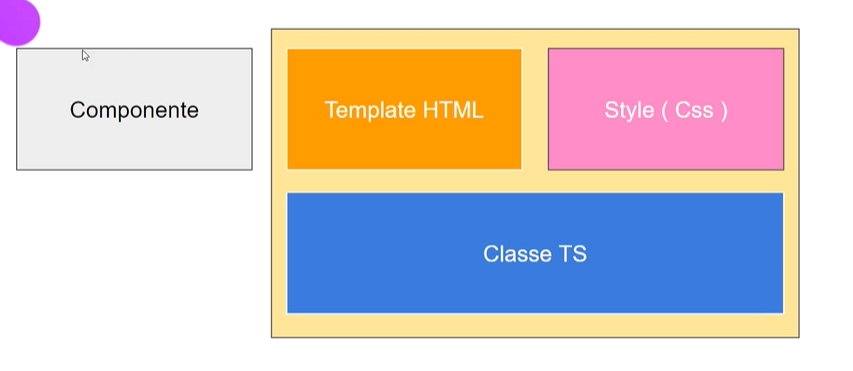
Comandos do ANGULAR CLI

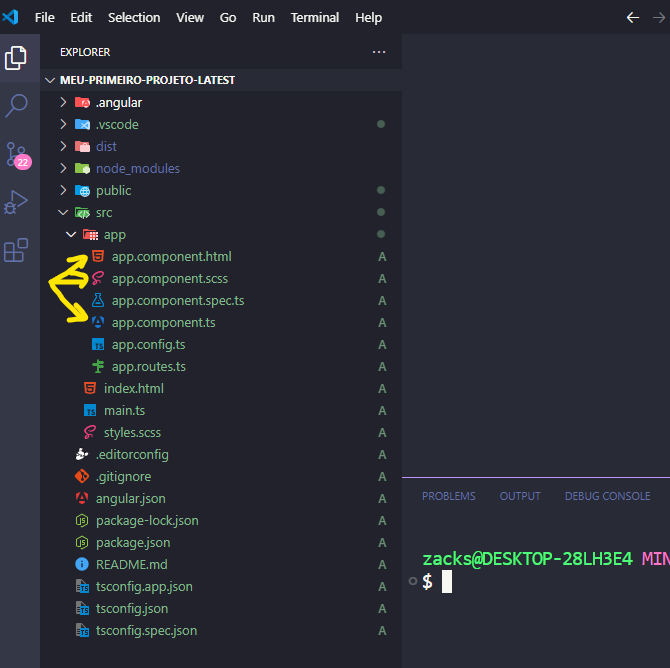
**NG BUILD** –Depois que o projeto estiver pronto este é o comando para gerar os arquivos que irão para o servidor(Deploy)

Este comando cria a pasta “dist” e dentro dela a pasta “browser”, onde os arquivos são salvos.



**COMPONENTES**

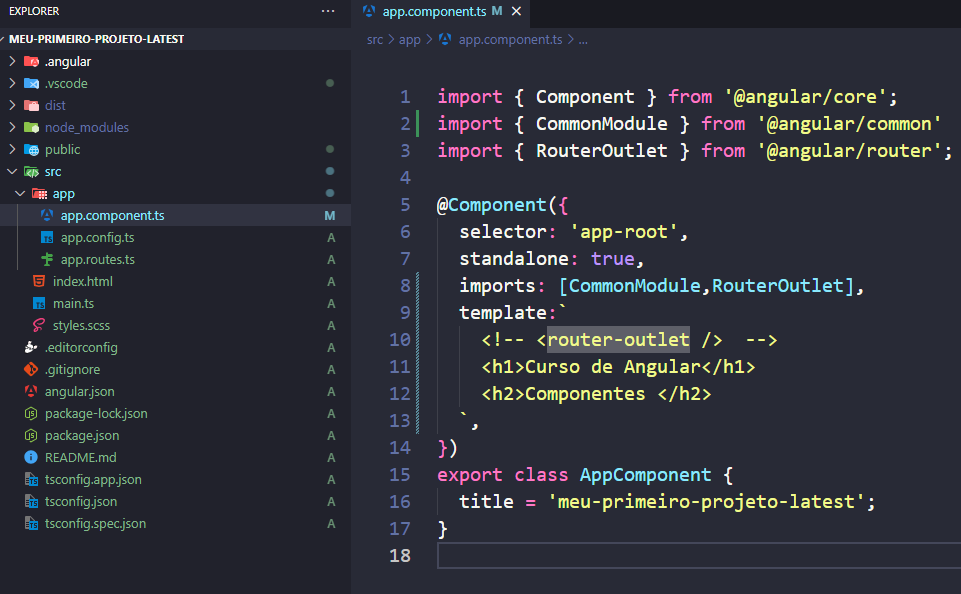
****



Decorator é @algumacoisa.

Na Aula do professor ele apagou os arquivos “app.component.html”, “app.component.scss” e “app.component.espec.ts”

E no decorator @Component adicionou o código abaixo:



**CRIANDO UM COMPNENTE**

Comando **ng g component components/new-component**

**1 -** No novo componente: **“Export”,**

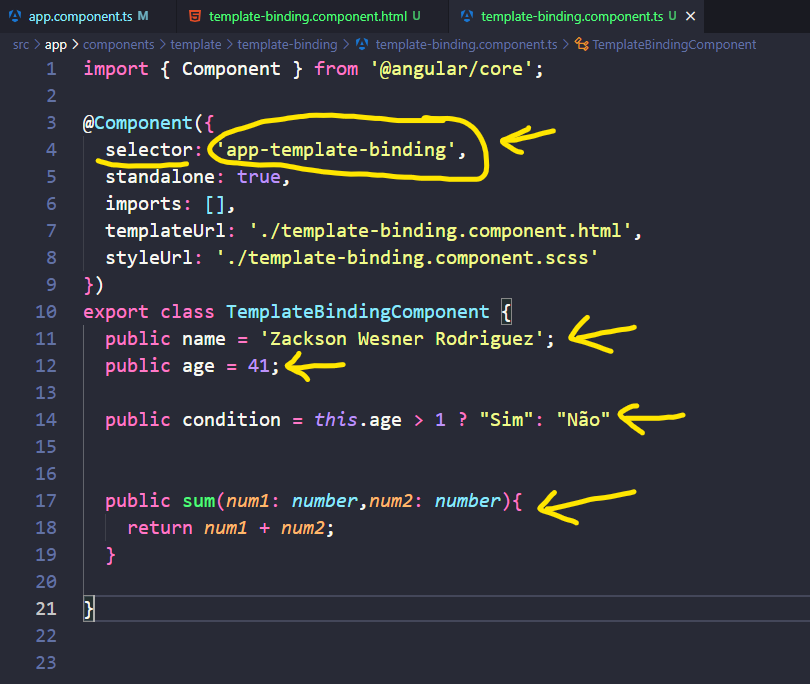
**2 -** no arquivo **app.components.ts -> no decorator @Component, imports**

