Condominium API

A **Condominium RESTful API** é uma API projetada para gerenciar informações de um condomínio, incluindo apartamentos, blocos e pessoas. Este projeto usa uma arquitetura em camadas para garantir uma separação de responsabilidades e permitir fácil manutenção e escalabilidade.

Requisitos

- .NET 8.0
- SQL Server (ou outro banco de dados configurável)

Estrutura do Projeto

A estrutura segue o padrão de arquitetura em camadas, com as seguintes pastas principais:

- **Condominium.API**: Camada de apresentação que contém os controladores da API.
- **Condominium.Domain**: Camada que define os modelos e interfaces principais do domínio.
- **Condominium.Infrastructure**: Camada de infraestrutura, incluindo o contexto de banco de dados e repositórios.

Detalhes das Camadas

Condominium.API

- **Controllers**: Controladores para gerenciar as requisições HTTP.
- **Dtos**: Objetos de Transferência de Dados (DTOs) para comunicação entre as camadas.
- **Mappers**: Mapeadores entre os modelos de domínio e os DTOs.

Condominium.Domain

1. **Interfaces**: Interfaces para os repositórios, como IApartmentRepository, IBlockRepository, IPeopleRepository.
2. **Models**: Definições de modelos de domínio como Apartment, Block e People.

A estrutura segue o padrão de arquitetura em camadas, com as seguintes pastas principais:

Condominium.API: Camada de apresentação que contém os controladores da API.

Condominium.Domain: Camada que define os modelos e interfaces principais do domínio.

Condominium.Infrastructure: Camada de infraestrutura, incluindo o contexto de banco de dados e repositórios.

Condominium.API

Controllers: Controladores para gerenciar as requisições HTTP.

Dtos: Objetos de Transferência de Dados (DTOs) para comunicação entre as camadas.

Mappers: Mapeadores entre os modelos de domínio e os DTOs.

Condominium.Domain

Interfaces: Interfaces para os repositórios, como IApartmentRepository, IBlockRepository, IPeopleRepository.

Models: Definições de modelos de domínio como Apartment, Block e People.

Condominium.Infrastructure

Data: Contexto do banco de dados (AppDbContext) e migrações.

Repositories: Implementações dos repositórios para acesso aos dados.

A Condominium RESTful API utiliza uma arquitetura em camadas, que ajuda a separar as responsabilidades e facilita a manutenção e expansão do sistema. Cada camada desempenha um papel específico no funcionamento da aplicação, conforme descrito abaixo:

1. Camada de Apresentação - Condominium.API

Essa camada é a responsável por expor os endpoints da API e receber as requisições HTTP dos clientes (como um navegador ou aplicativo). Ela inclui os seguintes elementos principais:

Controllers: Cada controlador lida com um conjunto específico de endpoints e é responsável por gerenciar as requisições e respostas. Por exemplo, o ApartmentController gerencia os endpoints relacionados aos apartamentos. Aqui, o

controlador processa a solicitação do usuário, chama a camada de domínio e retorna o resultado (geralmente em formato JSON).

DTOs (Data Transfer Objects): São objetos que facilitam a comunicação entre o cliente e o servidor, garantindo que somente os dados necessários sejam trocados. Por exemplo, o DTO pode conter apenas os campos que precisam ser mostrados para o usuário, ocultando informações desnecessárias.

Mappers: São responsáveis por converter os dados entre os DTOs e os modelos de domínio. Isso ajuda a manter uma separação clara entre o que é exposto na API e o que é armazenado no sistema, garantindo uma maior segurança e organização.

2. Camada de Domínio - Condominium.Domain

Esta é a camada central do sistema, que contém a lógica de negócios e as regras do domínio. Ela inclui:

Interfaces: As interfaces definem os métodos que devem ser implementados pelos repositórios, como IApartmentRepository, IBlockRepository e IPeopleRepository. Por exemplo, a IApartmentRepository pode definir métodos como GetApartmentById ou AddApartment, que serão usados para gerenciar os dados de apartamentos.

Modelos: São classes que representam as entidades principais do sistema, como Apartment, Block e People. Essas classes geralmente possuem propriedades quedefinem os atributos de cada entidade, como number, floor e description para um Apartment.

A camada de domínio é independente de frameworks específicos e não conhece detalhes de como os dados são armazenados, o que melhora a testabilidade e flexibilidade do sistema.

3. Camada de Infraestrutura - Condominium.Infrastructure

A camada de infraestrutura lida com os detalhes de persistência de dados. Esta camada é responsável por se conectar ao banco de dados e executar operações de leitura, escrita, atualização e exclusão. Ela inclui:

Contexto do Banco de Dados (AppDbContext): Esta classe, geralmente derivada de DbContext (no Entity Framework Core), representa a conexão com o banco de dados e define as tabelas (DbSets) correspondentes às entidades de domínio. Por exemplo, DbSet<Apartment> representa a tabela de apartamentos no banco de dados.

Repositórios: São classes que implementam as interfaces da camada de domínio e fornecem métodos específicos para manipulação de dados, como ApartmentRepository, BlockRepository e PeopleRepository. Cada repositório lida com uma entidade específica e permite realizar operações como adicionar, atualizar, buscar e deletar informações do banco de dados.

Fluxo de Funcionamento da API

Receber a Solicitação: O cliente faz uma solicitação HTTP para um endpoint da API, como POST /api/apartments para criar um novo apartamento.

Controller Processa a Solicitação: O controlador apropriado (ApartmentController) recebe a requisição, extrai os dados e chama os serviços da camada de domínio.

Validação e Regras de Negócio: A camada de domínio aplica quaisquer regras de negócio e validações necessárias. Por exemplo, ela pode verificar se o número do apartamento já existe antes de permitir a criação de um novo.

Interação com o Banco de Dados: Através da camada de infraestrutura, a API persiste ou recupera dados do banco de dados. A ApartmentRepository, por exemplo, pode ser chamada para salvar um novo registro de apartamento.

Retorno da Resposta: O controlador transforma o resultado em um DTO (usando mapeadores) e retorna a resposta ao cliente. A resposta geralmente está em formato JSON e contém informações relevantes sobre a operação realizada.

Exemplo de Funcionamento

O usuário faz um POST /api/apartments com dados de um novo apartamento.

O ApartmentController recebe a solicitação e chama a camada de domínio para validar os dados.

Após a validação, o ApartmentRepository salva o apartamento no banco de dados através do AppDbContext.

O controlador retorna uma resposta JSON com os dados do apartamento criado.

Swagger é um conjunto de ferramentas e uma especificação padrão para construir, documentar e consumir APIs REST de forma mais eficiente. Ele é amplamente usado para definir a estrutura de APIs, facilitando a criação de documentação clara e interativa, o que torna o desenvolvimento e o consumo de APIs mais fáceis e consistentes.

Componentes do Swagger

OpenAPI Specification: Uma especificação em formato JSON ou YAML que descreve todos os endpoints da API, incluindo os métodos HTTP (GET, POST, etc.), parâmetros de entrada, tipos de resposta e códigos de status. O OpenAPI, anteriormente chamado de Swagger Specification, é uma linguagem agnóstica e amplamente adotada para documentar APIs.

Swagger UI: Uma interface web interativa que gera automaticamente a documentação da API a partir da especificação OpenAPI. Ela permite que desenvolvedores visualizem e testem endpoints diretamente no navegador, o que facilita a validação e a experimentação com a API.

Swagger Editor: Um editor online que ajuda na criação e edição de especificações OpenAPI. Com ele, é possível visualizar a documentação em tempo real enquanto se escreve o código da API.

Exemplo utilização do Swagger