# **Documentação Técnica - Sistema de Agendamento de Consultas Médicas**

## **Introdução**

Este documento descreve o design da arquitetura e o desenvolvimento do Sistema de Agendamento de Consultas Médicas, desenvolvido como parte do Sub Hackathon do curso .NET da PÓSTECH.

### **Objetivo**

O objetivo deste projeto é desenvolver um sistema para agendamento de consultas médicas, permitindo que pacientes criem e gerenciem suas contas, agendem, reagendem ou cancelem consultas, e que médicos visualizem e gerenciem suas agendas de consultas.

## **Arquitetura**

O projeto foi desenvolvido seguindo o padrão de arquitetura Clean Architecture, que é uma arquitetura de software escalável e de fácil manutenção.

A Clean Architecture é composta por camadas bem definidas e independentes, cada uma com sua responsabilidade específica.

### **Camadas da Arquitetura**

1. **Domain**
   * Contém as entidades de domínio (Patient, Doctor, Appointment).
   * Contém enumerações e exceções compartilhadas.
   * Representa o núcleo da aplicação com regras de negócio e objetos fundamentais.
2. **Application**
   * Contém interfaces de serviços (IPatientService, IDoctorService, IAppointmentService, INotificationService).
   * Contém implementações de serviços (PatientService, DoctorService, AppointmentService, EmailNotificationService).
   * Esta camada orquestra as operações e aplica as regras de negócio.
3. **Infrastructure**
   * Contém o contexto do banco de dados (ApplicationDbContext) e implementações de repositórios (PatientRepository, DoctorRepository, AppointmentRepository).
   * Contém os testes unitários para repositórios e serviços.
   * A infraestrutura cuida de aspectos técnicos, como persistência de dados e comunicação externa.
4. **Web**
   * Contém os controladores (PatientController, DoctorController, AppointmentController) para lidar com as requisições HTTP.
   * O arquivo **Program.cs** configura e inicializa a aplicação.

A imagem abaixo ilustra a arquitetura geral do sistema:

A diagram of a clean architecture

Description automatically generated

## **Tecnologias Utilizadas**

* **Linguagem de Programação**: C#
* **Framework**: .NET Core
* **Banco de Dados**: Entity Framework Core (Code First)
* **Testes Unitários**: xUnit e Moq
* **Controle de Versão**: Git
* **Docker**: Utilizado para Dockerizar a aplicação (ainda não implementado neste projeto)

## **Desenvolvimento**

### **Models**

Foram definidos tres modelos principais para o sistema:

1. **AppointmentModel**: Representa uma consulta médica e possui os seguintes atributos:
   * AppointmentId
   * PatientId (chave estrangeira referenciando o paciente)
   * DoctorId (chave estrangeira referenciando o médico)
   * ScheduleID
   * CreatedAt
   * Comments
   * Status
2. **DoctorModel**: Representa um médico e possui os seguintes atributos:
   * Id
   * Name
   * Specialization
3. **PatientModel**: Representa um paciente e possui os seguintes atributos:
   * Id
   * Name
   * Email
   * PhoneNumber

### **Repositories**

Foram definidos três repositórios para cada modelo:

1. **PatientRepository**: Responsável pela comunicação com o banco de dados para operações relacionadas aos pacientes.
2. **DoctorRepository**: Responsável pela comunicação com o banco de dados para operações relacionadas aos médicos.
3. **AppointmentRepository**: Responsável pela comunicação com o banco de dados para operações relacionadas às consultas médicas.

### **Services**

Foram definidos quatro serviços de aplicação que implementam a lógica de negócios da aplicação:

1. **PatientService**: Contém os métodos para criar, listar, atualizar e excluir pacientes.
2. **DoctorService**: Contém os métodos para criar, listar, atualizar e excluir médicos.
3. **AppointmentService**: Contém os métodos para criar, listar, atualizar e excluir consultas médicas, além de cancelar consultas.
4. **NotificationService**: Responsável por enviar lembretes de consulta por e-mail.

### **Controllers**

Foram definidos três controladores para lidar com as requisições HTTP:

1. **PatientController**: Controlador responsável por lidar com as requisições relacionadas aos pacientes, como criar, listar, atualizar e excluir pacientes.
2. **DoctorController**: Controlador responsável por lidar com as requisições relacionadas aos médicos, como criar, listar, atualizar e excluir médicos.
3. **AppointmentController**: Controlador responsável por lidar com as requisições relacionadas às consultas médicas, como criar, listar, atualizar, excluir e cancelar consultas.

### **Testes Unitários**

Foram escritos testes unitários para garantir o correto funcionamento dos serviços de aplicação e dos repositórios. Foram utilizadas as bibliotecas xUnit e Moq para escrever os testes.

## **Conclusão**

O Sistema de Agendamento de Consultas Médicas foi desenvolvido seguindo as melhores práticas de arquitetura de software, utilizando a Clean Architecture para garantir um código limpo, escalável e de fácil manutenção. O projeto inclui modelos bem definidos, repositórios para acesso ao banco de dados, serviços de aplicação para implementar a lógica de negócios e controladores para lidar com as requisições HTTP.