**Prova Prática – Cadastro de Carnes, Compradores e Pedidos**

**Objetivo:**Crie um sistema clean arch usando c# com esses requisitos passo a passo mostrando as ferramentas e como usando como se eu fosse um leigo com esses requisitos: Desenvolver uma aplicação Web Full Stack com Razor Asp.Net Web App MVC no frontend e .NET 8 (C#) no backend, utilizando uma arquitetura em camadas e clean arch. A aplicação deve gerenciar carnes, compradores e pedidos, com relacionamento entre as entidades para garantir uma gestão eficaz do processo de compra.

**Backend (.NET 8)**

A API RESTful deve possuir as seguintes funcionalidades:

**Carnes:**

* **GET /carnes** – Listar todas as carnes cadastradas.
* **POST /carnes** – Cadastrar uma nova carne.
* **GET /carnes/{id}** – Buscar uma carne específica pelo ID.
* **PUT /carnes/{id}** – Editar os dados de uma carne existente.
* **DELETE /carnes/{id}** – Excluir uma carne (permitir exclusão apenas se não houver pedidos vinculados).

**Compradores:**

* **GET /compradores** – Listar todos os compradores cadastrados.
* **POST /compradores** – Cadastrar um novo comprador.
* **GET /compradores/{id}** – Buscar um comprador específico pelo ID.
* **PUT /compradores/{id}** – Editar os dados de um comprador existente.
* **DELETE /compradores/{id}** – Excluir um comprador.

**Pedidos:**

* **GET /pedidos** – Listar todos os pedidos, com as carnes e compradores associados.
* **POST /pedidos** – Criar um novo pedido, associando carnes e um comprador.
* **GET /pedidos/{id}** – Buscar um pedido específico pelo ID.
* **PUT /pedidos/{id}** – Editar um pedido existente.
* **DELETE /pedidos/{id}** – Excluir um pedido.

**Arquitetura em Camadas**

A aplicação deverá seguir a seguinte estrutura:

* **Controller:** É o projeto principal da aplicação, responsável por receber as requisições HTTP.
* **Service:** Projeto responsável por aplicar as regras de negócio e orquestrar o fluxo da aplicação.
* **Data:** Projeto responsável pelo acesso ao banco de dados, utilizando o Entity Framework Core para gerenciar repositórios, contextos e operações de CRUD.
* **Model:**  
  Projeto contendo as entidades do domínio (Carne, Comprador, Pedido etc.).

**Frontend (React)**

**Tela de Cadastro/Listagem de Carnes:**

* **Campos:**
  + Descrição da carne (campo de texto).
  + Combobox para selecionar a origem da carne (Bovina, Suína, Aves, Peixes).
* **Funcionalidade:**
  + Exibir a listagem de carnes cadastradas (Id, nome e origem).
  + Permitir edição e exclusão de carnes.
  + A exclusão só deve ser permitida se não houver pedidos associados à carne.

**Tela de Cadastro/Listagem de Compradores:**

* **Campos:**
  + Nome do comprador (campo de texto).
  + Documento (CPF/CNPJ) (campo de texto).
  + Combobox para selecionar a cidade e o estado.
* **Funcionalidade:**
  + Exibir a listagem de compradores cadastrados. (Id, nome e documento)
  + Permitir edição e exclusão de compradores.
  + A exclusão só deve ser permitida se não houver pedidos associados ao comprador.

**Tela de Criação/Listagem de Pedidos:**

* **Campos:**
  + Data do pedido (campo de data).
  + Combobox para selecionar o comprador.
  + Possibilidade de adicionar carnes ao pedido:
    - Combobox para selecionar a carne.
    - Campo livre para inserir o preço da carne.
    - Combobox para selecionar a moeda (Real, Dólar ou Euro).
* **Funcionalidade:**
  + Exibir a listagem de pedidos criados. (Id, comprador, valor total do pedido.)
  + Permitir edição e exclusão de pedidos.
  + A listagem de pedidos deve mostrar os valores em Real, utilizando uma API externa para cotação de moedas (link: [AwesomeAPI](https://docs.awesomeapi.com.br/api-de-moedas)).

**Banco de Dados (SQL Server)**

Deverá ser disponibilizado um script de banco de dados com:

* Criação do banco de dados.
* Criação das tabelas necessárias (Carne, Comprador, Pedido).
* Chaves primárias e estrangeiras para garantir a integridade dos dados.
* Inserts iniciais para popular as tabelas com informações de cidades e estados.

**Avaliação**

A avaliação será feita com base nos seguintes critérios:

* Arquitetura em camadas bem estruturada.
* Relacionamentos bem implementados, incluindo relacionamentos muitos-para-muitos entre carnes, compradores e pedidos.
* Código organizado, limpo e bem comentado.
* Usabilidade no frontend, utilizando Razor Asp.Net Web App MVC com formulários controlados, feedbacks visuais de sucesso e erro, e boas práticas de design.
* Boas práticas REST para as APIs e correto tratamento de erros. (Try Catch).

**Diferenciais**

A implementação de funcionalidades adicionais será um diferencial:

* Filtro por comprador ou data na listagem de pedidos.
* Validação de dados, como garantir que o preço seja positivo e que o nome do comprador seja obrigatório.
* Modal de confirmação de exclusão para evitar exclusões acidentais.
* Feedback visual (sucesso, erro) durante as interações.

**Entrega:**

* Código completo da aplicação frontend (React) e backend (.NET 8) no Github.
* README com instruções claras sobre como rodar o projeto localmente.