

[O que é o Angular?](#)

[Componente](#)

[Estrutura](#)

[Style](#)

[Instalação e criação projeto](#)

[Comando para instalar o CLI do Angular](#)

[Criar o projeto](#)

[ng new nome_projeto](#)

[Estruturas angular](#)

[Importar componente](#)

[Interpolação de dados](#)

[Css no Angular](#)

[Compartilhar dados entre componentes](#)

[Inserir o decorator input](#)

[Diretiva de classe](#)

[Command Line interface \(CLI\)](#)

[Diretivas](#)

[Bibliotecas próprias](#)

O que é o Angular?

- É uma plataforma de desenvolvimento robusta;
- Construída com typescript;
- Baseada em componentes;
- Possui arquitetura modular, isso possibilita reutilizar recursos;
- Oferece uma variedade de bibliotecas:
 - gerenciamento formulário;
 - comunicação cliente-servidor;
 - roteamento.
 - entre outros.

Pré requisitos

Os pré requisitos são:

- HTML;
- CSS
- Javascript;

- Json.

Componente

Os componentes são a base para criação de aplicações Angular, pois permite uma arquitetura de reaproveitamento.

Para identificar o componente podemos utilizar o sufixo *component* ao nome do arquivo. Exemplo:

meu-nome.componet.ts

Estrutura

- Um *decorator* define a configuração:
 - Um seletor que define qual tag name ao se referir ao nome do template;
 - Um template HTML que controla o que será renderizado;
- Typescript: definirá o comportamento do componente. Exemplos:
 - gerenciamento de estado;
 - métodos;
 - entrada de usuário.

```
// 📄 todo-list-item.component.ts
@Component({
  standalone: true,
  selector: 'todo-list-item',
  template: ` <li>(TODO) Read cup of coffee introduction</li> `,
  styles: ['li { color: papayawhip; }'],
})
export class TodoListItem {
  /* Component behavior is defined in here */
}
```

Style

Quando você precisa estilizar um componente, essas são duas propriedades que pode configurar dentro do `@Component` decorator.

1. Typescript class;
2. styleUrls properties.

```
@Component({
  selector: 'profile-pic',
  template: ``,
  styles: [
    `
      img {
        border-radius: 50%;
      }
    `,
  ],
})
export class ProfilePic {
  /* Your code goes here */
}
```

Instalação e criação projeto

Comando para instalar o CLI do Angular


<https://angular.dev/tools/cli/setup-local#dependencies>

```
npm install -g @angular/cli
```

Caso dê problema, podemos utilizar o comando

```
Set-ExecutionPolicy RemoteSigned
```

javascript

 Copy code

```
Set-ExecutionPolicy RemoteSigned
```

Criar o projeto

`ng new nome_projeto`

Executar o Angular

`ng serve`

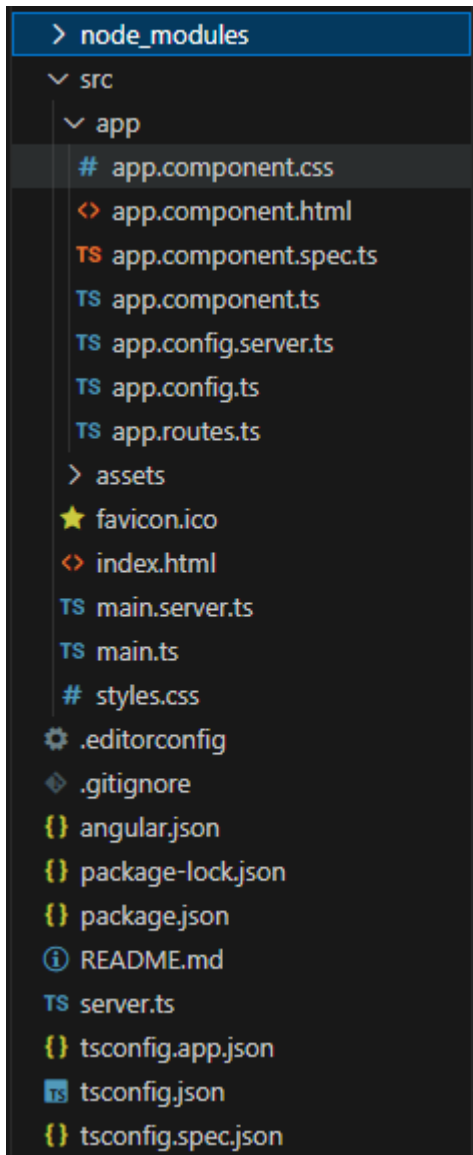
Estruturas angular

node_modules: dependências do projeto;

src: aplicação;

src/app: componentes, services e etc;

Os componentes são divididos entre: TS, HTML, CSS e testes.



