Projeto de Iniciação: Vitrine de Produtos com Angular

Contexto

O objetivo é colocar em prática os conceitos fundamentais que aprendemos no curso. Vamos construir uma aplicação do zero, focando na organização, manipulação de dados e interatividade que o Angular proporciona.

Você irá desenvolver o front-end de uma "Vitrine de Produtos", uma loja online simplificada. Para focar apenas no desenvolvimento front-end, utilizaremos uma API pública e gratuita que nos fornecerá todos os dados dos produtos.

API Recomendada: DummyJSON (`https://dummyjson.com/docs/products`) Esta API é excelente para iniciantes. Usaremos apenas os endpoints de consulta (`GET`):

- `https://dummyjson.com/products` (para listar todos os produtos)
- `https://dummyjson.com/products/1` (para buscar um produto específico pelo seu ID)

Objetivo Principal

Construir uma Single-Page Application (SPA) interativa e responsiva que permita aos usuários visualizar produtos, pesquisar por itens específicos, adicioná-los a um carrinho de compras local e preencher um formulário de finalização de compra (simulado).

Requisitos Funcionais (O que a aplicação deve fazer?)

A aplicação deverá ter as seguintes funcionalidades:

1. **Componente Inicial (Galeria de Produtos):**
    - Deve exibir uma galeria com todos os produtos vindos da API.
    - Cada produto deve ser um componente `CardProdutoComponent` reutilizável.
2. **Componente de Detalhes do Produto:**
    - Ao clicar em um produto na galeria, o usuário deve exibir um novo componente que mostra todas as informações daquele item (nome, descrição, preço, imagens, etc.).
    - Nesse componente, deve haver um botão "Adicionar ao Carrinho".
3. **Carrinho de Compras Simplificado:**
    - A aplicação deve ter um ícone de carrinho (geralmente no cabeçalho) que mostre a quantidade de itens adicionados.

4. **Formulário de "Checkout" (Formulário Simulado):**
    - Um componente com um formulário para que o usuário insira dados de contato e endereço para a "compra".
    - O formulário deve ter campos como "Nome Completo", "E-mail" e "Endereço".
    - Deve possuir validações (ex: campos obrigatórios, e-mail em formato válido).
    - O botão de "Finalizar Compra" só deve ser habilitado se o formulário estiver válido.

Requisitos Técnicos (Como a aplicação deve ser construída?)

Para implementar as funcionalidades acima, você **deve** aplicar os seguintes conceitos fundamentais do Angular:

1. **Organização do Código:**
    - Estruturar o projeto de forma clara, separando arquivos em pastas como `components`, `services`, `pages`, `models` (para as interfaces de dados, como `IProduct`).
2. **Manipulação de Dados:**
    - **Data Binding:** Utilizar `string interpolation` (`{{ }}`), `property binding` (`[ ]`) e `event binding` (`( )`).
    - **Pipes:** Utilizar o `currency` pipe para formatar o preço dos produtos e o `uppercase` pipe no título dos produtos na página de detalhes.
    - **Diretivas Estruturais:** Utilizar `@for` para renderizar a lista de produtos e `@if` para lógicas condicionais (ex: mostrar uma mensagem "Carregando..." enquanto os dados não chegam da API).
    - **Services e Injeção de Dependência:** Criar e injetar serviços onde for necessário, seguindo as melhores práticas.
3. **Interface Responsiva:**
    - A interface deve ser adaptável a diferentes tamanhos de tela (desktop e mobile).
    - Você pode escolher usar **Bootstrap**, **Angular Material** ou **CSS puro com Flexbox/Grid** para construir o layout.
4. **Consumo de API com `HttpClient` e `Observables`:**
    - Criar um `ProductService` que será o responsável por fazer as chamadas `GET` para a API usando o `HttpClient`.
    - Os métodos do serviço devem retornar `Observables` (`Observable<IProduct[]>` e `Observable<IProduct>`).
    - Os componentes devem se inscrever (`subscribe`) a esses `Observables` para receber os dados e exibi-los na tela.
5. **Gerenciamento de Estado com RxJS:**
    - Criar um `CartService` para gerenciar os itens do carrinho.
    - Dentro deste serviço, utilizar um `BehaviorSubject` do RxJS para manter a lista de produtos no carrinho.

    - O serviço deve expor um `Observable` público para que qualquer componente (como o cabeçalho ou a página do carrinho) possa se inscrever e ser notificado automaticamente quando o carrinho for alterado (um item for adicionado ou removido).
6. **Formulários Reativos (Reactive Forms):**
    - Utilizar um `FormControl` para o campo de busca de produtos na página inicial. Usar o `valueChanges` (um `Observable`) para reagir às digitações do usuário.
    - Construir o formulário da página de "Checkout" utilizando `FormGroup` e `FormBuilder`.
    - Aplicar validadores (`Validators.required`, `Validators.email`, `Validators.minLength`) aos campos e exibir mensagens de erro para o usuário.

Passos Sugeridos para o Desenvolvimento

1. **Setup:** Crie o projeto Angular e instale as dependências (Bootstrap ou Material, se for usar).
2. **Serviços:** Comece criando os serviços `ProductService` e `CartService`.
3. **Listagem:** Crie a página principal e o componente de card para listar os produtos.
4. **Roteamento:** Implemente o `RouterModule` e crie a rota para a página de detalhes do produto.
5. **Detalhes:** Desenvolva a página de detalhes, buscando um produto específico pelo ID.
6. **Carrinho:** Implemente a lógica de adicionar ao carrinho no `CartService` e faça o cabeçalho refletir a quantidade de itens.
7. **Busca:** Adicione o campo de busca e a lógica de filtro.
8. **Formulário:** Por último, crie a página de checkout com o formulário reativo.

Foco: O mais importante é a aplicação correta dos conceitos do Angular. A parte visual (CSS) pode ser simples, mas a aplicação deve ser funcional e bem estruturada.

Bom projeto!

Prazo para entrega 28/08/2025 23:59h

Formas de entrega: Link do GitHub pelo LMS ou Upload do projeto no LMS ou envio do projeto por link do GitHub ou anexado por e-mail roosevelt.franklin81@gmail.com