**3 –** chamar o novo componente no HTML com a tag **<app-new-component />**

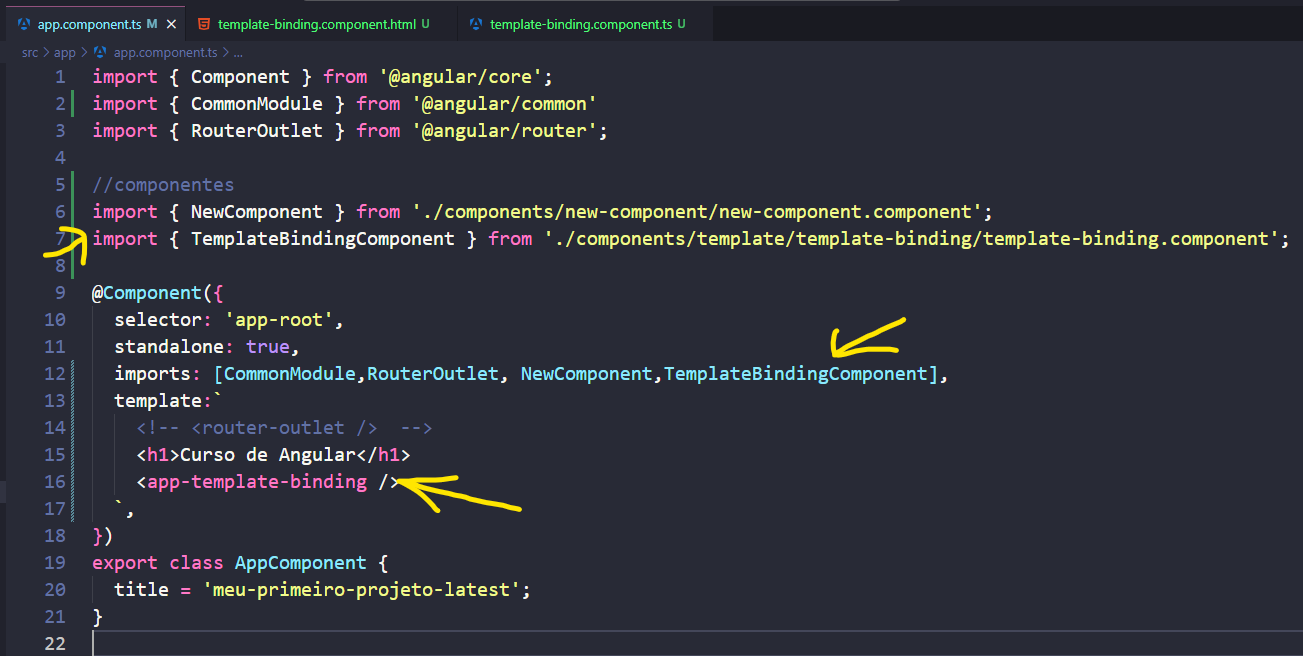
**INTERPOLATION**

Comando para criar novo componente

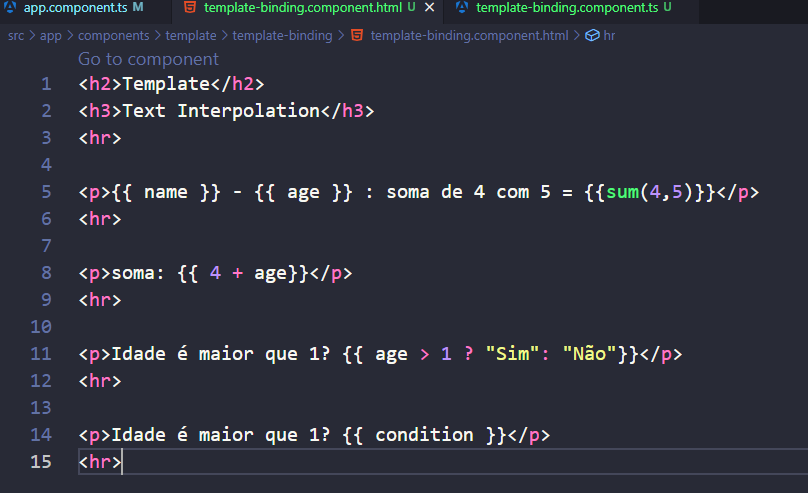
****

****

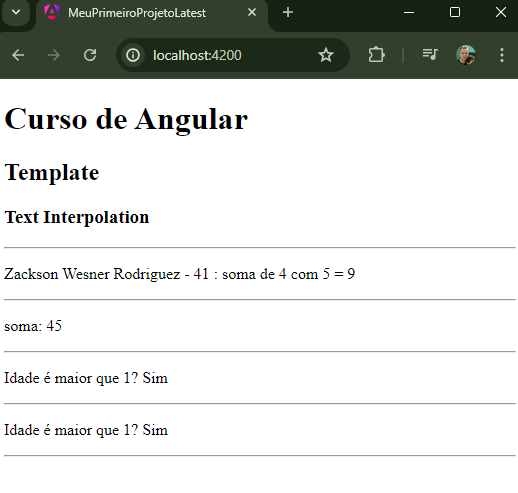
**Import, Imports, template**(< app-template-binding/>)

****

**Aqui ocorre a interpolação com uso de chaves duplas {{}}.**

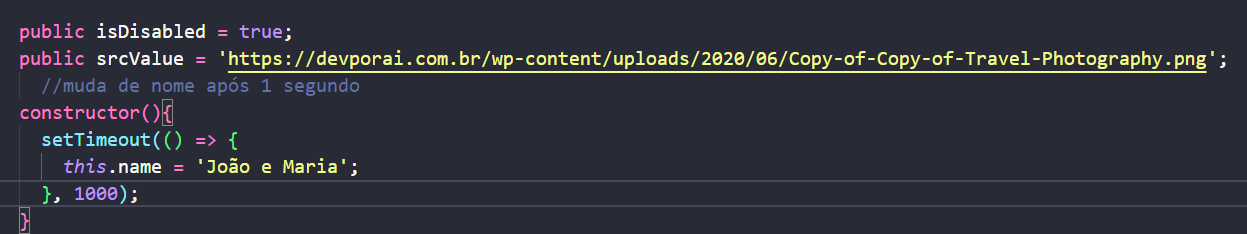
****

Resultado:

****

**PROPERTY BINDING**

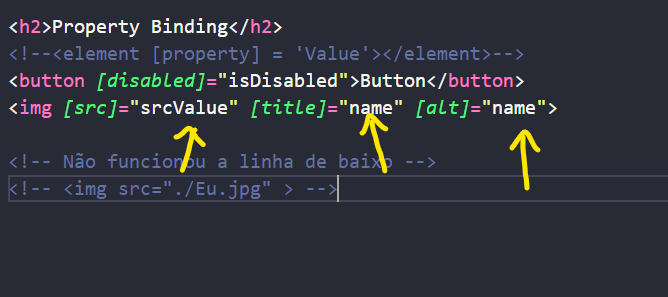
Em **template-binding.component.ts** foi adicionado os códigos:

****

O timeout acima é interessante, pois após o tempo de 1.000 milisegundos é alterado o valor da variável nome.

Em **template-binding.component.html** foi adicionado os códigos:

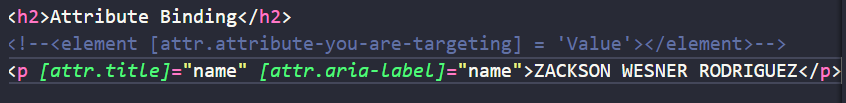
No HTML as variáveis são substituídas pelos seus respectiovos valores



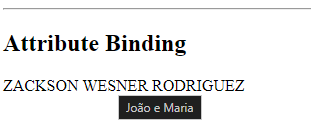
Resultado:



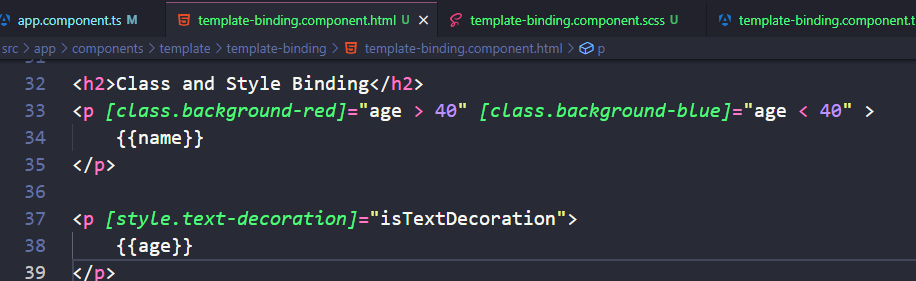
**ATRIBUTTE BINDING**

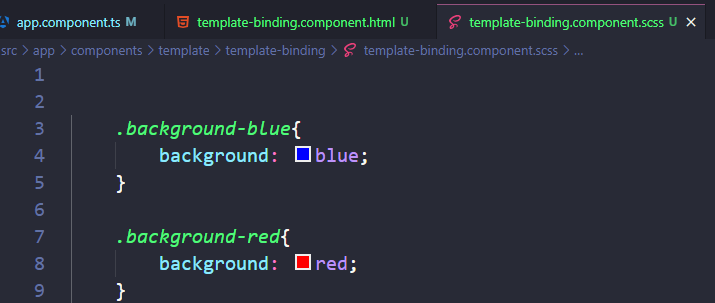
****

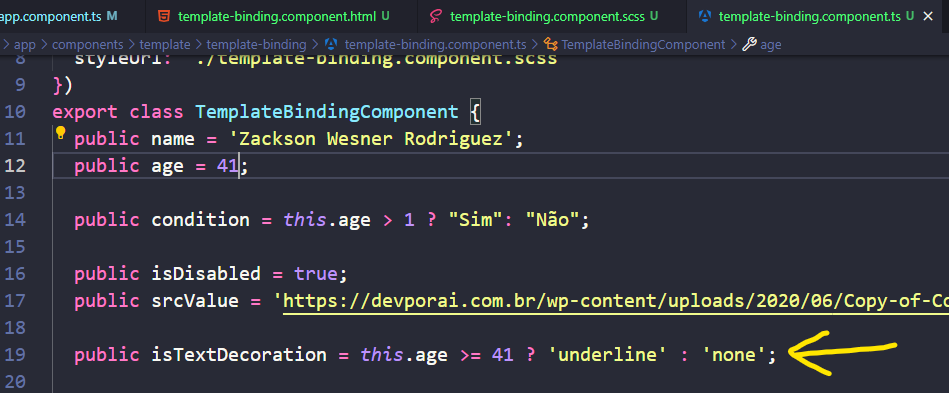
**RESULTADO**

****

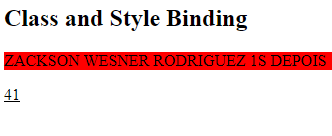
**CLASS AND STYLE BINDING**

****

****

****

**Resultado:**

****

**EVENT BINDING**

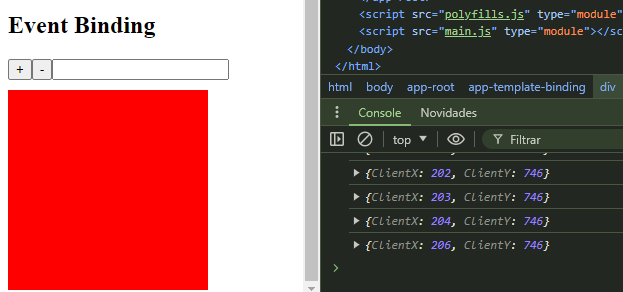
No HTML:

****

No TS:

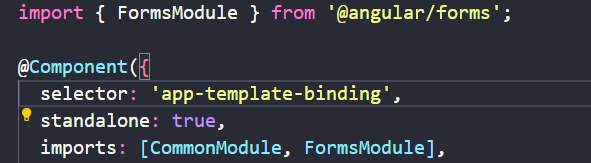
****

Na página exibida é possível identificar a posição X e Y do ponteiro do Mouse:

****

**Two-Way Binding**

Necessário importar no “imports do @component” o “FormsModule” no arquivo \*.ts.