Criar um componente

Os componentes são a base para criação de aplicações Angular, pois permite uma arquitetura de reaproveitamento.

Para identificar o componente podemos utilizar o sufixo *component* ao nome do arquivo

- Podemos utilizar o CLI;
- A importação do componente basta utilizar o seletor para utilizar.

ng generate

Comando:

ng generate component components/first-component

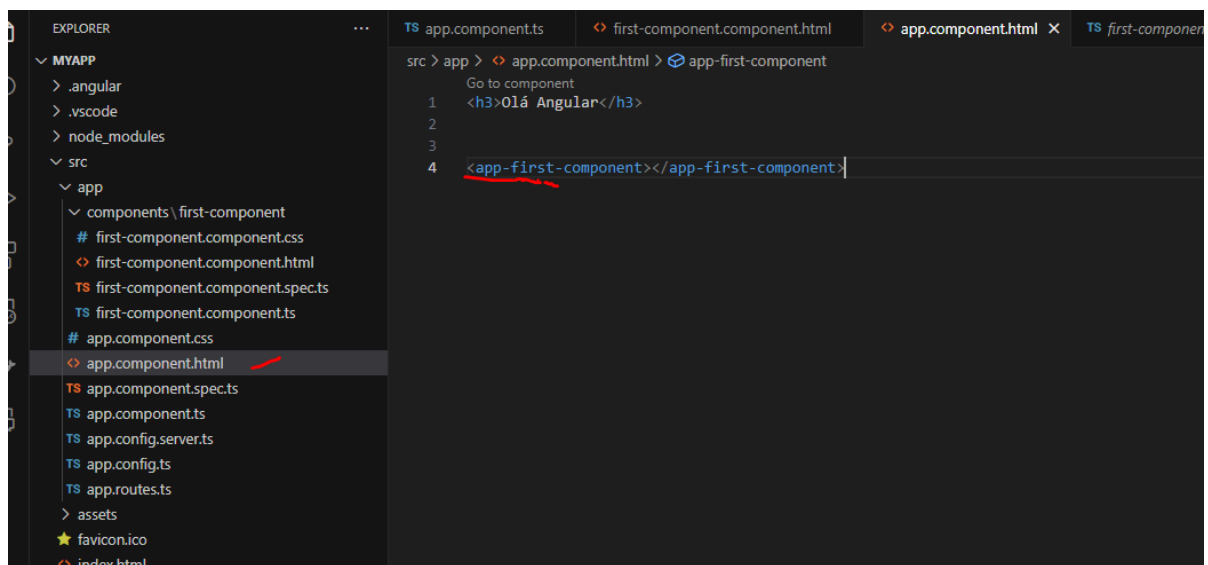
Importar componente

Para utilizar o componente primeiro precisamos configurar o import

1. Abra o app-component.ts e configure o import:

```
1 import { Component } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3 import { FirstComponentComponent } from '../components/first-component/first-component.component';
4
5 @Component({
6   selector: 'app-root',
7   standalone: true,
8   imports: [RouterOutlet, FirstComponentComponent],
9   templateUrl: './app.component.html',
10  styleUrls: ['./app.component.css']
11 })
12 export class AppComponent {
13   title = 'Thiago';
14 }
15
```

2. Inserir a tag no app-component.html para finalizarmos a inserção do component.



Interpolação de dados

- Criar variáveis para trabalhar as propriedades dinâmicas;
- A impressão é feita através do *double mustache*{{dado}} ;

Exemplo:

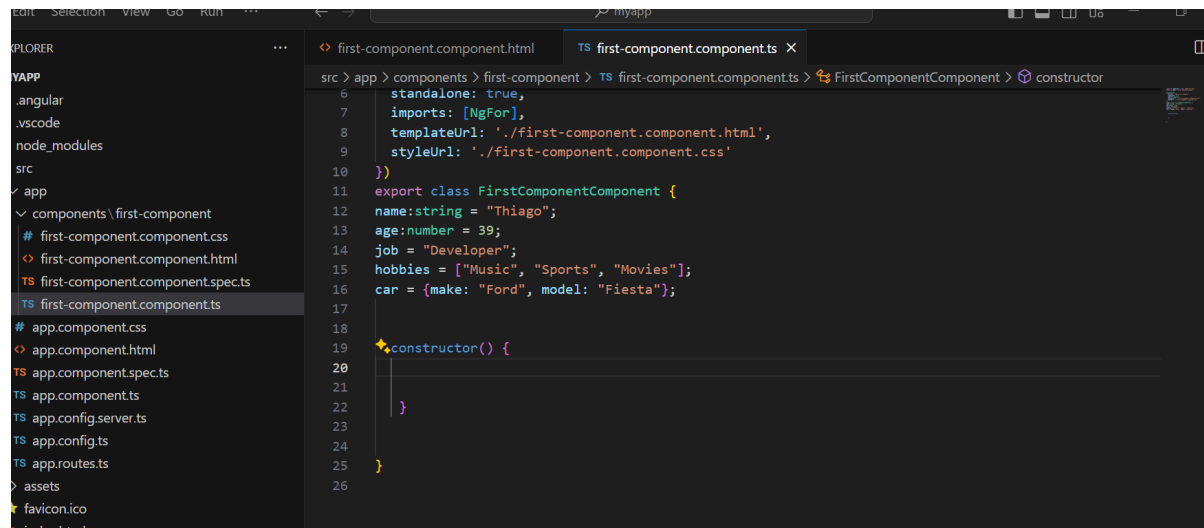
Podemos criar um atributo na classe do componente e utilizarmos na renderização;

Observação: vamos fazer o import do módulo CommonModule para poder utilizar a diretiva ngFor no exemplo.

```
import { CommonModule } from '@angular/common';
```

```
@Component({
  selector: 'app-first-component',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './first-component.component.html',
  styleUrls: ['./first-component.component.css']
})
```

arquivo: componet.ts



arquivo.html

```
first-component.component.html X TS first-component.component.ts
src > app > components > first-component > first-component.component.html > ul.ingredient-list
Go to component
1 <p>Alterar o componente</p>
2
3 <p>Olá {{name}}</p>
4
5 <p>Idade: {{age}}</p>
6 <p>Marca: {{car.make}}</p>
7 <p>Modelo: {{car.model}}</p>
8 <p>Hobbie: {{hobbies[1]}}</p>
9
10 <ul class="ingredient-list">
11   <li *ngFor="let task of hobbies">{{ task }}</li>
12 </ul>
```

Css no Angular

- Global: utilizado no arquivo style.css;
- scoped: pode ser utilizado a nível de component, para não replicar css desnecessário em outro component.

