Criar um repositório com o nome **GranShop**, marque a opção **ADD README** e na lista de opções do **git ignore** seleciona **Visual Studio**

Clonar repositório localmente

Abrir o repositório no **Visual Studio Code**

No terminal digite o comando abaixo:

```
dotnet new webapi -o GranShopAPI --use-controllers
```

No terminal execute os comandos abaixo:

```
cd GranShopAPI
```

```
dotnet add package Swashbuckle.AspNetCore
```

Abra o arquivo **GranVeiculosAPI.csproj** e altere a linha 5:

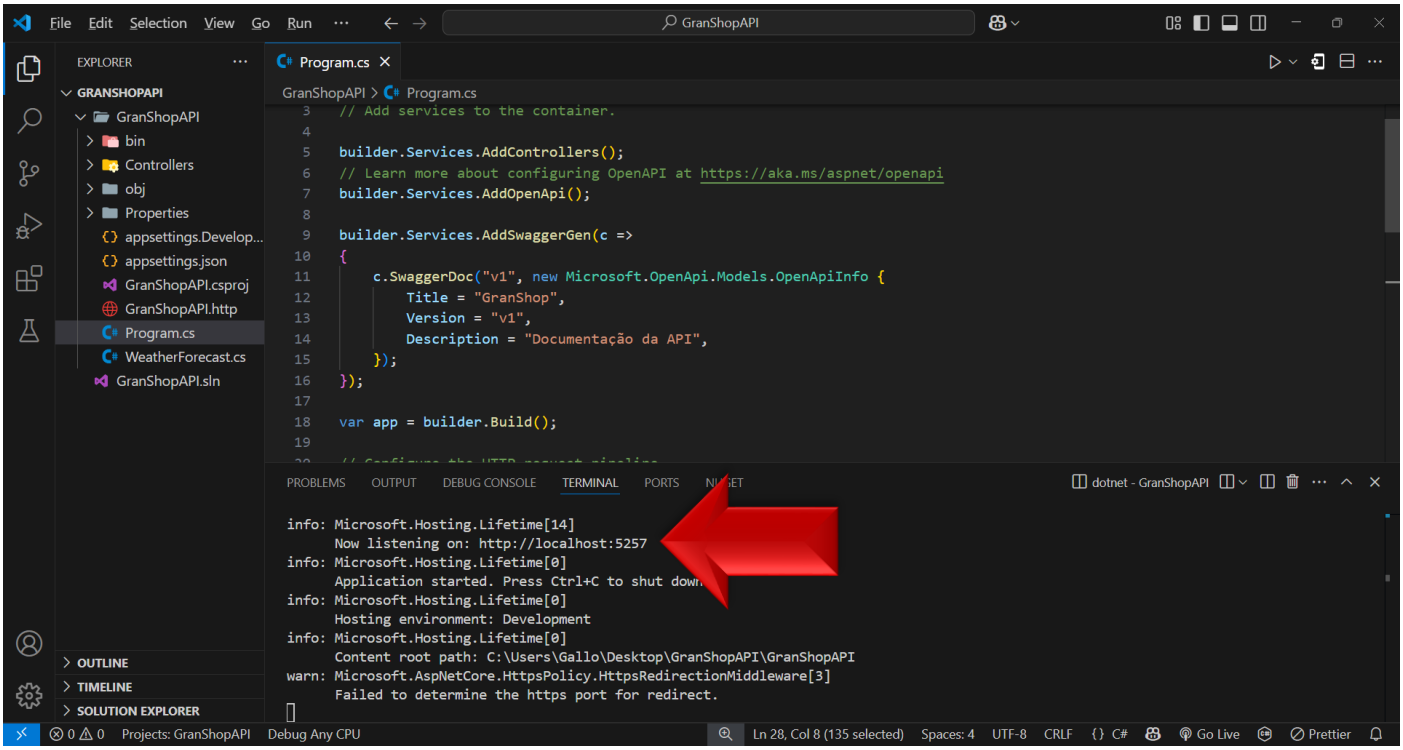
```
1  <Project Sdk="Microsoft.NET.Sdk.Web">
2
3    <PropertyGroup>
4      <TargetFramework>net9.0</TargetFramework>
5      <Nullable>disable</Nullable>
6      <ImplicitUsings>enable</ImplicitUsings>
7    </PropertyGroup>
8
9    <ItemGroup>
10     <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="9.0.5" />
11     <PackageReference Include="Swashbuckle.AspNetCore" Version="8.1.1" />
12   </ItemGroup>
13
14 </Project>
```

Agora abra o arquivo **Program.cs** e inclua as linhas, 9 a 16 e 24 a 28.

