UNIVERSIDADE LUTERANA DO BRASIL

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CAMPUS TORRES

PAULO HENRIQUE DOS SANTOS

Avaliação Semestral

Desenvolvimento de uma Web API para Gerenciamento de Pedidos e Fornecedores

TORRES

2024

UNIVERSIDADE LUTERANA DO BRASIL

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CAMPUS TORRES

PAULO HENRIQUE DOS SANTOS

Avaliação Semestral: Desenvolvimento de uma Web API em .NET 8 apresentada ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Luterana do Brasil, abordando o desenvolvimento de um sistema de gerenciamento de pedidos e fornecedores, utilizando as boas práticas de desenvolvimento de APIs RESTful e princípios de persistência de dados com Entity Framework Core.

Prof.º Lucas Rodrigues Schwartzhaupt Fogaça

TORRES

2024

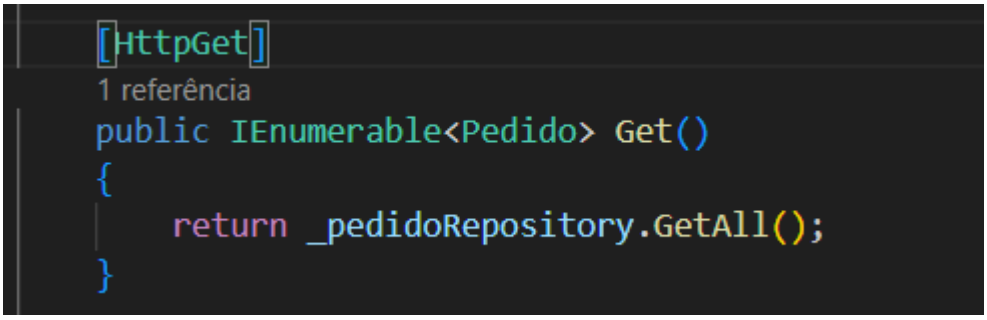
1. INTRODUÇÃO

A crescente demanda por aplicações web robustas e escaláveis tem impulsionado o desenvolvimento de APIs RESTful, uma abordagem amplamente adotada pela sua simplicidade e eficácia. Este projeto tem como objetivo explorar as boas práticas de desenvolvimento de APIs RESTful e ilustrar como essas práticas foram aplicadas na criação de uma Web API para gerenciamento de pedidos e fornecedores. A implementação das melhores práticas não só garante a funcionalidade correta da aplicação, mas também assegura manutenção e escalabilidade a longo prazo.

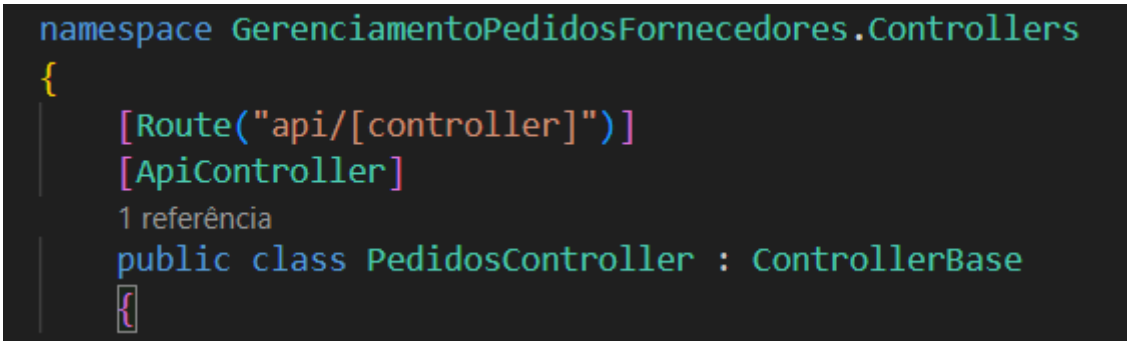
2. DESENVOLVIMENTO

- **Boas Práticas de Desenvolvimento de APIs RESTful**

Utilização de Métodos HTTP Adequados: A escolha correta dos métodos HTTP (GET, POST, PUT, DELETE) é fundamental para a semântica da API. Por exemplo, o método GET é usado para recuperar recursos, enquanto POST é utilizado para criar novos recursos.



Design de Endpoints Intuitivos e Simples: Os endpoints devem ser desenhados de maneira clara e intuitiva, refletindo a estrutura dos recursos que representam.



Mensagens de Resposta Apropriadas: Implementar respostas HTTP adequadas (200 OK, 201 Created, 400 Bad Request, 404 Not Found) para fornecer feedback apropriado ao cliente.

```
[HttpPut("{id}")]
0 referências
public IActionResult Put(int id, [FromBody] Pedido pedido)
{
    if (id != pedido.Id)
    {
        return BadRequest();
    }
    _pedidoRepository.Update(pedido);
    return NoContent();
}
```

Persistência de Dados: Utilizar o Entity Framework Core para gerenciar a persistência de dados, garantindo operações consistentes e seguras sobre o banco de dados.

```
namespace GerenciamentoPedidosFornecedores.Data
{
    9 referências
    public class AppDbContext : DbContext
    {
        0 referências
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

        6 referências
        public DbSet<Pedido> Pedidos { get; set; }
        6 referências
        public DbSet<Fornecedor> Fornecedores { get; set; }
    }
}
```

Injeção de Dependência: Facilitar a testabilidade e a manutenção do código através da injeção de dependência.

```
public class PedidosController : ControllerBase
{
    6 referências
    private readonly IPedidoRepository _pedidoRepository;

    0 referências
    public PedidosController(IPedidoRepository pedidoRepository)
    {
        _pedidoRepository = pedidoRepository;
    }
```

Aplicação das Boas Práticas no Projeto No desenvolvimento da Web API para gerenciamento de pedidos e fornecedores, foram seguidas as boas práticas de desenvolvimento para garantir um código limpo, organizado e fácil de manter.

Métodos HTTP Adequados: Cada operação CRUD foi mapeada para o método HTTP correspondente, garantindo a semântica correta das operações.

Design de Endpoints: Os endpoints foram nomeados de forma clara e intuitiva, facilitando o entendimento e a utilização da API.

Mensagens de Resposta: As respostas HTTP foram cuidadosamente implementadas para fornecer feedback significativo ao usuário da API.

Persistência de Dados: O Entity Framework Core com SQLite foi utilizado para gerenciar a persistência de dados de maneira eficiente.

Injeção de Dependência: A injeção de dependência foi aplicada para separar a lógica de acesso a dados dos controladores, aumentando a manutenibilidade e testabilidade do código.

3. **CONCLUSÃO**

O desenvolvimento de APIs RESTful seguindo as boas práticas abordadas é essencial para garantir a criação de sistemas robustos, escaláveis e fáceis de manter. No projeto de gerenciamento de pedidos e fornecedores, a aplicação dessas práticas foi fundamental para o sucesso do desenvolvimento, resultando em uma API clara, eficiente e preparada para evoluções futuras. Continuar a seguir essas diretrizes contribuirá significativamente para a qualidade das soluções desenvolvidas.