# Cuidado Pet API

**Backend Profissional para Gestão de Pets — ASP.NET Core + MySQL**

---

## Visão Geral

O **Cuidado Pet API** é o núcleo backend seguro e escalável para aplicações web e mobile de gestão de animais de estimação. Desenvolvido em **C# com ASP.NET Core WebAPI** e **MySQL**, oferece endpoints RESTful robustos para autenticação, cadastro de pets, controle de vacinas, alimentação, agendamentos, gamificação e integração com IA para assistência à saúde animal.

Projetado para excelência acadêmica e uso real, o sistema adota arquitetura modular, práticas modernas de segurança (JWT, WAF, rate limiting), documentação automática (Swagger/OpenAPI) e fácil integração com qualquer frontend (web, mobile, Cordova, etc).

---

## Sumário



---

## Motivação & Objetivo

- Centralizar e proteger dados clínicos, rotinas e agendamentos de pets;
- Garantir integração transparente com frontends modernos (web/mobile);
- Prover API RESTful segura, validada e documentada para uso acadêmico, profissional e real;
- Servir de referência em arquitetura backend, segurança, gamificação e integração com IA.

---

## Funcionalidades Principais

- **Autenticação JWT:**
  Registro, login e controle de sessão seguro via token;
- **Gestão de Usuários:**
  Perfil, atualização de dados, permissões por pet;
- **Cadastro e Gerenciamento de Pets:**
  CRUD completo, compartilhamento de acesso, pontos e badges;
- **Vacinas & Saúde:**
  Registro, consulta, controle de vencimento, perguntas para IA;
- **Alimentação:**
  Registro de refeições, histórico detalhado;
- **Agendamentos:**
  Marcação, remarcação, status e histórico de consultas (veterinário, banho, tosa, etc);
- **Gamificação:**
  Pontuação automática por ações, conquistas (“Super Pet”, “Veteran Pet”, “Ultimate Pet”);
- **Assistência Inteligente:**
  Integração com API externa de IA para dúvidas de saúde animal;
- **Documentação Automática:**
  Swagger/OpenAPI com Scalar para exploração e testes.

---

## Arquitetura

- **ASP.NET Core WebAPI:**
  Controllers organizados por domínio (auth, pets, care, feed, appointments, health, user);
- **Services:**
  Lógica de gamificação, pontos e badges centralizada;
- **Middleware:**
  WAF (firewall de aplicação), logging, rate limiting, CORS, HTTPS;
- **Banco de Dados MySQL:**
  Tabelas para usuários, pets, permissões, pontos, badges, vacinas, alimentação, agendamentos;
- **Validação:**
  DataAnnotations em todos os modelos de entrada;
- **Documentação:**
  OpenAPI/Swagger com Scalar Theme.

---

## Tecnologias Utilizadas

- **C# (ASP.NET Core WebAPI)**
- **MySQL (MySqlConnector)**
- **JWT (Microsoft.IdentityModel.Tokens)**
- **Serilog (logging estruturado)**
- **Swagger/OpenAPI + Scalar**
- **Rate Limiting (AspNetCoreRateLimit)**
- **WAF Middleware customizado**
- **Integração com IA (Genesis API)**

---

## Configuração & Execução

### Pré-requisitos

- .NET 7+ SDK
- MySQL Server
- (Opcional) Ferramenta de gerenciamento de banco (DBeaver, MySQL Workbench)

### Passos

1. **Banco de Dados:**
   - Crie o banco `pet_management` e as tabelas conforme os modelos dos arquivos.
   - Ajuste a string de conexão em `SqlSettings.cs` se necessário.
2. **Configuração JWT:**
   - Defina segredo, issuer e audience em `JwtSettings.cs` e/ou `appsettings.json`.
3. **Build & Execução:**

```
dotnet restore
dotnet build
dotnet run
```

4. **Acesso:**
   - Documentação Swagger: `http://localhost:5053/scalar/v1`
   - Endpoints da API: `http://localhost:5053/`

---

## Principais Endpoints

- **Autenticação:**
  `POST /auth/login` — Login
  `POST /auth/register` — Cadastro

- **Usuário:**
  `GET /user/user/profile` — Perfil
  `PUT /user/user/profile` — Atualizar perfil

- **Pets:**
  `GET /pets` — Listar pets
  `POST /pets` — Cadastrar pet
  `DELETE /pets/{id}` — Remover pet
  `POST /pets/{petId}/trust` — Compartilhar acesso

- **Vacinas & Saúde:**
  `GET /care/{petId}/vaccines` — Listar vacinas
  `POST /care/{petId}/vaccination` — Registrar vacina
  `POST /health/health/question` — Pergunta de saúde (IA)
  `GET /health/tips` — Dica de saúde

- **Alimentação:**
  `POST /feed/{petId}/feed` — Registrar refeição
  `GET /feed/{petId}/feed` — Listar refeições

- **Agendamentos:**
  `POST /appointments/appointments` — Agendar
  `GET /appointments/appointments` — Listar
  `GET /appointments/appointments/{id}` — Detalhes
  `POST /appointments/appointments/{id}/reschedule` — Reagendar
  `GET /appointments/appointments/{id}/status` — Status

*Consulte a documentação Swagger para detalhes de parâmetros e respostas.*

---

## Segurança & Boas Práticas

- **JWT obrigatório** em todos os endpoints sensíveis
- **WAF Middleware:** Bloqueio de SQLi, XSS, RCE, path traversal, etc.
- **Permissões por ação:** Controle rigoroso de acesso por usuário/pet
- **Rate Limiting:** Proteção contra abuso de requisições
- **CORS:** Configurável conforme ambiente
- **Validação e logging detalhados**
- **Queries sempre parametrizadas**

---

## Extensibilidade & Padrões

- **Princípios SOLID e Clean Code**
- **Async/await em todo acesso a dados**
- **Controllers enxutos, services para lógica**
- **Pronto para integração com qualquer frontend (web, mobile, Cordova, etc)**
- **Documentação automática e explorável**

---

## Licença


Distribuído sob Licença MIT, para fins educacionais, institucionais e profissionais.


---

**Este backend representa um case de referência em arquitetura, segurança, integração e usabilidade para sistemas de gestão de pets, pronto para integração com frontends modernos e publicação em ambientes reais ou acadêmicos.**