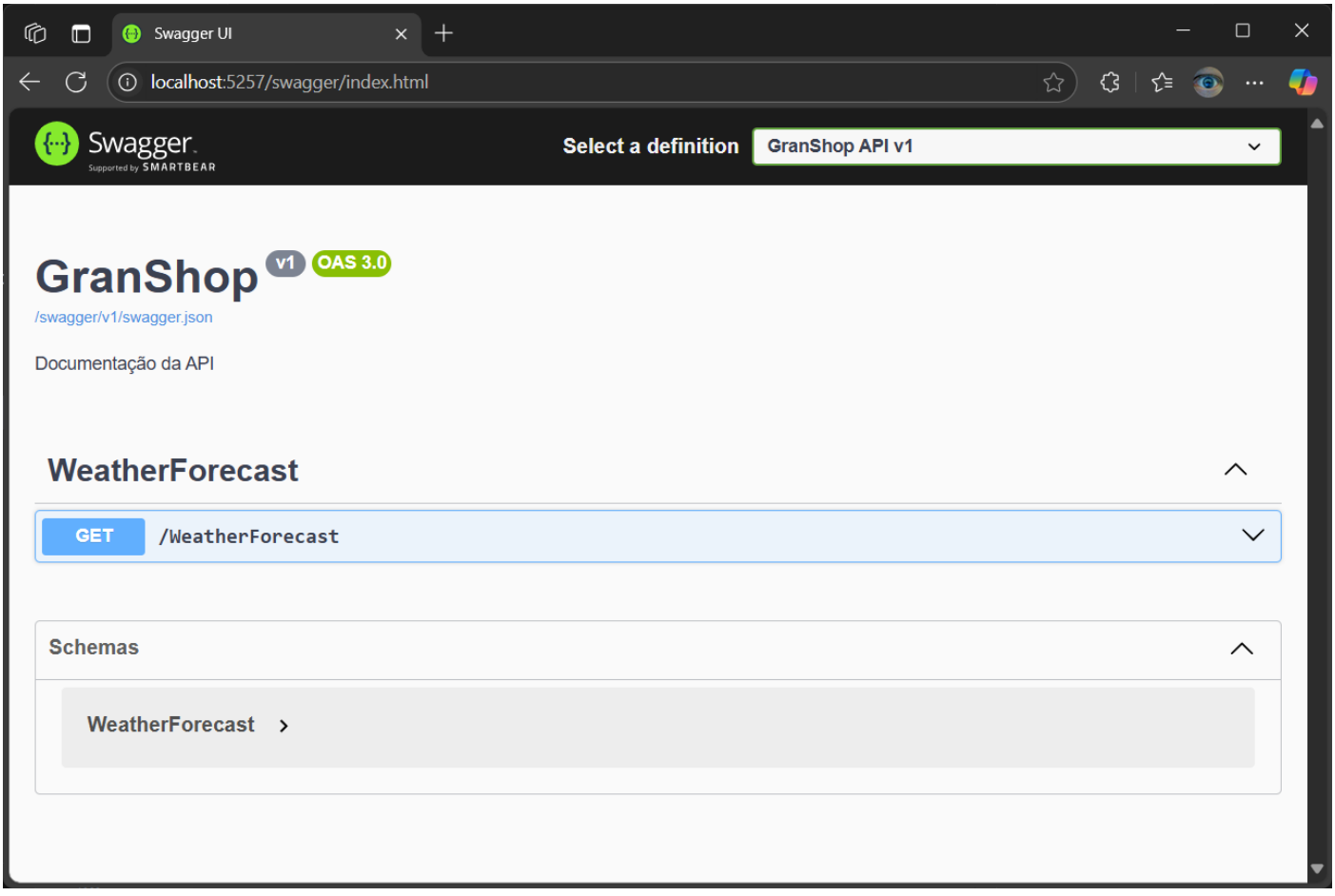
```
1  var builder = WebApplication.CreateBuilder(args);
2
3  // Add services to the container.
4
5  builder.Services.AddControllers();
6  // Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi
7  builder.Services.AddOpenApi();
8
9  builder.Services.AddSwaggerGen(c =>
10 {
11     c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo {
12         Title = "GranShop",
13         Version = "v1",
14         Description = "Documentação da API",
15     });
16 });
17
18 var app = builder.Build();
19
20 // Configure the HTTP request pipeline.
21 if (app.Environment.IsDevelopment())
22 {
23     app.MapOpenApi();
24     app.UseSwagger();
25     app.UseSwaggerUI(c =>
26     {
27         c.SwaggerEndpoint("/swagger/v1/swagger.json", "GranShop API v1");
28     });
29 }
30
31 app.UseHttpsRedirection();
32
33 app.UseAuthorization();
34
35 app.MapControllers();
36
37 app.Run();
38
```

