**WEB API .NET CORE**

Una API es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.

Para ver los tipos de proyectos disponlbles, si aparece web api

**dotnet new --list**

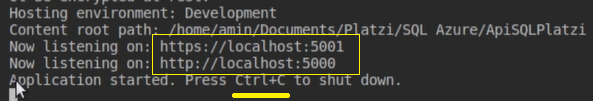
Para instalar las plantilla de Web Api:  
**dotnet new --install “Microsoft.DotNet.Web.ProjectTemplates.5.0”**

Para crear un project webApi:  
**dotnet new webapi -n “ProjectSQLAzure”**

cd ProjectSQLAzure

Para compilar el proyecto:

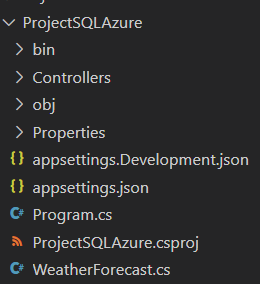
**dotnet run**



http://localhost:5000/api/<controlador>

<http://localhost:5000/api/values>

-CLI DOTNET CORE:  
[**https://bit.ly/2BcxKmF**](https://bit.ly/2BcxKmF)



Instale:

dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Tools

Al ejecutar el comando dotnet ef me daba un error que no reconocía la instrucción:

Lo solucione de la siguiente manera:

* dotnet tool install --global dotnet-ef
* dotnet ef migrations add InitialCreat

CONTROLLER

using Microsoft.AspNetCore.Mvc;

namespace ProjectSQLAzure.Controllers;

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

    private static readonly string[] Summaries = new[]

    {

        "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

    };

    private readonly ILogger<WeatherForecastController> \_logger;

    public WeatherForecastController(ILogger<WeatherForecastController> logger)

    {

        \_logger = logger;

    }

    [HttpGet(Name = "GetWeatherForecast")]

    public IEnumerable<WeatherForecast> Get()

    {

        return Enumerable.Range(1, 5).Select(index => new WeatherForecast

        {

            Date = DateTime.Now.AddDays(index),

            TemperatureC = Random.Shared.Next(-20, 55),

            Summary = Summaries[Random.Shared.Next(Summaries.Length)]

        })

        .ToArray();

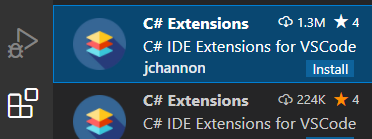
    }

}

**Modelos**

Cree carpeta Models

Instalar C# Extensions



Instale:

* dotnet add package Microsoft.EntityFrameworkCore
* dotnet add package Microsoft.EntityFrameworkCore.SqlServer
* dotnet add package Microsoft.EntityFrameworkCore.Tools

Verifique todo bien

* dotnet restore
* dotnet build

Right click / New Class / Contact.cs

using System.ComponentModel.DataAnnotations;

namespace Test.Models

{

    public class Contacts

    {

        [Key]

        public int id {get; set;}

        public string nombre {get; set;} = “”;

        public int telefono {get; set;}

    }

}

DBContext vincular

En Models cree ContactContext.cs

using Microsoft.EntityFrameworkCore;

namespace ProjectSQLAzure.Models

{

    public class ContactContext: DBContext

    {

public ContactContext(DbContextOptions<ContactContext> options) : base(options)

{

        }

        // Donde todo viene de SQL

        public DbSet<Contacts> ContactSet{get; set;}

    }

}

En appsetting.json

  "ConnectionString": {

      "Conn": "-----"

  },

En program.cs

using Microsoft.EntityFrameworkCore;

using Test.Models;

var builder = WebApplication.CreateBuilder(args);

var Conn = builder.Configuration.GetConnectionString("Conn");

builder.Services.AddDbContext<ContactContext>(options =>

    options.UseSqlServer(Conn));

Controlador

Cierre y salve todo

dotnet build

dotnet run

[ApiController]

[Route("api/autores")]

**public** **class** AutoresController : ControllerBase

{

    [HttpGet]

**public** ActionResult<List<Autor>> Get()

    {

        //Por ahora una lista estática hasta que creemos la bd

**return** **new** List<Autor>()

        {

**new** Autor { Id = 1, Nombre = "Gerson" },

**new** Autor { Id = 2, Nombre = "Luis" },

**new** Autor { Id = 3, Nombre = "Antonio" }

        };

    }

}

[HttpGet]

**public** **async** Task<ActionResult<List<Autor>>> Get()

{

**return** **await** context.Autor.ToListAsync();

}

ApplicationDbContext.cs

**public** **class** ApplicationDbContext : DbContext //hacemos que herede de DbContext y generamos constructor con options

{

**public** ApplicationDbContext(DbContextOptions options) : **base**(options)

    {

    }

**public** DbSet<Autor> Autor { **get**; **set**; } //el nombre de esta propiedad será el que tenga la tabla generada en la BD

}

Startup.cs

**public** **void** ConfigureServices(IServiceCollection services)

        {

            services.AddControllers();

            //Configurando la conexión a la BD y el contexto de datos

            services.AddDbContext<ApplicationDbContext>(options =>

                options.UseSqlServer(Configuration.GetConnectionString("defaultConnection")));

            services.AddSwaggerGen(c =>

            {

                c.SwaggerDoc("v1", **new** OpenApiInfo { Title = "WebAPIBiblioteca", Version = "v1" });

            });

        }

CONTROLADOR

[HttpPost]

**public** **async** Task<ActionResult> Post(Autor autor)

{

    context.Add(autor);

