Código estudiante

https://www.youtube.com/watch?v=Pb6Ohi9OQzU&list=WL&index=58

Arranca haciendo la BBDD en el Sql Server

Crea procedimientos almacenados.

1. Selecciona todo de la tabla de Departamento

create procedure sp\_ListaDepartamentos

as

begin

select idDepartamento, nombre from CEDepartamento

end

1. Selecciona luego info de ambas tablas a la vez, por medio de un inner join:

create procedure sp\_ListaEmpleados

as

begin

set dateformat dmy

select

e.idEmpleado, e.nombreCompleto,

d.idDepartamento, d.nombre,

e.sueldo,

convert(char(10),e.fechaContrato,103) as 'fechaContrato'

from CEEmpleado as e

inner join CEDepartamento as d on e.idDepartamento= d.IdDepartamento

end

A la vez, juguetea con el formato de fechaContrato. El 103 alude a cierta configuración (sobre si separan barras o puntos o que va primero, si el mes o el día).

1. Para guardar los empleados:

create procedure sp\_GuardarEmpleado(

@nombreCompleto varchar (50),

@idDepartamento int,

@sueldo int,

@fechaContrato varchar(10)

)

as

begin

insert into CEEmpleado (nombreCompleto, idDepartamento, sueldo, fechaContrato)

values

(@nombreCompleto, @idDepartamento,@sueldo,convert(date,@fechaContrato))

end

Sigue haciéndole conversiones a fechaContrato.

1. Editar empleado.

create procedure sp\_EditarEmpleado(

@idEmpleado int,

@nombreCompleto varchar (50),

@idDepartamento int,

@sueldo int,

@fechaContrato varchar(10)

)

as

begin

set dateformat dmy

update CEEmpleado set

nombreCompleto=@nombreCompleto,

idDepartamento=@idDepartamento,

sueldo=@sueldo,

@fechaContrato=convert(date,@fechaContrato)

where idEmpleado=@idEmpleado

end

1. Eliminar empleado

create procedure sp\_EliminarEmpleado(

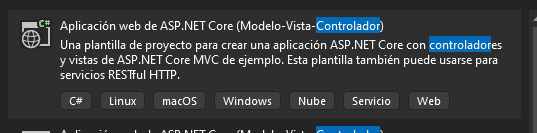
@idEmpleado int)

as

begin

delete from CEEmpleado where idEmpleado=@idEmpleado

end



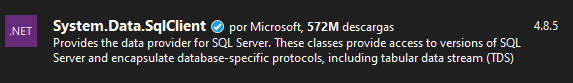
Desactiva configurar para https porque no quiere instalar el certificado.

Pone la cadena de conexión appsetings.Json

"CadenaDeConexion": "Server=(local); DataBase=BolsaDeDatos; Trusted\_Connection=True; TrustServerCertificate=True"

En español porque soy re piola

Instala los paquetes para poder trabajar con Ado.Net.



Agrega clases al modelo: Departamento y empleado. En la segunda establece una relación con Departamento. O sea, los empleados tienen departamento, pero no al revés:

public class Departamento

{

public int idDepartamento { get; set; }

public string nombre { get; set; }

}

public class Empleado

{

public int idEmpleado { get; set; }

public string nombreCompleto { get; set; }

public Departamento departamento { get; set; }

public int sueldo { get; set; }

public string fechaContrato { get; set; }

}

Crea carpeta repositorio, y ahí otras 2, contrato e implementación. En la primera crea una interfaz.

Como es una interfaz genérica (que podrá recibir tanto departamentos como empleados), por eso es <T> y especifica que T es una clase.

public interface IGenericRepository<T> where T:class

{

Task<List<T>> GetLista();

Task<bool> Guardar(T modelo);

Task<bool> Editar(T modelo);

Task<bool> Eliminar(int id);

}

En implementación crea clase DepartamentoRepository.

public class DepartamentoRepository:IGenericRepository<Departamento>

{

private readonly string \_cadenaSql = "";

//Como parámetro recibe la configuración, para poder recibir la cadena de conexión.

public DepartamentoRepository(IConfiguration configuracion)

{

\_cadenaSql = configuracion.GetConnectionString("CadenaDeConexion");

}

public async Task<List<Departamento>> GetLista()

{

List<Departamento> \_lista=new List<Departamento>();

using (var conexion = new SqlConnection(\_cadenaSql))

{

SqlCommand cmd = new SqlCommand("sp\_ListaDepartamentos", conexion);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

using (var dr = await cmd.ExecuteReaderAsync())

{

while (await dr.ReadAsync())

{

\_lista.Add(new Departamento

{

idDepartamento = Convert.ToInt32(dr["idDepartamento"]),

nombre = dr["nombre"].ToString()

});

}

}

}

return \_lista;

}

Acá mete los 4 métodos declarados en la interfaz, aunque en los hechos hace solo 1.

