



Configurações Iniciais – Angular.js

Nome: Carlos Eduardo Cerqueira Leite

Cargo: Desenvolvedor de Software

São José do Rio Preto, São Paulo, Brasil

Sumário

1	Comandos.....	3
2	Primeiros passos.....	4
3	Criação de novos componentes.....	8
4	Adicionando Interfaces.....	10
4.1	Aplicação dos dados em estrutura de repetição	11
5	Tratamento em caso de loading extenso	12
6	Estilização do Projeto.....	13

1 Comandos

Ng: O comando **ng** é parte do Angular CLI (Command Line Interface), uma ferramenta de linha de comando usada para criar, desenvolver e gerenciar aplicativos Angular de forma mais eficiente. Para instalar o Angular CLI utilize **npm install -g @angular/cli**.

Ng s: Inicia o servidor.

Ng g c: Comando utilizado para gerar componentes, sua utilização deve ser feita junto com um caminho relativo de pastas. Exemplo: **ng g c modules/portfolio/pages/home**.

Ng add @angular/material: Adiciona a biblioteca Angular Material ao projeto

2 Primeiros passos

De início, primeiro instalaremos a Angular CLI para obter as configurações iniciais do projeto, para isso, execute o comando **npm install -g @angular/cli**.

Acompanhe a seguir os primeiros passos para iniciar um projeto com Angular.js:

```
PS C:\Users\Carlos\Desktop\portfolioField> ng new portfolioAngular
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. No
Global setting: disabled
Local setting: No local workspace configuration file.
Effective status: disabled
? Which stylesheet format would you like to use? SCSS [ https://sass-lang.com/documentation/syntax#scss
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No
CREATE portfolioAngular/angular.json (2918 bytes)
CREATE portfolioAngular/package.json (1086 bytes)
CREATE portfolioAngular/README.md (1097 bytes)
CREATE portfolioAngular/tsconfig.json (936 bytes)
CREATE portfolioAngular/.editorconfig (290 bytes)
CREATE portfolioAngular/.gitignore (590 bytes)
CREATE portfolioAngular/tsconfig.app.json (277 bytes)
CREATE portfolioAngular/tsconfig.spec.json (287 bytes)
CREATE portfolioAngular/.vscode/extensions.json (134 bytes)
CREATE portfolioAngular/.vscode/launch.json (490 bytes)
CREATE portfolioAngular/.vscode/tasks.json (980 bytes)
CREATE portfolioAngular/src/main.ts (256 bytes)
CREATE portfolioAngular/src/favicon.ico (15086 bytes)
CREATE portfolioAngular/src/index.html (315 bytes)
CREATE portfolioAngular/src/styles.scss (81 bytes)
CREATE portfolioAngular/src/app/app.component.html (20239 bytes)
CREATE portfolioAngular/src/app/app.component.spec.ts (975 bytes)
CREATE portfolioAngular/src/app/app.component.ts (326 bytes)
CREATE portfolioAngular/src/app/app.component.scss (0 bytes)
CREATE portfolioAngular/src/app/app.config.ts (235 bytes)
CREATE portfolioAngular/src/app/app.routes.ts (80 bytes)
CREATE portfolioAngular/src/assets/.gitkeep (0 bytes)
? Installing packages (npm)...
```

Figura 1 - Criação de novo projeto Angular.js e seleção de preferências

Note que o formato stylesheet selecionado foi SCSS.

```
PS C:\Users\Carlos\Desktop\portfolioField\portFolioAngular> ng s

Initial Chunk Files | Names          | Raw Size
polyfills.js        | polyfills      | 83.46 kB |
main.js             | main           | 22.10 kB |
styles.css          | styles         | 96 bytes |

| Initial Total | 105.65 kB

Application bundle generation complete. [1.485 seconds]
Watch mode enabled. Watching for file changes...
  → Local: http://localhost:4200/
  → press h + enter to show help
```

Figura 2 - Utilização do comando ng s

Agora que o servidor já foi iniciado, delete os 3 próximos arquivos, eles não serão utilizados:

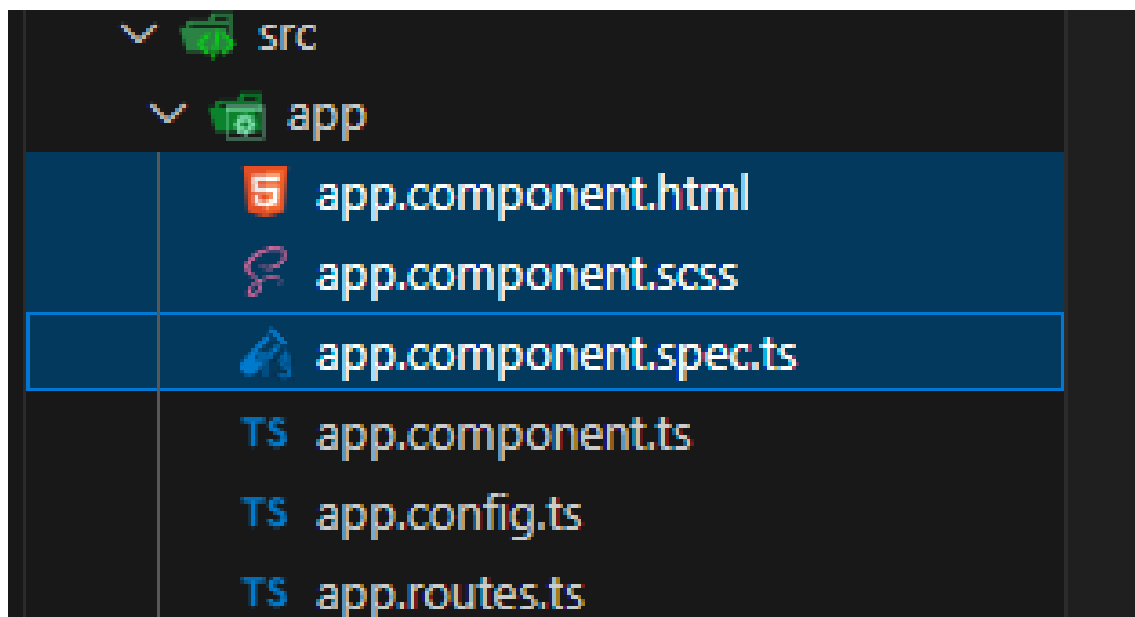


Figura 3 - Deleção de arquivos

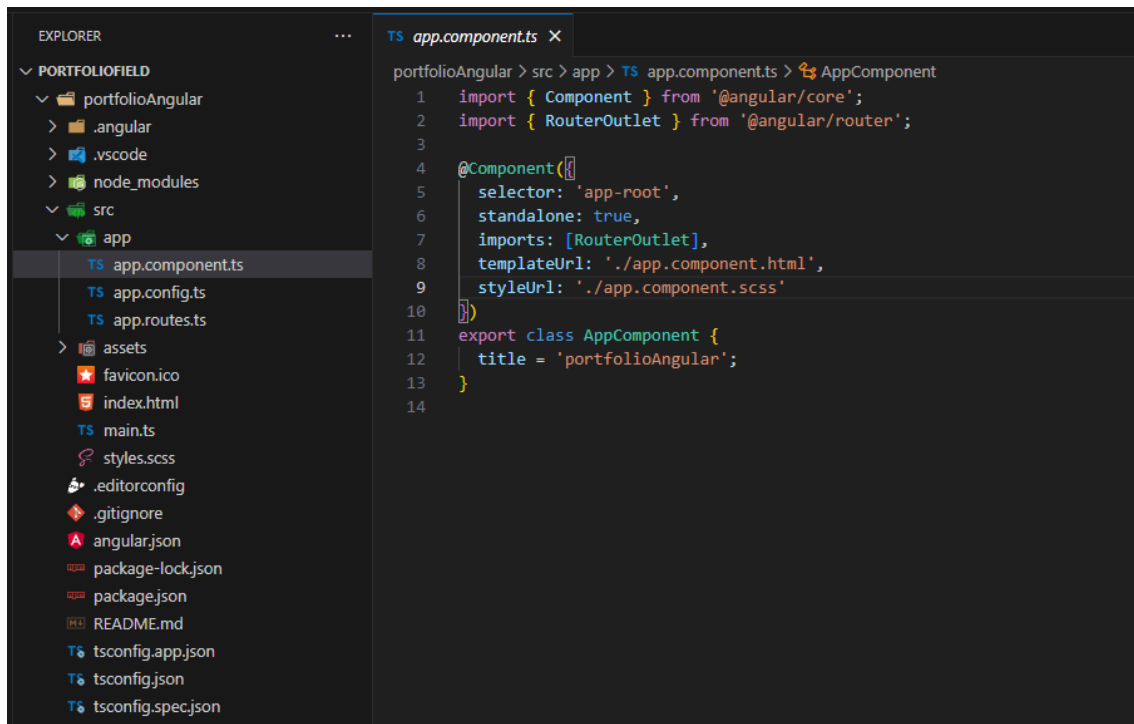


Figura 4 - Alteração do app.component.ts

Em **app.component.ts** apague a linha “styleUrl” e “templateUrl”, pois se referem aos arquivos deletados e insira a linha **template: `<router-outlet></router-outlet>`**, apague também o title de AppComponent.



Figura 5 - Pós-alterações no app.components.ts

O `<router-outlet>` é uma diretiva no Angular que é usada para indicar onde o Angular deve renderizar os componentes correspondentes às rotas definidas na sua aplicação.

A seguir, criaremos a estrutura de pastas, utilizaremos uma estrutura que consiste em criar uma pasta **app/modules**, seguida pela pasta **portfolio** (possui esse nome pois o projeto em questão trata-se de um portfólio), dentro da pasta **portfolio** adicionaremos as pastas **componentes**, **enum**, **interface** e **pages**. Essa estrutura foi escolhida devido a escalabilidade que proporciona.

