**Angular**

**O que é?**

Angular é uma plataforma de desenvolvimento front-end, nessa plataforma esta um framework baseado em componentes, algumas bibliotecas e uma switch que auxilia no desenvolvimento, no build e nos testes.

**Ferramentas:**

Visual studio code, Node.js e angular.

**Como instalar o Angular de forma geral:**

Va para o cmd e utilize o comando:

npm i -g @angular/cli

No visual studio adicione a extensão angular language servisse.

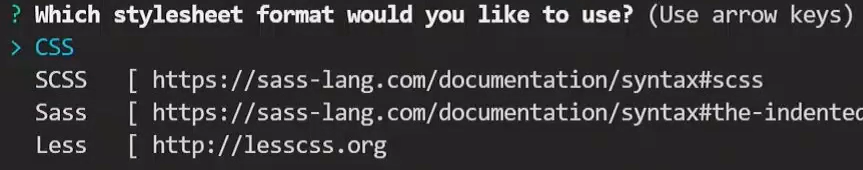
**Criando projeto angular:**

Após criarmos uma pasta acessamos o terminal no local da pasta e usamos o comando:

ng new nomedoprojeto

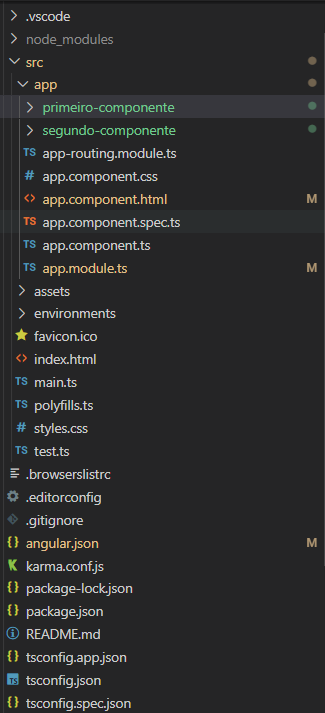


Isso é uma pergunta se queremos adicionar a biblioteca angular routing, que serve para fazermos o roteamento das nossas paginas.



Pergunta referente ao formato de arquivos de stylesheet utilizado.

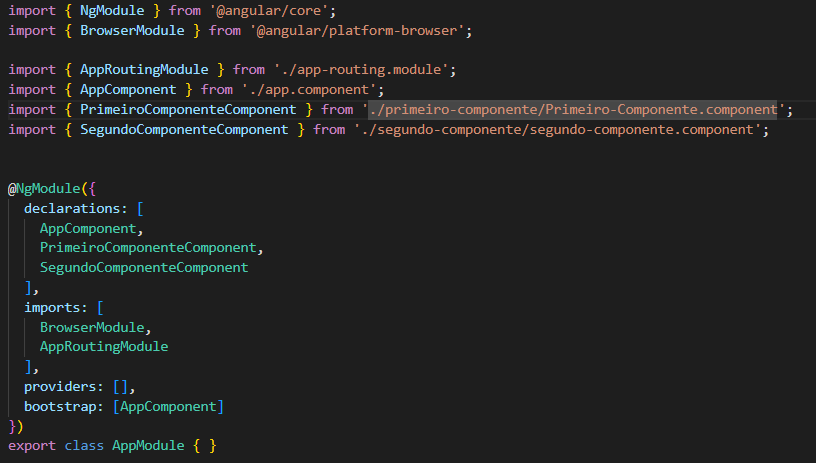
**Estrutura do projeto:**



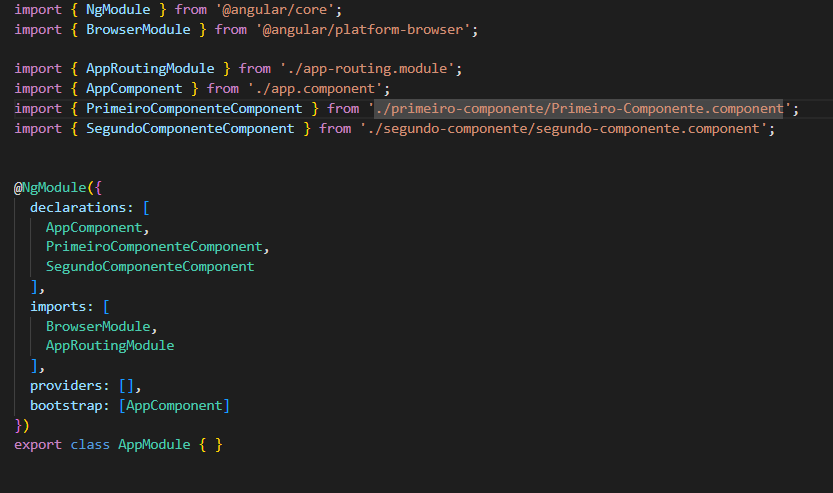
No arquivo package são colocadas as dependências do projeto.

