DESENVOLVIMENTO DE UMA API REST PARA GERENCIAMENTO DE TUTORES E PETS UTILIZANDO C# E ENTITY FRAMEWORK CORE

INTRODUÇÃO

Este artigo tem como objetivo descrever o desenvolvimento de uma aplicação Web API RESTful utilizando a linguagem C# e o Entity Framework Core, com persistência de dados em banco SQLite. A aplicação foi implementada como atividade avaliativa (AP1) e visa permitir o gerenciamento de tutores e seus respectivos pets. O sistema conta com funcionalidades de cadastro, listagem, atualização e exclusão (CRUD), demonstrando a aplicação prática de conceitos como arquitetura MVC, repositórios e injeção de dependência.

DESENVOLVIMENTO

Requisitos Funcionais Atendidos:

1. **Cadastro de Tutores e Pets**

   A aplicação permite o cadastro de tutores e pets através das rotas HTTP POST. O método Post do TutorController recebe um objeto Tutor e o adiciona no banco por meio do repositório AddTutorAsync. O mesmo acontece no PetController, com a verificação se o Tutor existe antes de cadastrar o pet, atendendo ao requisito de integridade relacional.

2. **Listagem de Dados**

   A listagem de todos os tutores e pets é feita via método GET (GetAllAsync). As rotas GET api/tutor e GET api/pet retornam todos os registros. Também é possível buscar por ID com GET api/tutor/{id} e GET api/pet/{id}, implementados com retorno 404 caso o item não exista.

3. **Atualização de Registros**

   A rota PUT permite atualizar dados existentes. O ID passado na URL é validado e os dados enviados são persistidos com os métodos UpdateAsync dos repositórios, respeitando o padrão Repository.

4. **Exclusão de Dados**

   A exclusão de um tutor ou pet é realizada via DELETE. O sistema verifica se o item existe antes de tentar removê-lo. A exclusão só ocorre se o item for encontrado, garantindo a robustez do sistema.

5. **Documentação com Swagger**

   A aplicação foi configurada para gerar automaticamente a interface Swagger em ambiente de desenvolvimento, permitindo testar todas as rotas da API diretamente no navegador.

Requisitos Não Funcionais Atendidos:

1. **Persistência com SQLite**

   A escolha do banco SQLite atende ao requisito de simplicidade e leveza, ideal para aplicações de pequeno porte ou acadêmicas.

2. **Arquitetura Limpa e Padrões de Projeto**

   A aplicação utiliza o padrão Repository para separação da lógica de acesso a dados e o padrão MVC (Model-View-Controller), embora não haja "View" neste caso específico por se tratar de uma API.

3. **Injeção de Dependência**

   A injeção de dependência foi configurada no Program.cs com os serviços dos repositórios IPetRepository e ITutorRepository, garantindo a flexibilidade e testabilidade da aplicação.

4. **Migrations Automatizadas**

   O código inclui a linha db.Database.Migrate() no início da aplicação para aplicar migrações automaticamente ao iniciar, facilitando o gerenciamento do banco de dados.

CONCLUSÃO

Durante o desenvolvimento da aplicação, um dos principais desafios foi lidar com erros relacionados à criação e atualização do banco de dados SQLite, especialmente ao trabalhar com migrações do Entity Framework. Em alguns momentos, foi necessário excluir o banco manualmente e refazer as migrações para resolver conflitos e garantir que a estrutura estivesse correta. Outro ponto desafiador foi entender como a injeção de dependência e os repositórios funcionavam na prática, principalmente ao integrar os dados entre as entidades Tutor e Pet. Apesar das dificuldades iniciais, a organização do código com o padrão Repository e a estrutura em camadas contribuiu para uma maior clareza no projeto. Além disso, o uso do Swagger facilitou bastante os testes durante o desenvolvimento, tornando o processo de validação das rotas mais ágil

REFERÊNCIAS BIBLIOGRÁFICAS

- MICROSOFT DOCS. Entity Framework Core. Disponível em: https://learn.microsoft.com/ef/core

- MICROSOFT DOCS. ASP.NET Core Web API. Disponível em: https://learn.microsoft.com/aspnet/core/web-api

- MICROSOFT CORPORATION. *CRUD Operations in ASP.NET Core*. Disponível em: https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/crud.