ULBRA - UNIVERSIDADE LUTERANA DO BRASIL CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

SANDRA HELOISA GUINCHESKI DE SOUZA

APLICAÇÃO WEB API UTILIZANDO .NET 8

TORRES, RIO GRANDE DO SUL
2024

SUMÁRIO

1. RESUMO

O projeto explora as práticas de desenvolvimento de APIs RESTful com foco nas tecnologias e padrões usados no .NET, que incluem o uso do Entity Framework Core e o padrão Repository. Esse artigo aborda como essas práticas são implementadas em um projeto que gerencia pedidos e fornecedores em um sistema web. O projeto desenvolvido segue as práticas de estruturação de APIs que garantem modularidade, escalabilidade e fácil manutenção. Também é discutido os benefícios da utilização do padrão Repository e do ORM Entity Framework Core, bem como suas implementações e integrações na prática.

2.INTRODUÇÃO

No contexto atual de desenvolvimento de software, a construção de APIs RESTful tem se tornado uma necessidade crescente para integração de sistemas distribuídos e para promover a escalabilidade de aplicações. As APIs RESTful são amplamente adotadas devido à sua simplicidade, flexibilidade e ao uso padronizado de métodos HTTP para comunicação entre cliente e servidor. O uso de boas práticas de desenvolvimento é essencial para garantir a eficácia e a manutenção do sistema ao longo do tempo. Este trabalho tem como objetivo apresentar essas práticas, relacionadas à implementação de uma API RESTful utilizando o .NET, especificamente com o padrão Repository e o Entity Framework Core, e a maneira como essas abordagens podem ser aplicadas em um projeto prático para gerenciar pedidos e fornecedores.

3. DESENVOLVIMENTO

3.1 BOAS PRÁTICAS NO DESENVOLVIMENTO DE APIs RESTful

O desenvolvimento de APIs RESTful deve seguir práticas recomendadas para garantir a clareza, a escalabilidade e a manutenção da aplicação. A documentação RestfulAPI.net sugere várias boas práticas, entre as quais se destacam:

Uso Adequado dos Verbos HTTP. Cada verbo HTTP (GET, POST, PUT, DELETE) tem um propósito específico e deve ser utilizado de forma correta. O GET é usado para recuperar dados, o POST para criação, o PUT para atualização e o DELETE para remoção de recursos.

Estrutura de URLs deve ser intuitiva e seguir um padrão lógico. No projeto, as URLs dos recursos como pedidos e fornecedores são compostas de forma simples e expressiva, sem a necessidade de verbos, já que o verbo HTTP já define a ação.

Códigos de Status HTTP Adequados. O retorno de códigos de status HTTP como 200 OK, 201 Created, 404 Not Found e 400 Bad Request é essencial para garantir a comunicação correta sobre o estado da requisição.

Validação de Entrada e Tratamento de Erros. A validação dos dados de entrada é uma prática fundamental para garantir que a API receba apenas dados válidos. Em casos de falha, a API deve retornar mensagens de erro claras e explicativas, permitindo que o desenvolvedor identifique e corrija problemas de forma eficiente.

Documentação da API. A documentação interativa, como a fornecida pelo Swagger, é crucial para a compreensão e utilização da API por outros desenvolvedores. No projeto em questão, o Swagger foi configurado para gerar uma documentação detalhada e interativa da API, facilitando os testes e a compreensão dos endpoints disponíveis.

3.2 O PADRÃO REPOSITORY NO .NET

O padrão Repository é um padrão de design utilizado para encapsular a lógica de acesso a dados, separando a lógica de negócios da manipulação direta do banco de dados. A documentação da Microsoft sobre o padrão Repositpry no .NET destaca que esse padrão ajuda a centralizar as operações de persistência de dados e facilita a manutenção do sistema.

No projeto desenvolvido, PedidoRepository e FornecedorRepository são responsáveis por gerenciar o acesso aos dados, realizando operações como criação, leitura, atualização e exclusão de pedidos e fornecedores. O uso do repositório promove a modularidade e facilita o teste da aplicação, além de permitir que a lógica de acesso a dados seja reutilizada de forma consistente em toda

aplicação. A separação da lógica de negócios da camada de persistência ajuda a manter o código limpo e organizado.

3.3 ENTITY FRAMEWORK CORE

O Entity Framework Core (EF Core) é uma ferramenta poderosa para mapear objetos de domínio para banco de dados relacionais em aplicações .NET. COmo descrito na documentação oficial do EF Core, o framework permite que os desenvolvedores trabalhem com dados de maneira orientada a objetos, sem a necessidade de escrever SQL diretamente.

No projeto, o AppDbContext foi configurado pra usar o SQLite como banco de dados, permitindo que os dados dos pedidos e fornecedores fossem armazenados e recuperados de maneira eficiente. O EF Core também facilita a migração do banco de dados, garantindo que as alterações na estrutura de dados fossem facilmente refletidas no banco sem a necessidade de intervenção manual.

Além disso, o EF Core oferece suporte a consultas assíncronas, o que melhora o desempenho e evita bloqueios durante operações de leitura e escrita no banco de dados. O uso do DbContext foi crucial para a integração entre a aplicação e o banco de dados, permitindo a realização de operações CRUD de maneira simples e eficiente.

3.4 APLICAÇÃO PRÁTICA NO PROJETO

A implementação do projeto seguiu as boas práticas mencionadas anteriormente, com uma estrutura de API RESTful bem definida, utilizando verbos HTTP apropriados para cada operação. Os repositórios foram utilizados para separar a lógica de acesso a dados a lógica de negócios, garantindo que os controladores permanecessem focados na manipulação dos pedidos e fornecedores.

O Entity Framework Core foi usado para interagir com o banco de dados, realizando operações de persistência de maneira transparente e eficiente. A configuração do SQLite como banco de dados foi simples e eficaz, e o uso de migrations permitiu que as alterações no modelo de dados fossem facilmente aplicadas ao banco.

A documentação da API foi gerada automaticamente pelo swagger, permitindo que os desenvolvedores testassem e explorassem os endpoints de forma interativa. O uso de Swagger ajudou a garantir que todos os recursos da API fossem bem documentados, facilitando a integração com outros sistemas e garantindo a compreensão do funcionamento da API.

4. CONCLUSÃO

O desenvolvimento de uma API RESTful eficaz exige a adoção de boas práticas que garantam a escalabilidade, segurança e manutenibilidade do sistema. No projeto desenvolvido, foram aplicadas essas boas práticas, utilizando os princípios do REST, o padrão Repository e o Entity Framework Core. A separação de responsabilidades entre a camada de dados e a lógica de negócios foi alcançada, resultando em uma aplicação modular e fácil de manter. A documentação gerada pelo swagger e o uso de metodologias como o Entity Framework Core e o padrão Repository garantiram um processo de desenvolvimento ágil e eficiente, facilitando a integração e os testes da API.

5. REFERÊNCIAS

1. RestfulAPI.net. (n.d.). *Boas práticas em APIs RESTful*. Disponível em: https://restfulapi.net/
2. Microsoft Docs. (n.d.). *Padrão Repository no .NET*. Disponível em: https://learn.microsoft.com/pt-br/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design
3. Microsoft Docs. (n.d.). *Entity Framework Core - Documentação*. Disponível em: https://learn.microsoft.com/pt-br/ef/core/