No arquivo angular são colocadas as configurações do projeto, como sourceRoot que determina a pasta do projeto e prefix que determina o prefixo do projeto.

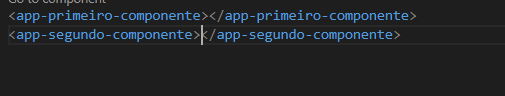
Como no prefix do angular.json esta app a pasta app ira conter nossos componentes.



No arquivo app.module.ts contém os componentes que importamos.



No arquivo app.component.html possui a aplicação dos componentes na tela.



Na pasta assets são colocados arquivos de mídia como imagens por exemplo.

**Components:**

Componentes são “partes de códigos html estilizados e reutilizáveis”.

Estrutura do componente:

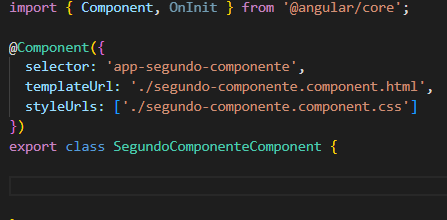
Devemos importar a classe componente do angular/core.

Usamos o @Component({}) para especificarmos o componente.

Usamos o método export class nomedeimportação{} para especificarmos como exportamos o componente.



No componente temos que especificar o seletor o template e o style o template e o style podem ser usados através de url para outros arquivos:



**Podemos criar a estrutura de um componente através do comando:**

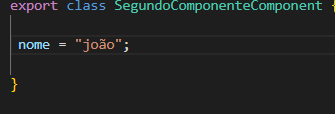
ng generated component nomedocomponente

**Interpolação de texto:**

É a forma que inserimos dados da classe no template:

Utilizamos {{nomedavariavel}} para dizermos o nome da variável na classe.





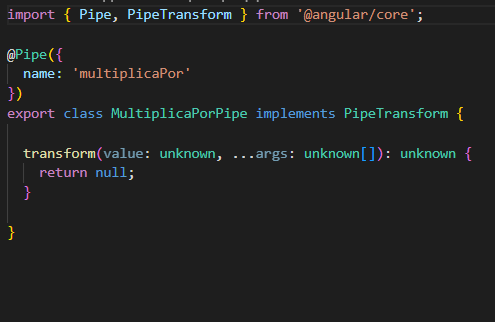
Podemos usar **|** para formatarmos o texto o angular chama de **pipes** e existem diversas formatações possíveis

<https://angular.io/guide/pipes>

**Pipe personalizado:**

Podemos criar uma classe Typescript com um pipe personalizado usando o seguinte comando:

ng generate pipe nomedopipe



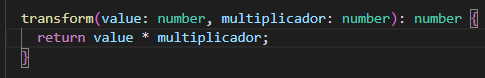
Essa classe pipe também é adicionado no app.module.ts

Sintaxe do transform:

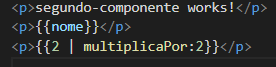
transform(variável: tipoprimitivo, ...args: tipoprimitivo[]): tipoprimitivo {

retun retorno;

}



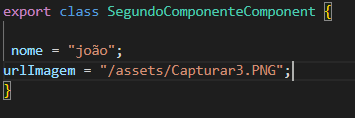
Exemplo de aplicação do pipe:



**Property\_Binding:**

É uma forma de linkarmos propriedades de componentes html com variáveis das classes componente:

Varivéis:



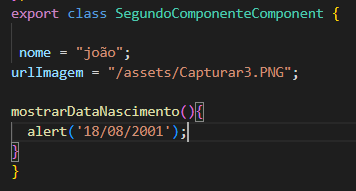
Property\_binding:



**Event\_Binding:**

É uma forma de linkarmos um método nas classes a um componente HTML:

Método:



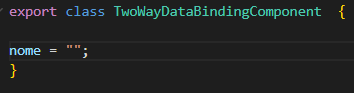
Event\_Binding:



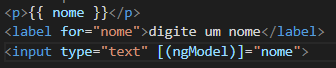
**Two\_Way\_Data\_Binding:**

Para utilizarmos um input em uma variável utilizamos o data\_binding:

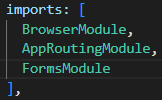
Variável:



Data\_binding:



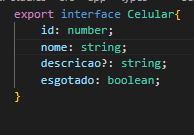
Porém devemos importar a biblioteca **FormsModule** no app.module.ts:



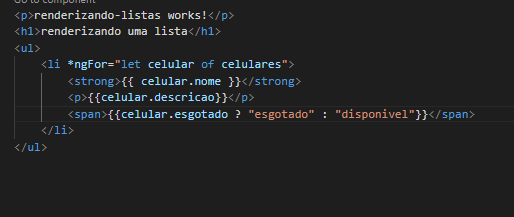
**Renderizando lista:**

Para renderizarmos uma lista primeiro criamos uma.

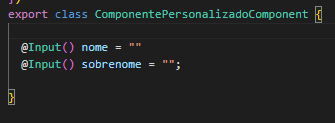
Criamos a interface em um arquivo ts separado com as propriedades que serão criadas na classe do componente:







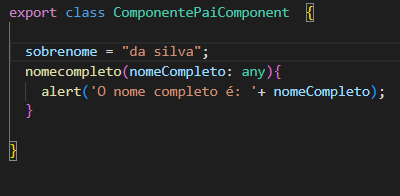
**Componente Personalizado:**

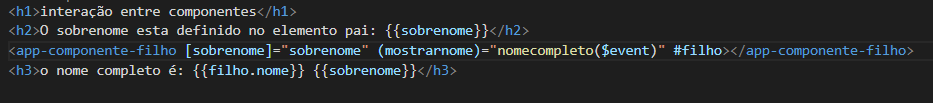


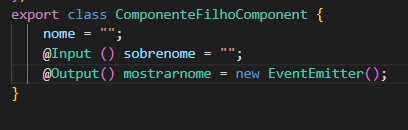


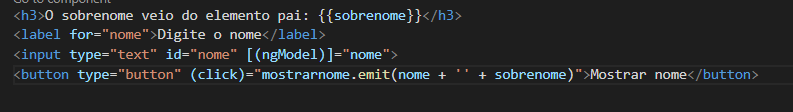


**Comunicação entre componentes:**









**Serviços:**

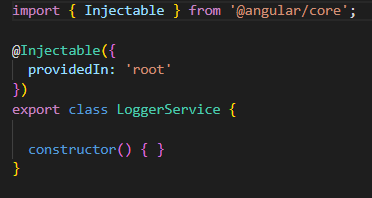
Um serviço basicamente é uma funcionalidade que poderá ser utilizada em diversos componentes.

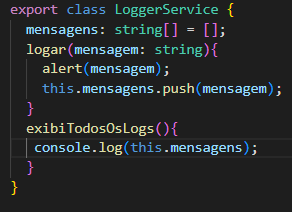
Para gerar um serviço de forma automática se utiliza o comando:

ng generate service nomedoserviço

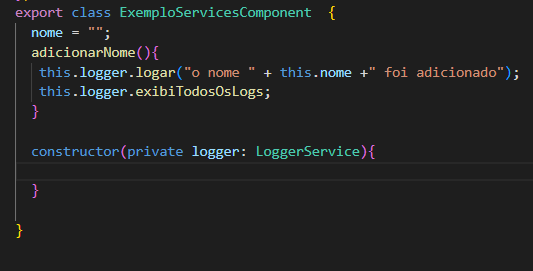
isso irá gerar um arquivo typescript:

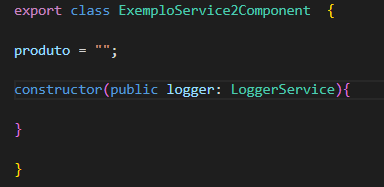






Aplicando o servisse:



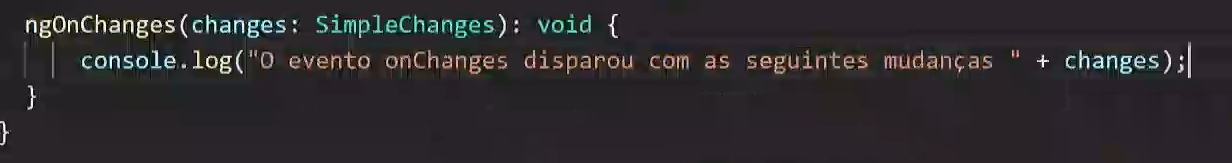


No caso de atribuído um public ao serviço podemos utiliza-lo no template:

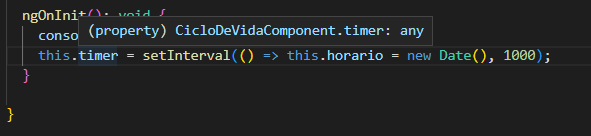


**Ciclo de vida dos componentes:**

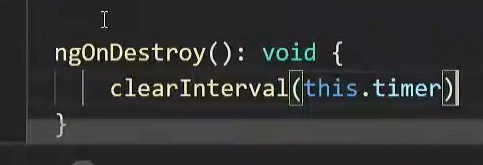
O primeiro evento a ser disparado no componente é o:



O segundo evento a ser disparado no componente é o:

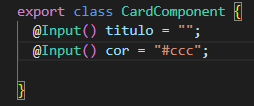


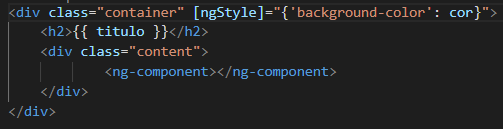
O ultimo evento a ser disparado acontece quando o componente sai de tela e é o:

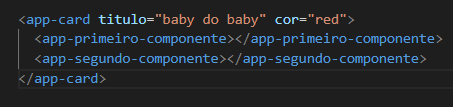


**Ng\_Content:**

É utilizado quando queremos colocar componentes dentro de outros.







**Adicionando\_Roteamento:**

É importante marcar N na hora de criar o projeto indicando que não quer importar e configurar a biblioteca de roteamento de forma automática.



Devemos instalar a biblioteca angular router através do comando:

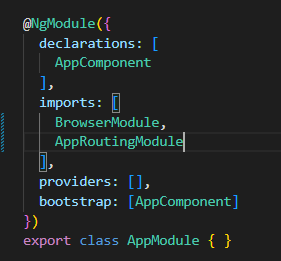
npm i @angular/router

Agora devemos criar um modulo responsável pelo roteamento:

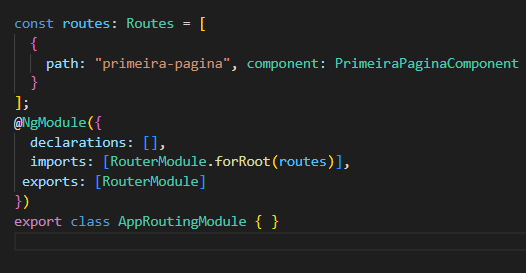
ng generate module app-routing

esse arquivo deve ir na pasta app.

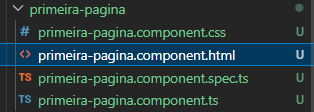
Devemos também acrescentar essa classe no app.module:



Após isso vamos acessar essa classe:



Com essa configuração acrescentamos a página “primeira-pagina” ao componente PrimeiraPaginaComponent:



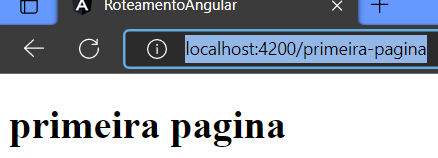
Após isso no arquivo app.component.html adicionamos a tag:



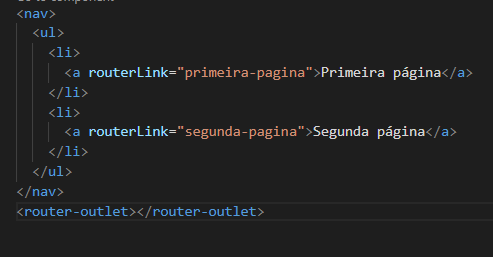
Depois disso apenas colocamos o server para rodar através do comando:

ng serve –open

após isso temos a pagina <http://localhost:4200/> como a pagina app.component.html e a pagina <http://localhost:4200/primeira-pagina> como a pagina do componente que criamos:



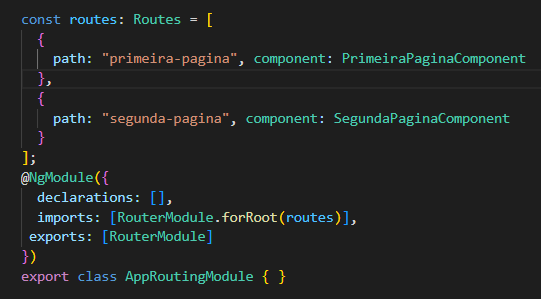
**Navegando entre páginas:**



O atributo routerLink navega entre as páginas roteadas.

