





Workshop Angular 7

Na aula 1 aprendemos o que é o Angular e como ele trabalha explorando a sua arquitetura e entendendo os seus elementos.

Por fim, criamos nossa aplicação Angular, geramos módulos e componentes, mas como fazemos para acessarmos nossos componentes, seria possível configurar uma rota específica para cada um?

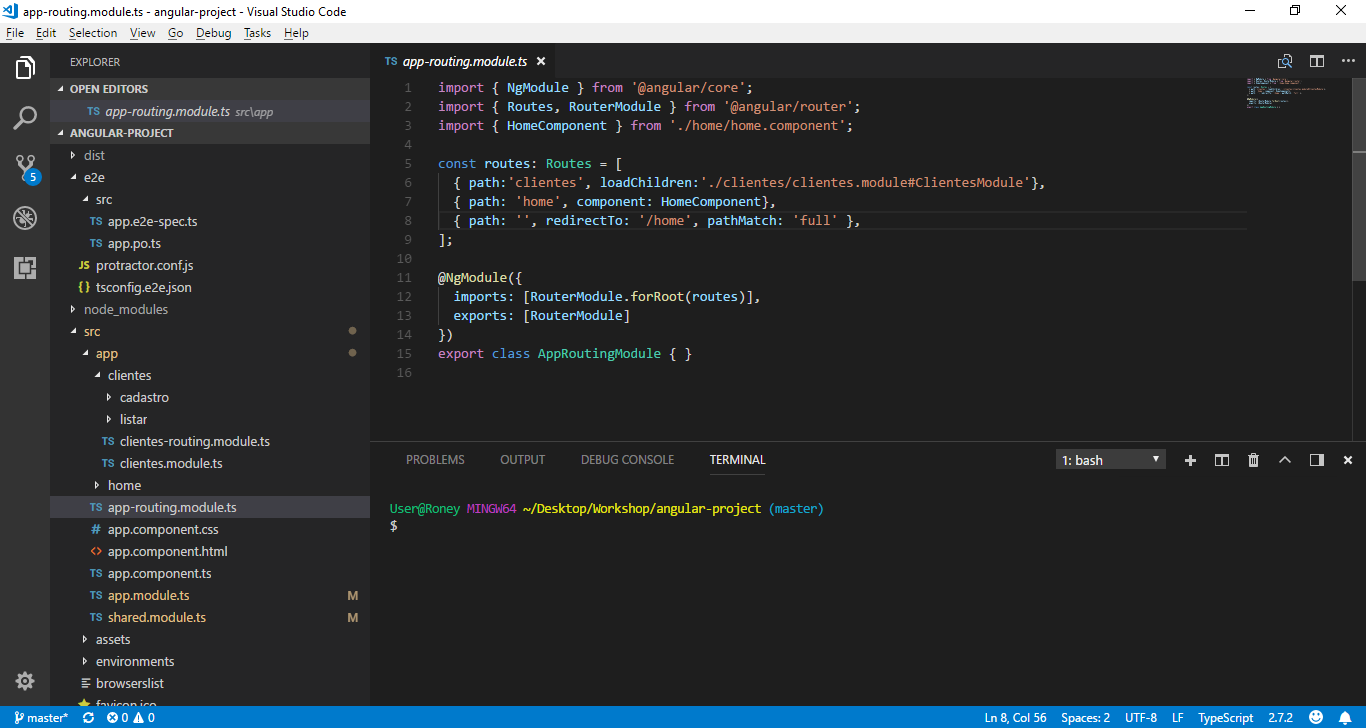
**Criando rotas**

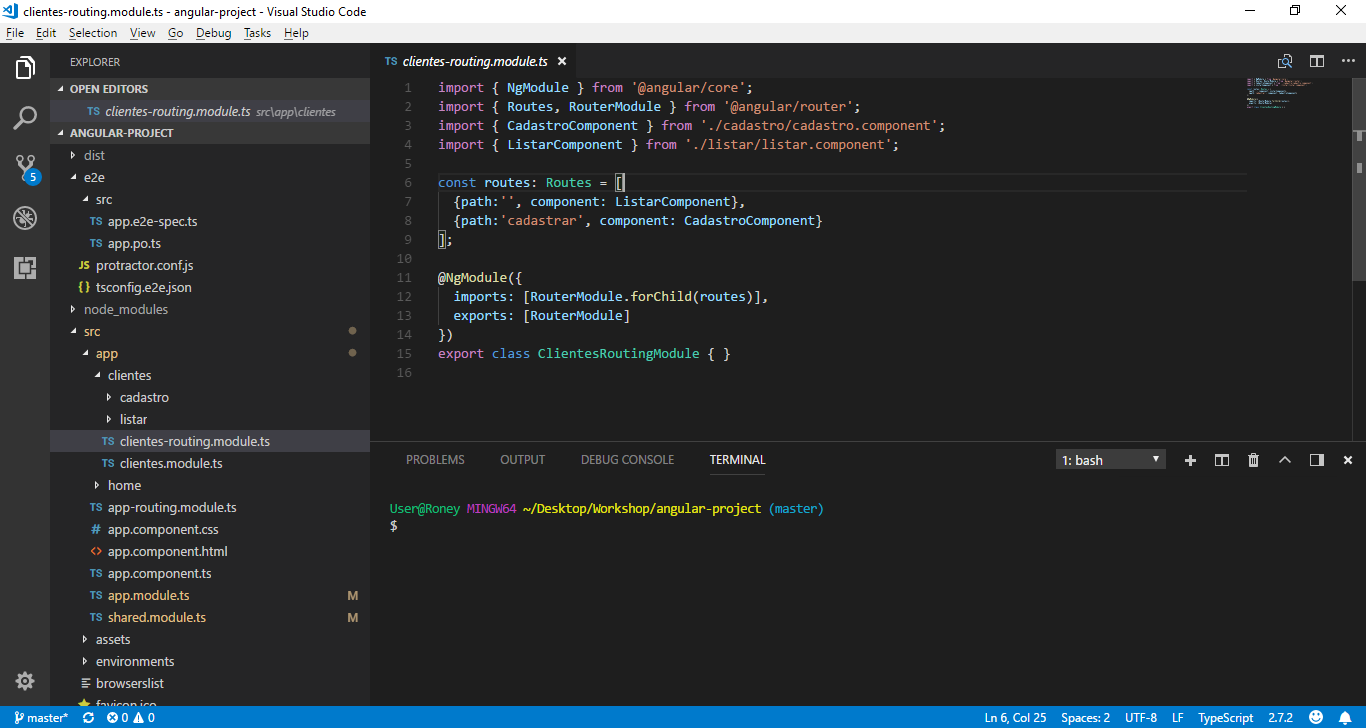
As rotas da aplicação, por padrão são configuradas em modulos denominados como routing.module.ts

Quando criamos nosso projeto, executamos o comando de novo projeto seguido do comando –routing gerando automaticamente o arquivo app.routing.module.ts.

Podemos destacar o modulo app.routing.module.ts como o roteador principal da aplicação sendo ele o responsável por dar as rotas iniciais dos outros módulos da aplicação e os routing.modules de cada um destes módulos são responsáveis por declarar as rotas específicas dos seus componentes.