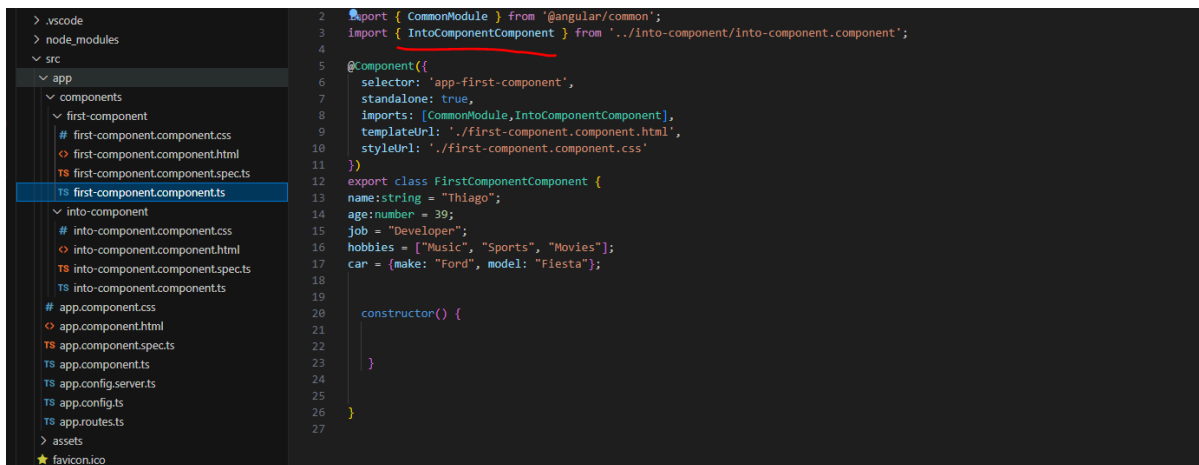
Compartilhar dados entre componentes

- É possível compartilhar dados entre os componentes;
- Precisaremos utilizar o decorator @input.

1. Para nosso teste, vamos criar outro component.

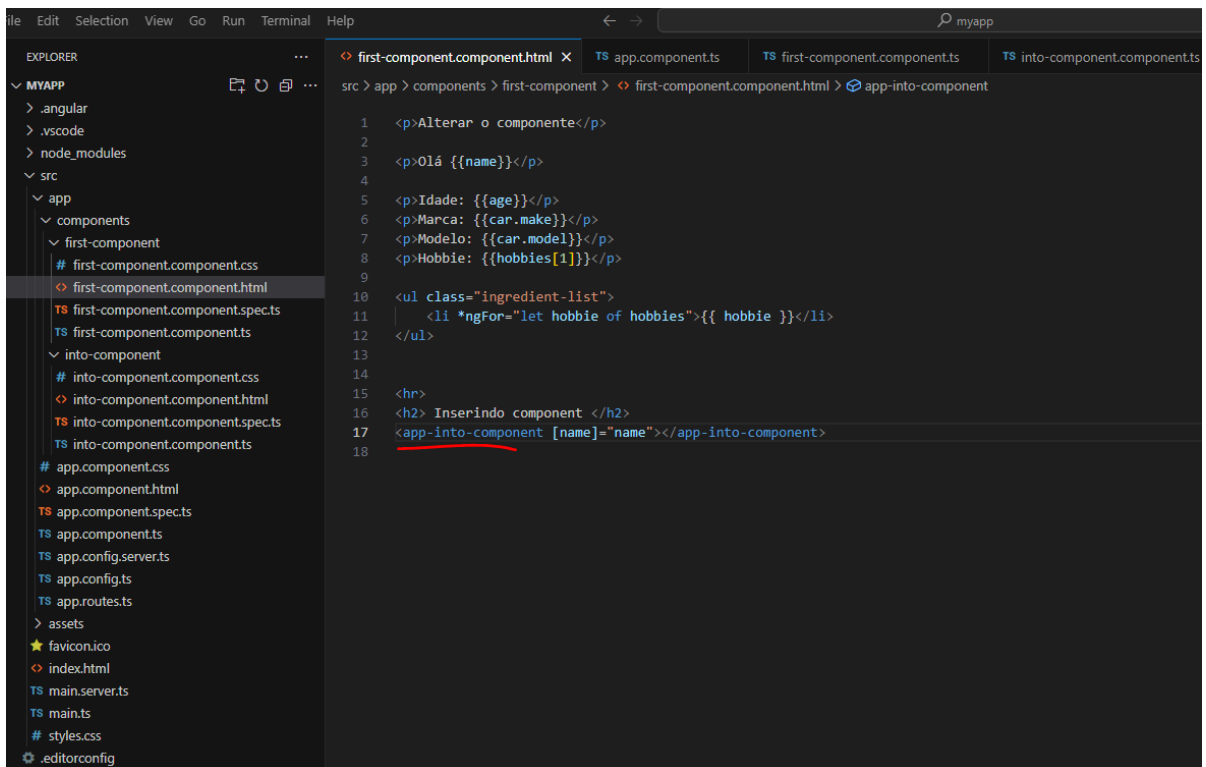
ng g component components/into-component

2. Importar o component dentro do component que deixa inseri-lo.



```
2 import { CommonModule } from '@angular/common';
3 import { IntoComponentComponent } from '../into-component/into-component.component';
4
5 @Component({
6   selector: 'app-first-component',
7   standalone: true,
8   imports: [CommonModule, IntoComponentComponent],
9   templateUrl: './first-component.component.html',
10  styleUrls: ['./first-component.component.css']
11 })
12 export class FirstComponentComponent {
13   name:string = "Thiago";
14   age:number = 39;
15   job = "Developer";
16   hobbies = ["Music", "Sports", "Movies"];
17   car = {make: "Ford", model: "Fiesta"};
18
19   constructor() {
20
21   }
22
23 }
24
25
26
27
```

