Para criar um sistema de orçamento com o fluxo que você descreveu, **não é recomendável criar apenas um único componente**. Embora seja tecnicamente possível, isso pode levar a um componente muito grande e difícil de manter, o que vai contra as boas práticas de desenvolvimento no Angular.

A abordagem ideal é **dividir o sistema em múltiplos componentes**, cada um responsável por uma parte específica do fluxo. Isso melhora a organização, facilita a manutenção e permite a reutilização de componentes em outras partes do sistema, caso necessário.

**Divisão Recomendada de Componentes**

Com base no fluxo descrito, você pode dividir o sistema em **três componentes principais**, além de um componente pai para gerenciar o fluxo geral:

**1. Componente Pai: OrcamentoComponent**

* Este componente será responsável por gerenciar o fluxo geral do orçamento.
* Ele será o contêiner que controla os dados compartilhados entre os componentes filhos e a navegação entre as etapas.

**2. Componente Filho 1: DadosClienteComponent**

* Responsável por coletar os dados do cliente, como nome, telefone, e-mail, etc.
* Envia os dados do cliente para o componente pai.

**3. Componente Filho 2: MontarRefeicaoComponent**

* Responsável por montar cada refeição, permitindo que o usuário selecione os ingredientes e quantidades.
* Permite adicionar múltiplas refeições.
* Envia as refeições montadas para o componente pai.

**4. Componente Filho 3: ResumoOrcamentoComponent**

* Responsável por exibir o resumo do orçamento, incluindo os dados do cliente e as refeições montadas.
* Permite ao usuário revisar e confirmar o orçamento.

**Estrutura do Sistema**

A estrutura do sistema ficaria assim:

src/app/

orcamento/

orcamento.component.ts <-- Componente Pai

orcamento.component.html

orcamento.component.scss

dados-cliente/

dados-cliente.component.ts <-- Coleta os dados do cliente

dados-cliente.component.html

dados-cliente.component.scss

montar-refeicao/

montar-refeicao.component.ts <-- Monta as refeições

montar-refeicao.component.html

montar-refeicao.component.scss

resumo-orcamento/

resumo-orcamento.component.ts <-- Exibe o resumo do orçamento

resumo-orcamento.component.html

resumo-orcamento.component.scss

**Como Funciona o Fluxo**

1. **Componente Pai (OrcamentoComponent):**
   * Gerencia o estado geral do orçamento (dados do cliente, refeições montadas, etc.).
   * Controla a navegação entre os componentes filhos.
2. **Componente Filho 1 (DadosClienteComponent):**
   * Coleta os dados do cliente e os envia para o componente pai.
   * Exemplo de dados:
   * {
   * nome: "João Silva",
   * telefone: "(11) 99999-9999",
   * email: "joao@email.com"
   * }
3. **Componente Filho 2 (MontarRefeicaoComponent):**
   * Permite que o usuário monte refeições personalizadas.
   * Cada refeição contém:
     + Tipo de refeição (ex.: Almoço, Jantar).
     + Ingredientes e suas quantidades.
   * Envia as refeições montadas para o componente pai.
   * Exemplo de dados:
   * [
   * {
   * tipoRefeicao: "Almoço",
   * ingredientes: [
   * { tipo: "Proteína", nome: "Frango", quantidade: 200 },
   * { tipo: "Carboidrato", nome: "Arroz", quantidade: 150 },
   * { tipo: "Legume", nome: "Brócolis", quantidade: 100 }
   * ]
   * },
   * {
   * tipoRefeicao: "Jantar",
   * ingredientes: [
   * { tipo: "Proteína", nome: "Peixe", quantidade: 150 },
   * { tipo: "Carboidrato", nome: "Batata Doce", quantidade: 200 }
   * ]
   * }
   * ]
4. **Componente Filho 3 (ResumoOrcamentoComponent):**
   * Recebe os dados do cliente e as refeições do componente pai.
   * Exibe o resumo do orçamento, incluindo o preço total.
   * Permite confirmar ou editar o orçamento.

