DOCUMENTAÇÃO TÉCNICA DO PROJETO – API ESTOQUE

Título do Projeto: Sistema de Gerenciamento de Estoque

Grupo: Rafael dos Santos, João Pedro dos Santos

Tópico 1: Resumo do Projeto e do Problema Abordado

O projeto "ApiEstoque" consiste em uma API RESTful desenvolvida para a gestão de acervo de produtos em um estoque. O objetivo principal é fornecer uma interface robusta e eficiente para as operações CRUD (Create, Read, Update, Delete) de itens, visando agilidade no controle de inventário em diversos contextos comerciais. A API é construída utilizando C# com o framework .NET e o Entity Framework Core para acesso a dados, com o banco de dados SQL Lite para persistência das informações.

O problema central que o projeto aborda é a necessidade de um sistema eficiente para gerenciar o inventário de produtos. Empresas de diversos portes frequentemente enfrentam desafios na organização, rastreamento e análise de seus estoques. Um controle ineficiente pode levar a perdas financeiras por excesso ou falta de produtos, dificuldade em identificar itens populares ou parados, e imprecisão nos relatórios de vendas e compras. Este sistema visa mitigar esses problemas, oferecendo uma solução centralizada e programática para a manipulação dos dados de estoque.

Tópico 2: Lista das Funcionalidades Implementadas

O sistema "ApiEstoque" implementa as seguintes funcionalidades principais:

API RESTful Completa (CRUD para Produtos):

Consulta de Produtos (GET): Permite obter a lista completa de todos os produtos em estoque ou um produto específico por seu ID.

Cadastro de Produtos (POST): Possibilita a adição de novos produtos ao estoque.

Edição de Produtos (PUT): Permite a atualização de informações de um produto existente, como nome, quantidade, preço e descrição.

Exclusão de Produtos (DELETE): Habilita a remoção de um produto do estoque com base em seu ID.

Frontend Funcional e Interativo:

Tela de Cadastro de Dados.

Tela de Listagem de Produtos.

Opções de Edição na Listagem.

Opções de Remoção na Listagem.

Tela de Análise/Relatório de Estoque.

Cálculo do Valor Total do Estoque.

Identificação de Produtos com Baixo Estoque.

Interface Responsiva.

Tópico 3: Descrição Técnica das Funcionalidades e da Integração com a API

O projeto é dividido em duas camadas principais: o backend (API RESTful em C#) e o frontend (HTML, CSS, JavaScript).

Backend (API RESTful em C# com .NET e Entity Framework Core):

Rotas da API: As operações são definidas em classes estáticas separadas na pasta Rotas para melhor organização:

ROTA_GET.cs: Contém os métodos MapGet para listar todos os produtos (/api/produtos) e consultar um produto por ID (/api/produtos/{id}). Inclui também um redirecionamento da rota raiz / para /produtos.

ROTA_POST.cs: Implementa o método MapPost para adicionar novos produtos, persistindo-os no banco de dados.

ROTA_PUT.cs: Adicionado para lidar com a atualização de produtos. Recebe o ID na URL e os dados atualizados no corpo da requisição. Ele busca o produto existente e atualiza suas propriedades antes de salvar as mudanças no banco. Retorna 204 No Content em caso de sucesso.

ROTA_DELETE.cs: Gerencia a exclusão de produtos por ID via método MapDelete.

Frontend (HTML, CSS, JavaScript):

O frontend é uma aplicação web estática que interage com a API através de requisições HTTP).

HTML: Estrutura as diferentes telas da aplicação (index.html para listagem/edição/remoção, cadastro.html para criação, relatorio.html para análise).

CSS: (Conforme o arquivo style.css original fornecido) Responsável pela estilização visual da interface, garantindo uma apresentação clara e funcional dos elementos.

JavaScript: Gerencia a lógica de interação do usuário e a comunicação com a API:

Tópico 4: Endpoints

A API "ApiEstoque" oferece os seguintes endpoints para gerenciar produtos:

A aplicação foi estruturada com rotas separadas para cada tipo de operação. Abaixo estão as rotas implementadas:

MÉTODO	ROTA	DESCRIÇÃO
GET	/api/produtos	Retorna uma lista de todos os produtos no estoque.
GET	/api/produtos/{id}	Retorna um produto específico pelo seu ID.
POST	/api/produtos	Adiciona um novo produto ao estoque.
PUT	/api/produtos/{id}	Atualiza um produto existente pelo seu ID.

DELETE	/api/produtos/{id}	Remove um produto existente pelo seu ID

4. Organização do Código

O projeto foi dividido em componentes com responsabilidades bem definidas, promovendo clareza e modularidade:

- Models/Produto.cs – Define a estrutura da entidade principal.
- Models/ProdutosContext.cs – Responsável pela configuração da base de dados e integração com o Entity Framework.
- Rotas/ROTA_GET.cs – Contém os métodos de listagem e consulta por ID.
- Rotas/ROTA_POST.cs – Responsável pelo cadastro de novos produtos.
- Rotas/ROTA_PUT.cs – Responsável pela atualização do cadastro de produtos.
- Rotas/ROTA_DELETE.cs – Gerencia a remoção de produtos existentes.
- Program.cs – Arquivo de inicialização da aplicação, onde são registrados os serviços e rotas.

5. Justificativa Técnica

A modelagem da entidade principal, o produto, foi feita com base nos atributos essenciais para identificação, catalogação e referência, garantindo coerência com o contexto funcional de sistemas de estoque reais. Essa modelagem orientou toda a arquitetura da aplicação, incluindo a configuração da base de dados, a definição das rotas e a estrutura do código. A separação entre os arquivos responsáveis pelas rotas, o modelo de dados e o contexto do banco adequa clareza na manutenção e favorece a evolução futura da aplicação, caso novas funcionalidades precisem ser incorporadas como atualização de registros ou autenticação de usuários.