🧠 Guia Rápido: ASP.NET Core Web API

📌 Visão Geral

- **Framework:** ASP.NET Core 9.0
- **Paradigma:** APIs RESTful
- **Linguagem:** C#
- **Banco de Dados:** SQL Server (via Entity Framework Core)
- **Ferramentas de Teste:** Postman, SwaggerWikipédia, a enciclopédia livre

🗂️ Estrutura de um Projeto Web API

- **Controllers:** Classes que herdam de `ControllerBase` e contêm os endpoints da API.
- **Models:** Representam as entidades de dados.
- **DTOs (Data Transfer Objects):** Objetos utilizados para transferência de dados entre a API e os clientes.
- **Services:** Contêm a lógica de negócios.
- **Repositories:** Responsáveis pelo acesso aos dados.
- **Startup.cs / Program.cs:** Configurações da aplicação, incluindo serviços e middlewares.

🔄 Operações CRUD com Entity Framework Core

1. **Create (POST):** Adiciona um novo recurso.
2. **Read (GET):** Recupera recursos existentes.
3. **Update (PUT/PATCH):** Atualiza recursos existentes.
4. **Delete (DELETE):** Remove recursos existentes.

Exemplo de Endpoint GET:

```
csharp
CopiarEditar
[HttpGet("{id}")]
public async Task<ActionResult<Product>> GetProduct(int id)
{
```

```
    var product = await _context.Products.FindAsync(id);
    if (product == null)
    {
        return NotFound();
    }
    return product;
}
```

🩹 Testando a API com Postman

- **GET:** Recupera dados.
- **POST:** Envia dados para criação.
- **PUT/PATCH:** Atualiza dados existentes.
- **DELETE:** Remove dados.Wikipédia, a enciclopédia livre+2Wondershare PDFelement+2Wikipédia, a enciclopédia livre+2Adobe+3Wikipédia, a enciclopédia livre+3Smallpdf+3Smallpdf+4Wikipédia, a enciclopédia livre+4PDF24 Tools+4Smallpdf+10iLovePDF - Online tools for PDF+10PDFChef+10

Dicas:

- Utilize o Swagger para visualizar e testar os endpoints da API.
- Verifique os códigos de status HTTP retornados para garantir o comportamento esperado.

⚠️ Tratamento de Erros

- **Validação de Dados:** Utilize atributos como `[Required]`, `[StringLength]`, etc., para validar os modelos.
- **Tratamento de Exceções:** Implemente middlewares para capturar e tratar exceções globalmente.
- **Respostas de Erro:** Retorne respostas padronizadas com informações úteis para o cliente.

Exemplo de Middleware de Tratamento de Erros:

```
csharp
CopiarEditar
```

```
app.UseExceptionHandler(errorApp =>
{
    errorApp.Run(async context =>
    {
        context.Response.StatusCode = 500;
        context.Response.ContentType = "application/json";
        var error = new { message = "Ocorreu um erro interno." };
        await context.Response.WriteAsJsonAsync(error);
    });
});
```

🛠️ Dicas para Evitar Problemas Comuns

- **CORS:** Configure o CORS corretamente para permitir requisições de diferentes origens.
- **Injeção de Dependência:** Registre todos os serviços e repositórios no contêiner de injeção de dependência.
- **Migrations:** Utilize o Entity Framework Core para gerenciar as migrações do banco de dados.
- **Segurança:** Implemente autenticação e autorização para proteger os endpoints da API.