Salve tudo e para executar o projeto no terminal execute o comando abaixo:

```
dotnet watch run
```



Pressione [Ctrl] e clique no endereço em exibição na imagem acima para abrir o navegador padrão. Em seguida na frente do endereço escreva **"/swagger"** para execução do **SWAGGER**, que é uma biblioteca de documentação de APIs, que vem instalada por padrão nas APIs do **SDK .Net**. Clique em **GET** e depois **Try Out** e **Execute** para observar o funcionamento da API.



Para parar a execução pressione no terminal **Ctrl+C.**

Cria duas pastas dentro da pasta do projeto (**GranShopAPI**), com os nomes:

- **Models**
- **Data.**

Execute os comandos abaixo, um por vez, no terminal para instalação dos pacotes de conexão com o banco de dados:

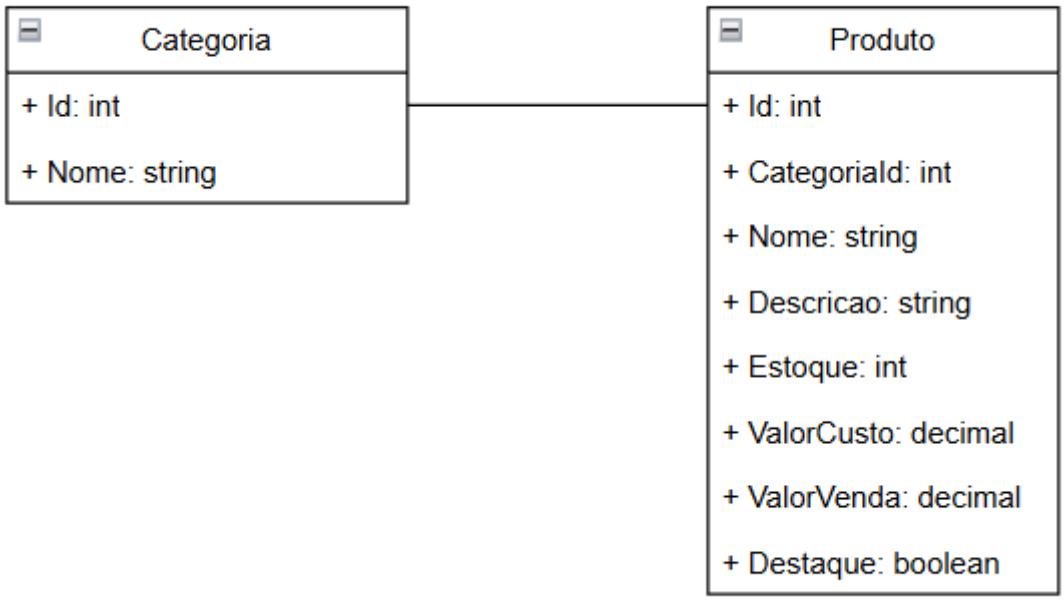
```
dotnet add package Microsoft.EntityFrameworkCore --version 9.0.5
```

```
dotnet add package Microsoft.EntityFrameworkCore.Design --version 9.0.5
```

```
dotnet add package Microsoft.EntityFrameworkCore.Tools --version 9.0.5
```

```
dotnet add package MySql.EntityFrameworkCore --version 9.0.3
```

