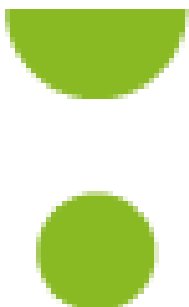




# TI-Capital Humano

Desarrollador .Net

Crear Aplicación Web  
con .Net Core y  
EntityFramework



## Contenido

1.	Crear Aplicación Web MVC.....	3
1.1.	Crear Proyecto.....	3
1.2.	Agregar Paquetes Nugets.....	3
1.3.	Mapear la Base de datos con EntityFrameWork.....	4
1.4.	Crear el Controlador .....	5
1.5.	Ejecutar la Aplicación Web MVC .....	7

# 1. Crear Aplicación Web MVC

Crear una Aplicación Web – MVC con EntityFramework, utilizando .Net Core 6., con acciones de un CRUD, utilizando únicamente las plantillas y el scaffolding.

Las acciones CRUD serán para la tabla Estados de la base de datos InstitutoTich

## 1.1. Crear Proyecto

Crear un Proyecto tipo Aplicación web ASP .Net Core (Modelo-Vista-Controlador), con el nombre de CRUDEstadosMVCCore



Eliminar la configuración para https

## Información adicional

Aplicación web de ASP.NET Core (Modelo-Vista-Controlador) C# Linux

Framework ⓘ  
.NET 6.0 (Compatibilidad a largo plazo) ▼

Authentication de campo ⓘ  
Ninguno ▼

☐ Configurar para HTTPS ⓘ

☐ Habilitar Docker ⓘ

Sistema operativo de Docker ⓘ  
Linux ▼

☐ No usar instrucciones de nivel superior ⓘ

## 1.2. Agregar Paquetes Nugets

Como se utilizará EntityFramework agregar los siguientes paquetes:

Microsoft.EntityFrameworkCore.SqlServer ver 6.0.29

Microsoft.EntityFrameworkCore.Tools ver 6.0.29

### 1.3. Mapear la Base de datos con EntityFrameWork

Utilizando ingeniería inversa ejecutar el proceso de scaffolding para obtener las clases de tipo de entidad y una clase DbContext basada en un esquema de base de datos. Utilizar el comando Scaffold-DbContext de las herramientas de consola del administrador de paquetes (PMC) de EF Core

Primeramente crear la carpeta **Models** dentro del proyecto para que colocar ahí la clase de contexto y la de entidad resultante. En esta caso solo se tomará la Tabla Estados de la Base de Datos InstitutoTich

```
Scaffold-DbContext "Server=MSI;Database=InstitutoTich;User
ID=sa;password=Pass2017;" Microsoft.EntityFrameworkCore.SqlServer -Tables
"Estados" -Context EdoContext -ContextDir Models -OutputDir Models -
NoPluralize -Force
```

Toda vez que se há creado la clase de DbContext y la de Entidad

Se deberá ajustar el método OnConfiguring de la clase de Contexto, ya que en ella el scaffolding deja las credenciales de la base de datos, mismas que deberán quedar en el archivo de configuración **appsettings.json**

```
public partial class EdoContext : DbContext
{
    // 0 referencias
    public EdoContext()
    {
    }

    // 0 referencias
    public EdoContext(DbContextOptions<EdoContext> options)
        : base(options)
    {
    }

    // 0 referencias
    public virtual DbSet<Estados> Estados { get; set; } = null!;

    // 0 referencias
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        if (!optionsBuilder.IsConfigured)
        {
            // To protect potentially sensitive information in your connection string, you should move it out of s
            optionsBuilder.UseSqlServer("Server=MSI;Database=InstitutoTich;User ID=sa;password=Pass2017;")
        }
    }
}
```

La cadena de conexión se pasará al archivo de configuración

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "InstitutoTich": "Server=MSI;Database=InstitutoTich;User ID=sa;password=Pass2017;"
  }
}
```

Además en el archivo **program.cs** se deberá cargar esta cadena de conexión como sigue:

```
using Microsoft.EntityFrameworkCore;
using CRUEstadosMVCCore.Models;

var builder = WebApplication.CreateBuilder(args);

