

## Rotas Aninhadas

Rotas podem possuir rotas filhas. Essas rotas filhas são rotas que mostram componentes dentro de outro componente que também são rotas. Por exemplo, existe uma página de um site que mostra os dados de um produto utilizando a rota **/product**. Mas essa rota pode ter uma sub-rota para mostrar as opiniões dos compradores. Essa rota poderia ser a rota **/product/overview**. Ou seja, a rota **overview** é uma rota filha de **product**. Sendo assim, essa rota está dentro de outra.

Para informar que uma rota tem rotas filhas, basta utilizar o parâmetro **children** na definição da sua rota. Esse parâmetro recebe um array de outras rotas, assim como vimos antes.

```
1  const appRoutes: Routes = [  
2    {  
3      path: '',  
4      pathMatch: 'full',  
5      component: Page1Component  
6    },  
7    {  
8      path: 'page2',  
9      component: Page2Component,  
10     children: [  
11       {  
12         path: 'filho1',  
13         component: Filho1Component  
14       }  
15     ]  
16   }  
17 ]
```

Agora, definimos que a rota **page2** tem uma rota filha chamada **filho1**. Rotas filhas possuem as mesmas propriedades que as rotas pai, ou seja, uma sub-rota também pode possuir sub-rotas e qualquer outra propriedade que for necessária.

Agora, para acessarmos as sub-rotas da page2, devemos colocar, dentro do template HTML do componente Page2 o componente **router-outlet**. Vamos colocar um link para acessarmos a sub-rota.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in a light-colored font and is numbered from 1 to 10 on the left side. The code defines the HTML template for a component, including a paragraph, a list with a link, and a router-outlet.

```
1 <p>page2 works!</p>
2
3 <ul>
4   <li>
5     <a routerLink="/page2/filho1">Acessar rota filha</a>
6   </li>
7 </ul>
8
9 <router-outlet></router-outlet>
10
```

Quando esse link for clicado dentro do componente Page2, o componente Filho1 aparecerá.

**OBS:** o routerLink para a sub-rota não precisa estar, exatamente, dentro do componente pai da rota. Pode estar em qualquer outro lugar. Isso é apenas um exemplo ilustrativo de sua utilização.

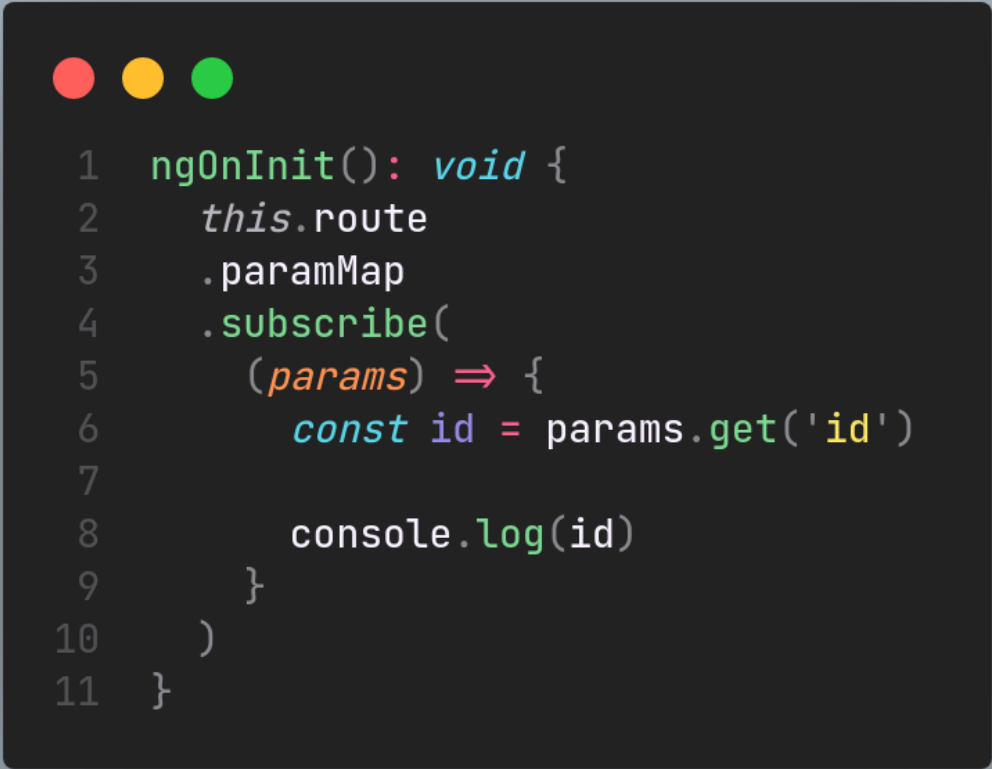
## Rotas Filhas com parâmetros

Rotas filhas também podem receber parâmetros. Vamos fazer com que a rota filha que criamos receba um parâmetro chamado **id** e, dentro do componente Filho1 ele faça um console.log do valor informado.



```
1  const appRoutes: Routes = [  
2    {  
3      path: '',  
4      pathMatch: 'full',  
5      component: Page1Component  
6    },  
7    {  
8      path: 'page2',  
9      component: Page2Component,  
10     children: [  
11       {  
12         path: 'filho1/:id',  
13         component: Filho1Component  
14       }  
15     ]  
16   }  
17 ]
```

Agora vamos fazer a lógica dentro do componente Filho1 para mostrar o valor do parâmetro id na tela. Vamos injetar um objeto ActivatedRoute no componente e, dentro do ngOnInit, fazer uma lógica para recuperar o valor.



```
1  ngOnInit(): void {
2    this.route
3    .paramMap
4    .subscribe(
5      (params) => {
6        const id = params.get('id')
7
8        console.log(id)
9      }
10   )
11 }
```

Nós vimos anteriormente que, para pegar um valor de um parâmetro, nós utilizamos o código **this.route.snapshot.paramMap.get()**. Por qual motivo não utilizamos dessa vez? 😞

Não utilizamos dessa vez pois o método visto antes não consegue observar alterações nos valores. Se caso o valor do parâmetro `id` fosse alterado, nesse caso, o valor não seria substituído e ficaria sempre no primeiro valor digitado.

Mas, como dessa vez estamos usando um **Observable**, o componente conseguirá recuperar a alteração dos dados dos parâmetros, evitando assim, possíveis erros e inconsistências dentro da aplicação.

Quando acessamos a propriedade **paramMap**, sem utilizar a propriedade **snapshot** antes, ele é um **Observable**. Ao fazermos o **subscribe** nesse observable, ele nos retorna um objeto para recuperarmos os novos dados. E basta utilizar o método **get()** informando o nome do parâmetro. Se existir, ele retornará o valor do parâmetro, se não, retornará **null**.