Projeto CRUD com ASP.NET Core (Minimal API) – Explicação Completa para Iniciantes

# 1. Produto.cs – Modelo da Tabela

public class Produto  
{  
 public int Id { get; set; } // Identificador único (chave primária)  
 public string Nome { get; set; } = string.Empty; // Nome do produto  
 public decimal Preco { get; set; } // Preço do produto  
}

📌 Essa classe define a estrutura da tabela no banco de dados.

# 2. AppDbContext.cs – Gerenciador do Banco (DbContext)

using Microsoft.EntityFrameworkCore;  
  
public class AppDbContext : DbContext  
{  
 public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }  
  
 public DbSet<Produto> Produtos => Set<Produto>();  
}

🔍 Explicação:  
- DbContext é a ponte entre sua aplicação e o banco.  
- DbSet<Produto> representa a tabela Produtos.

# 3. Program.cs – Arquivo Principal da API

using Microsoft.EntityFrameworkCore;  
  
var builder = WebApplication.CreateBuilder(args);  
  
builder.Services.AddEndpointsApiExplorer(); // Gera documentação automática da API  
builder.Services.AddSwaggerGen(); // Gera Swagger visual  
builder.Services.AddDbContext<AppDbContext>(options =>  
 options.UseSqlite("Data Source=produtos.db")); // Banco de dados SQLite  
  
var app = builder.Build();  
  
app.UseSwagger();  
app.UseSwaggerUI();  
  
app.MapGet("/produtos", async (AppDbContext db) =>  
 await db.Produtos.ToListAsync());  
  
app.MapGet("/produtos/{id}", async (int id, AppDbContext db) =>  
 await db.Produtos.FindAsync(id) is Produto p ? Results.Ok(p) : Results.NotFound());  
  
app.MapPost("/produtos", async (Produto produto, AppDbContext db) =>  
{  
 db.Produtos.Add(produto);  
 await db.SaveChangesAsync();  
 return Results.Created($"/produtos/{produto.Id}", produto);  
});  
  
app.MapPut("/produtos/{id}", async (int id, Produto input, AppDbContext db) =>  
{  
 var produto = await db.Produtos.FindAsync(id);  
 if (produto is null) return Results.NotFound();  
  
 produto.Nome = input.Nome;  
 produto.Preco = input.Preco;  
  
 await db.SaveChangesAsync();  
 return Results.Ok(produto);  
});  
  
app.MapDelete("/produtos/{id}", async (int id, AppDbContext db) =>  
{  
 var produto = await db.Produtos.FindAsync(id);  
 if (produto is null) return Results.NotFound();  
  
 db.Produtos.Remove(produto);  
 await db.SaveChangesAsync();  
 return Results.NoContent();  
});  
  
app.Run();

# Comandos do Terminal (Bash)

✅ Criar o projeto:  
dotnet new web -n MinhaApi  
  
✅ Adicionar os pacotes do Entity Framework Core + SQLite:  
dotnet add package Microsoft.EntityFrameworkCore.Sqlite  
dotnet add package Microsoft.EntityFrameworkCore.Tools  
  
✅ Criar a Migration inicial:  
dotnet ef migrations add InitialCreate  
  
✅ Aplicar a Migration:  
dotnet ef database update  
  
✅ Rodar o projeto:  
dotnet run  
  
✅ Acessar a API no navegador via Swagger:  
http://localhost:5000/swagger/index.html