Crie na pasta **Models** as classes do diagrama abaixo:



Crie na pasta **Data** a classe **AppDbContext**, conforme código abaixo:

```
1  using GranShopAPI.Models;
2  using Microsoft.EntityFrameworkCore;
3
4  namespace GranShopAPI.Data;
5
6  public class AppDbContext : DbContext
7  {
8      public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
9      {
10     }
11
12     public DbSet<Categoria> Categorias { get; set; }
13     public DbSet<Produto> Produtos { get; set; }
14 }
15
```

Altere o arquivo **appsettings.json**:

```
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft.AspNetCore": "Warning"
6      }
7    },
8    "AllowedHosts": "*",
9    "ConnectionStrings": {
10     "Conexao": "server=localhost;port=3306;database=granshop;uid=root;pwd=''"
11   }
12 }
13
```

Agora faça as alterações abaixo no arquivo, **Program.cs.**

Linhas 1 e 2;
Linhas 7 a 10;
Linhas 27 a 32;

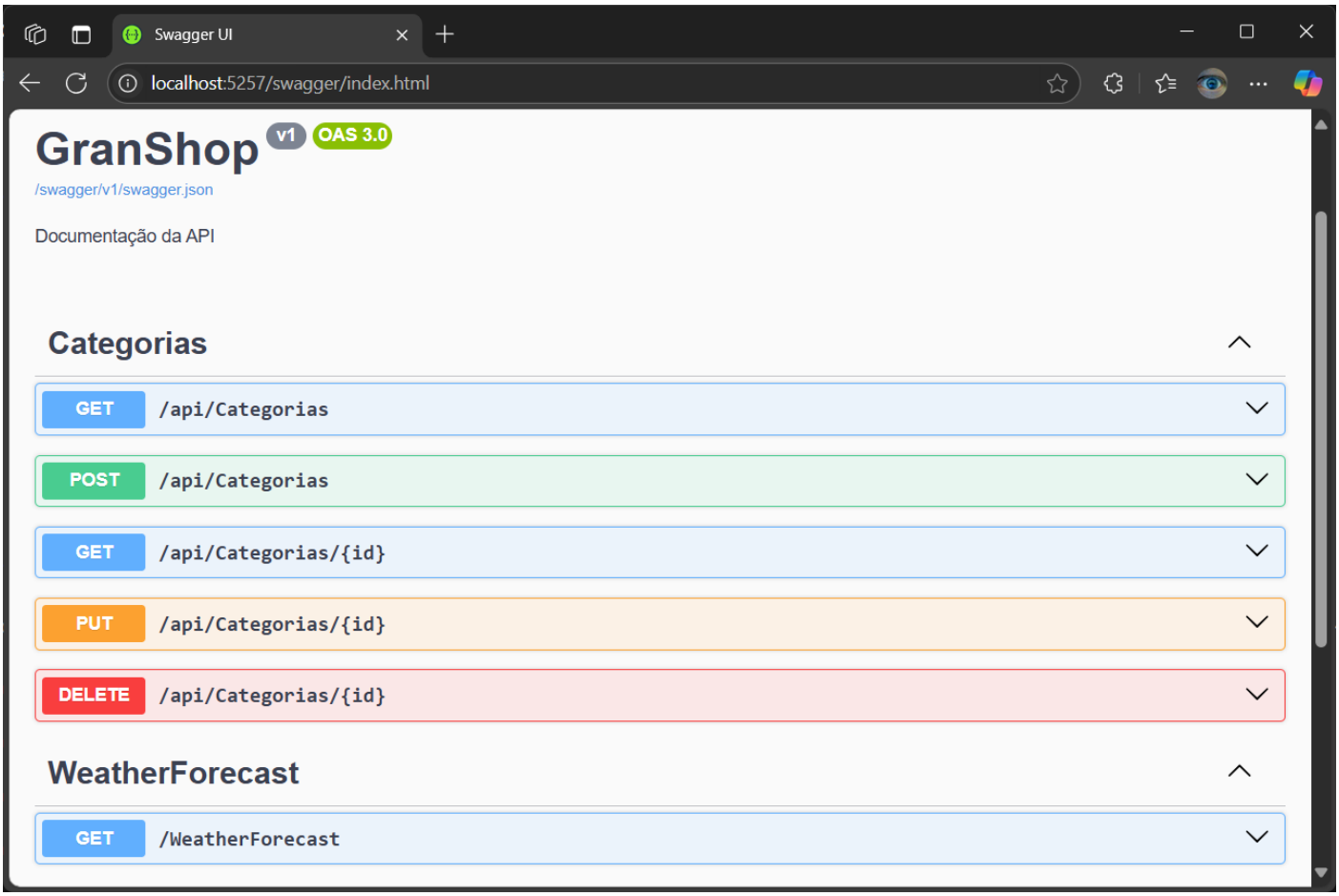
```
using GranShopAPI.Data;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
string conexao = builder.Configuration.GetConnectionString("Conexao");
builder.Services.AddDbContext<AppDbContext>(options =>
    options.UseMySQL(conexao)
);

builder.Services.AddControllers();
// Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi
builder.Services.AddOpenApi();

builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo {
        Title = "GranShop",
        Version = "v1",
        Description = "Documentação da API",
    });
});

var app = builder.Build();

using (var scope = app.Services.CreateScope())
{
    var dbContext = scope.ServiceProvider
        .GetRequiredService<AppDbContext>();
    await dbContext.Database.EnsureCreatedAsync();
}

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.MapOpenApi();
    app.UseSwagger();
    app.UseSwaggerUI(c =>
    {
        c.SwaggerEndpoint("/swagger/v1/swagger.json", "GranShop API v1");
    });
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