**Exemplo de Implementação**

**1. Componente Pai (OrcamentoComponent)**

**TypeScript**

import { Component } from '@angular/core';

@Component({

selector: 'app-orcamento',

templateUrl: './orcamento.component.html',

styleUrls: ['./orcamento.component.scss']

})

export class OrcamentoComponent {

dadosCliente: any = null;

refeicoes: any[] = [];

// Recebe os dados do cliente do componente filho

salvarDadosCliente(dados: any) {

this.dadosCliente = dados;

}

// Recebe as refeições montadas do componente filho

salvarRefeicoes(refeicoes: any[]) {

this.refeicoes = refeicoes;

}

}

**HTML**

<div>

<h1>Orçamento de Marmitas</h1>

<!-- Etapa 1: Dados do Cliente -->

<app-dados-cliente

(dadosCliente)="salvarDadosCliente($event)"

></app-dados-cliente>

<!-- Etapa 2: Montar Refeições -->

<app-montar-refeicao

(refeicoes)="salvarRefeicoes($event)"

></app-montar-refeicao>

<!-- Etapa 3: Resumo do Orçamento -->

<app-resumo-orcamento

[dadosCliente]="dadosCliente"

[refeicoes]="refeicoes"

></app-resumo-orcamento>

</div>

**2. Componente Filho 1 (DadosClienteComponent)**

**TypeScript**

import { Component, EventEmitter, Output } from '@angular/core';

@Component({

selector: 'app-dados-cliente',

templateUrl: './dados-cliente.component.html',

styleUrls: ['./dados-cliente.component.scss']

})

export class DadosClienteComponent {

@Output() dadosCliente = new EventEmitter<any>();

cliente = {

nome: '',

telefone: '',

email: ''

};

salvar() {

this.dadosCliente.emit(this.cliente);

}

}

**HTML**

<div>

<h2>Dados do Cliente</h2>

<input [(ngModel)]="cliente.nome" placeholder="Nome do Cliente" />

<input [(ngModel)]="cliente.telefone" placeholder="Telefone" />

<input [(ngModel)]="cliente.email" placeholder="E-mail" />

<button (click)="salvar()">Salvar Dados</button>

</div>

**3. Componente Filho 2 (MontarRefeicaoComponent)**

**TypeScript**

import { Component, EventEmitter, Output } from '@angular/core';

@Component({

selector: 'app-montar-refeicao',

templateUrl: './montar-refeicao.component.html',

styleUrls: ['./montar-refeicao.component.scss']

})

export class MontarRefeicaoComponent {

@Output() refeicoes = new EventEmitter<any[]>();

novaRefeicao = {

tipoRefeicao: '',

ingredientes: []

};

refeicoesCadastradas: any[] = [];

adicionarRefeicao() {

this.refeicoesCadastradas.push({ ...this.novaRefeicao });

this.novaRefeicao = { tipoRefeicao: '', ingredientes: [] };

this.refeicoes.emit(this.refeicoesCadastradas);

}

}

**HTML**

<div>

<h2>Montar Refeições</h2>

<select [(ngModel)]="novaRefeicao.tipoRefeicao">

<option value="Almoço">Almoço</option>

<option value="Jantar">Jantar</option>

</select>

<button (click)="adicionarRefeicao()">Adicionar Refeição</button>

</div>

**4. Componente Filho 3 (ResumoOrcamentoComponent)**

**TypeScript**

import { Component, Input } from '@angular/core';

@Component({

selector: 'app-resumo-orcamento',

templateUrl: './resumo-orcamento.component.html',

styleUrls: ['./resumo-orcamento.component.scss']

})

export class ResumoOrcamentoComponent {

@Input() dadosCliente: any;

@Input() refeicoes: any[];

}

**HTML**

<div>

<h2>Resumo do Orçamento</h2>

<p>Cliente: {{ dadosCliente?.nome }}</p>

<p>Telefone: {{ dadosCliente?.telefone }}</p>

<p>E-mail: {{ dadosCliente?.email }}</p>

<h3>Refeições</h3>

<ul>

<li \*ngFor="let refeicao of refeicoes">

{{ refeicao.tipoRefeicao }}

</li>

</ul>

</div>

**Vantagens de Dividir em Componentes**

1. **Organização:**
   * Cada parte do fluxo é gerenciada por um componente específico.
2. **Reutilização:**
   * Componentes como DadosClienteComponent podem ser usados em outros fluxos.
3. **Manutenção:**
   * Alterações em uma etapa não afetam as outras.
4. **Escalabilidade:**
   * Facilita a adição de novas funcionalidades no futuro.

Essa abordagem modular é ideal para sistemas com fluxos complexos como o seu. 🚀