**await** context.SaveChangesAsync();

**return** Ok();

}

[HttpPut("{id:int}")]//especificamos un parámetro de ruta

**public** **async** Task<ActionResult> Put(Autor autor, **int** id)

{

**if** (autor.Id != id)

**return** BadRequest("El id del autor no coincide");

**var** existe = **await** context.Autor.AnyAsync(x => x.Id == id);

**if** (!existe)

**return** NotFound();

    context.Update(autor);

**await** context.SaveChangesAsync();

**return** Ok();

}

[HttpDelete("{id:int}")]

**public** **async** Task<ActionResult> Delete(**int** id)

{

**var** existe = **await** context.Autor.AnyAsync(x => x.Id == id);

**if** (!existe)

**return** NotFound();

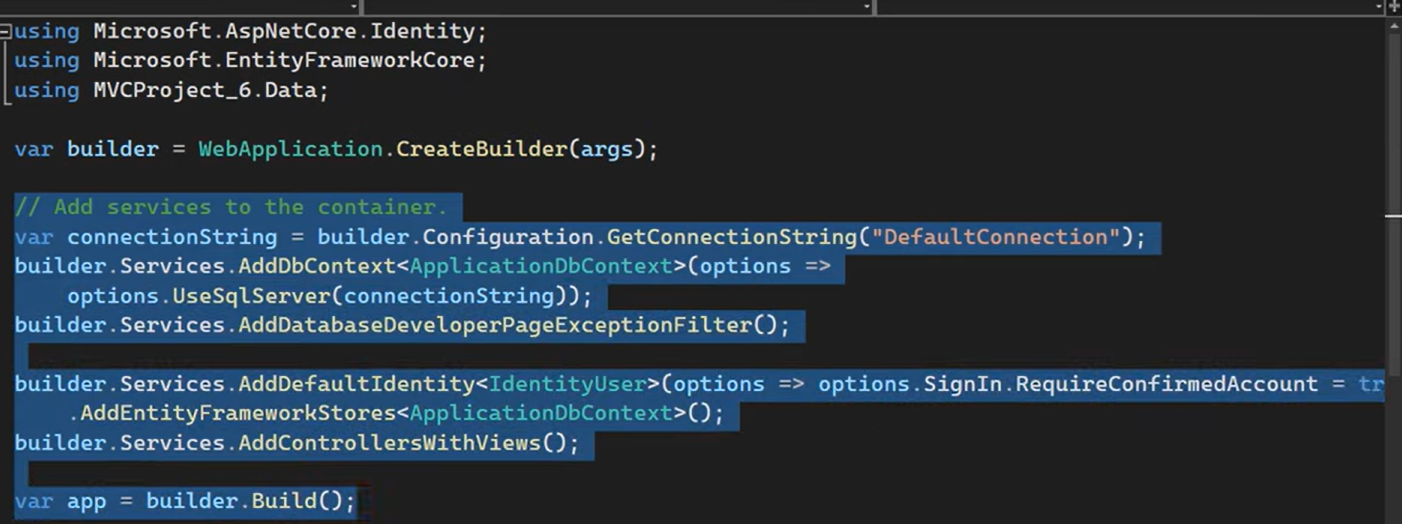
    context.Remove(**new** Autor() { Id = id });

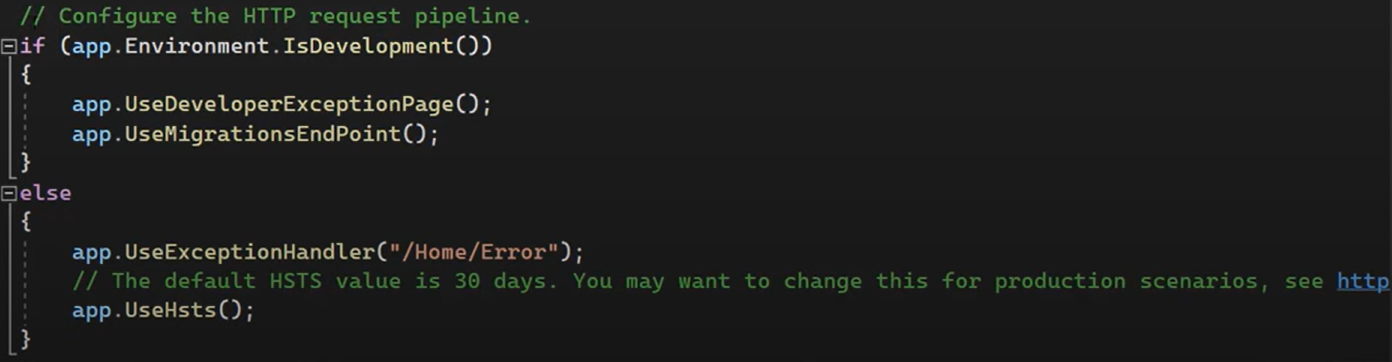
**await** context.SaveChangesAsync();

**return** Ok();

}

Program.cs





**Controllers**

[Route("api/[controller]")]

[ApiController]

public class ValuesController : ControllerBase

{

    // GET api/values

    [HttpGet]

    public ActionResult<IEnumerable<string>> Get()

    {

        return new string[] { "value1", "value2" };

    }

    // GET api/values/5

    [HttpGet("{id}")]

    public ActionResult<string> Get(int id)

    {

        return "Welcome to AspSolution.net";

    }

    // POST api/values

    [HttpPost]

    public void Post([FromBody] string value)

    {

    }

    // PUT api/values/5

    [HttpPut("{id}")]

    public void Put(int id, [FromBody] string value)

    {

    }

    // DELETE api/values/5

    [HttpDelete("{id}")]

    public void Delete(int id)

    {

    }

}

**Output**

