Construindo uma Web API REST com .Net Core

Wellington Júnior
Matheus Gurgel

.NET Core

REQUISITOS DE SOFTWARE

- SDK
  - .Net Core 2.1 >
  - Entity Framework
  - My SQL Server
- Ferramentas
  - Visual Studio Code
  - Postman
  - Browser a sua escolha
  - MySQL Workbench (opcional)
- Outras Ferramentas
  - Swagger UI
- Extensões VS Code
  - C# (essencial)
  - .Net Core Extension Pack
  - .Net Core Tools
  - C# Extensions
  - Essential ASP.NET Core Snippets
  - Auto-Using for C#
  - Beautify

REQUISITOS HUMANOS

- Essenciais
  - Lógica de Programação
  - Orientação a Objetos
  - Banco de Dados
- Ferramentas
  - Visual Studio Code
  - Postman
  - MySQL
- Desejáveis
  - Serviços REST
  - Injeção de dependências
  - C#
  - Entity Framework Core
  - Swagger
  - MVC

SUMÁRIO

Introdução

- .Net Core
- Serviços REST
- Postman
- Swagger / OpenAPI
- NuGet

.Net Core

- .NET Core é um framework livre e de código aberto
- Multi plataforma
- Sucessor de código aberto do .NET Framework.
- Desenvolvido principalmente pela Microsoft e lançado com a Licença MIT.
- Disponível no github https://github.com/dotnet/core

Serviços REST

- Interface pela qual os clientes podem se comunicar com uma Web API.
- Funciona sobre o protocolo HTTP(s)
- Requisições GET, POST, PUT, PATCH, DELETE

Postman

- O Postman é uma plataforma de colaborativa desenvolvimento de APIs
- Permite fazer requisições HTTP de modo fácil e prático

Swagger / OpenAPI

- É uma biblioteca para documentação de APIs REST
- Pode gerar um cliente para consumir a API automaticamente
- Multiplataforma
- Open Source

NuGet

- Galeria de Pacotes https://www.nuget.org

Construindo nossa primeira API

- Comandos dotnet
- Criação e execução do projeto
- Acesso a dados

Comandos dotnet [options]

- Ajuda: -h
  - dotnet [option] -h
- criar novo projeto
  - dotnet new webapi
- rodar o projeto: run
  - dotnet run
- assistir arquivos e auto compilar caso seja modificado: watch
  - dotnet watch run
- restaurar pacotes do projeto: restore
  - dotnet restore
- Adicionar pacotes ou referencias: add
  - dotnet add package [package reference]

Criando Nossa Primeira Web API

- dotnet new webapi

Executando o Projeto

- dotnet [watch] run

Acessando Dados

- Primeiro acesso a API: GET no Postman

Modificando Dados

- Adicionando Swagger UI
- POST, PUT e DELETE
- Validações de Modelo

Adicionando Swagger UI (**dotnet 2.1**)

- Instalando o pacote
  - *dotnet add package Swashbuckle.AspNetCore --version 1.1.0*
- Modificando a classe *Startup.cs*
  - Adicionar **AddSwaggerGen** ao service e Invocar método **SwaggerDoc**

```
services.AddMvc();
services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new Info { Title = "My API", Version = "v1" });
});
```

  - Habilitar o **Middleware** do Swagger no Configure

```
app.UseSwagger();
app.UseSwaggerUI(c =>
{
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");
});
```

Adicionando Swagger UI (**dotnet 3.0**)