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

builder.Services.AddDbContext<EdoContext>(opcion => {
    opcion.UseSqlServer(builder.Configuration.GetConnectionString("InstitutoTich"));
});
var app = builder.Build();
```

Agregar las referencias para la clase UseSqlServer y para la clase de contexto

El `builder.Configuration` obtendrá la cadena de conexión del archivo de configuración que se encuentre con el nombre proporcionado, en este caso `InstitutoTich`, y el `builder.Services.AddDbContext` se la pasa a la clase de contexto.

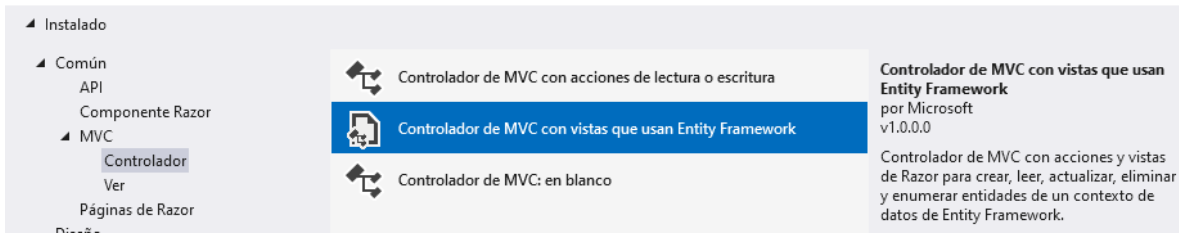
Hecho lo anterior, se deberá ajustar el método `OnConfiguring` de la clase de Contexto, dejando el cuerpo vacío.

```
0 referencias
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
}
```

## 1.4. Crear el Controlador

Crear el controlador de MVC con vistas que usan Entity Framework

Agregar nuevo elemento con scaffolding



Seleccionar la clase de modelo y la clase de contexto, en este caso Estados y EdoContext respectivamente

✕

## Agregar Controlador de MVC con vistas que usan Entity Fra

Clase de modelo Estados (CRUDEstadosMVCCore.Models)

Clase de contexto de datos EdoContext (CRUDEstadosMVCCore.Models) +

**Vistas**

☒

Generar vistas

☒

Hacer referencia a bibliotecas de scripts

☒

Usar página de diseño

...

(Dejar en blanco si se define en un archivo \_viewstart de Razor)

Nombre de controlador EstadosController

Agregar
Cancelar

Con esto el scaffolding creará el controlador con los métodos de un CRUD y las vistas correspondientes, para el modelo seleccionado

```

using Microsoft.EntityFrameworkCore;
using CRUDEstadosMVCCore.Models;

namespace CRUDEstadosMVCCore.Controllers
{
    1 referencia
    public class EstadosController : Controller
    {
        private readonly EdoContext _context;

        0 referencias
        public EstadosController(EdoContext context) {...}

        // GET: Estados
        3 referencias
        public async Task<IActionResult> Index() {...}

        // GET: Estados/Details/5
        0 referencias
        public async Task<IActionResult> Details(int? id) {...}

        // GET: Estados/Create
        0 referencias
        public IActionResult Create() {...}

        // POST: Estados/Create ...
        [HttpPost]
        [ValidateAntiForgeryToken]
        0 referencias
        public async Task<IActionResult> Create([Bind("Id,Nombre")] Estados estados) {...}

        // GET: Estados/Edit/5
        0 referencias
        public async Task<IActionResult> Edit(int? id) {...}

        // POST: Estados/Edit/5 ...
        [HttpPost]
        [ValidateAntiForgeryToken]
        0 referencias
        public async Task<IActionResult> Edit(int id, [Bind("Id,Nombre")] Estados estados) {...}

        // GET: Estados/Delete/5
        0 referencias
        public async Task<IActionResult> Delete(int? id) {...}

        // POST: Estados/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        0 referencias
        public async Task<IActionResult> DeleteConfirmed(int id) {...}

        1 referencia
        private bool EstadosExists(int id) {...}
    }
}

```

## 1.5. Ejecutar la Aplicación Web MVC

Con lo hecho anteriormente esta lista la Aplicación Web con acciones y vistas de CRUD

localhost:5158

CRUDEstadosMVCCoré Home Privacy

## Index

[Create New](#)

Nombre	
Aguascalientes	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Baja California	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Baja California Sur	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Campeche	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Chihuahua	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Chiapas	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Coahuila	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Colima	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Ciudad de México	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Durango	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>