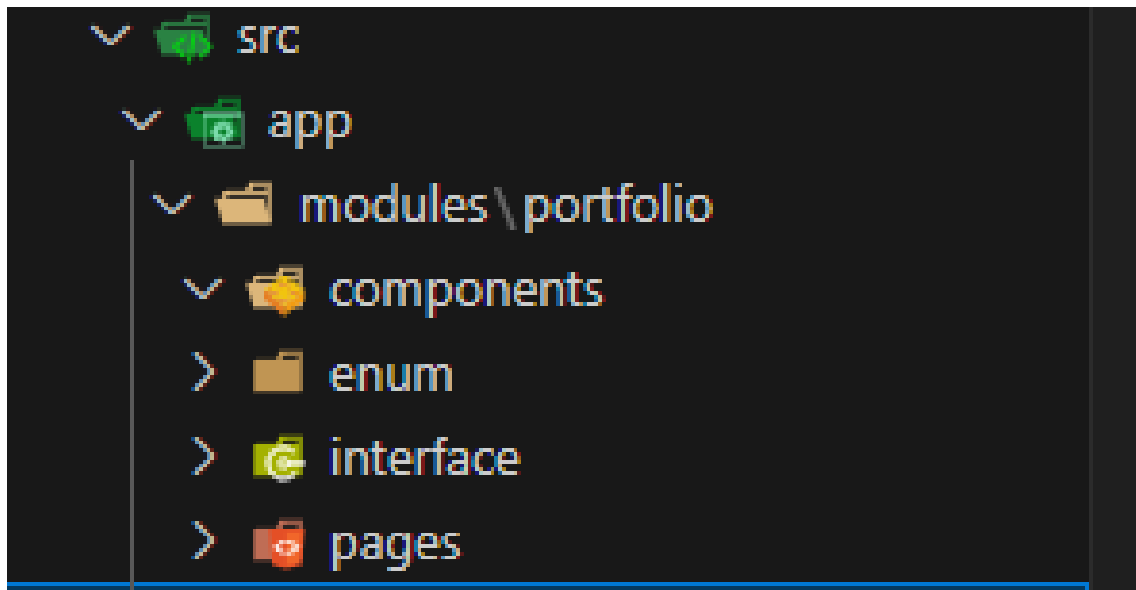


Figura 6 - Estrutura de pastas utilizada

A seguir utilizaremos o comando **ng g c**, onde **g** é uma abreviação para **generate** e **c** é a abreviação para **componente**, portanto geraremos o componente no caminho de pastas indicado:

```
PS C:\Users\Carlos\Desktop\portfolioField\portfolioAngular> ng g c modules/portfolio/pages/home
CREATE src/app/modules/portfolio/pages/home/home.component.html (20 bytes)
CREATE src/app/modules/portfolio/pages/home/home.component.spec.ts (605 bytes)
CREATE src/app/modules/portfolio/pages/home/home.component.ts (239 bytes)
CREATE src/app/modules/portfolio/pages/home/home.component.scss (0 bytes)
PS C:\Users\Carlos\Desktop\portfolioField\portfolioAngular>
```

Figura 7 - Utilização de comando ng g c

Ao utilizar esse comando, foram gerados os seguintes arquivos:

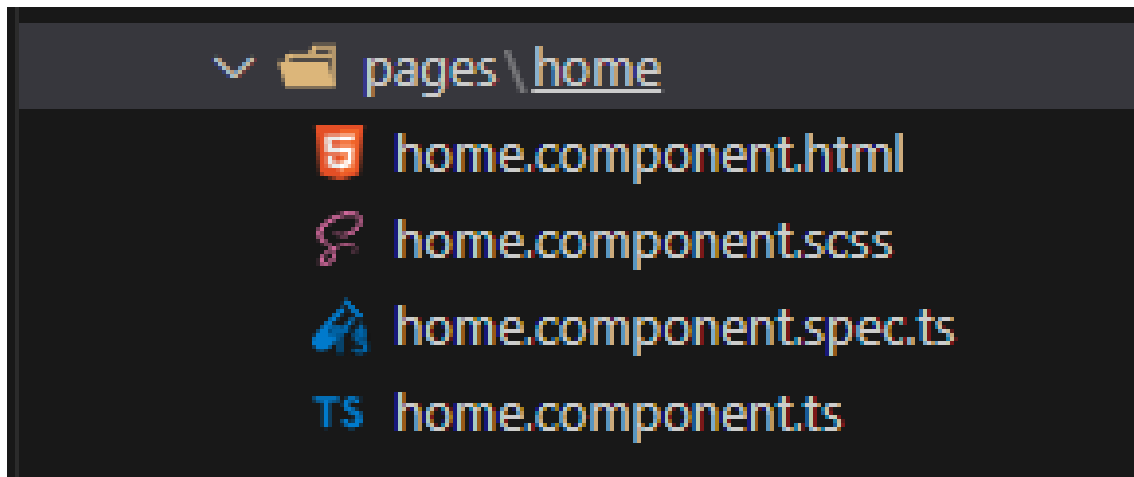


Figura 8 - Arquivos gerados através do comando `ng g c modules/portfolio/pages/home`

Atualizando agora o arquivo de rotas:

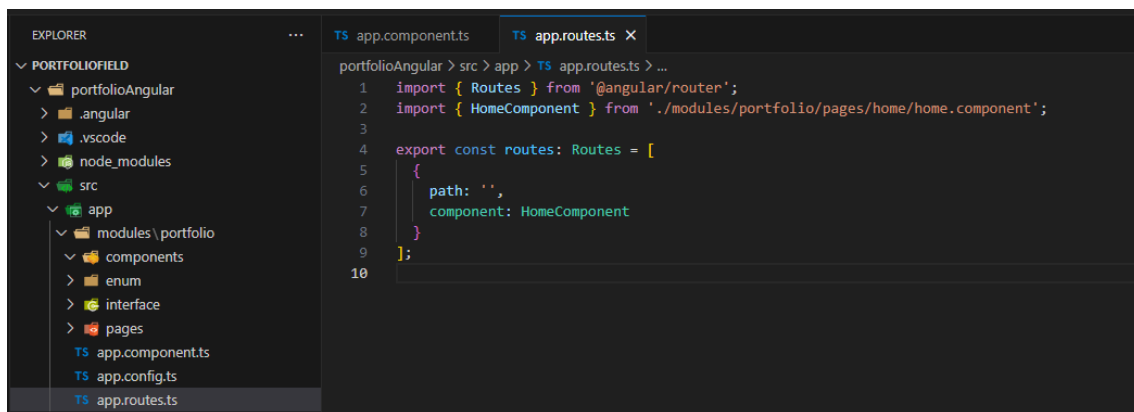


Figura 9 - Atualização do arquivo `app.routes.ts`

Feitos esses passos, a rota “ correspondente à nossa Home agora está sendo exibida para o usuário.

3 Criação de novos componentes

Para criar novos componentes, faremos uso do comando `ng g c modules/portfolio/componentes/ (Nome do Componente)`, criaremos então o componente `header`

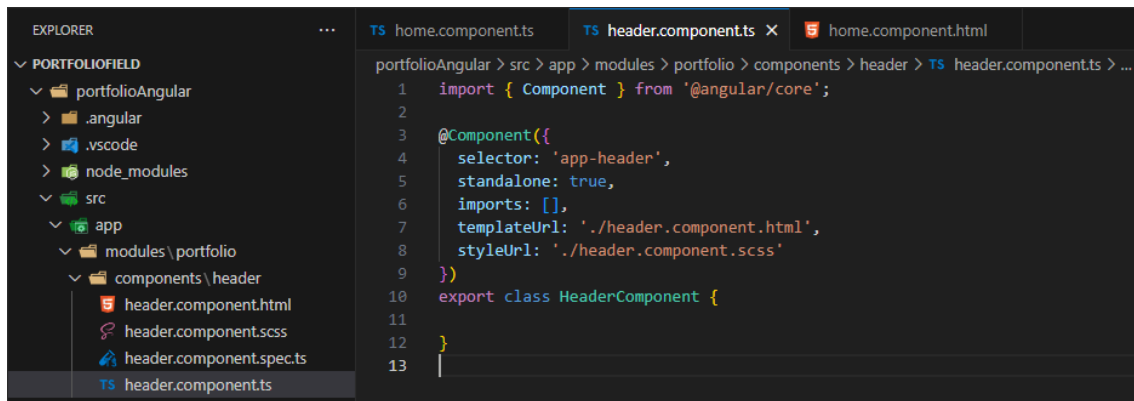


Figura 10 - Criação do componente header

Note que a propriedade **selector** tem como valor **app-header**, portanto, para utilizar o componente header no arquivo HTML da home (**home.component.html**), utilizaremos **<app-header />**.

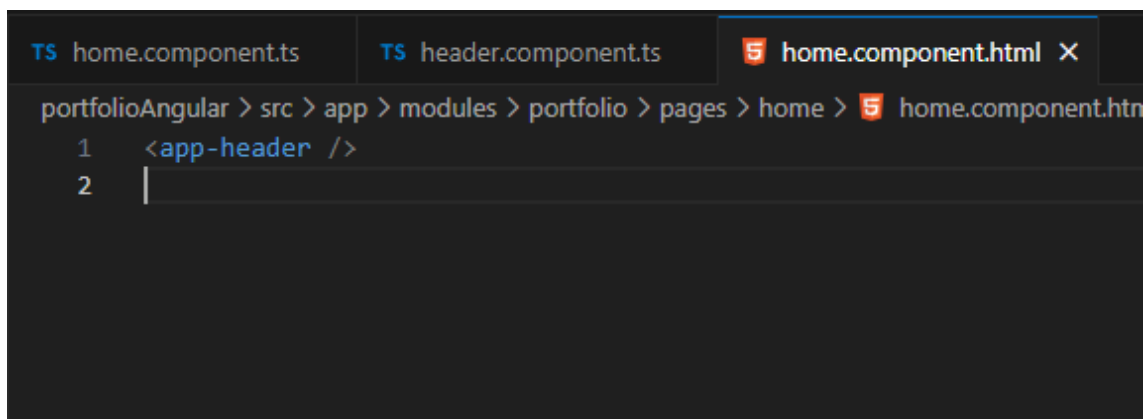


Figura 11 - Utilização do componente Header na Home

Referenciado o seletor em **home.component.html**, criaremos agora o HTML referente ao **header** em **header.component.html**:

```
header.component.html x TS home.component.ts
portfolioAngular > src > app > modules > portfolio > components > header > header.component.html > aside > div.container > section > figure
1 <header>
2   
9   <h1>Carlos Leite</h1>
10 </header>
11
12 <aside>
13   <div class="container">
14     <p>JavaScript | TypeScript | Angular.js | React.js | React Native | Node.js </p>
15     <p>São José do Rio Preto, São Paulo, Brasil</p>
16     <p>Atuo como freelancer a 8 meses, trabalhando principalmente com tecnologias JavaScript</p>
17
18   <nav>...
19   </nav>
20
21   <section>
22     <figure>...
23     </figure>
24
25     <figure>...
26     </figure>
27   </section>
28   </div>
29 </aside>
30
```

