Documento de Arquitetura de Software (DAS) - Gestão de Oficinas

1. Introdução

Este Documento de Arquitetura de Software (DAS) descreve a estrutura de alto nível, os componentes, os relacionamentos e os princípios de design do sistema **Gestão de Oficinas**. O objetivo é comunicar as decisões arquiteturais e fornecer uma visão técnica para o desenvolvimento e manutenção.

2. Visão Arquitetural

O sistema adota a **Arquitetura Limpa (Clean Architecture)**, que é um padrão de design que separa o software em camadas concêntricas, onde as dependências fluem de fora para dentro. Isso garante que a lógica de negócio (o núcleo) seja independente da interface do usuário, do banco de dados e de frameworks externos.

2.1. Diagrama de Camadas (Clean Architecture)

Camada	Projeto .NET	Dependências
Entidades (Core)	`GestaoOficinas.Domain`	Nenhuma (Entidades de Negócio)
Casos de Uso (Business Logic)	`GestaoOficinas.Application`	`GestaoOficinas.Domain`
Adaptadores (Data Access)	`GestaoOficinas.Infrastructure`	`GestaoOficinas.Domain`
Frameworks/UI (External)	`GestaoOficinas.API`	`GestaoOficinas.Application`, `GestaoOficinas.Infrastructure`

3. Componentes e Tecnologias

O sistema é construído sobre o ecossistema .NET, utilizando uma arquitetura limpa (API) e uma aplicação web de front-end (MVC).

Componente	Tecnologia	Função
Backend API	**ASP.NET Core 8**	Fornece os endpoints RESTful para todas as operações do sistema.
Frontend Web	**ASP.NET Core MVC**	Interface de usuário para interação com o sistema.
Banco de Dados	**PostgreSQL**	Armazenamento persistente dos dados.
ORM	**Entity Framework Core (EF Core)**	Mapeamento Objeto-Relacional e acesso a dados.
Autenticação	**JWT (JSON Web Token)**	Mecanismo de segurança para proteger os endpoints da API.
Mapeamento	**AutoMapper**	Conversão de objetos entre Entidades e DTOs.
Testes	**xUnit, Microsoft.AspNetCore.Mvc.Testing**	Frameworks para testes de integração e unitários.

4. Visão de Implantação (Deployment View)

O sistema é projetado para ser implantado em um ambiente de nuvem, seguindo um modelo de três camadas lógicas:

1. **Camada de Apresentação (Client Side):** O `GestaoOficinas.Web` (ou qualquer aplicação cliente, como um SPA ou mobile) que consome a API.
2. **Camada de Aplicação (Server Side - API):** O `GestaoOficinas.API`, que é o servidor de aplicação.

3. **Camada de Dados (Database):** O servidor PostgreSQL.

Decisão de Implantação: A API e o Frontend Web podem ser implantados separadamente ou no mesmo contêiner (se o Frontend consumir a API internamente). Recomenda-se o uso de **Contêineres Docker** para garantir a portabilidade e a consistência do ambiente entre desenvolvimento, teste e produção.

5. Padrões de Design Aplicados

O projeto faz uso de padrões de design cruciais para a manutenibilidade e escalabilidade:

Padrão	Aplicação no Projeto	Benefício
Repository	Interfaces (`IAlunoRepository`) na camada `Domain` e implementações na camada `Infrastructure`.	Abstrai a lógica de acesso a dados, permitindo a troca do banco de dados sem afetar a lógica de negócio.
Service	Classes de serviço (`AlunoService`) na camada `Application`.	Centraliza a lógica de negócio e os casos de uso, promovendo o princípio da Responsabilidade Única (SRP).
Dependency Injection (DI)	Configuração centralizada no `Program.cs` da API.	Reduz o acoplamento entre os componentes, facilitando a substituição de implementações (ex: o uso de `InMemoryDatabase` nos testes).
Data Transfer Object (DTO)	Classes DTOs na camada `Application`.	Garante que apenas os dados necessários sejam transferidos entre as camadas, protegendo as entidades de domínio.

6. Relacionamento de Dados (Modelo Lógico)

O modelo de dados é relacional e foi projetado para suportar as regras de negócio de gestão de oficinas e escolas.

Relacionamentos Chave:

- **Inscrição:** Relacionamento N:M entre `Aluno` e `Turma`.
- **Presença:** Relacionamento N:M entre `Aluno` e `Chamada`.
- **OficinaTutor:** Relacionamento N:M entre `Oficina` e `Professor` (para tutores).
- **Integridade Referencial:** O EF Core está configurado para impor restrições de chave estrangeira, com algumas regras de exclusão em cascata ou restrição (`DeleteBehavior.Restrict`) para proteger dados críticos (ex: não permitir a exclusão de um Professor responsável por uma Oficina ativa).

Este documento, em conjunto com o DRS, fornece a base para a compreensão técnica e funcional do sistema.