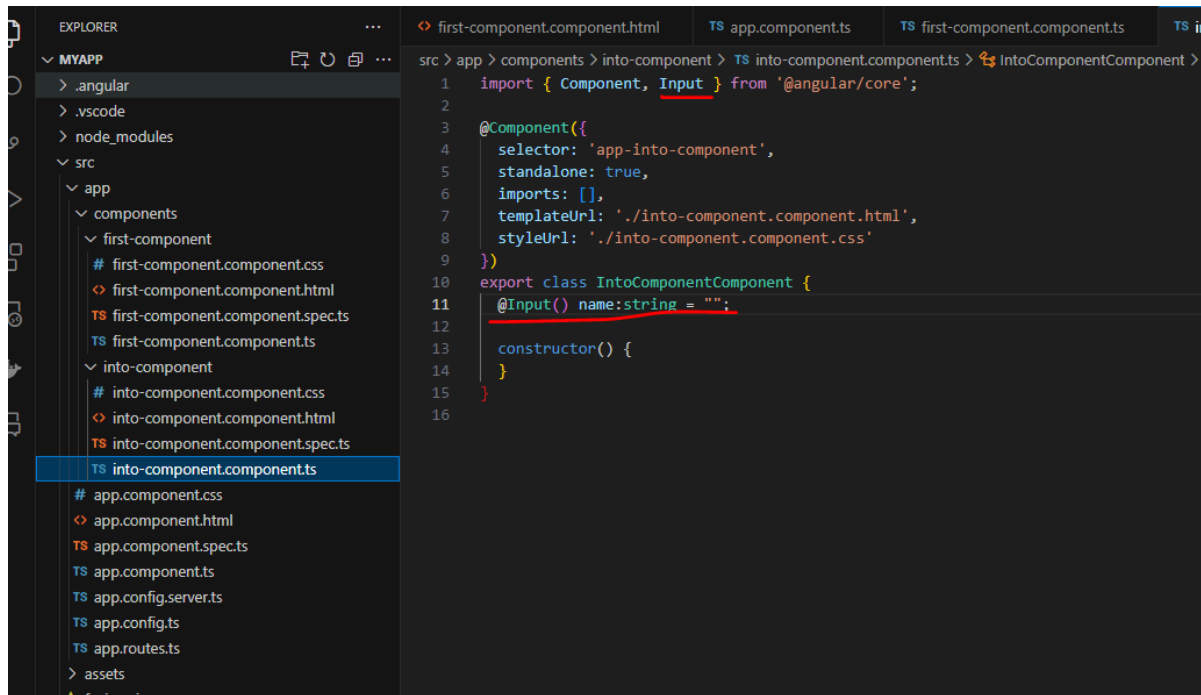
3. inserir o seletor



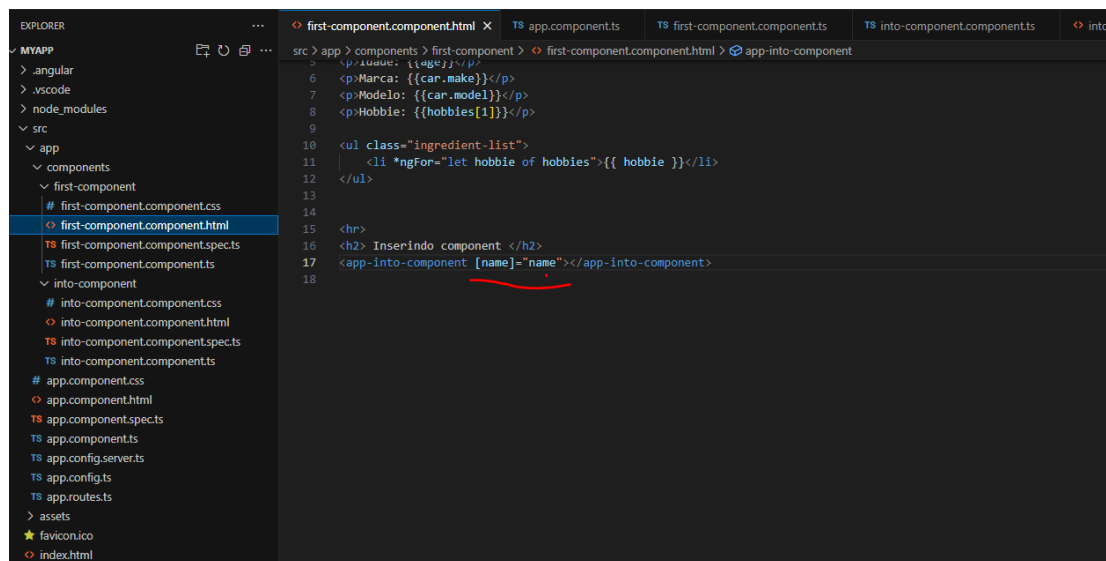
```
1 <p>Alterar o componente</p>
2
3 <p>Olá {{name}}</p>
4
5 <p>Idade: {{age}}</p>
6 <p>Marca: {{car.make}}</p>
7 <p>Modelo: {{car.model}}</p>
8 <p>Hobbie: {{hobbies[1]}}</p>
9
10 <ul class="ingredient-list">
11   <li *ngFor="let hobbie of hobbies">{{ hobbie }}</li>
12 </ul>
13
14
15 <hr>
16 <h2> Inserindo component </h2>
17 <app-into-component [name]="name"></app-into-component>
18
```

Inserir o decorator input

1. No component que receberá o dado compartilhado, inserir o decorator Input
2. Dentro da classe filho, que irá receber o objeto compartilhado, inserir o nome do objeto que será compartilhada.



3. Para finalizar, ao inserir o componente, podemos passar como parâmetro para component filho.



Diretivas

- Utilizado para aplicar estilos em elemento;
- Permite estender o HTML com novos comportamentos e funcionalidades
- Prefixo sempre ng... Ex: ngif, ngFor..

Diretiva de classe

1 - Vamos criar um component

ng g component components/directives

component.html

```
<h2>Diretiva de estilo</h2>

<p [ngStyle]="{'background-color': color, 'font-size': size + 'px',
'color' : 'red'}">Estilo com ngStyle</p>

<ul class="ingredient-list">
  <li *ngFor="let hobbie of hobbies">{{ hobbie }}</li>
</ul>

<h2>Diretiva de classe</h2>

<p [ngClass]="{ 'errorClass': hasError }">Estilo com ngClass</p>

<p [ngClass]="classes">Texto com sucesso</p>
```

componente.ts

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
