CHALLENGE - ODONTOPREV

Turmas - TDSPB e TDSPS

análise e desenvolvimento de sistemas

ADVANCED BUSINESS DEVELOPMENT WITH .NET

Larissa Araújo

RM 96496

Larissa Lopes

RM 552628

Luna Faustino

RM 552473

Sumário

[1. Objetivo do Projeto 2](#_Toc193408835)

[2. Escopo 2](#_Toc193408836)

[3. Requisitos Funcionais 2](#_Toc193408837)

[4. Requisitos Não Funcionais 3](#_Toc193408838)

[5. Arquitetura 3](#_Toc193408839)

[6. MONOLÍTICO X MICROSERVIÇOS 3](#_Toc193408840)

[Simplicidade e Facilidade de Desenvolvimento 4](#_Toc193408841)

[Desempenho 4](#_Toc193408842)

[Manutenção e Deploy 4](#_Toc193408843)

[Justificativa para Não Usar Microservices 4](#_Toc193408844)

[7. Arquitetura do sistema 5](#_Toc193408845)

[Camadas da Aplicação 5](#_Toc193408846)

[Estrutura de Pastas 5](#_Toc193408847)

[Configuração e Inicialização 6](#_Toc193408848)

[Injeção de Dependência 6](#_Toc193408849)

[Desenvolvimento e Ambiente 6](#_Toc193408850)

[8. LINKS 6](#_Toc193408851)

# Objetivo do Projeto

Devido ao alto número de sinistros, que ocorrem quando há um uso excessivo ou indevido de serviços odontológicos, a OdontoPrev está enfrentando um desafio. Isso inclui consultas e tratamentos desnecessários, além de fraudes, o que resulta em aumento de custos.

Pensando nisso, propomos adicionar funções ao aplicativo da OdontoPrev, que ajudem a monitorar os hábitos de saúde bucal dos pacientes e analisar exames de Raio-X, utilizando Inteligência Artificial, para melhorar a triagem e o diagnóstico preventivo.

# Escopo

O projeto terá como funcionalidades principais:

* Monitoramento diário dos hábitos alimentares e de higiene bucal dos pacientes: por meio de perguntas sobre alimentação, higiene bucal e hábitos.
* Gamificação: o usuário poderá ganhar pontos, que poderão ser trocados por serviços ou descontos oferecidos pela OdontoPrev.
* Análise preditiva de Raio-X: através do upload de exames de Raio-X, um sistema de IA fará análises automaticamente desses arquivos, com o objetivo de prever possíveis problemas dentários e identificar padrões que indiquem tratamentos desnecessários ou fraudes.

# Requisitos Funcionais

* Sistema de perguntas diárias aos pacientes para monitoramento.
* Sistema de pontuação baseado em check-ins, upload de Raio-x, informações fornecidas e outras ações.
* Upload de Raio-X e análise preditiva com base nas imagens.
* Geração de relatórios detalhados.
* Monitoramento de alterações no banco de dados.

# Requisitos Não Funcionais

* O sistema deve ser escalável para suportar milhares de usuários simultaneamente.
* Deve garantir a segurança e a privacidade dos dados dos pacientes, com armazenamento seguro de imagens e dados pessoais.
* Alta disponibilidade para evitar interrupções no serviço.

# Arquitetura

A solução será implementada seguindo os princípios da Clean Architecture, garantindo uma separação clara das responsabilidades e facilitando a escalabilidade e manutenibilidade do sistema.

* Apresentação: Responsável pela interface do usuário (UI) e controle das interações no aplicativo da OdontoPrev, utilizando ASP.NET Core MVC.
* Aplicação: Gerencia os casos de uso e serviços da aplicação, como o monitoramento de hábitos dos pacientes, sistema de gamificação, e a análise preditiva de exames de Raio-X, implementando as regras de negócio essenciais para o funcionamento do sistema.
* Domínio: Representa as principais entidades do projeto, como Paciente, Dentista, Plano, Raio-X, Análise de Raio-X, Perguntas, Respostas e Extrato de Pontos, encapsulando toda a lógica de negócio associada.
* Infraestrutura: Gerencia o acesso a dados, utilizando Entity Framework Core para integração com banco de dados Oracle SQL, além de outras integrações necessárias. O projeto será implementado em .NET Core 6.0.

