo Identificación</label>
                    <select class="form-select" id="cboTipoIdent"></select>
                </div>
                <div class="mb-3">
                    <label class="form-label">Cedula</label>
                    <input type="number" class="form-control" id="txtCedula">
                </div>
                <div class="mb-3">
                    <label class="form-label">Ciudad</label>
                    <select class="form-select" id="cboCiudad"></select>
                </div>


            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Cerrar</button>
                <button type="button" class="btn btn-primary btn-guardar">Guardar</button>
            </div>
        </div>
    </div>
</div>
```

18. Creamos todos los métodos y funciones en el archivo Index.js para que funcione toda la lógica de la página.
    NOTA: Tener en cuenta que los métodos Http pueden devolver los parámetros en minúscula.

```javascript
const modeloEmpleado = {
    id: 0,
    nombre: "",
    idTipoIdent: 0,
    cedula: "",
    idCiudad: 0
}
//funcion que se encarga de llenar la tabla que creamos en el archivo html
function MostrarEmpleados() {
    fetch("/Home/listaEmpleados")//El Home es el controlador, no es necesario poner todo el nombre osea HomeController
        .then(response => {
            return response.ok ? response.json() : Promise.reject(response)
        })
        .then(responseJson => {
            if (responseJson.length > 0) {
                $("#tablaEmpleados tbody").html("");

                responseJson.forEach((empleado) => {
                    $("#tablaEmpleados tbody").append(
                        $("<tr>").append(
                            $("<td>").text(empleado.nombre),
                            $("<td>").text(empleado.refTipoIdent.tipoIdentificacion),
                            $("<td>").text(empleado.cedula),
                            $("<td>").text(empleado.refCiudad.nombreCiudad),
                            $("<td>").append(                                   //la variable dataEmpleado guarda la informacion del modelo
                                $("<button>").addClass("btn btn-primary btn-sm btn-editar").text("Editar").data("dataEmpleado", empleado),
                                $("<button>").addClass("btn btn-danger btn-sm ms-2 btn-eliminar").text("Eliminar").data("dataEmpleado", empleado)
                            )
                        )
                    )
                })
            }
        })
}
```

```
//creamos el metodo cuando la pagina esta cargada
document.addEventListener("DOMContentLoaded", function () {
    MostrarEmpleados();

    fetch("/Home/listaTiposIdent")
        .then(response => {
            return response.ok ? response.json() : Promise.reject(response)
        })
        .then(responseJson => {
            if (responseJson.length > 0) {
                responseJson.forEach((tipoIdent) => {
                    $("#cboTipoIdent").append(
                        $("<option>").val(tipoIdent.id).text(tipoIdent.tipoIdentificacion)
                    )
                })
            }
        })

    fetch("/Home/listaCiudades")
        .then(response => {
            return response.ok ? response.json() : Promise.reject(response)
        })
        .then(responseJson => {
            if (responseJson.length > 0) {
                responseJson.forEach((ciudad) => {
                    $("#cboCiudad").append(
                        $("<option>").val(ciudad.id).text(ciudad.nombreCiudad)
                    )
                })
            }
        })

    //La siguiente logica se utiliza para cargar una fecha
    //$("#txtFecha").datepicker({
    //    format: "dd/mm/yyyy",
    //    autoclose: true,
    //    todayHighlight:  true
    //})

}, false)
```

```
function MostrarModal() {
    $("#txtNombre").val(modeloEmpleado.nombre);
    $("#cboTipoIdent").val(modeloEmpleado.idTipoIdent == 0 ? $("#cboTipoIdent option:first").val() : modeloEmpleado.idTipoIdent);
    $("#txtCedula").val(modeloEmpleado.cedula);
    $("#cboCiudad").val(modeloEmpleado.idCiudad == 0 ? $("#cboCiudad option:first").val() : modeloEmpleado.idCiudad);
    $("#modalEmpleado").modal("show");
}