Luego, hace los 4 en EmpleadoRepository:

public class EmpleadoRepository : IGenericRepository<Empleado>

{

private readonly string \_cadenaSql = "";

public EmpleadoRepository(IConfiguration configuracion)

{

\_cadenaSql = configuracion.GetConnectionString("CadenaDeConexion");

}

public async Task<List<Empleado>> GetLista()

{

List<Empleado> \_lista = new List<Empleado>();

using (var conexion = new SqlConnection(\_cadenaSql))

{

conexion.Open();

SqlCommand cmd = new SqlCommand("sp\_ListaEmpleados", conexion);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

using (var dr = await cmd.ExecuteReaderAsync())

{

while (await dr.ReadAsync())

{

\_lista.Add(new Empleado

{

idEmpleado = Convert.ToInt32(dr["idEmpleado"]),

nombreCompleto = dr["nombreCompleto"].ToString(),

departamento = new Departamento()

{

idDepartamento = Convert.ToInt32(dr["idDepartamento"]),

nombre = dr["nombre"].ToString()

},

sueldo = Convert.ToInt32(dr["sueldo"]),

fechaContrato = dr["fechaContrato"].ToString(),

});

}

}

}

return \_lista;

}

public async Task<bool> Editar(Empleado modelo)

{

using (var conexion = new SqlConnection(\_cadenaSql))

{

conexion.Open();

SqlCommand cmd = new SqlCommand("sp\_EditarEmpleado", conexion);

cmd.Parameters.AddWithValue("@nombreCompleto", modelo.nombreCompleto);

cmd.Parameters.AddWithValue("@idEmpleado", modelo.idEmpleado);

cmd.Parameters.AddWithValue("@idDepartamento", modelo.departamento.idDepartamento);

cmd.Parameters.AddWithValue("@sueldo", modelo.sueldo);

cmd.Parameters.AddWithValue("@fechaContrato", modelo.fechaContrato);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

int filasAfectadas = await cmd.ExecuteNonQueryAsync();

//Lo que sigue se relaciona a lo que retorna SQL Server:

//Si elimina, edita o crea, retorna 1. Por eso si es mayor que 0, es exitoso.

if (filasAfectadas > 0)

{

return true;

}

else { return false; }

}

}

public async Task<bool> Guardar(Empleado modelo)

{

using (var conexion = new SqlConnection(\_cadenaSql))

{

conexion.Open();

SqlCommand cmd = new SqlCommand("sp\_GuardarEmpleado", conexion);

cmd.Parameters.AddWithValue("@nombreCompleto", modelo.nombreCompleto);

cmd.Parameters.AddWithValue("@idDepartamento", modelo.departamento.idDepartamento);

cmd.Parameters.AddWithValue("@sueldo", modelo.sueldo);

cmd.Parameters.AddWithValue("@fechaContrato", modelo.fechaContrato);

cmd.CommandType = System.Data.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(\_cadenaSql))

{

conexion.Open();

SqlCommand cmd = new SqlCommand("sp\_EliminarEmpleado", conexion);

cmd.Parameters.AddWithValue("@idEmpleado", id);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

int filasAfectadas = await cmd.ExecuteNonQueryAsync();

if (filasAfectadas > 0)

{

return true;

}

else { return false; }

}

}

}

Program.Cs:

builder.Services.AddScoped<IGenericRepository<Departamento>, DepartamentoRepository>();

builder.Services.AddScoped<IGenericRepository<Empleado>, EmpleadoRepository>();

Luego trabaja con los controladores (HomeController):

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

private readonly IGenericRepository<Departamento> \_departamentoRepository;

private readonly IGenericRepository<Empleado> \_empleadoRepository;

//Construttore

public HomeController(ILogger<HomeController> logger,

IGenericRepository<Departamento> departamentoRepository,

IGenericRepository<Empleado> empleadoRepository)

{

\_logger = logger;

\_departamentoRepository = departamentoRepository;

\_empleadoRepository = empleadoRepository;

}

Crea los 5 métodos ya trabajados:

[HttpGet]

public async Task <IActionResult> ListaDepartamentos()