# MONOLÍTICO X MICROSERVIÇOS

O projeto está utilizando uma abordagem monolítica para a arquitetura da API. Aqui estão algumas justificativas para essa escolha:

## Simplicidade e Facilidade de Desenvolvimento

* Menor Complexidade Inicial: Projetos monolíticos são geralmente mais simples de desenvolver e implementar inicialmente. Todos os componentes da aplicação estão em um único projeto, o que facilita a gestão e o desenvolvimento.
* Facilidade de Testes e Debugging: Com uma única base de código, é mais fácil realizar testes e debugging, pois todas as partes da aplicação estão no mesmo lugar.

## Desempenho

* Comunicação Interna Rápida: Em uma aplicação monolítica, a comunicação entre diferentes partes do sistema é mais rápida, pois ocorre dentro do mesmo processo, sem a necessidade de chamadas de rede.

## Manutenção e Deploy

* Deploy Simples: O deploy de uma aplicação monolítica é mais simples, pois envolve apenas um único artefato. Isso reduz a complexidade de configuração e gerenciamento de múltiplos serviços.
* Menor Sobrecarga Operacional: Não há necessidade de gerenciar múltiplos serviços, o que reduz a sobrecarga operacional e a necessidade de ferramentas de orquestração e monitoramento complexas.

## Justificativa para Não Usar Microservices

* Complexidade Adicional: A arquitetura de microservices adiciona complexidade significativa em termos de comunicação entre serviços, gestão de dados distribuídos, e orquestração de serviços.
* Overhead de Comunicação: Microservices introduzem overhead de comunicação entre serviços, o que pode impactar a performance, especialmente em sistemas que não necessitam de escalabilidade extrema.
* Necessidade de Infraestrutura Avançada: Implementar microservices requer uma infraestrutura mais avançada para gerenciamento de serviços, balanceamento de carga, e monitoramento, o que pode ser desnecessário para projetos menores ou menos complexos.

# Arquitetura do sistema

A arquitetura da solução é baseada em uma abordagem monolítica, utilizando o .NET 8. Detalhada a seguir:

## Camadas da Aplicação

Camada de Apresentação (Presentation Layer)

* Controllers: Utiliza o ASP.NET Core para criar controladores que expõem endpoints HTTP. Esses controladores são responsáveis por receber as requisições, processá-las e retornar as respostas apropriadas.
* Swagger: Integrado para documentação e teste dos endpoints da API.
* Camada de Aplicação (Application Layer)
* AutoMapper: Utilizado para mapear objetos entre diferentes camadas, facilitando a conversão de DTOs para modelos de domínio e vice-versa.

Camada de Dados (Data Layer)

* Entity Framework Core: Utilizado como ORM para interagir com o banco de dados Oracle. A configuração do contexto de dados (DataContext) é feita no Program.cs.
* Repositórios: Implementações de repositórios que encapsulam a lógica de acesso a dados. Interfaces e implementações de repositórios são registradas no contêiner de injeção de dependência.

## Estrutura de Pastas

* Controllers: Contém os controladores da API.
* Data: Contém o contexto de dados (DataContext) e configurações relacionadas ao Entity Framework Core.
* Dtos: Contém os Data Transfer Objects (DTOs) utilizados para transferir dados entre a camada de apresentação e a camada de aplicação.
* Mappings: Contém perfis de mapeamento do AutoMapper.
* Models: Contém os modelos de domínio que representam as entidades do banco de dados.
* Repositories
* Interfaces: Contém as interfaces dos repositórios.
* Implementations: Contém as implementações das interfaces dos repositórios.

## Configuração e Inicialização

* Program.cs: Arquivo principal de configuração e inicialização da aplicação. Aqui são configurados os serviços, como AutoMapper, Entity Framework Core, Swagger, e os repositórios. Também é onde a aplicação é construída e executada.
* • O ConfigurationManager é inicializado com a configuração da aplicação e registrado como Singleton no contêiner de injeção de dependência.

## Injeção de Dependência

* Scoped Services: Os repositórios são registrados como serviços scoped, o que significa que uma nova instância é criada para cada requisição HTTP.
* Segurança e Proteção de Dados
* HTTPS Redirection: A aplicação está configurada para redirecionar todas as requisições HTTP para HTTPS.
* Data Protection: O serviço de proteção de dados é adicionado para garantir a segurança dos dados sensíveis.

## Desenvolvimento e Ambiente

* Ambiente de Desenvolvimento: Durante o desenvolvimento, o Swagger é habilitado para facilitar a documentação e o teste dos endpoints da API.

# LINKS

<https://github.com/larissalaga/OdontoPrev.Net-Sprint3>