$(document).on("click", ".btn-nuevo", function () {
    modeloEmpleado.id = 0;
    modeloEmpleado.nombre = "";
    modeloEmpleado.idTipoIdent = 0;
    modeloEmpleado.cedula = "";
    modeloEmpleado.idCiudad = 0
    MostrarModal();
})

$(document).on("click", ".btn-editar", function () {

    const empleado = $(this).data("dataEmpleado");// Se obtiene la informacion del modelo
    modeloEmpleado.id = empleado.id;
    modeloEmpleado.nombre = empleado.nombre;
    modeloEmpleado.idTipoIdent = empleado.refTipoIdent.id;
    modeloEmpleado.cedula = empleado.cedula;
    modeloEmpleado.idCiudad = empleado.refCiudad.id
    MostrarModal();
})
```

```
$(document).on("click", ".btn-guardar", function () {

    const modelo = {
        id: modeloEmpleado.id,
        nombre: $("#txtNombre").val(),
        refTipoIdent: {
            id: $("#cboTipoIdent").val()
        },
        cedula: $("#txtCedula").val(),
        refCiudad: {
            id: $("#cboCiudad").val()
        }
    }

    if (modeloEmpleado.id == 0) {
        fetch("/Home/guardarEmpleado", {
            method: "POST",
            headers: { "Content-Type": "application/json; charset=utf-8" },
            body: JSON.stringify(modelo)
        })
            .then(response => {
                return response.ok ? response.json() : Promise.reject(response)
            })
            .then(responseJson => {
                if (responseJson.valor) {
                    $("#modalEmpleado").modal("hide");
                    Swal.fire("Listo!", "Empleado fue creado", "success")
                    MostrarEmpleados();
                }
                else
                    Swal.fire("Lo sentimos", "No se pudo crear", "error")
            })
    }
```

```
    else {
        fetch("/Home/editarEmpleado", {
            method: "PUT",
            headers: { "Content-Type": "application/json; charset=utf-8" },
            body: JSON.stringify(modelo)
        })
            .then(response => {
                return response.ok ? response.json() : Promise.reject(response)
            })
            .then(responseJson => {
                if (responseJson.valor) {
                    $("#modalEmpleado").modal("hide");
                    Swal.fire("Listo!", "Empleado fue actualizado", "success")
                    MostrarEmpleados();
                }
                else
                    Swal.fire("Lo sentimos", "No se pudo actualizar", "error")
            })
    }

})
```

```javascript
$(document).on("click", ".btn-eliminar", function () {

    const empleado = $(this).data("dataEmpleado");
    Swal.fire({
        title: "Esta seguro?",
        text: `Eliminar empleado "${empleado.nombre}"`,
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#3085d6",
        cancelButtonColor: "#d33",
        confirmButtonText: "Si, eliminar",
        cancelButtonText: "No, volver"
    })
        .then((result) => {
            if (result.isConfirmed) {
                fetch(`/Home/eliminarEmpleado?id=${empleado.id}`, {
                    method: "DELETE"
                })
                    .then(response => {
                        return response.ok ? response.json() : Promise.reject(response)
                    })
                    .then(responseJson => {
                        if (responseJson.valor) {
                            Swal.fire("Listo!", "Empleado fue eliminado", "success");
                            MostrarEmpleados();
                        }
                        else
                            Swal.fire("Lo sentimos", "No se pudo eliminar", "error");
                    })
            }
        })
})
```

Con esto terminamos el aplicativo.

TEMAS A TENER EN CUENTA

1. **set nocount on; Si esta configurado de esta manera en el procedimiento almacenado no va a funcionar la lógica de filasAfectadas en los repositorios**
2. **Al trabajar en los archivos de JavaScript posiblemente las variables que vienen en los métodos Http lleguen en minúsculas y por lo tanto no hace el patch correcto al cargar la información.**
3. **https://getbootstrap.com/docs/5.0/getting-started/introduction/ en el link anterior encontramos toda la documentación y ejemplos para trabajar con los objetos html que podemos incluir en nuestra pagina.**