```

Clique com o botão direito na pasta **Controllers** e escolha a opção, **New C# > Api Controller,** coloque o nome **CategoriasController** e altere conforme a imagem:

```
using GranShopAPI.Data;
using GranShopAPI.Models;
using Microsoft.AspNetCore.Mvc;

namespace GranShopAPI.Controllers;

[ApiController]
[Route("api/[controller]")]
public class CategoriasController(AppDbContext db) : ControllerBase
{
    private readonly AppDbContext _db = db;

    [HttpGet]
    public IActionResult Get()
    {
        return Ok(_db.Categorias.ToList());
    }

    [HttpGet("{id}")]
    public IActionResult Get(int id)
    {
        var categoria = _db.Categorias.Find(id);
        if (categoria == null)
            return NotFound();
        return Ok(categoria);
    }

    [HttpPost]
    public IActionResult Create([FromBody] Categoria categoria)
    {
        if (!ModelState.IsValid)
            return BadRequest("Categoria informada com problemas");
        _db.Categorias.Add(categoria);
        _db.SaveChanges();
        return CreatedAtAction(nameof(Get), categoria.Id, new { categoria });
    }

    [HttpPut("{id}")]
    public IActionResult Edit(int id, [FromBody] Categoria categoria)
    {
        if (!ModelState.IsValid || id != categoria.Id)
            return BadRequest("Categoria informada com problemas");
        _db.Categorias.Update(categoria);
        _db.SaveChanges();
        return NoContent();
    }

    [HttpDelete("{id}")]
    public IActionResult Delete(int id)
    {
        var categoria = _db.Categorias.Find(id);
        if (categoria == null)
            return NotFound();
        _db.Categorias.Remove(categoria);
        _db.SaveChanges();
        return NoContent();
    }
}

```

IMPORTANTE: Deste ponto em diante, é necessário a execução do aplicativo XAMPP e a inicialização do banco de dados MYSQL (clicar no botão START do MySQL).

Execute e teste o **CRUD** de **Categorias** de sua **API.**



Apague os arquivos **WeatherForecast.cs** e **WeatherForecastController.cs**, estes arquivos eram apenas exemplos criados pelo **template** do **dotnet** e não tem relação com nosso projeto.

AGORA É SUA VEZ, CRIE O CONTROLLER DE PRODUTOS, SEGUINDOS OS PASSOS ATÉ AQUI E DAS AULAS PRÁTICAS