📚 Recursos Adicionais

- **Documentação Oficial:**
  - Tutorial: Criar uma Web API com ASP.NET Core
  - Tratamento de Erros em Web APIs com ASP.NET Core
- **Cursos Online:**
  - Curso Web API ASP.NET Core Essencial (.NET 8 / .NET 9) - Udemy
- **Artigos e Tutoriais:**
  - ASP.NET Core 9 Web API CRUD com Entity Framework Core
  - Melhores Práticas para Tratamento de Respostas e Erros em APIs ASP.NET Core

🧠 Guia Rápido: ASP.NET Core Web API

📌 Visão Geral

- **Framework:** ASP.NET Core 9.0
- **Paradigma:** APIs RESTful
- **Linguagem:** C#
- **Banco de Dados:** SQL Server (via Entity Framework Core)
- **Ferramentas de Teste:** Postman, SwaggerWikipédia, a enciclopédia livre

🗂️ Estrutura de um Projeto Web API

- **Controllers:** Classes que herdam de `ControllerBase` e contêm os endpoints da API.
- **Models:** Representam as entidades de dados.
- **DTOs (Data Transfer Objects):** Objetos utilizados para transferência de dados entre a API e os clientes.
- **Services:** Contêm a lógica de negócios.
- **Repositories:** Responsáveis pelo acesso aos dados.
- **Startup.cs / Program.cs:** Configurações da aplicação, incluindo serviços e middlewares.

🔄 Operações CRUD com Entity Framework Core

1. **Create (POST):** Adiciona um novo recurso.
2. **Read (GET):** Recupera recursos existentes.
3. **Update (PUT/PATCH):** Atualiza recursos existentes.
4. **Delete (DELETE):** Remove recursos existentes.

Exemplo de Endpoint GET:

```
csharp
CopiarEditar
[HttpGet("{id}")]
public async Task<ActionResult<Product>> GetProduct(int id)
{
    var product = await _context.Products.FindAsync(id);
```

```
    if (product == null)
    {
        return NotFound();
    }
    return product;
}
```

🧪 Testando a API com Postman

- **GET:** Recupera dados.
- **POST:** Envia dados para criação.
- **PUT/PATCH:** Atualiza dados existentes.
- **DELETE:** Remove dados.Wikipédia, a enciclopédia livre+2Wondershare PDFelement+2Wikipédia, a enciclopédia livre+2Adobe+3Wikipédia, a enciclopédia livre+3Smallpdf+3Smallpdf+4Wikipédia, a enciclopédia livre+4PDF24 Tools+4Smallpdf+10iLovePDF - Online tools for PDF+10PDFChef+10

Dicas:

- Utilize o Swagger para visualizar e testar os endpoints da API.
- Verifique os códigos de status HTTP retornados para garantir o comportamento esperado.

⚠️ Tratamento de Erros

- **Validação de Dados:** Utilize atributos como `[Required]`, `[StringLength]`, etc., para validar os modelos.
- **Tratamento de Exceções:** Implemente middlewares para capturar e tratar exceções globalmente.
- **Respostas de Erro:** Retorne respostas padronizadas com informações úteis para o cliente.

Exemplo de Middleware de Tratamento de Erros:

```
csharp
CopiarEditar
```

```
app.UseExceptionHandler(errorApp =>
{
    errorApp.Run(async context =>
    {
        context.Response.StatusCode = 500;
        context.Response.ContentType = "application/json";
        var error = new { message = "Ocorreu um erro interno." };
        await context.Response.WriteAsJsonAsync(error);
    });
});
```

🛠️ Dicas para Evitar Problemas Comuns

- **CORS:** Configure o CORS corretamente para permitir requisições de diferentes origens.
- **Injeção de Dependência:** Registre todos os serviços e repositórios no contêiner de injeção de dependência.
- **Migrations:** Utilize o Entity Framework Core para gerenciar as migrações do banco de dados.
- **Segurança:** Implemente autenticação e autorização para proteger os endpoints da API.

📚 Recursos Adicionais

- **Documentação Oficial:**
  - Tutorial: Criar uma Web API com ASP.NET Core
  - Tratamento de Erros em Web APIs com ASP.NET Core
- **Cursos Online:**
  - Curso Web API ASP.NET Core Essencial (.NET 8 / .NET 9) - Udemy
- **Artigos e Tutoriais:**
  - ASP.NET Core 9 Web API CRUD com Entity Framework Core
  - Melhores Práticas para Tratamento de Respostas e Erros em APIs ASP.NET Core