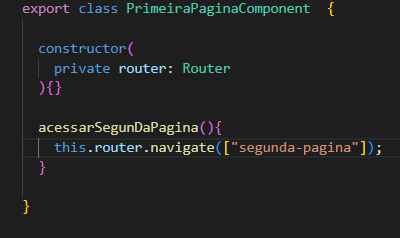
As tags que estiverem anteriores a tag <router-outlet> seram passadas para as páginas subsequentes.

Routing:

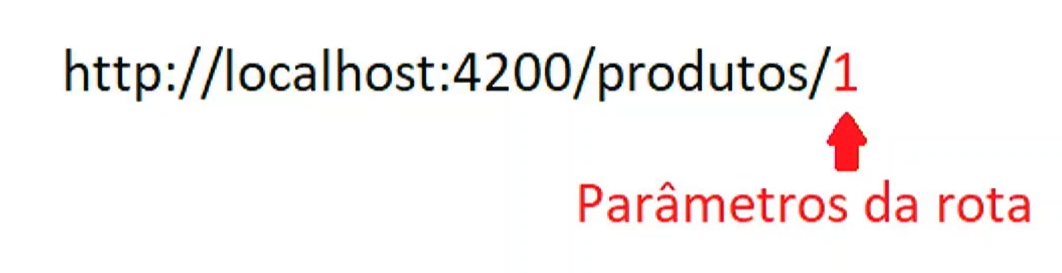


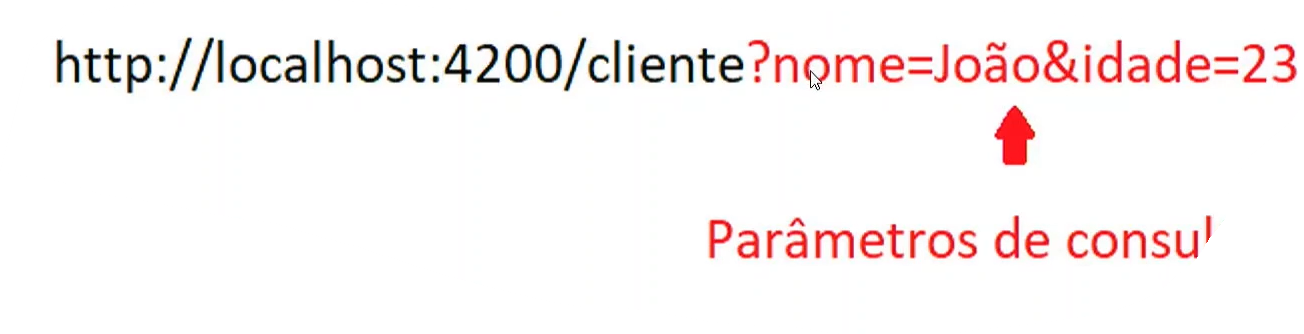
Movendo através de métodos:



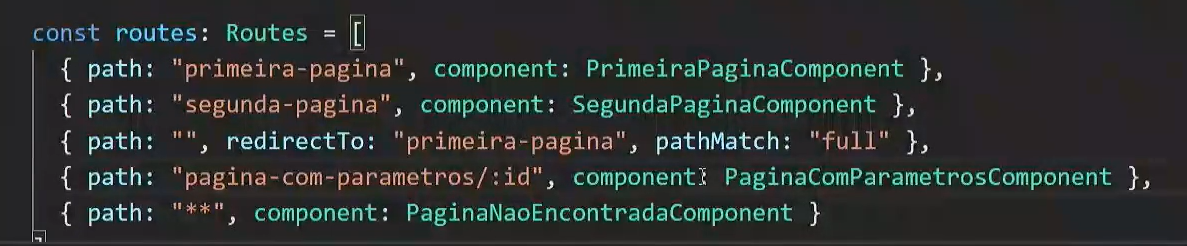


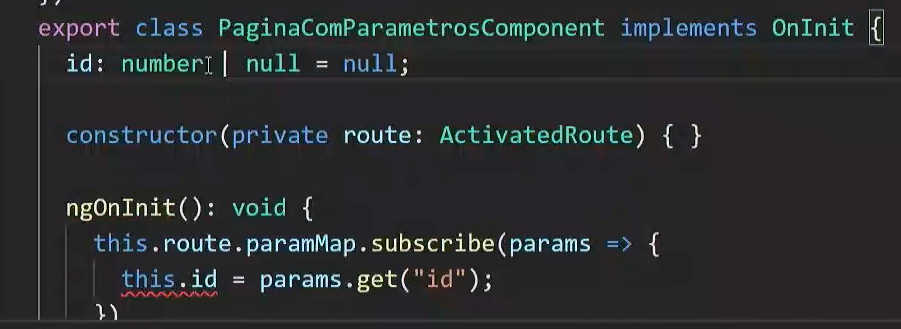
**Parâmetros de URL:**

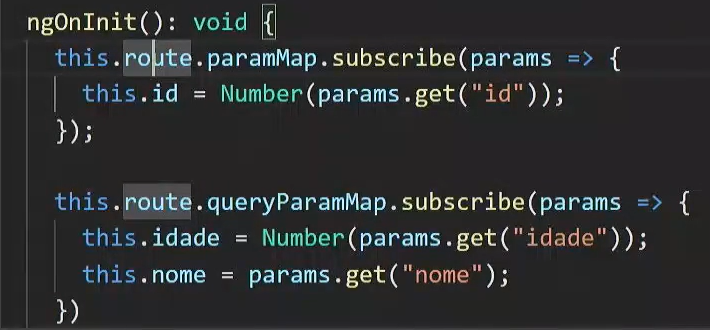


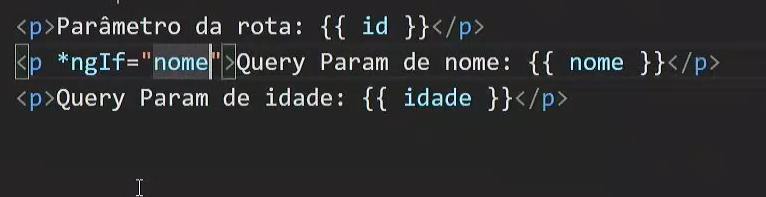


Para trabalharmos com parâmetros de rota devemos criar o path correspondente:



Classe do componente gerado pelo id:  






**Diferença module e componente:**

Um modulo irá conter diversos componentes.

**Lazy Loading:**

Devemos primeiro criar um modulo para o lazy loading:

ng generate module lazy-loading --route=lazy-loading --module=app.module

interessante observer que nesse commando a rota para acessar o module é lazy-loading e ele está atrelado ao modulo principal.

Interessante demonstrar o path gerado nesse comando:



O lazy loading já está funcionando quando acessamos o module app não é carregado o modulo lazy-loading, apenas quando acessamos a url atrelada.

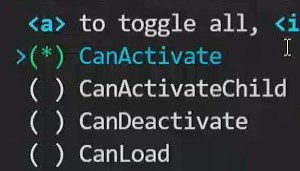
**Route\_guard:**

É uma forma de protegermos nossas páginas.

Devemos criar um arquivo guard:

ng generate guard nomedoarquivo

com isso ira aparecer para selecionarmos o tipo de guard que sera usado:



Usaremos o CanActivate.

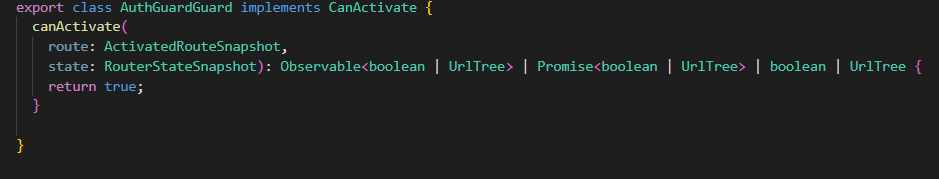
Para usarmos o guard em uma pagina devemos modificar o path do componente:

Exemplo:

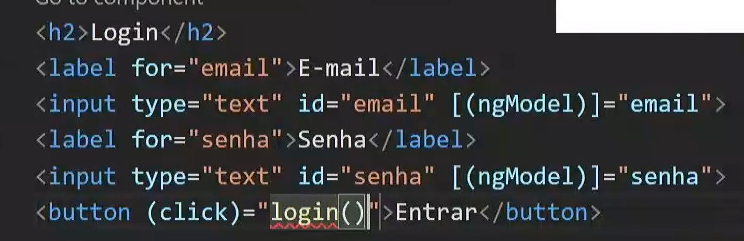


Ou seja, passamos o tipo de guard e depois a classe do guard que vamos usar.