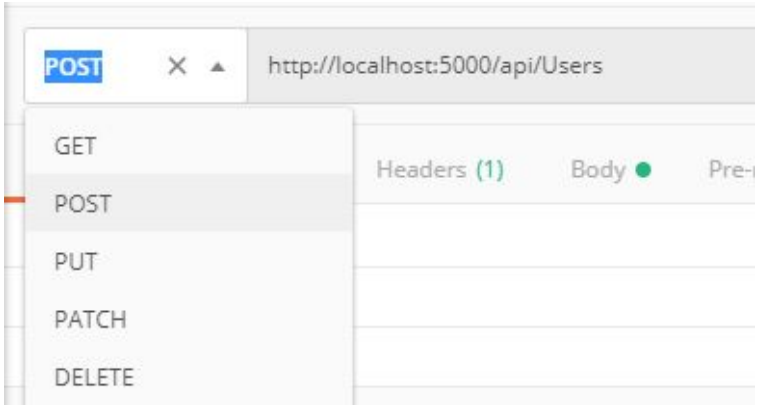
- Instalando o pacote
  - `dotnet add package Swashbuckle.AspNetCore -v 5.0.0-rc4`
- Modificando a classe *Startup.cs*
  - Adicionar **AddSwaggerGen** ao service e Invocar método **SwaggerDoc**

```
//using ValeWebLivre.Models;
services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "My API", Version = "v1" });
});
```

  - Habilitar o **Middleware** do Swagger no Configure

```
app.UseSwagger();
app.UseSwaggerUI(c =>
{
c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");
c.RoutePrefix = string.Empty;
});
```

POST, PUT, DELETE



Validações de Modelo

- [EmailAddress], [Required], [MaxLenght()], etc.

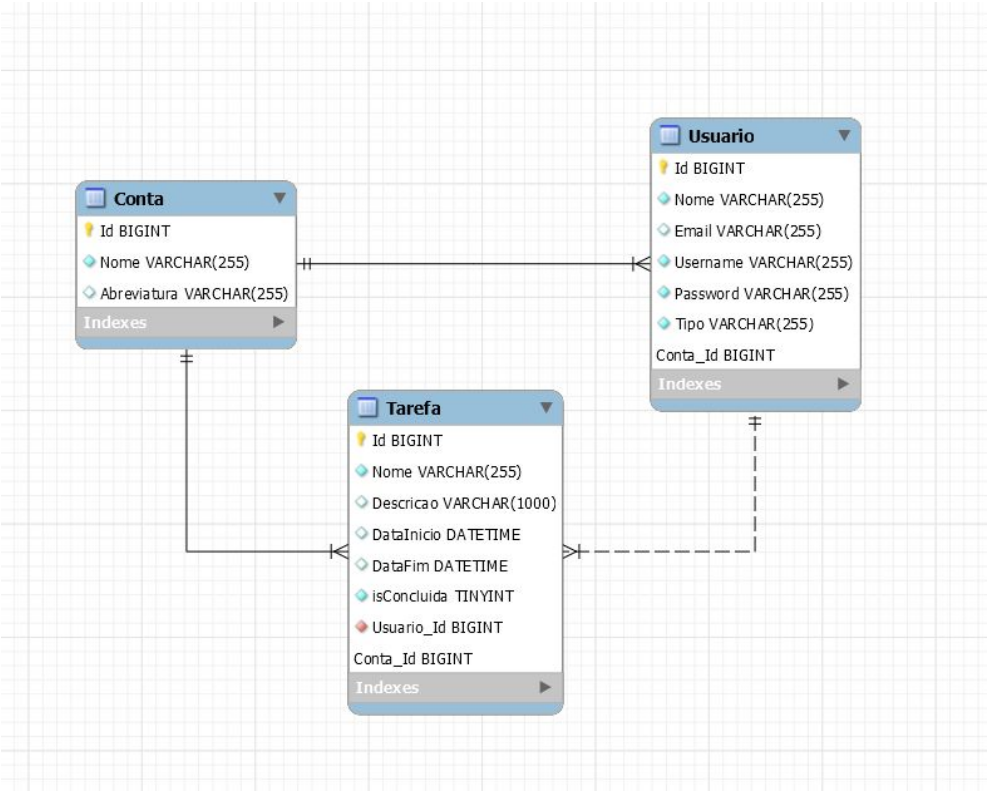
```
if (!ModelState.IsValid)
{
    return BadRequest(ModelState);
}
```

