Projeto Acapra BeckEnd

Sistema de adoção para Pets resgatados

Equipe de Desenvolvimento:
João Vitor Pereira
Leonardo Tachini
Luis Felipe Barnabe
Luis Felipe Peirão
Maria de Fatima Groh

31/10/2025

Sumário

1) Visão geral

O back-end do Projeto Acapra é uma API REST desenvolvida em .NET com C#, responsável por fornecer os dados e operações utilizados pelo front-end. Ele gerencia informações sobre usuários, animais e processos de adoção, mantendo regras de negócio e persistência em um banco de dados relacional.

2) Stack principal

- Framework: ASP.NET Core Web API (.NET 7 ou superior)
- Linguagem: C#
- ORM: Entity Framework Core
- Banco de dados: MySQL ou MariaDB
- Documentação de API: Swagger (gerado automaticamente)
- Autenticação: JWT (JSON Web Token)
- Padrão de arquitetura: Clean Architecture (camadas API, Application, Domain e Infra)

3) Estrutura de pastas

/ (raiz)
— Acapra.API
— Acapra.Application
— Acapra.Domain
— Acapra.Infra
— Banco de Dados
— AcapraBackEnd.sln

4) Como executar o projeto

Pré-requisitos

- .NET SDK 7.0 ou superior
- MySQL 8.0+ (ou MariaDB equivalente)
- Ferramenta global `dotnet-ef`

Passo a passo

 # 1. Restaurar dependências
dotnet restore

 # 2. Configurar ambiente (appsettings ou secrets)
 # Exemplo de configuração:
 # Acapra.API/appsettings.Development.json

 # 3. Aplicar migrações e criar o banco de dados
dotnet ef database update \
--project Acapra.Infra \
--startup-project Acapra.API

 # 4. Executar o servidor de desenvolvimento
dotnet run --project Acapra.API

A API ficará disponível em:

- `http://localhost:...` (HTTP)
- `https://localhost:...` (HTTPS)

Exemplo de configuração

```
{
"ConnectionStrings": {
"Default": "Server=localhost;Port=3306;Database=acapra;User
Id=acapra;Password=acapra;SslMode=None"
},
"Jwt": {
"Issuer": "acapra.local",
"Audience": "acapra.clients",
"Key": "chave-super-secreta"
},
"Swagger": { "Enabled": true },
"Logging": { "LogLevel": { "Default": "Information" } }
}
```

5) Principais componentes do código

- `Acapra.API`

- Contém os Controllers responsáveis por expor os endpoints da API.
- Program.cs: configura serviços, autenticação JWT, CORS e Swagger.
- Exemplos de controllers esperados:
    - `AnimalsController.cs` → CRUD de animais.
    - `AdoptionsController.cs` → operações de adoção.
    - `UsersController.cs` → autenticação e gestão de usuários.

- `Acapra.Application`

- Centraliza a lógica de aplicação e casos de uso.
- Contém os DTOs, Services e Validators.
- Intermedia comunicação entre API e domínio.

- `Acapra.Domain`

- Define as **entidades principais** e suas regras de negócio.
- Exemplo:
  public class Animal {
  public int Id { get; set; }
  public string Nome { get; set; }
  public string Especie { get; set; }
  public int Idade { get; set; }
  public string Status { get; set; }
  }

- `Acapra.Infra`

- Implementa os repositórios e o DbContext do Entity Framework Core.
- Contém as migrations responsáveis por gerar o banco de dados.
- Exemplo de configuração:
  public class AcapraDbContext : DbContext {
  public DbSet<Animal> Animais { get; set; }
  public DbSet<Usuario> Usuarios { get; set; }
  public DbSet<Adocao> Adocoes { get; set; }
  }

6) Arquitetura e fluxo de requisição

graph LR
A[Controller] --> B[Service / Application Layer]
B --> C[Domain Entities]
B --> D[Infrastructure / Repositories]
D <--> DB[(Database)]

1. O Controller recebe a requisição HTTP.
2. O Application Service processa regras e validações.
3. O Domain define o comportamento das entidades.
4. A Infraestrutura executa as operações de persistência.

7) Rotas principais

Método	Rota	Descrição
GET	/api/v1/animals	Retorna todos os animais cadastrados
GET	/api/v1/animals/{id}	Retorna detalhes de um animal
POST	/api/v1/animals	Cadastra um novo animal
PUT	/api/v1/animals/{id}	Atualiza um animal existente
DELETE	/api/v1/animals/{id}	Remove um animal
POST	/api/v1/adoptions	Cria uma solicitação de adoção
GET	/api/v1/adoptions	Lista adoções registradas

8) Resumo do Modelo de dados

Entidade	Principais campos	Relações
Animal	Id, Nome, Especie, Idade, Sexo, Porte, Status	1-N com Adoção
Usuário	Id, Nome, Email, SenhaHash, Role	1-N com Adoção
Adoção	Id, AnimalId, UsuarioId, Status, DataSolicitacao	N-1 com Animal e Usuário
Instituição	Id, Nome, Endereco, Contato	Independente

9) Segurança e autenticação

- JWT Bearer Tokens configurados em `Program.cs`.
- Endpoints protegidos por `[Authorize]`.
- Validação de credenciais e roles (`Admin`, `User`).
- CORS habilitado para o domínio do front-end (porta do Vite).

10) Erros e logs

- Middleware global captura exceções e retorna respostas JSON padronizadas.
- Erros de validação → `400 Bad Request`.
- Entidade não encontrada → `404 Not Found`.
- Exceções internas → `500 Internal Server Error`.
- Logs gerados via `Microsoft.Extensions.Logging` (console e debug).

11) Documentação da API (Swagger)

Durante o desenvolvimento, o Swagger é habilitado automaticamente.

- URL padrão: `https://localhost:5001/swagger`
- Mostra todos os endpoints, parâmetros e modelos de requisição/resposta.

12) Publicação e deploy

 # Compilar o projeto
dotnet build

 # Gerar build de produção
dotnet publish Acapra.API -c Release -o out

- O projeto pode ser hospedado em servidores Linux (Kestrel + Nginx), Windows (IIS) ou via Docker.
- Para deploy com Docker, usar multi-stage build (`sdk` + `aspnet runtime`).

13) Conclusão

O back-end do Acapra fornece uma base sólida para o funcionamento do sistema, estruturado de forma modular e escalável. A separação entre camadas garante clareza e manutenção facilitada, enquanto o uso do Entity Framework Core e da arquitetura REST torna a comunicação com o front-end eficiente.

Sua configuração simples permite iniciar rapidamente o ambiente de desenvolvimento e conectar o front-end à API, tornando o conjunto uma aplicação completa para gerenciamento de adoções e cadastro de animais.