Figura 12 - HTML referente ao HeaderComponent

Logicamente, esse será o procedimento padrão para adicionar novos componentes nessa arquitetura escolhida para esse projeto.

4 Adicionando Interfaces

Confira a seguir como incluiremos interfaces ao projeto e no que elas serão utilizados.

A interface em TypeScript é um recurso poderoso que permite definir contratos para objetos em termos de tipos. No seu exemplo, a interface **IKnowledge** define um contrato que especifica quais propriedades e seus tipos devem estar presentes em um objeto que implementa essa interface, no caso definiremos os tipos para **src** e **alt**.

Ao criar interfaces nesse projeto, os arquivos possuíram um prefixo “I” antes de seus nomes.

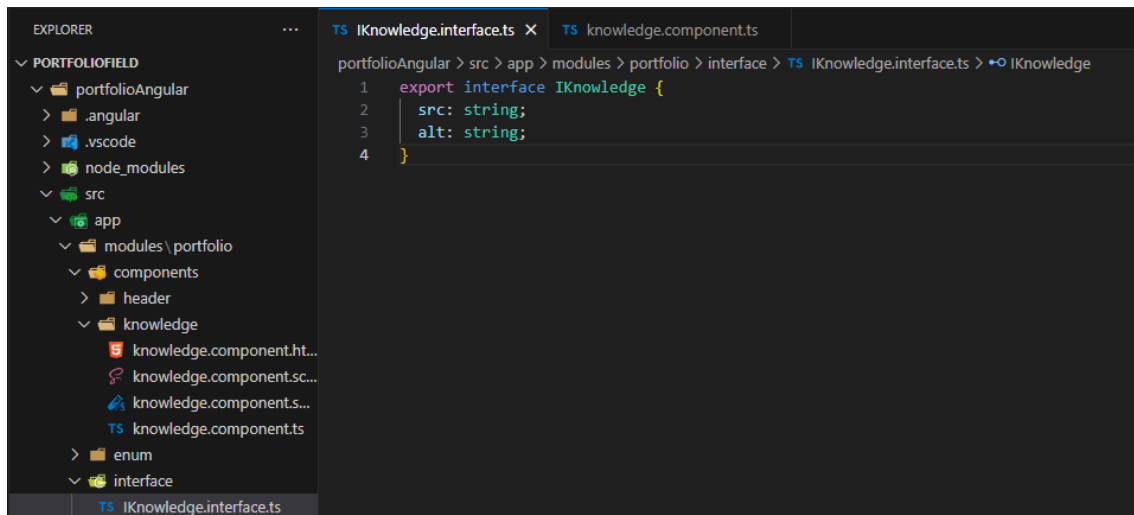


Figura 13 - Interface IKnowledge

Observe a seguir a utilização dessa interface:

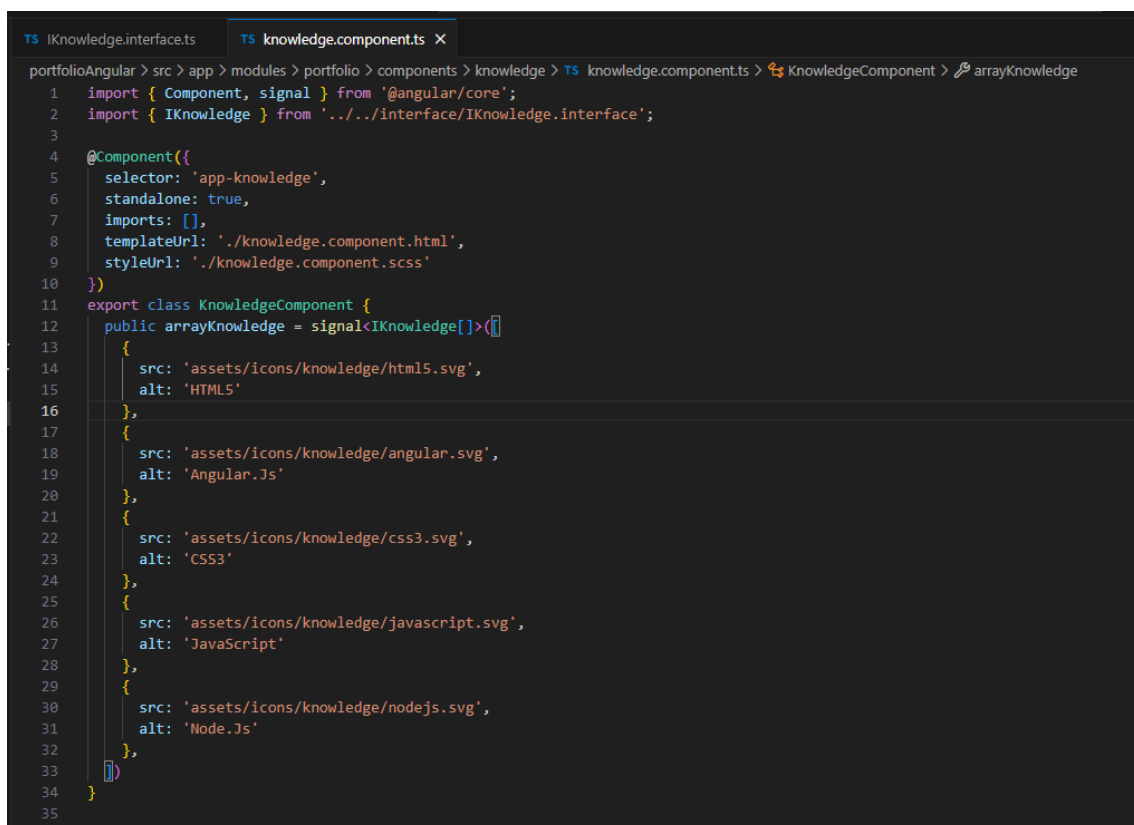
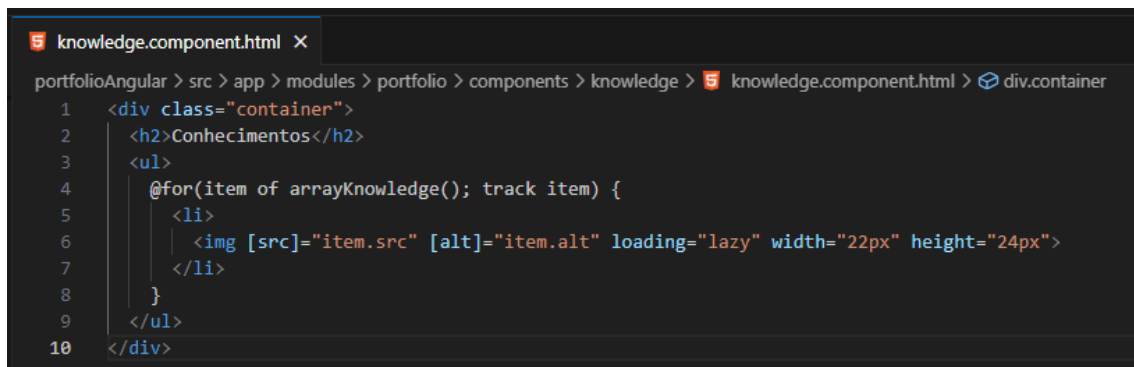


Figura 14 - Tipagem de dados através da interface

4.1 Aplicação dos dados em estrutura de repetição

Observe a seguir a estrutura de repetição utilizada para renderizar elementos que serão repetidos diversas vezes:



```
knowledge.component.html X
portfolioAngular > src > app > modules > portfolio > components > knowledge > knowledge.component.html > div.container
1 <div class="container">
2   <h2>Conhecimentos</h2>
3   <ul>
4     @for(item of arrayKnowledge(); track item) {
5       <li>
6         <img [src]="item.src" [alt]="item.alt" loading="lazy" width="22px" height="24px">
7       </li>
8     }
9   </ul>
10 </div>
```

Figura 15 - Estrutura de repetição em HTML

Note que essa estrutura de repetição é pertencente ao Angular.js, impondo que **item** sejam os itens do array e **track item** realiza o rastreamento dos itens da lista.

5 Tratamento em caso de loading extenso

Conforme evidenciado no capítulo anterior, o Angular.js possui algumas características próprias em algumas estruturas. Observe a seguir a estrutura implementada no componente **Projects** renderizado em **home.component.html**, que exibirá elementos com carregamento potencialmente lento e por isso é interessante que traga algum retorno ao usuário quanto ao status do loading:

```
home.component.html X
portfolioAngular > src > app > modules > portfolio > pages > home > home.component.html > main
1 <app-header />
2
3 <main>
4   <app-knowledge />
5   <app-experiences />
6
7   @defer(on viewport) {
8     <app-projects />
9   } @placeholder {
10    <section>
11      <h2>Carregando Projetos...</h2>
12    </section>
13  } @error {
14    <section>
15      <h2>Ocorreu um erro ao carregar os projetos. Tente novamente...</h2>
16    </section>
17  }
18 </main>
```

Figura 16 - Estrutura para feedback de loading

6 Estilização do Projeto

Para estilização, adicionaremos ao projeto a biblioteca UI Angular Material, que fornece alguns recursos interessantes para o design do site. Para isso, utilizaremos o comando **ng add @angular/material**.

```
PS C:\Users\Carlos\Desktop\portfolioField\portfolioAngular> ng add @angular/material
i Using package manager: npm
✓ Found compatible package version: @angular/material@17.1.2.
✓ Package information loaded.

The package @angular/material@17.1.2 will be installed and executed.
Would you like to proceed? Yes
✓ Packages successfully installed.
? Choose a prebuilt theme name, or "custom" for a custom theme: Indigo/Pink [ Preview: https://material.angular.io?theme=indigo-pink ]
? Set up global Angular Material typography styles? Yes
? Include the Angular animations module? Include and enable animations
UPDATE package.json (1114 bytes)
✓ Packages installed successfully.
UPDATE src/app/app.config.ts (347 bytes)
UPDATE angular.json (3054 bytes)
UPDATE src/index.html (538 bytes)
UPDATE src/styles.scss (182 bytes)
PS C:\Users\Carlos\Desktop\portfolioField\portfolioAngular>
```