Configuração padrão da classe:

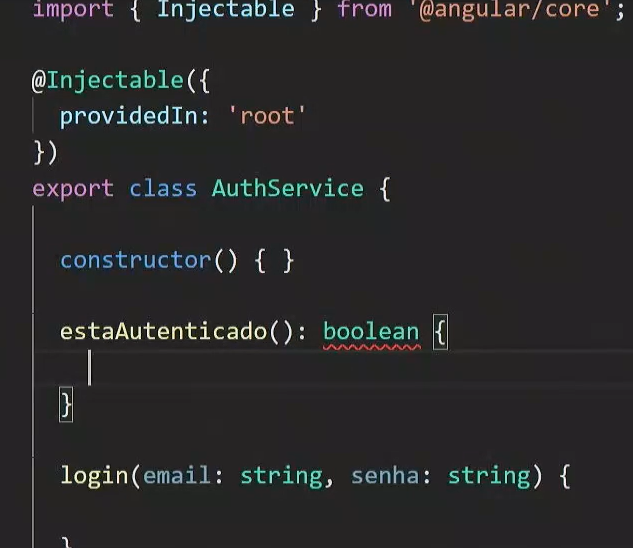


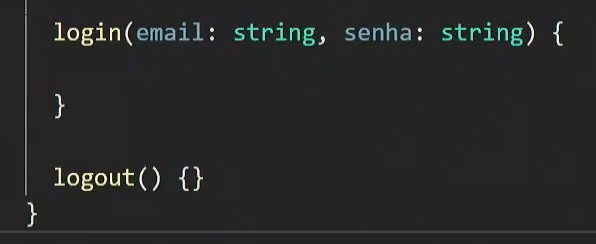
Devemos então criar uma forma de acessar essa página, para isso vamos criar um componente de login:





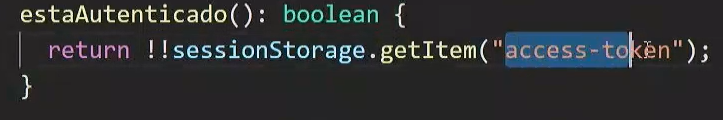
Devemos criar um serviço de autenticação para ser usado no componente de login:

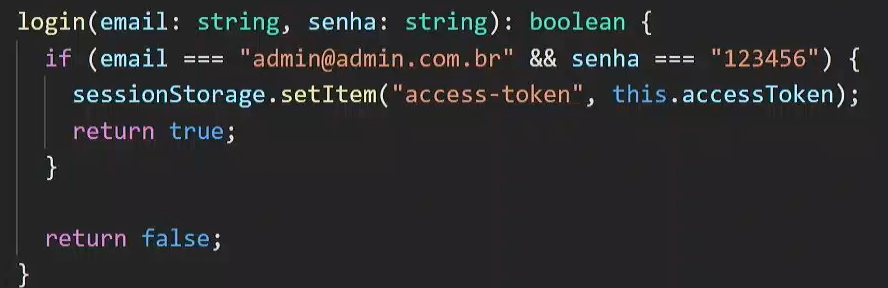




LocalStorage e SessionStorage:

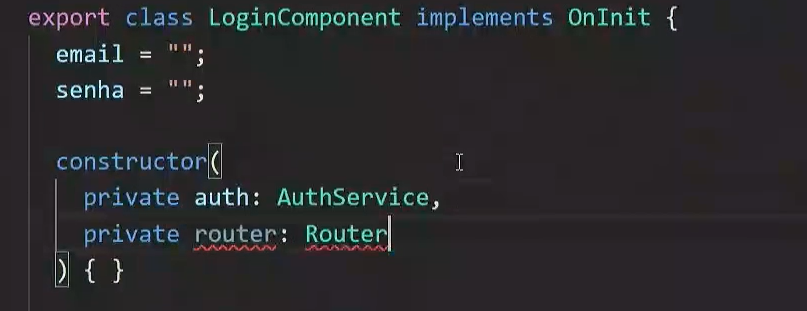
São uma forma de manter informações, o LocalStorage ficara salvo no local, ou seja, mesmo que o usuário feche o navegador ou ate desligue o computador essas informações serão mantidas. Já o SessionStorage, as informações serão perdidas quando a janela do navegador for fechada.

