Sistema de Gerenciamento de Clínicas Veterinárias

1. Introdução

Este trabalho apresenta o processo de desenvolvimento de uma API Web voltada para o gerenciamento de clínicas veterinárias. A aplicação foi criada como parte da avaliação da disciplina de Modelagem de Software no curso de Análise e Desenvolvimento de Sistemas. O sistema tem como foco principal a gestão de tutores e seus respectivos animais de estimação, oferecendo funcionalidades completas de cadastro, consulta, atualização e exclusão de registros.

O cenário utilizado considera o funcionário da clínica como usuário principal do sistema. Toda a modelagem foi baseada em diagramas de caso de uso, de classe e na definição prévia dos requisitos do projeto.

2. Desenvolvimento

2.1 Requisitos Funcionais

- **1 - Cadastro de Tutor:** É possível registrar novos tutores através do endpoint `POST /api/tutores`.
- **2 - Atualização de Tutor:** Atualizações são feitas via `PUT /api/tutores/{id}`.
- **3 - Remoção de Tutor:** A exclusão de tutores ocorre por meio do `DELETE /api/tutores/{id}`.
- **4 - Consulta de Tutores:** Os dados podem ser consultados com `GET /api/tutores` e `GET /api/tutores/{id}`.
- **5 - Cadastro de Pet:** Pets são cadastrados vinculados a um tutor usando o endpoint `POST /api/pets`.
- **6 - Atualização de Pet:** As informações dos pets podem ser atualizadas com `PUT /api/pets/{id}`.
- **7 - Remoção de Pet:** A exclusão é feita via `DELETE /api/pets/{id}`.
- **8 - Consulta de Pets:** Os dados dos animais podem ser acessados através de `GET /api/pets` e `GET /api/pets/{id}`.

2.2 Requisitos Não Funcionais

- **1 - Persistência:** O sistema utiliza o Entity Framework Core com banco de dados SQLite, configurado via `AppDbContext` com `UseSqlite`.

- **2 - Controle de Versão:** Todo o código-fonte está versionado em um repositório público no GitHub, com commits bem organizados.

- **3 - Arquitetura e Boas Práticas:** O projeto segue o padrão Repository e faz uso de injeção de dependência, configurada no `Program.cs`.

- **4 - Validações:** A entrada de dados é validada com anotações como `[Required]`, `[Phone]` e `[EmailAddress]`.

3. Conclusão

Ao longo do desenvolvimento, foi possível aplicar diversos conceitos essenciais da engenharia de software, como separação de responsabilidades, uso de padrões de projeto, arquitetura REST e práticas de validação de dados. Alguns dos desafios enfrentados incluíram a configuração correta de migrações, o relacionamento entre entidades e a implementação de validações específicas.

O resultado foi um sistema modular, bem estruturado, que atende aos requisitos propostos e segue boas práticas de desenvolvimento, tornando-se um exemplo sólido de aplicação prática dos conhecimentos adquiridos ao longo do curso.

4. Referências

1. Microsoft Docs. *Entity Framework Core Documentation*. Disponível em: https://learn.microsoft.com/en-us/ef/core/

2. https://ulbra.instructure.com/courses/16192/modules/items/391890

3. Lerman, J. *Programming Entity Framework*. O'Reilly Media, 2010.

4. https://ulbra.instructure.com/courses/16192/modules/items/398555