# DOCUMENTAÇÃO TÉCNICA DO PROJETO – API DE CADASTRO DE PACIENTES

Título do Projeto: Sistema de Gerenciamento de Pacientes

Grupo: Dionathan Boing Mesquita, Marlon Zorzi Kososki, Julia Vitoria Medeiros De Almeida, Ricardo Augusto Matias da Luz.

## 1. Objetivo do Projeto

Este projeto visa desenvolver uma API RESTful completa para a gestão eficiente de cadastros de pacientes, aplicando os conceitos de modelagem de dados, rotas HTTP e integração com banco de dados. O sistema oferece funcionalidades completas para operações CRUD — criação, leitura, atualização e remoção — de registros clínicos. É ideal para clínicas, hospitais e consultórios que necessitam de um gerenciamento organizado e seguro das informações de seus pacientes.

## 2. Estrutura da Solução

### 2.1 Modelagem de Dados

A entidade central é o Paciente, modelada com atributos essenciais para fins clínicos e administrativos:  
  
- Identificação pessoal: Nome, CPF, Data de Nascimento.  
- Contato: Telefone, E-mail, Endereço.  
- Dados médicos: Tipo Sanguíneo e Alergias.  
  
Essa modelagem foi implementada como classe no domínio da aplicação e utilizada na configuração do banco de dados SQLite, garantindo integridade e consistência dos dados.

### 2.2 Arquitetura da API

A API adota o padrão Minimal API do .NET 9, com uma estrutura RESTful. Cada endpoint representa uma operação específica, promovendo clareza e modularidade. O banco de dados está integrado via Entity Framework Core, com migrações automatizadas. Além disso, a documentação da API é gerada de forma interativa com o uso do Swagger, facilitando testes e entendimento da estrutura por parte dos desenvolvedores.  
  
Tecnologias utilizadas:  
- .NET 9  
- Entity Framework Core  
- SQLite  
- Swagger (Swashbuckle)

## 3. Endpoints da API

A seguir, os endpoints implementados:

|  |  |  |
| --- | --- | --- |
| Método | Rota | Descrição |
| GET | /pacientes | Retorna todos os pacientes cadastrados |
| GET | /pacientes/{id} | Retorna um paciente específico pelo ID |
| POST | /pacientes | Adiciona um novo paciente |
| PUT | /pacientes/{id} | Atualiza os dados de um paciente existente |
| DELETE | /pacientes/{id} | Remove um paciente do sistema |

## 4. Organização do Código

O projeto foi estruturado de forma modular, com separação clara de responsabilidades:  
  
- Models/Paciente.cs – Define a entidade principal, com atributos clínicos e administrativos.  
- Data/AppDbContext.cs – Configuração do banco de dados SQLite via Entity Framework Core.  
- Endpoints/GetPacientes.cs – Listagem completa.  
- Endpoints/GetPacienteById.cs – Consulta por ID.  
- Endpoints/PostPaciente.cs – Cadastro de novos pacientes.  
- Endpoints/PutPaciente.cs – Atualização de registros.  
- Endpoints/DeletePaciente.cs – Remoção de pacientes.  
- Program.cs – Arquivo de inicialização que registra dependências, configura o pipeline, mapeia os endpoints e executa a aplicação.

## 5. Justificativa Técnica

A escolha pela arquitetura Minimal API foi motivada pela busca de simplicidade, alta performance e facilidade de manutenção. O uso do Entity Framework Core com SQLite permite rápida configuração e execução local, ideal para ambientes de desenvolvimento e pequenos sistemas. O Swagger fornece uma interface de testes interativa essencial para validação e inspeção dos endpoints implementados.