Problemas na Duplicação de Componentes no Angular 19

A duplicação de componentes no Angular pode gerar **erros** e **comportamentos inesperados**, especialmente quando você tenta declarar o mesmo componente em múltiplos lugares ou módulos. Esses problemas geralmente surgem devido à forma como o Angular organiza e gerencia os componentes dentro de seus **módulos**.

Aqui estão os principais problemas e como evitá-los:

---

1. Declaração Duplicada em Módulos

Problema:

Se você declarar o mesmo componente em mais de um módulo no Angular, ocorrerá um erro de compilação, como:

- Type AppComponent is part of the declarations of 2 modules.

Isso acontece porque o Angular exige que cada componente seja declarado **apenas uma vez** e faça parte de um único módulo.

Solução:

Certifique-se de declarar o componente em apenas um módulo. Se precisar usá-lo em outro módulo, **exporte-o** do módulo original e **importe-o** no módulo desejado.

Exemplo de Correção:

1. Declare o componente em um único módulo:

```
import { NgModule } from '@angular/core';

import { CommonModule } from
'@angular/common'; // Para diretivas e pipes
comuns do Angular

import { CardComponent } from
'../card/card.component'; // Componente
standalone
```

```
import { CardButtonComponent } from
'../card-button/card-button.component'; //
Componente standalone

import { CardRoxoComponent } from
'../card-roxo/card-roxo.component'; //
Componente standalone

import { CardRoxoButtonComponent } from
'../card-roxo-button/card-roxo-button.componen
t'; // Componente standalone



@NgModule({

  imports: [

    CommonModule, // Módulo para usar
diretivas como *ngIf, *ngFor, etc.

    CardComponent, // Importando standalone
components

    CardButtonComponent,

    CardRoxoComponent,

    CardRoxoButtonComponent

  ],

  exports: [

    CardComponent, // Exportando para que
outros módulos possam usá-los
```

```
    //CardButtonComponent,
    CardRoxoComponent,
    //CardRoxoButtonComponent
  ]
})
export class CardsModule {}
```

2. Importe-o no component onde deseja utilizá-lo:

```
import { Component } from '@angular/core';
import { RouterOutlet } from
'@angular/router';
import { CardsModule } from
'./cards/cards.module';


@Component({
  selector: 'app-root',
  imports: [RouterOutlet, CardsModule],
  templateUrl: './app.component.html',
  styleUrl: './app.component.scss'
```

```
})
export class AppComponent {
  title = 'componentes-estilizacoes';
}
```

2. Uso Indevido de Componentes Duplicados no HTML

Problema:

Mesmo sem duplicar declarações nos módulos, você pode acabar criando **componentes redundantes ou com comportamento conflitante** no HTML. Por exemplo:

- <app-meu-componente></app-meu-componente>
- <app-meu-componente></app-meu-componente>

Se o componente contiver lógica que manipula estados compartilhados ou faz chamadas de API, duplicá-lo no HTML pode causar conflitos ou sobrecarga.

Solução:

Se você precisa reutilizar o mesmo componente várias vezes, mas de forma independente, **garanta que ele seja projetado para suportar estados isolados ou compartilhe estados conscientemente**. Use `@Input` e `@Output` para passar e emitir dados, e evite estados globais desnecessários.

---

3. Conflito de Seletores

Problema:

Se dois componentes diferentes usarem o mesmo `selector` no `@Component`, o Angular não saberá qual deles renderizar. Isso pode causar erros no console ou comportamento inesperado.

```
@Component({
  selector: 'app-meu-componente',
  template: '<p>Componente 1</p>',
})
export class MeuComponente1 {}

@Component({
  selector: 'app-meu-componente',
  template: '<p>Componente 2</p>',
})
export class MeuComponente2 {}
```

Erro gerado:

- Component with selector 'app-meu-componente' is already defined.

Solução:

Certifique-se de que cada componente tenha um `selector` único:

```
@Component({
  selector: 'app-meu-componente-1',
  template: '<p>Componente 1</p>',
})
export class MeuComponente1 {}

@Component({
  selector: 'app-meu-componente-2',
  template: '<p>Componente 2</p>',
})
export class MeuComponente2 {}
```

4. Duplicação Acidental em Testes

Problema:

No desenvolvimento de testes, você pode acidentalmente declarar o mesmo componente várias vezes, especialmente ao configurar os módulos de teste. Isso gera erros semelhantes ao declarar componentes em múltiplos módulos.

Solução:

Declare o componente apenas uma vez no módulo de teste:

```
TestBed.configureTestingModule({
  declarations: [MeuComponenteComponent], // Apenas uma vez
});
```

Se o componente for parte de outro módulo, importe o módulo em vez de declarar o componente novamente:

```
TestBed.configureTestingModule({
  imports: [MeuComponenteModule], // Importa o módulo original
});
```

Resumo de Boas Práticas para Evitar Problemas

1. **Declare cada componente em apenas um módulo.**
2. **Use `exports` para disponibilizar componentes para outros módulos.**
3. **Evite duplicar seletores; mantenha-os únicos em toda a aplicação.**
4. **Projete componentes reutilizáveis com `@Input` e `@Output` para evitar conflitos de estado.**
5. **Tenha atenção especial ao configurar módulos em testes.**