****

**NgClass e NgStyle - Diretivas de atributo**

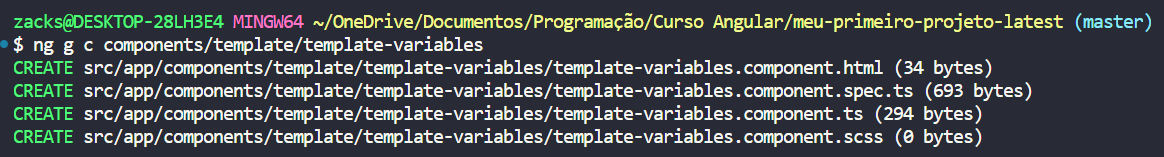
No HTML:

****

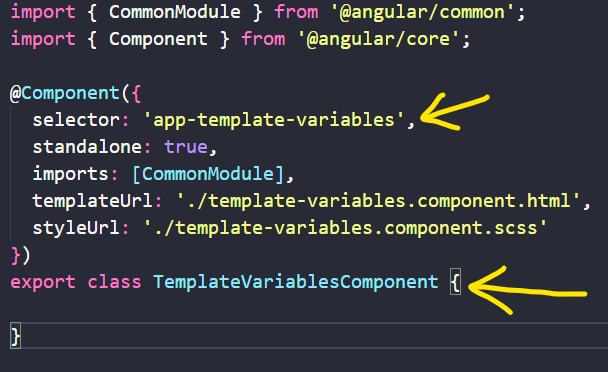
**Templates Variables**

Criar a pasta “template-variables

Fazer comando no Terminal:

****

**Copia o selector gerado:**

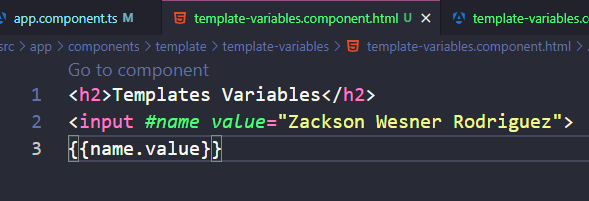
****

Adicionar no arquivo app.component.ts as informações dos imports e do template para o componente recém criado funcionar:

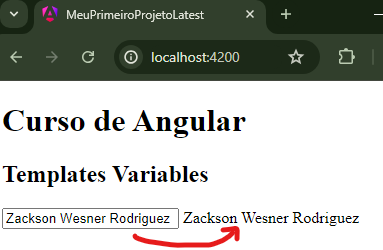


Agora vamos para o código relacionado ao TEMPLATE VARIABLES:

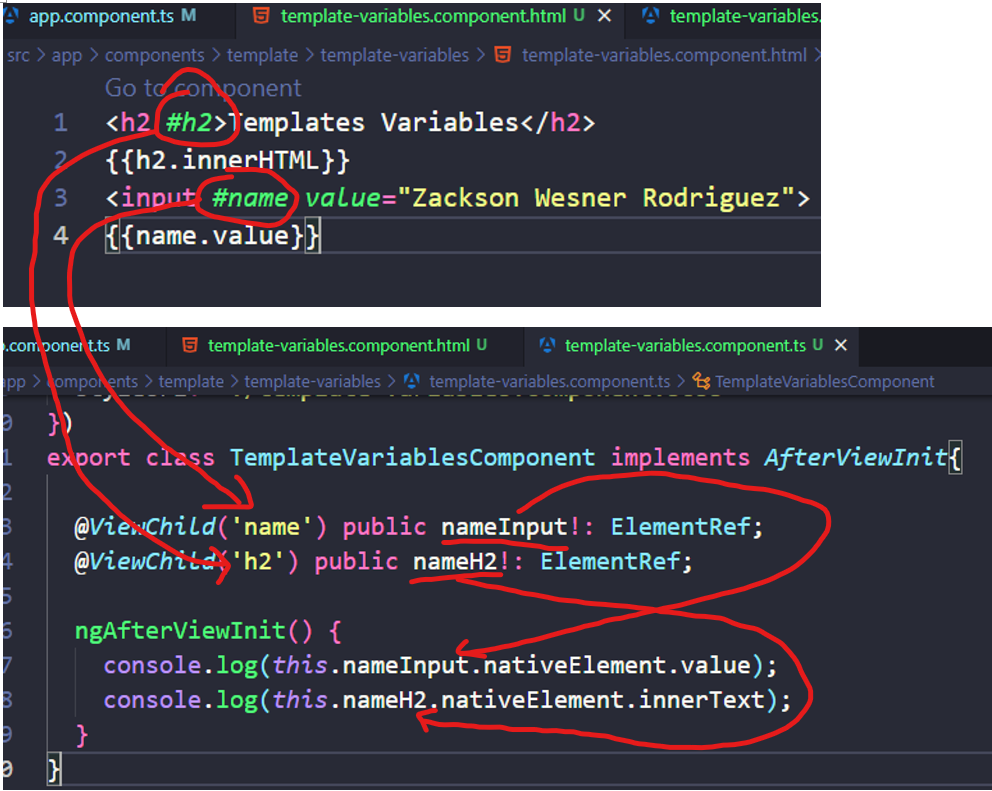
A variável #name declarada no HTML, pode ser resgatada no ANGULAR com a propriedade do INPUT, no caso abaixo a propriedade VALUE.



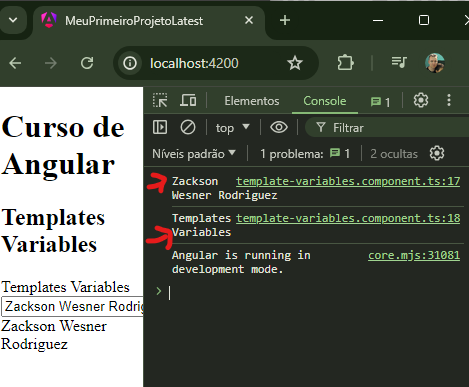
Saída na página:



Forma de resgatar as variáveis mediante JAVASCRIPT(arquivo \*.ts, no caso “template-variables.component.ts”):



Saída no console:

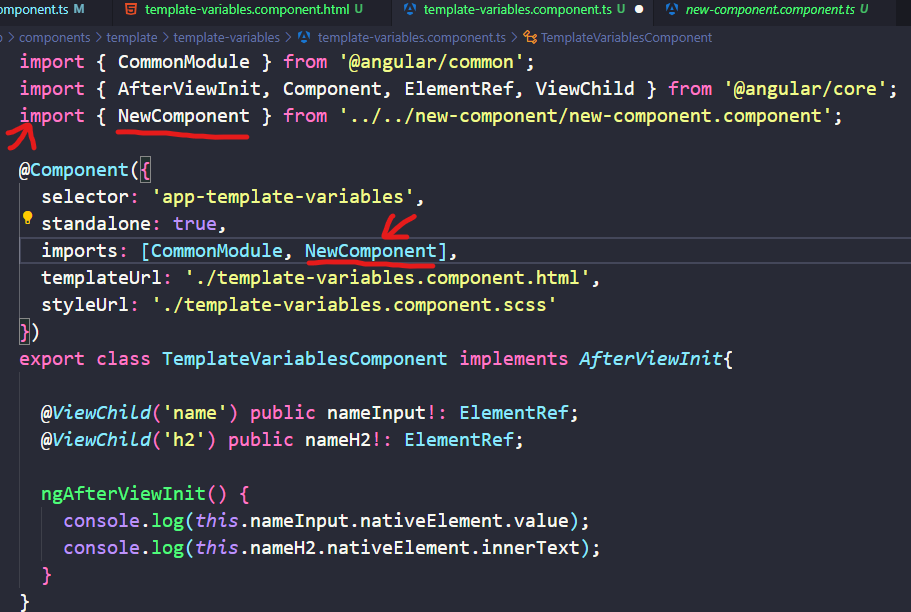


Também é possível resgatar valores quando há um componente dentro de outro componente:

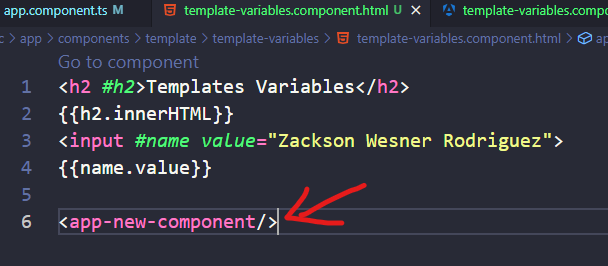
Exemplo:

Preparando o exemplo:

Primeiro importar o componente



Depois chamá-lo no template:



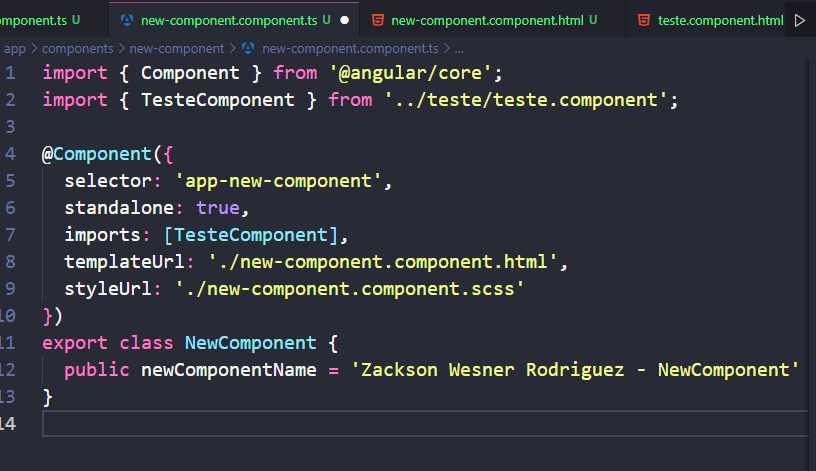
Saída:



Agora o exemplo em si:

Como referenciar:

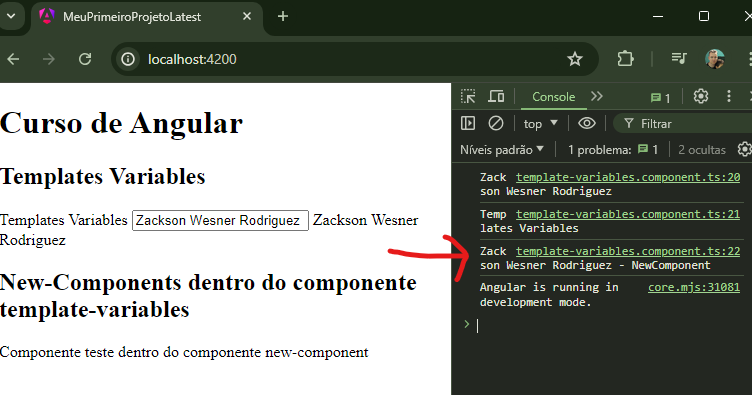
Primeiro colocar uma variável pública a ser exportada na classe do novo componente:



Como chamar:

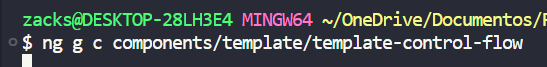


Como fica a saída no CONSOLE:

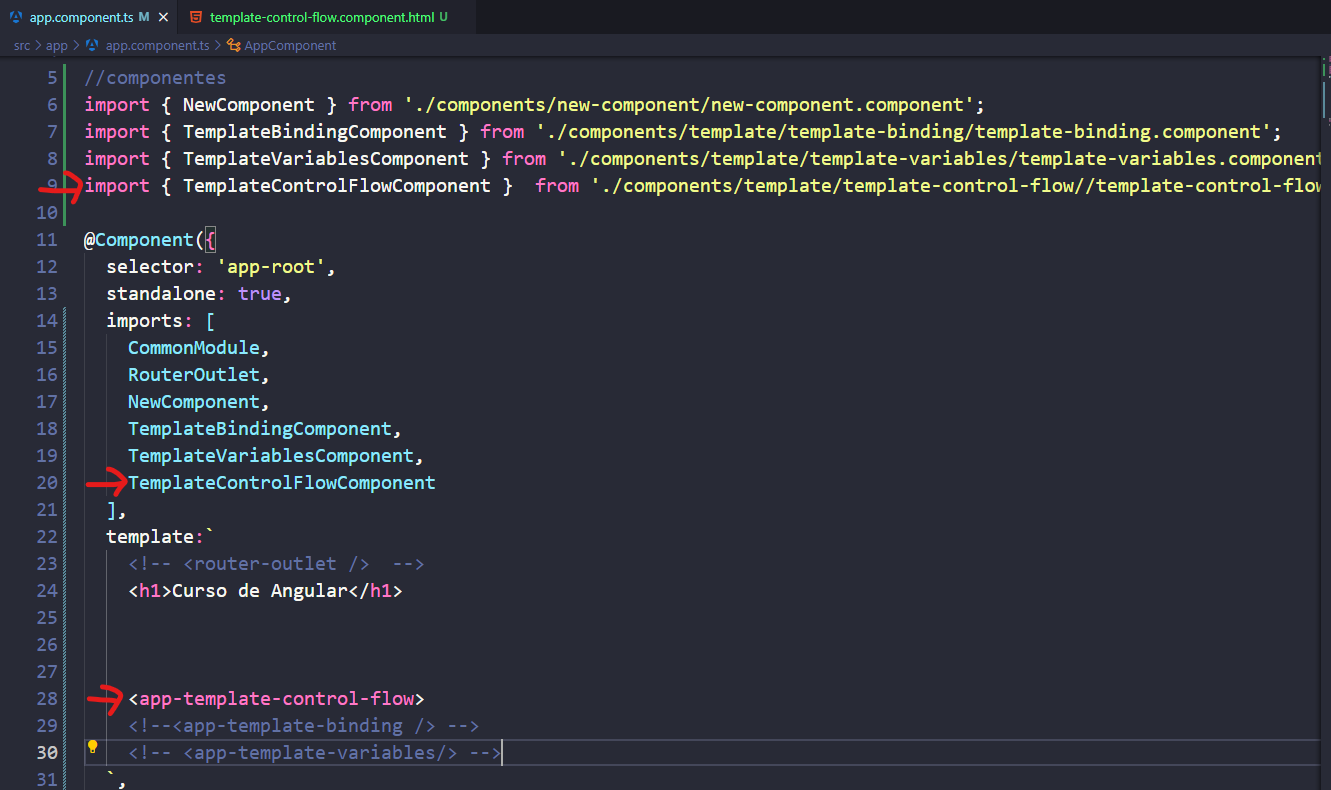


**Control Flow - IF**

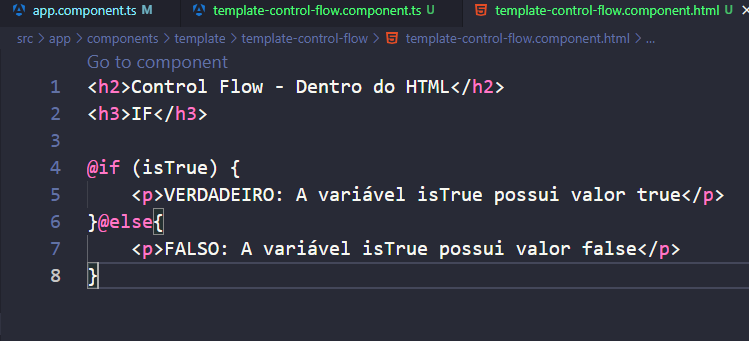
Criar novo componente como no exemplo abaixo:



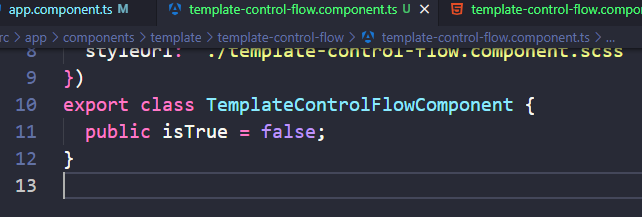
Importar classe do novo compenente para o arquivo app.component.ts:



Código @if no template do componente



Variável necessária no arquivo TS para funcionar o @if

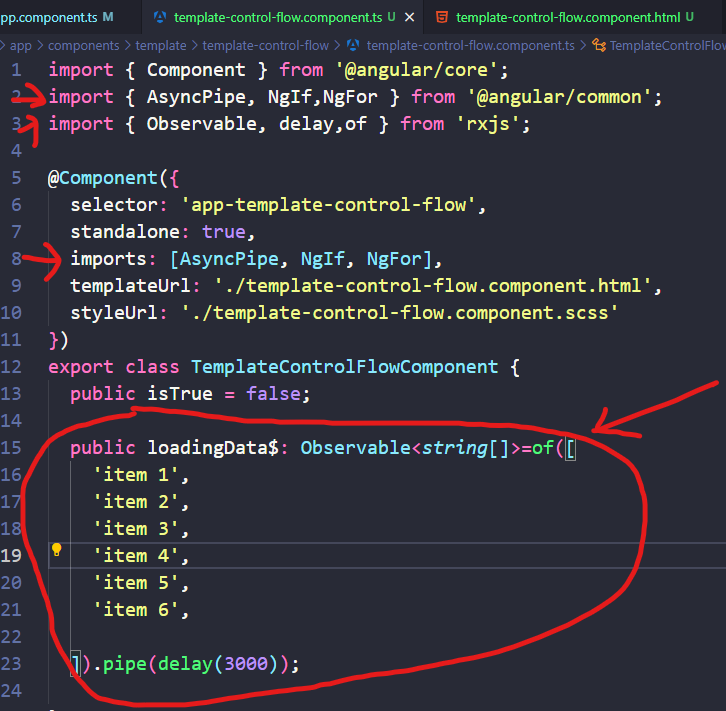


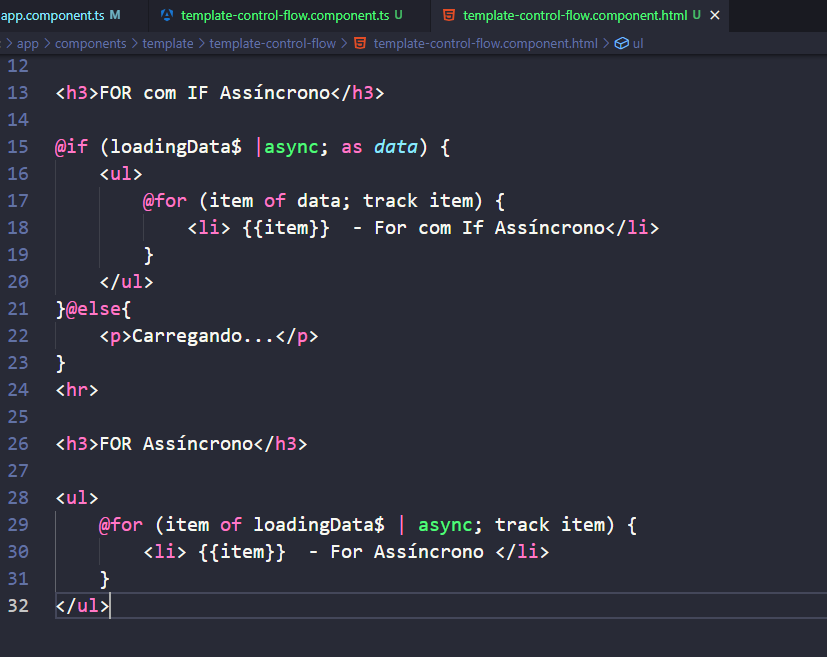
Saída:

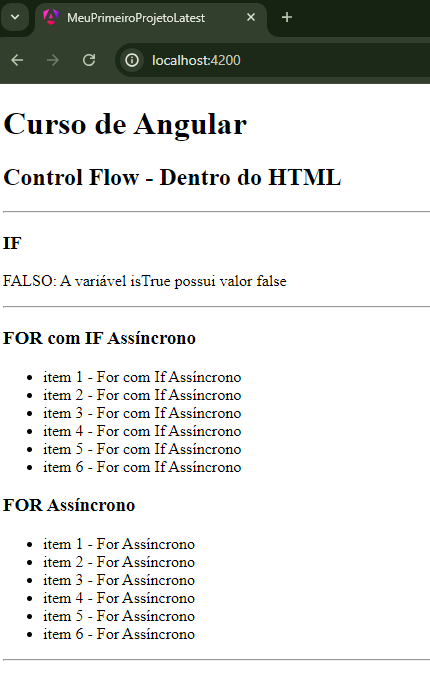


**Control Flow - FOR**

Adicionar as informações abaixo no arquivo TS:

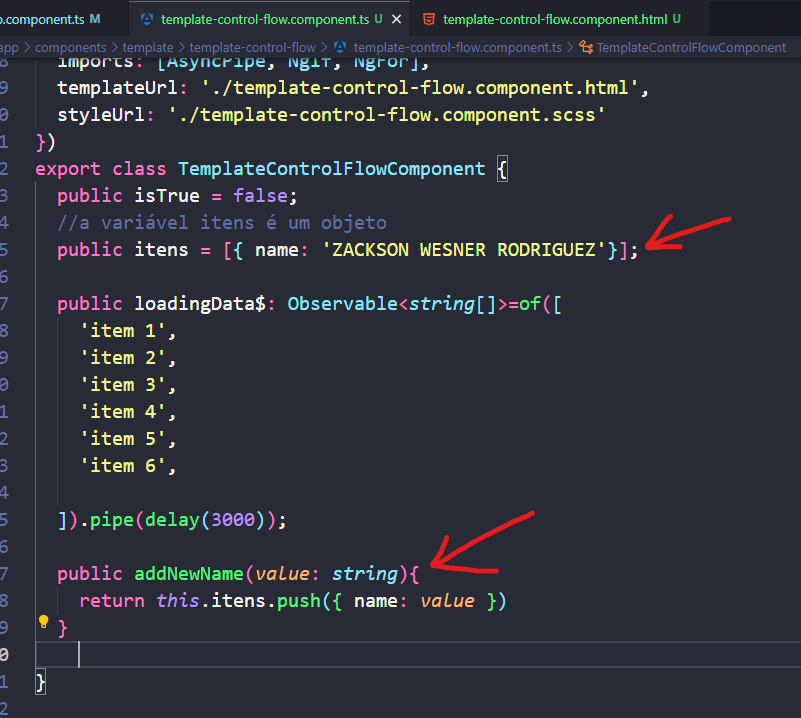




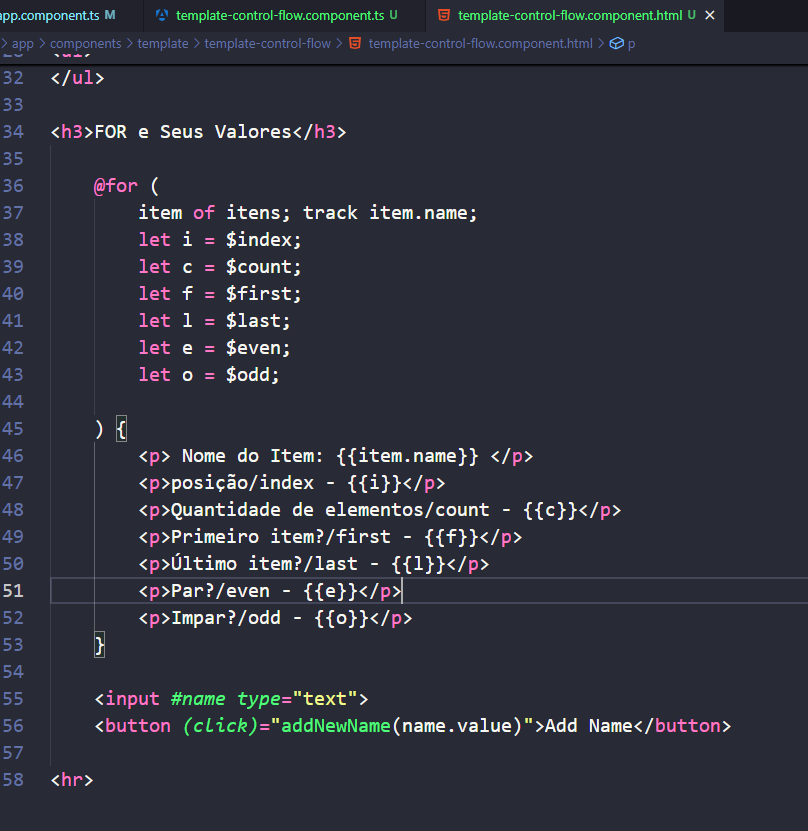


**FOR e Seu Valores**

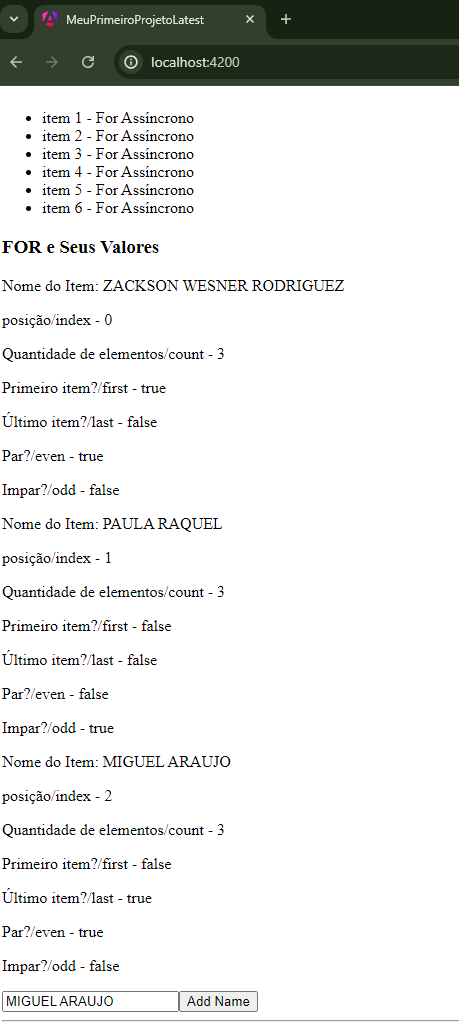
Criar uma variável objeto(itens) e uma função addNewName como abaixo:



No template:



Saída:



**FOR e Empty**

Código no HTML:



Código no arquivo TS:



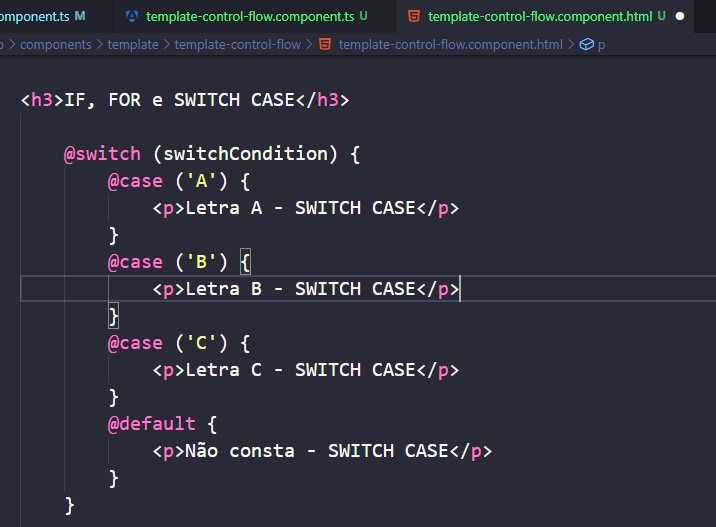
Saída:



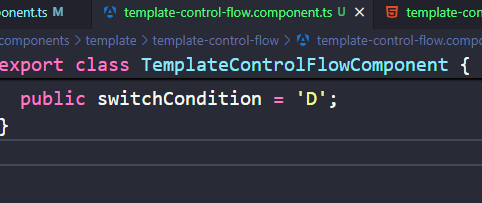
Para usar “@SWITCH” é necessário importar:



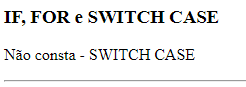
No HTML:



No TS apenas declaro uma variável para testar:

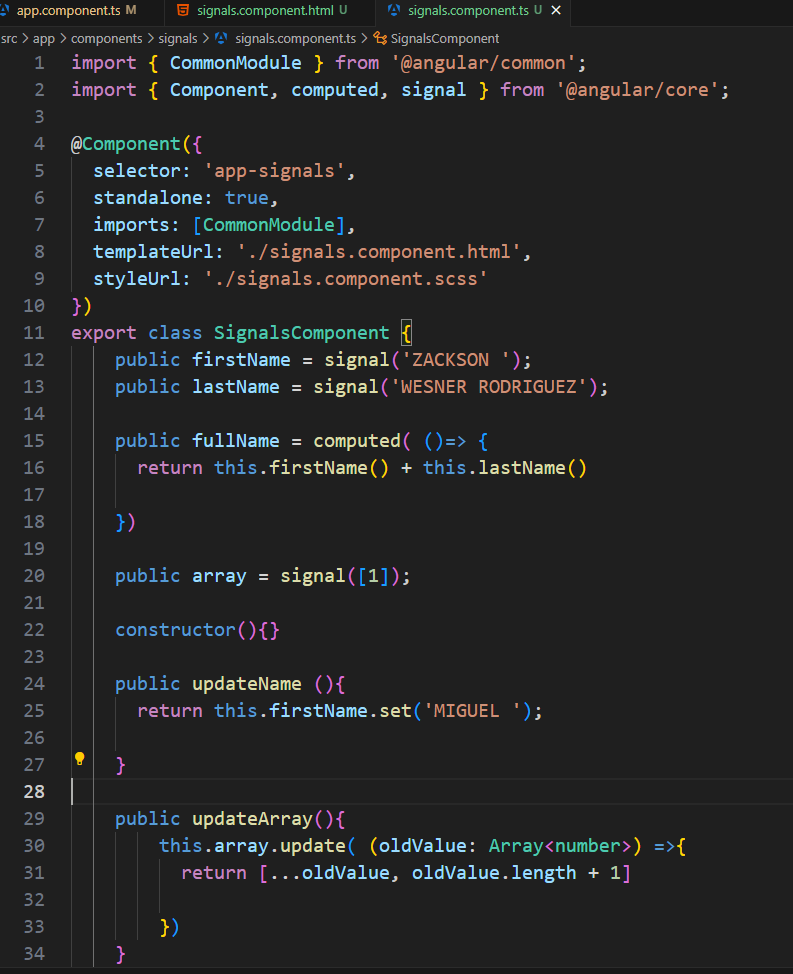


Saída:

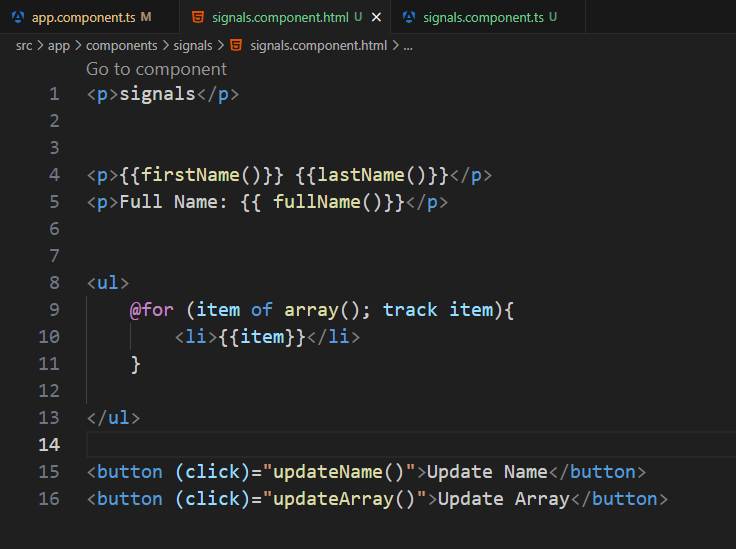


**SIGNALS**

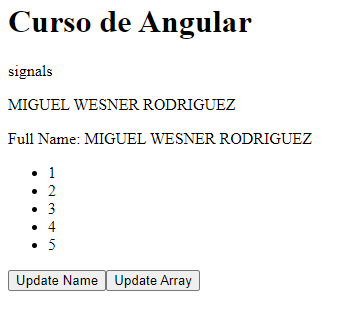
Arquivo \*.ts:



Template



Saída:



Signals - Effect

Dentro do SIGNALS existe o effect, que serve para monitorar mudanças nas variáveis signals. Só executa a função effect se houver mudança no signal, se não nada é executado.