Camada de Modelo

- Criação do Models
- Acesso a entidades modelo

Entidades do Modelo

- Conta, Usuario, Tarefa



Acesso às entidades Modelo

Banco de Dados e Integração

- Entity Framework Core
- Configuração e Integração MySQL
- LINQ e Expressões Lambda
- CRUD

Entity Framework Core

- É uma versão leve, extensível, de software livre e multiplataforma do Entity Framework.
- pode servir como um ORM (Mapeador de Objeto Relacional)
- elimina a necessidade de grande parte do código de acesso aos dados
- Disponível no github https://github.com/aspnet/EntityFrameworkCore

Configuração e Integração MySQL (**dotnet 2.1**)

- Instalar pacote do Pomelo
  - dotnet add package Pomelo.EntityFrameworkCore.MySql --version 2.2.6
- Criar classe DbContext

```
namespace ProjetoWebVale.Context {
    1 reference | You, a few seconds ago | 1 author (You)
    public class ProjetoWebValeDbContext : DbContext {
        0 references
        public ProjetoWebValeDbContext (DbContextOptions<ProjetoWebValeDbContext> options) : base (options) { }
        0 references
        public DbSet<EntityName> EntityNames { get; set; }
    }
}       You, 4 days ago • adicionando banco de dados mysql
```

- Adicionar context no services no Startup.cs

```
services.AddDbContext<ProjetoWebValeDbContext>(options =>
    options.UseMySql(Configuration.GetConnectionString("DefaultConnection")));
```

- Adicionar string de conexão no appsettings.json

```
"ConnectionStrings": {
  "DefaultConnection": "server=localhost;port=3306;database=webvaledb;uid=root;password=root"
},
```

Configuração e Integração MySQL (**dotnet 3.0**)

- Instalar pacote do Pomelo
  - dotnet add package Pomelo.EntityFrameworkCore.MySql --version 3.0.0
- Criar classe DbContext

```
using Microsoft.EntityFrameworkCore;

You, 6 minutes ago | 1 author (You)
namespace ValeWebLivre.Models {
    6 references | You, 6 minutes ago | 1 author (You)
    public class ValeWebLivreDbContext : DbContext {
        0 references
        public ValeWebLivreDbContext (DbContextOptions<ValeWebLivreDbContext> options) : base (options) { }
        2 references
        public DbSet<Usuario> Usuarios { get; set; }
    }
}       You, 6 minutes ago • projeto criado
```

- Adicionar context no services no Startup.cs

```
// using Microsoft.EntityFrameworkCore;
services.AddDbContext<ValeWebLivreDbContext>(options =>
    options.UseMySql(Configuration.GetConnectionString("DefaultConnection")));
```

- Adicionar string de conexão no appsettings.json

```
"ConnectionStrings": {
  "DefaultConnection": "server=localhost;port=3306;database=webvaledb;uid=root;password=root"
},
```

LINQ e Expressões Lambda

- Where()
- FirstOrDefault()
- LastOrDefault()
- GroupBy()
- OrderBy()
- Sum()
- Distinct()
- Include()

CRUD

- Create -> POST
- Read -> GET
- Update -> PUT
- Delete -> DELETE

Aplicando Filtros

- context.Entity.Where(c => condição);

Aplicando Paginação

- Skip()
- Take()

Referências

- https://docs.microsoft.com/pt-br/dotnet/core/
- https://docs.microsoft.com/pt-br/ef/core/
- https://www.devmedia.com.br/rest-tutorial/28912
- https://swagger.io/tools/swagger-ui/
- https://www.getpostman.com
- https://docs.microsoft.com/pt-br/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-3.0&tabs=visual-studio
- https://docs.microsoft.com/pt-br/aspnet/core/mvc/models/validation?view=aspnetcore-3.0