**Declarando Rotas em app-routing.module.ts**



**Declarando Rotas em clientes-routing.module.ts**

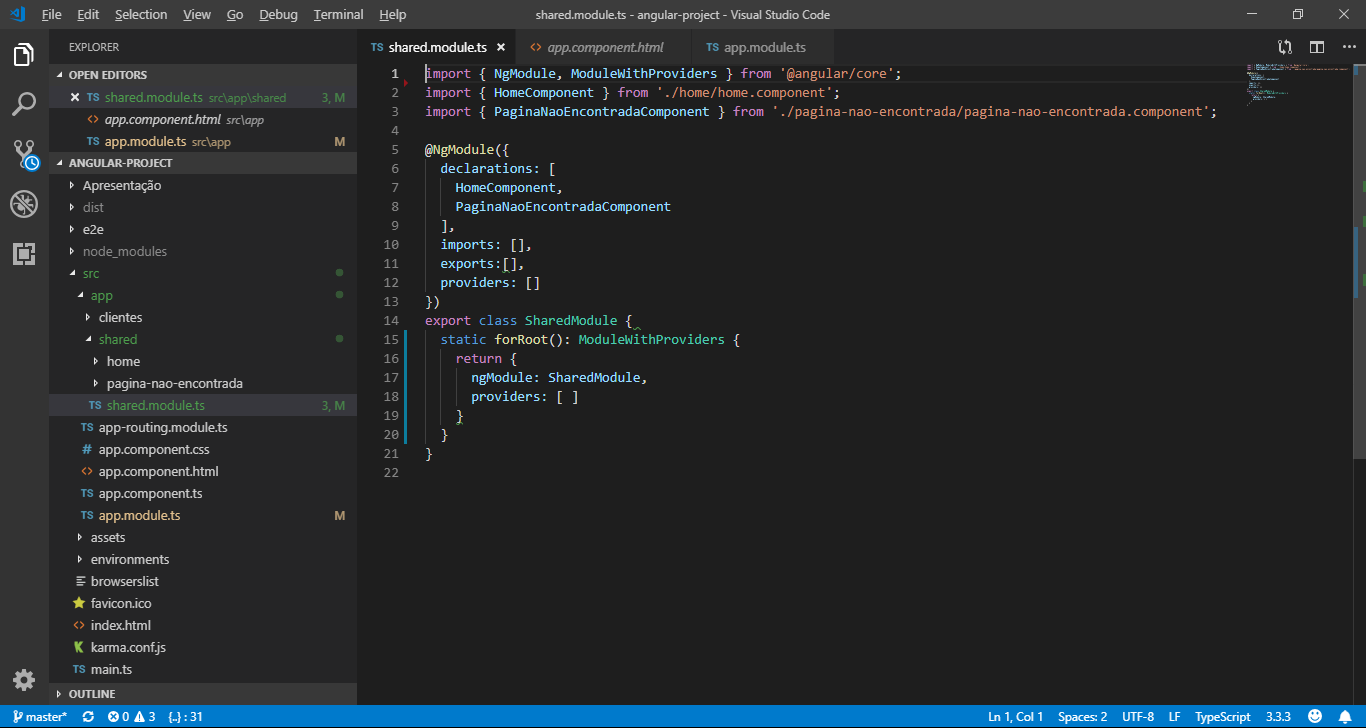
**Trabalhando com Shared.module**

Por questões de organização de código, é comum a prática de criar um Módulo apenas para declarar, importar, prover e exportar componentes, pipes, diretivas, módulos para serem compartilhados para toda a aplicação. Geralmente damos um nome para este módulo: shared.module ou core.module.

**Utilizando o método estático .forRoot()**

Algumas vezes necessitamos compartilhar serviços para que toda a aplicação possa utilizar, mas nem sempre há a necessidade de expor um serviço com múltiplas instancias a cada vez que é solicitado.

