# Documento de arquitetura Health & Med

## Introdução

Este documento descreve a arquitetura de um sistema de agendamento de consultas médicas online desenvolvido em .NET com C# , utilizando MongoDB como banco de dados e Angular como framework front-end. A aplicação tem como objetivo fornecer uma plataforma eficiente e segura para que os pacientes consigam se cadastrar e fazer seus agendamentos de consultas médicas totalmente online e listagem dos médicos disponíveis. O sistema foi projetado seguindo os princípios da Clean Architecture, garantindo uma separação clara de responsabilidades e promovendo a manutenção, escalabilidade e testabilidade do código. A arquitetura é dividida em várias camadas, incluindo API, APPLICATION, DOMAIN, INFRA.DATA, INFRA.IOC, e TESTS, cada uma responsável por um aspecto específico da funcionalidade e fluxo de dados da aplicação.

## Objetivo do sistema

O sistema de agendamento de consultas médicas online foi concebido para atender às seguintes necessidades:

• Consulta de disponibilidade dos médicos: Permitir que os pacientes consigam verificar quais médicos estão disponíveis;

• Agendamento de consultas: Facilitar o agendamento de consultas com os médicos disponíveis na data e horário que seja mais adequado ao paciente;

• Autenticação de médicos e pacientes: Garantir a segurança dos dados dos usuários através de um mecanismo robusto de autenticação baseado em tokens JWT (JSON Web Tokens).

• Escalabilidade e Performance: Suportar um grande volume de dados e requisições simultâneas, utilizando MongoDB como banco de dados para garantir alta performance e escalabilidade.

• Facilidade de Manutenção e Evolução: Utilizar uma arquitetura modular e desacoplada para permitir a fácil adição de novas funcionalidades e manutenção do sistema.

## Justificativa para a arquitetura e tecnologias escolhidas

A escolha do MongoDB como banco de dados foi motivada por sua flexibilidade no armazenamento de documentos JSON, alta performance em operações de escrita e capacidade de escalabilidade horizontal. Essas características o tornam ideal para uma aplicação que lida com dados financeiros complexos e variáveis, como o gerenciamento de ativos e transações.

O uso do C# e do .NET no backend fornece uma base sólida para o desenvolvimento, graças à robustez da linguagem, suporte para práticas de desenvolvimento orientadas a objetos, alta performance, e uma ampla gama de bibliotecas e ferramentas para segurança e comunicação com bancos de dados. Além disso, o framework .NET Core é multiplataforma, permitindo fácil implantação em diferentes ambientes, incluindo containers Docker.

**Angular** é conhecido por sua forte estrutura modular, facilitando a organização do código em módulos de funcionalidades bem definidos e oferecendo as ferramentas necessárias para o desenvolvimento da aplicação. Componentes, serviços, e módulos podem ser estruturados de acordo com os princípios da arquitetura limpa.

## Estrutura do documento

Este documento de arquitetura está organizado nas seguintes seções:

1. Descrição Geral da Arquitetura: Uma visão geral da arquitetura do sistema, detalhando cada camada e suas responsabilidades.

2. Modelo de Dados: Definição das entidades principais, incluindo suas propriedades e relacionamentos.

3. Fluxo de Dados e Casos de Uso: Descrição dos principais fluxos de dados e casos de uso da aplicação, incluindo autenticação, criação de portfólios, e registro de transações.

4. Definição dos Endpoints da API: Detalhamento dos endpoints disponíveis na API RESTful, incluindo métodos, parâmetros e exemplos de requisições e respostas.

5. Camadas da Aplicação: Explicação detalhada das responsabilidades de cada camada da arquitetura (API, Application, Domain, Infra.Data, Infra.IOC, Tests).

6. Casos de Teste: Definição dos casos de teste para garantir a qualidade e a robustez da aplicação.

Este documento é destinado a desenvolvedores, arquitetos de software, engenheiros de DevOps e outros profissionais interessados em entender a estrutura e os

fundamentos técnicos do sistema de agendamento de consultas. Ele serve como guia para o desenvolvimento, manutenção e evolução contínua da aplicação.

## Estrutura da aplicação

O projeto será dividido nas seguintes camadas:

**API**: Responsável por expor os endpoints da aplicação via HTTP usando controllers.

**APPLICATION**: Contém os casos de uso, serviços e DTOs (Data Transfer Objects). Esta camada orquestra o fluxo de dados entre a API e a camada DOMAIN.

**DOMAIN**: Contém as entidades, interfaces de repositório e regras de negócio.

**INFRA.DATA**: Implementação dos repositórios e contextos de banco de dados, bem como qualquer lógica de persistência específica.