Figura 17 - Instalação da biblioteca Angular Material

Além disso, a arquitetura que utilizaremos no código é conhecida como **SMACSS** (Scalable and Modular Architecture for CSS), escolhida por sua alta escalabilidade e facilidade de manutenção.

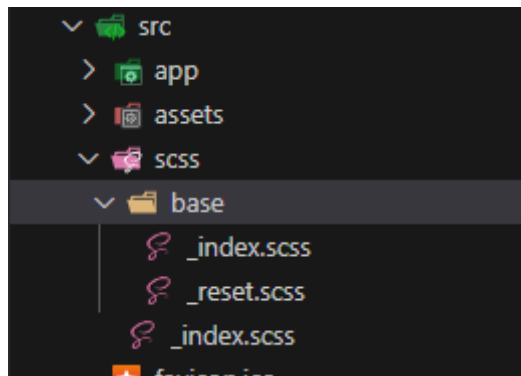


Figura 18 - Arquitetura inicial SMACSS

Note que os arquivos SCSS estão sendo nomeados com o prefixo `_`, essa decisão foi tomada para facilitar as buscas por arquivos SCSS nas sugestões de busca fornecidas pela IDE VsCode.

Note também que dentro da pasta **src** criamos a pasta **scss** e dentro da mesma, criamos um arquivo **_index.scss** e uma pasta **base**, esse arquivo será responsável por buscar o arquivo **_index.scss** dentro da pasta **base**, conforme mostra a imagem a seguir:

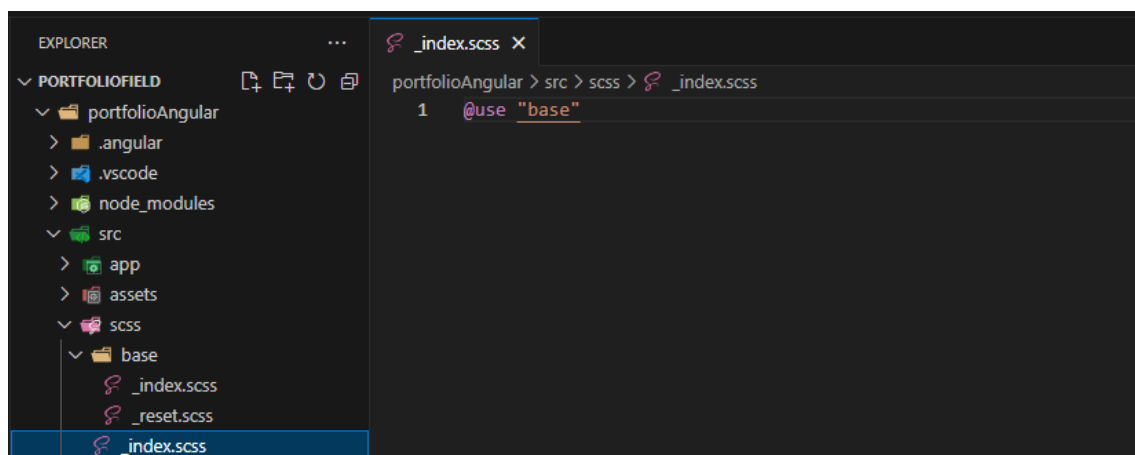


Figura 19 - Arquivo `_index.scss` responsável pela busca de outros arquivos

A princípio pode ser um pouco confuso, mas pensando em escalabilidade de código, essa é uma arquitetura eficiente e de fácil manutenção.

Agora observe o arquivo **_index.scss** existente dentro da pasta **base**:

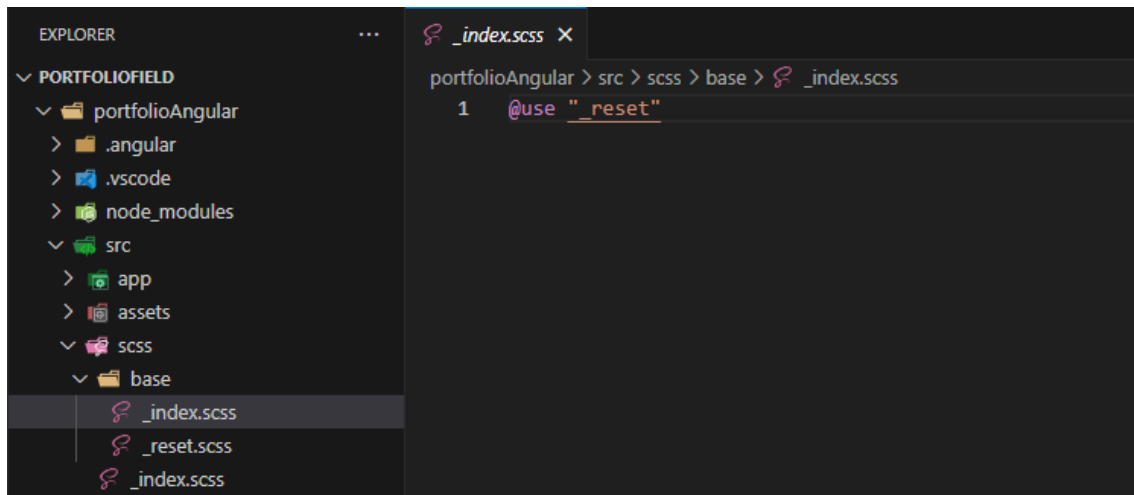


Figura 20 - Arquivo `_index.scss` presente dentro da pasta `base`

Note que, por hora, sua única função é buscar aquilo que está presente no arquivo `_reset.scss`, que, por sua vez, contém as configurações iniciais geradas na instalação da biblioteca Angular Material. Observe:

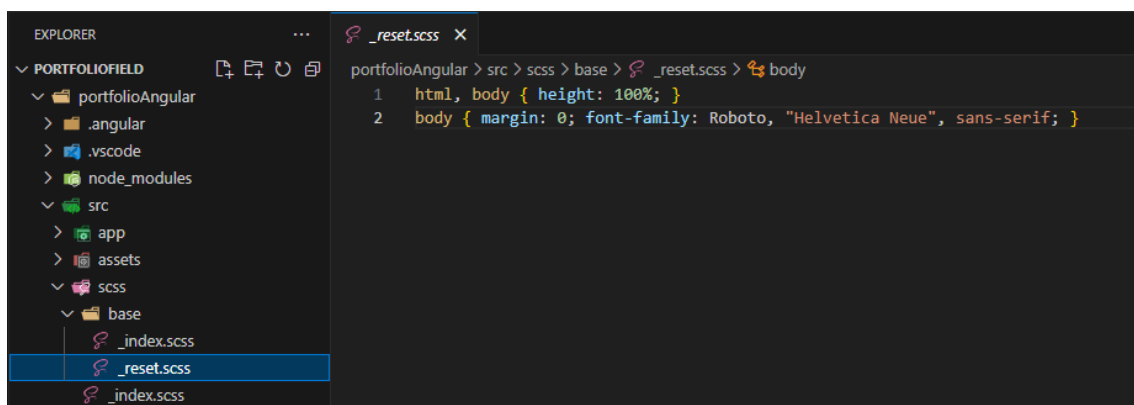


Figura 21 - Conteúdo do arquivo `_reset.scss`

Agora observe um pouco além na estrutura do código, no arquivo `styles.scss`:

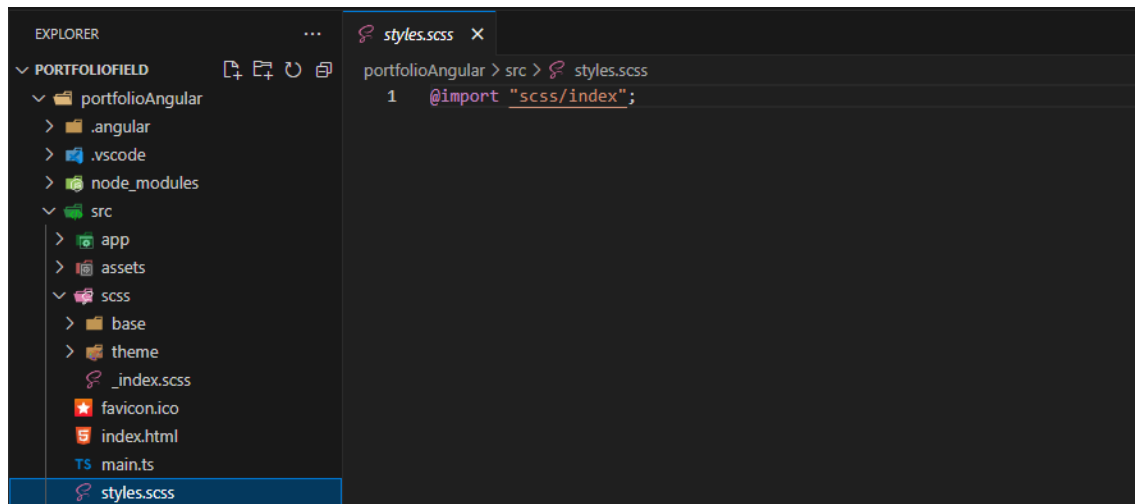


Figura 22 - Conteúdo do arquivo styles.scss

O arquivo **styles.scss** foi gerado na utilização do comando **npm install -g @angular/cli** e sua função é buscar os arquivos SCSS existentes no projeto e/ou somente contê-los, ao instalar a biblioteca Angular Material, as definições que agora estão no arquivo **_reset.scss** foram geradas nesse arquivo.

Note que o **import** utiliza a palavra **index** no caminho indicado para busca, essa palavra não é necessária pois **index** é um alias e sua existência já é pressuposta pelo SCSS, portanto sua retirada na escrita não afeta o produto final.