{

List<Departamento> listadepartamentos = await \_departamentoRepository.GetLista();

//LLama al método getlista y lo retorna pero con el mensajito este de status.

return StatusCode(StatusCodes.Status200OK, listadepartamentos);

}

[HttpGet]

public async Task<IActionResult> ListaEmpleados()

{

List<Empleado> listaEmpleados = await \_empleadoRepository.GetLista();

return StatusCode(StatusCodes.Status200OK, listaEmpleados);

}

[HttpPost]

public async Task<IActionResult> GuardarEmpleado([FromBody] Empleado modelo)

{

bool resultado= await \_empleadoRepository.Guardar(modelo);

if (resultado)

{

return StatusCode(StatusCodes.Status200OK, new { valor = resultado, msg = "Ok" });

}

else

{

return StatusCode(StatusCodes.Status500InternalServerError, new { valor = resultado, msg = "Error" });

}

}

[HttpPut]

public async Task<IActionResult> EditarEmpleado([FromBody] Empleado modelo)

{

bool resultado = await \_empleadoRepository.Guardar(modelo);

if (resultado)

{

return StatusCode(StatusCodes.Status200OK, new { valor = resultado, msg = "Ok" });

}

else

{

return StatusCode(StatusCodes.Status500InternalServerError, new { valor = resultado, msg = "Error" });

}

}

[HttpDelete]

public async Task<IActionResult> EliminarEmpleado([FromBody] int idEmpleado)

{

bool resultado = await \_empleadoRepository.Eliminar(idEmpleado);

if (resultado)

{

return StatusCode(StatusCodes.Status200OK, new { valor = resultado, msg = "Ok" });

}

else

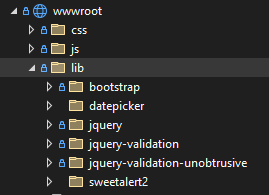
{

return StatusCode(StatusCodes.Status500InternalServerError, new { valor = resultado, msg = "Error" });

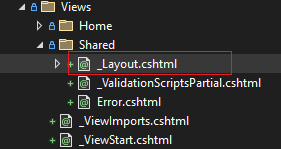
}

}

Deja 2 plugins que se envían a la carpeta lib (donde está bootstap y Jquery).



Sweetalert2 y Datepicker



Ahí agrega las referencias a los estilos importados.

HTML

La extensión de archivo CSHTML es un archivo ASP.Net Web Page desarrollado originalmente por Microsoft Corporation para Microsoft Visual Studio.  . cshtml es un archivo HTML C# que el motor Razor Markup utiliza en el lado del servidor para representar los archivos de la página web en el navegador del usuario.

DATAZO. Para agregar la referencia basta con pinchar el archivo .css y llevarlo hasta el html!!!

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

<**link** href="~/lib/sweetalert2/sweetalert2.min.css" rel="stylesheet" />

Abajo, en la parte de los scripts, agrega los 2 respectivos scripts

Crea nuevo hoja de Index.Js

Va a Index.cshtml y agrega la referencia al index.js

CSHTML ES EL HTML DE CS.

@section Scripts {

<**script** src="~/js/index.js"></**script**>

}

Luego se pone a armar el html (o cshtml). CopyPastea de boostrap un msg de alerta. Botón. Tabla, Modal, FormControls

@{

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 boton-NuevoEmpleado">Nuevo empleado</button>

<table class="table table-striped" id="tablaEmpleados">

<thead>

<tr>Nombre Completo</tr>

<tr>Departamento</tr>

<tr>Sueldo</tr>

<tr>Fecha contrato</tr>

<tr></tr>

</thead>

<tbody></tbody>

</table>

</div>

</div>

<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">

<h1 class="modal-title fs-5" id="exampleModalLabel">Modal Empleado</h1>

<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 Completo</label>

<input type="text" class="form-control" id="txtNombreCompleto" autocomplete="off">

</div>

<div class="mb-3">

<label class="form-label">Departamento</label>

<select class="form-select" id="cboDepartamento"></select>

</div>

<div class="mb-3">

<label class="form-label">Sueldo</label>

<input type="number" class="form-control" id="txtSueldo">

</div>

<div class="mb-3">

<label class="form-label">Fecha de contrato</label>

<input type="text" class="form-control" id="txtFechaContrato">

</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 boton-guardarCambiosEmpleado">Guardar</button>

</div>

</div>

</div>

</div>

@section Scripts {

<**script** src="~/js/index.js"></**script**>

}

Toca turno a Index.Js

JAVASCRIPT