**INFRA.IOC**: Configuração da Injeção de Dependência (Dependency Injection).

**TESTS**: Contém os testes unitários e de integração.

**Entidades e regras de negócio**

1. **Usuario**
    a. **GUID:** Identificador único
    b. **Nome:** String
    c. **Cpf:** String
    d. **Email:** String
    e. **Senha:** String
    f. **TipoCadastro**: Char(1)

2. **Medico**
    a. **GUID:** Identificador único
    b. **UsuarioGUID**: String
    c. **CRM:** String

3. **Agendamento**
    a. **GUID**: Identificador único
    b. **Data**: Datetime
    c. **MedicoGUID**: String
    d. **UsuarioGUID**: String

4. **Disponibilidade**

a. **GUID:** Identificador único
b. **MedicoGUID**: String
c. **DiaSemana**: Integer **(0 é domingo 6 é sábado)**
d. **HorarioInicio**: Time
e. **HorarioFim**: Time
f. **DataEspecifica:** Date **NULLABLE** (Pode ser nulo, para casos de disponibilidade apenas 1 dia da semana)
g. **Repeticao:** String
h. **Observacao:** String

5. **Notificacoes**
    a. **GUID**: Identificador único
    b. **CorpoEmail**: String
    c. **Assunto**: String
    d. **Destinatario:** String
    e. **UsuarioGUID**: String

## Endpoints

- **Autenticação**

```
Endpoint para autenticação de usuários
```

**POST: /Usuario/autenticar**

```
Requisição:
{
    "email": "usuario@exemplo.com",
    "senha": "senha123"
}
Resposta:
{
    "token": "JWT_TOKEN"
}
```

- **Cadastrar**

```
Cadastrar um novo usuário
```

**POST: /Usuario/cadastrar-paciente**

- Requisição:

```
{
    "nome": "João da Silva",
    "email": "joao@exemplo.com",
    "cpf": 123456789,
    "senha": "senha123",
    "tipoCadastro": "P",
}
```

- Resposta:

```
{
    "id": "1",
    "nome": "João da Silva",
    "email": "joao@exemplo.com",
    "cpf": 123456789,
    "senha": "senha123",
    "tipoCadastro": "P",
}
```

**GET: /Usuario/listar/medico**

Listar todos os médicos

- Resposta:

```
[
    {
            "id": "1",
      "nome": "João da Silva",
      "email": "joao@exemplo.com",
      "cpf": 123456789,
      "senha": "senha123",
            "crm": "123455678",
      "tipoCadastro": "M",
        },
]
```

**GET: /Usuario/listar/disponibilidade/medico/{medicoId}**

Listar todos os horários disponíveis dos médicos

- Resposta:

```
[
    {
            "data": "20/10/2024",
            "horarios": [
                {
                    dataConsulta: "20/10/2024 15:00",
                    disponivel: true,
                },
                {
                    dataConsulta: "20/10/2024 14:30",
                    disponivel: false,
                },
            ]
    }
]
```

**POST: /Usuario/cadastrar-medico**

Cadastrar um novo médico

- Requisição:

```
{
    "nome": "João da Silva",
    "email": "joao@exemplo.com",
    "cpf": "123456789",
    "senha": "senha123",
    "crm": "123455678",
    "tipoCadastro": "M",
}
```

- Resposta:

```
{
    "id": "1",
    "nome": "João da Silva",
    "email": "joao@exemplo.com",
    "cpf": 123456789,
    "senha": "senha123",
        "crm": "123455678",
    "tipoCadastro": "M",
}
```

**POST: /Paciente/marcar-consulta**
Cadastrar uma consulta médica

- Requisição:

{
**dataConsulta: string**
**idMedico string**
**idPaciente string**

**}**

- **Resposta:**

**{**
**dataConsulta: string**
**idMedico string**
**idPaciente string**
}

**GET: /Paciente/listar-consultas-agendadas/{GUID}**
Listar consultas

- Requisição:

{
**GUID: string**
}

- Resposta:

[{
**dataConsulta: string**
**idMedico string**
**idPaciente string**
}]

**GET: /Medico/listar**

Listar médicos disponíveis

- Resposta:

[{

| | |
|---|---|
| **nome:** | **string** |
| **CPF** | **string** |
| **CRM** | **String** |
| **TempoDeConsulta** | **Integer** |
| **email** | **string** |

}]

**POST: /Medico/liberar-agenda**

Liberação de agenda para o dia selecionado

Requisição

{

| | |
|---|---|
| **idMedico** | **string** |
| **dataLiberar** | string |

}

- Resposta:

{

| | |
|---|---|
| **idMedico** | **string** |
| **dataLiberar** | string |

}

**POST:** /Medico/cadastrar-periodo-atendimento

Cadastrar período de atendimento

Requisição

{

| | |
|---|---|
| **idMedico** | **string** |
| **diaDaSemana** | **integer** |
| **inicio** | **string** |

**fim** **string**
}

- Resposta:

{
**id** **string**
**idMedico** **string**
**diaDaSemana** **integer**
**inicio** **string**
**fim** **string**
}

**GET:** /Medico/listar-periodo-atendimento/{id}

Listar período de atendimento

Requisição
{

**GUID string**

}

- Resposta:

[{
**id** **string**
**idMedico** **string**
**diaDaSemana** **integer**
**inicio** **string**
**fim** **string**
}]