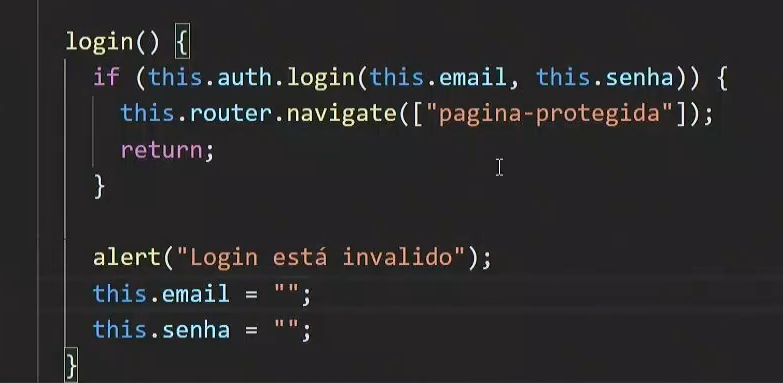




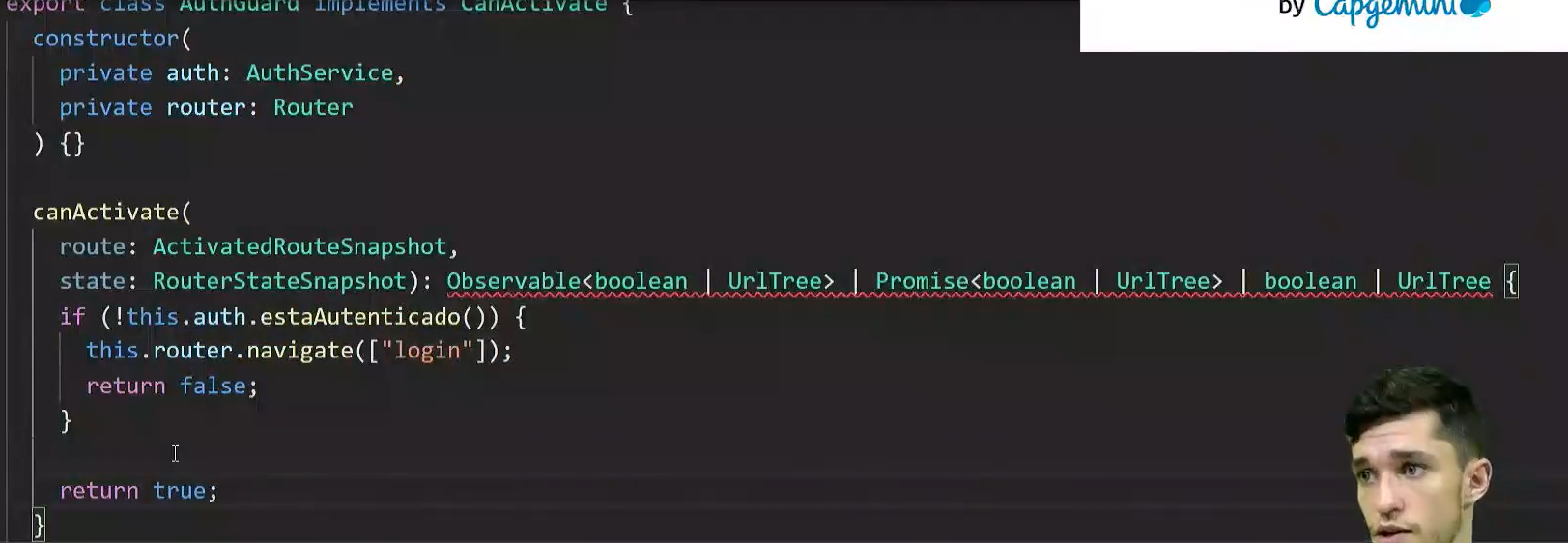


Após isso vamos configurar o componente:





Mesmo com isso nos apenas configuramos a autenticação e o salvamento das informações porem ainda não temos acesso a pagina protegida precisamos configurar o guard:



**Criando a build do projeto:**

Para criar a build do projeto utilizamos o comando:

ng build --base-href=”url que irá ficar” --output-path nomedapasta

exemplo url github pages: <https://usuario.github.io/nomedorepositorio/>

exemplo:

ng build --base-href=”<https://usuario.github.io/nomedorepositorio/>” --output-path docs

**Configurando o git:**

Identidade:

git config --global user.name “Fulano de tal”

git config --global user.email [fulanodetal@example.com](mailto:fulanodetal@example.com)

**Publicando projetos no github pages:**

Após ter criado o build do projeto e configurado o git, vamos seguir os passos:

Comando:

git init -irá inicializar um repositório vazio

git add

git commit -m “build”

git branch -M main

git remote add origin <https://github.com/username/repositorio.git>

git push -u origin main