Arquitetura da Solução

1. Estrutura Geral da Arquitetura

A solução segue o padrão de separação de responsabilidades:
Front-End: Desenvolvido em **ASP.NET MVC** com Razor, para apresentação e interação do usuário.
Back-End: Uma **API REST** em C# com .NET Core 6.0+.
Segue o padrão **Clean Architecture**, separando em camadas: **Distribution, Application, Domain e Infra**.
Banco de Dados: SQL Server 2016 para persistência de dados.

2. Estrutura do Back-End

Abaixo estão as camadas e responsabilidades principais:

Camada 1 - Distribution

Responsável por expor os endpoints da API.
Utiliza o **MediatR** para delegar comandos e consultas, desacoplando a API da lógica de negócio.
JWT Bearer para autenticação, com dois papéis de usuário: **administrador** e **usuário comum**.

Camada 2 - Application

Implementa a lógica de negócio usando o padrão **CQRS** (Command Query Responsibility Segregation):
Comandos (Command): Executam operações de escrita (criação, atualização, exclusão).
Consultas (Query): Realizam operações de leitura.
Utiliza o **AutoMapper** para transformar objetos, como VOs em DTOs ou entidades.
Contém **Handlers** do MediatR para tratar comandos e consultas.

Camada 3 - Domain

Contém as entidades de domínio e os VOs (**Value Objects**).

Implementa regras de negócio principais, respeitando os princípios do **DDD (Domain-Driven Design)**.
Contém validações e métodos que são específicos das regras de domínio.

Camada 4 - Infra

Responsável pelo acesso ao banco de dados, serviços externos e configurações.
Entity Framework Core:
Configurações centralizadas no método `OnModelCreating` para mapeamento das entidades.
Repositórios para encapsular operações no banco.
Configuração e armazenamento de dependências utilizando **Microsoft.Extensions.DependencyInjection**.

3. Tecnologias e Como Elas Ajudam

a. MediatR

Motivo: Reduz o acoplamento entre a API e a lógica de negócio.
Como atende: Permite que a API apenas dispare comandos ou consultas sem se preocupar com os detalhes da lógica.

b. CQRS

Motivo: Separa as operações de leitura e escrita, tornando o código mais organizado e escalável.
Como atende: Facilita a manutenção e melhora a performance em operações complexas.

c. AutoMapper

Motivo: Evita código repetitivo ao mapear objetos (VO para DTO).
Como atende: Facilita a conversão de entidades para objetos que serão retornados ao cliente.

d. JWT Bearer Authentication

Motivo: Oferece segurança para a API com autenticação baseada em tokens.

Como atende: Protege os endpoints e fornece autenticação com dois papéis distintos.

e. Entity Framework Core com OnModelCreating

Motivo: Proporciona um controle centralizado de mapeamentos e configurações de entidades.
Como atende: Simplifica o gerenciamento de esquemas de banco de dados.

4. Estrutura do Projeto Back-End

Pasta: Distribution

Controllers: Endpoints da API.
Configurações: Autenticação JWT, validação de dependências.

Pasta: Application

Commands: Comandos para operações de escrita (e.g., criar, atualizar).
Queries: Consultas para operações de leitura.
Handlers: Implementações do MediatR para processar comandos e consultas.

Pasta: Domain

Entidades: Classes que representam o domínio.
Value Objects: Objetos de valor reutilizáveis.
Regras de Negócio: Métodos e validações específicas.

Pasta: Infra

Repositórios: Implementação de acesso ao banco de dados.
Mappings: Configurações do EF Core no `OnModelCreating`.
Dependências: Configuração de serviços no **Dependency Injection**.

5. Segurança com JWT

A API utiliza **JWT Bearer Authentication** para autenticação:
O token JWT é gerado após login bem-sucedido com **usuário: administrador** e **senha: senha@123**.

Os tokens permitem acessar endpoints protegidos.
O middleware valida o token antes de processar as requisições.

6. Requisitos Atendidos

Desacoplamento: MediatR reduz dependências entre camadas.
Organização: Clean Architecture separa responsabilidades.
Segurança: JWT protege os endpoints.
Performance: CQRS melhora o desempenho em consultas complexas.
Manutenibilidade: AutoMapper reduz código repetitivo.