New terminal

Create a web API by running dotnet new:

cd..

cd temp

dotnet new web -o PizzaStore -f net6.0

cd PizzaStore

|  |  |
| --- | --- |
| Install the Swashbuckle package: | dotnet add package Swashbuckle.AspNetCore --version 6.2.3 |
| Install the EntityFrameworkCore.InMemory package: | dotnet add package Microsoft.EntityFrameworkCore.SqlServer |

appsettings.json

  "ConnectionStrings": {

    "Conexion": "data source=(localdb)\\MSSQLLocalDB;Initial Catalog=Test;Integrated Security=True;"

  }

program.cs

using Microsoft.OpenApi.Models;

using Microsoft.EntityFrameworkCore;

using PizzaStore.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();

ConfigurationManager configuration = builder.Configuration;

var conn = configuration.GetConnectionString("Conexion");

builder.Services.AddDbContext<PizzaDb>(options => options.UseSqlServer(conn));

builder.Services.AddSwaggerGen(c =>

    {

          c.SwaggerDoc("v1", new OpenApiInfo {

                Title = "PizzaStore API", Description = "Making the Pizzas you love", Version = "v1"

          });

     }

);

var app = builder.Build();

app.UseSwagger(); app.UseSwaggerUI(c =>

    {

          c.SwaggerEndpoint("/swagger/v1/swagger.json", "PizzaStore API V1");

     }

);

app.MapGet("/", () => "Hello World!");

// LIST

app.MapGet("/pizzas", async (PizzaDb db) => await db.Pizzas.ToListAsync());

// ONE ITEM

app.MapGet("/pizza/{id}", async (PizzaDb db, int id) => await db.Pizzas.FindAsync(id));

// INSERT

app.MapPost("/AddPizza", async (PizzaDb db, Pizza pizza) =>

{

    await db.Pizzas.AddAsync(pizza);

    await db.SaveChangesAsync();

    return Results.Created($"/pizza/{pizza.Id}", pizza);

});

app.MapPost("/InicioPizza", async (PizzaDb db, Pizza pizza) =>

{

    Pizza pizza1 = new Pizza { Id = 1, Name = "pizza1", Description="another" };

    await db.Pizzas.AddAsync(pizza1);

    await db.SaveChangesAsync();

    Pizza pizza2 = new Pizza { Id = 2, Name = "pizza2", Description="another222" };

    await db.Pizzas.AddAsync(pizza2);

    await db.SaveChangesAsync();

    return Results.Created($"/pizza/{pizza.Id}", pizza);

});

app.MapPut("/UpdPizza/{id}", async (PizzaDb db, Pizza updatepizza, int id) =>

{

    var pizza = await db.Pizzas.FindAsync(id);

    if (pizza is null) return Results.NotFound();

    pizza.Name = updatepizza.Name;

    pizza.Description = updatepizza.Description;

    await db.SaveChangesAsync();

    return Results.NoContent();

});

// DELETE

app.MapDelete("/DelPizza/{id}", async (PizzaDb db, int id) =>

{

  var pizza = await db.Pizzas.FindAsync(id);

  if (pizza is null)

  {

    return Results.NotFound();

  }

  db.Pizzas.Remove(pizza);

  await db.SaveChangesAsync();

  return Results.Ok();

});

app.Run();

Models\pizzaDb.cs

using Microsoft.EntityFrameworkCore;

namespace PizzaStore.Models

{

    class PizzaDb : DbContext

    {

        public PizzaDb(DbContextOptions<PizzaDb> options) : base(options) {

        public DbSet<Pizza> Pizzas { get; set; }

    }

}

Models\pizzas.cs

    public class Pizza

    {

      public int Id { get; set; } = 1;

      public string? Name { get; set; }

      public string? Description { get; set; }

  }

}

app.MapGet("/LINQ", (PizzaDb db) =>

{

      List<Persona> Persona = db.Persona.ToList<Persona>();

      List<Books> Libros = db.Books1.ToList<Books>();

      var query = from A in Libros

                  join B in Persona

                   on A.Author equals B.id

            select new

            {

                B.Nombre,

                A.Titulo

           };

        return query;

});

app.MapGet("/StoredProc", (PizzaDb db) =>

{

string query = "exec GetListado @Json out";

string query = "exec GetListado @Json out";

SqlParameter param = new SqlParameter("Json", SqlDbType.NVarChar, -1)

{

Direction = ParameterDirection.Output

};

db.Database.ExecuteSqlRaw(query, param);

string json = param.Value.ToString();

return json;

});