

ULBRA - TORRES
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

AS - API EM C#

Miguel Elias Bernardes

Lucas Rodrigues Schwartzhaupt Fogaça

Torres - RS

2024

INTRODUÇÃO

Este projeto foi desenvolvido com o objetivo de criar uma API para gerenciar fornecedores e pedidos de forma simples e eficiente. Ele permite que você registre, atualize, exclua e consulte fornecedores e pedidos. A ideia principal foi usar tecnologias modernas, como o ASP.NET Core e o Entity Framework Core, para construir uma aplicação leve e modular.

A API também conta com o Swagger, que ajuda na visualização e teste das rotas de forma prática. Para o banco de dados, usamos o SQLite, uma opção ideal para projetos menores e para desenvolvimento local.

DESENVOLVIMENTO

Tecnologias utilizadas

- **ASP.NET Core 7.0:** O "coração" da aplicação. Usamos este framework para criar a API de forma rápida e organizada.
- **Entity Framework Core:** Ferramenta que facilita o trabalho com o banco de dados, transformando tabelas em objetos e vice-versa.
- **SQLite:** Banco de dados pequeno e prático, perfeito para testes e projetos menores.
- **Swagger:** Ferramenta que documenta automaticamente a API e permite fazer testes diretamente pelo navegador.

Estrutura do Projeto

O projeto foi dividido de forma organizada para facilitar a leitura e manutenção:

- **Controllers:** São os arquivos que definem as rotas da API, como onde criar, atualizar ou deletar dados.
 - **FornecedoresController:** Gerencia tudo relacionado aos fornecedores.
 - **PedidosController:** Gerencia os pedidos.

"No método **Post**, adicionei lógica para alterar dados de um pedido já criado anteriormente no sistema como mostra abaixo!

```
[HttpPost]
0 references
public async Task<IActionResult> Post([FromBody] Pedido pedido)
{
    await _repository.AddAsync(pedido);
    return CreatedAtAction(nameof(GetById), new { id = pedido.Id }, pedido);
}
```

- **Models:** Contêm as classes que representam o formato dos dados, como os campos de um pedido ou fornecedor.
 - **Fornecedor:** Representa os fornecedores no sistema.
 - **Pedido:** Representa os pedidos feitos.

```

12 references
public class Pedido
{
    3 references
    public int Id { get; set; }
    0 references
    public DateTime Data { get; set; }
    0 references
    public decimal ValorTotal { get; set; }
    0 references
    public string Status { get; set; }
    0 references
    public string Descricao { get; set; }

    0 references
    public int? FornecedorId { get; set; }
    2 references
    public Fornecedor? Fornecedor { get; set; }
}

```

- **Repositories:** São responsáveis por acessar e manipular os dados no banco.
 - **FornecedorRepository** e **PedidoRepository:** Implementam as ações de leitura, escrita, atualização e exclusão no banco.
- **Banco de Dados e Migrações:** O Entity Framework foi usado para criar e atualizar o banco de dados de forma automática, gerando as tabelas com base nas classes criadas.

gerenciadorpedidos.db

Filter 5 tables... Rows: 1

TABLES		Id	Data	Descricao	Fornec...	Sta
> Fornecedores						
> Pedidos		1	2	2024-12-04 01:31:21.112	nome ta	NULL co
> __EFMigrationsHistory		2				
> __EFMigrationsLock						
> sqlite_sequence						

CONCLUSÃO

O projeto foi um sucesso em termos de aprendizado e funcionalidade. Ele mostra como ferramentas modernas como o ASP.NET Core e o Entity Framework Core podem ser usadas para criar uma API robusta e organizada.

Com ele, foi possível entender melhor como estruturar um projeto, trabalhar com banco de dados de forma prática e documentar tudo de maneira clara. Além disso, ferramentas como o Swagger facilitaram muito o teste da API.

O sistema está pronto para ser usado e expandido no futuro, adicionando novas funcionalidades ou até migrando para um banco de dados maior em produção.