@Component({
  selector: 'app-directives',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './directives.component.html',
  styleUrls: ['./directives.component.css']
})
export class DirectivesComponent {
  size = 50;
  font = 'Arial';
  color = 'yellow';
  hobbies = ["Music", "Sports", "Movies"];

  isActive = true;
  hasError = true
```

```
classes=["text-success", "text-danger", "text-special"]
}
```

component.css

Command Line interface (CLI)

comando	detalhe
ng build	Compila a aplicação Angular dentro do diretório de saída
ng serve	construir a aplicação
ng generate	Gerar ou modificar os arquivos baseados na semântica.
ng test	Rodar os teste de unidade
ng e2e	Compilar e executar os testes

Diretivas

nome	descrição	exemplo	modulo
ngStyle	trabalhar com styles inline	<pre><p [ngStyle]="{'background-color': color, 'font-size': size + 'px', 'color' : 'red'}">Estilo com ngStyle</p></pre>	<pre>import { CommonModule } from '@angular/common';</pre>
ngIf	Trabalhar com condicionais		

Bibliotecas próprias

Angular router	Navegação e roteamento avançado.
Angular forms	Sistema para formulário validação
Angular HttpClient	Comunicação servidor cliente
Angular Animations	Animações baseadas no estados da aplicação
Angular PWA	Constrói a aplicação progressivamente incluindo service worker e web application manifest.
Angular schematics	Scaffolding automático. Atualizar ferramentas para simplificar o desenvolvimento em larga escala.

Referência

<https://angular.io/guide/what-is-angular>