Documentação Backend - Hackaton

1. Visão Geral

O backend do projeto Hackaton é desenvolvido em .NET 8.0, utilizando uma arquitetura em camadas (N-Tier) com padrões de design modernos e boas práticas de desenvolvimento.

2. Tecnologias Principais

2.1 Framework e Linguagem

- **.NET 8.0**: Framework mais recente da Microsoft
- **C#**: Linguagem de programação principal
- **ASP.NET Core**: Framework web para construção de APIs RESTful

2.2 Banco de Dados

- **SQL Server**: Banco de dados relacional
- **Entity Framework Core**: ORM (Object-Relational Mapping)
- **Migrations**: Sistema de controle de versão do banco de dados

2.3 Segurança

- **JWT (JSON Web Tokens)**: Autenticação e autorização
- **BCrypt**: Hash de senhas
- **Bearer Authentication**: Esquema de autenticação
- **CORS**: Configurado para permitir requisições cross-origin

2.4 Documentação

- **Swagger/OpenAPI**: Documentação automática da API
- **Health Checks**: Monitoramento de saúde da aplicação

3. Estrutura do Projeto

3.1 Camadas

1. **API (Hackaton.Api)**
    - Controllers
    - Configurações

    - Middlewares
    - Segurança
2. **Application (Hackaton.Application)**
    - Serviços
    - DTOs
    - Interfaces de Serviço
3. **Domain (Hackaton.Domain)**
    - Entidades
    - Interfaces de Repositório
    - Regras de Negócio
4. **Infrastructure (Hackaton.Infrastructure)**
    - Repositórios
    - Contexto do Banco de Dados
    - Implementações de Persistência

3.2 Controllers

- **AdminController**: Gerenciamento administrativo
- **AuthController**: Autenticação e autorização
- **PacientesController**: Gestão de pacientes
- **MedicosController**: Gestão de médicos
- **ConsultasController**: Gestão de consultas
- **AgendasController**: Gestão de agendas
- **TestAuthController**: Testes de autenticação

4. Funcionalidades Principais

4.1 Autenticação e Autorização

- Login com JWT
- Refresh Token
- Proteção de rotas
- Hash seguro de senhas

4.2 Gestão de Usuários

- CRUD de Médicos
- CRUD de Pacientes
- Perfis de acesso

4.3 Agendamento

- Gestão de Agendas
- Marcação de Consultas
- Disponibilidade de Horários

5. Configurações e Middlewares

5.1 Middlewares Implementados

- Tratamento global de exceções
- CORS
- Autenticação JWT
- Health Checks
- Swagger/OpenAPI

5.2 Configurações de Segurança

- Validação de tokens JWT
- Configuração de CORS
- Proteção contra ataques comuns

6. Dependências Principais

- - Microsoft.AspNetCore.OpenApi (8.0.8)
- - Swashbuckle.AspNetCore (6.5.0)
- - Microsoft.EntityFrameworkCore.Design (8.0.8)
- - Microsoft.AspNetCore.Authentication.JwtBearer (8.0.4)
- - System.IdentityModel.Tokens.Jwt (7.4.0)
- - BCrypt.Net-Next (4.0.3)

7. Endpoints da API

7.1 Autenticação

- POST /api/auth/login
- POST /api/auth/refresh-token

7.2 Médicos

- GET /api/medicos
- POST /api/medicos

- PUT /api/medicos/{id}
- DELETE /api/medicos/{id}

7.3 Pacientes

- GET /api/pacientes
- POST /api/pacientes
- PUT /api/pacientes/{id}
- DELETE /api/pacientes/{id}

7.4 Consultas

- GET /api/consultas
- POST /api/consultas
- PUT /api/consultas/{id}
- DELETE /api/consultas/{id}

7.5 Agendas

- GET /api/agendas
- POST /api/agendas
- PUT /api/agendas/{id}
- DELETE /api/agendas/{id}

8. Monitoramento e Saúde

- Endpoint de Health Check: `/health`
- Swagger UI: `/swagger`
- Logs de exceções
- Monitoramento de performance

9. Boas Práticas Implementadas

- Injeção de Dependência
- Repository Pattern
- Service Layer Pattern
- Tratamento de Exceções
- Validação de Dados
- Separação de Responsabilidades

10. Considerações de Segurança

- Autenticação JWT
- Hash de Senhas com BCrypt
- Proteção contra CSRF
- Validação de Tokens
- Headers de Segurança
- CORS Configurado
- Tratamento de Exceções Seguro