Então para resolver um caso que possa se tornar um problema para aplicação posteriormente, declararemos um método estático chamado forRoot() onde incluiremos todos nossos serviços que devem ser Singletons para aplicação.



Com o nosso novo modulo de componentes compartilhados, em nosso app.module podemos importar nosso shared.module chamando diretamente o método forRoot().

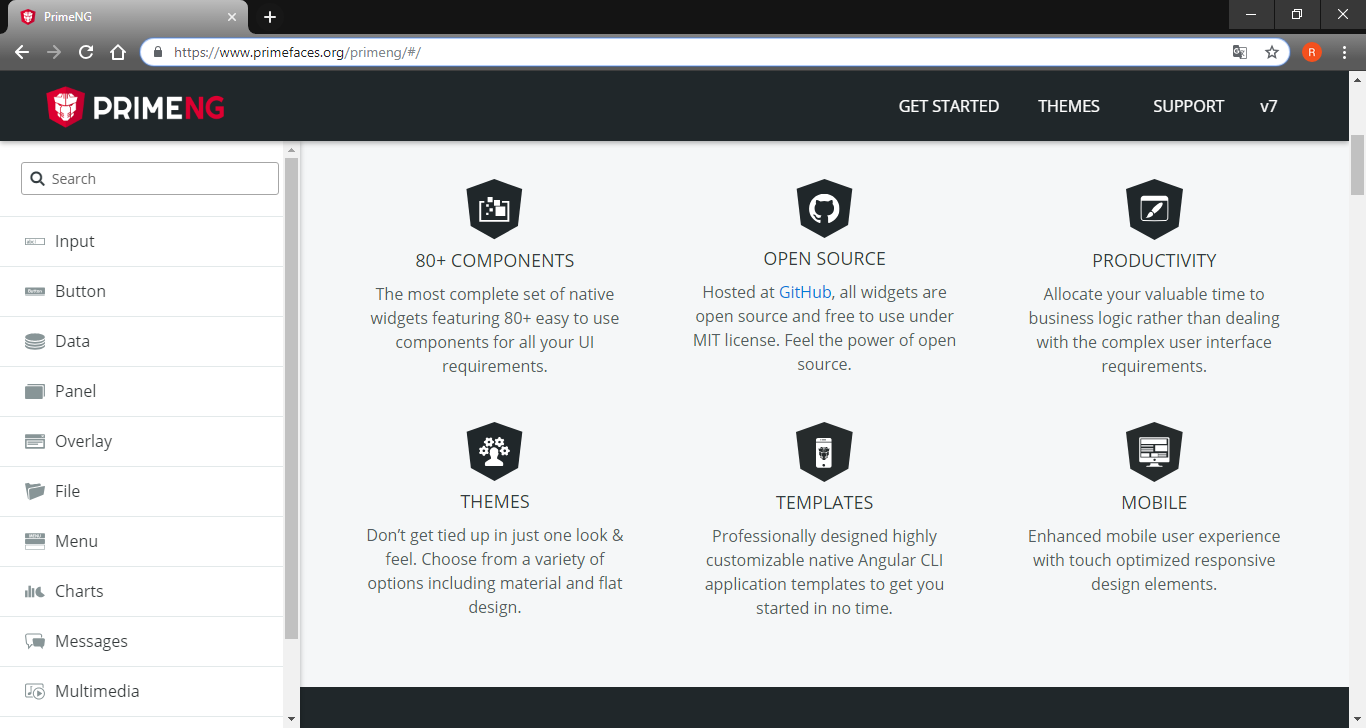
**Exercício 2**

**Realizando um CRUD**

**Exercício 3**

Explorando o PrimeNG

PrimeNG é uma coleção de componentes de UI ricos para Angular. Todos os widgets são de código aberto e gratuitos para uso sob a licença MIT. O PrimeNG é desenvolvido pela [PrimeTek Informatics](http://www.primetek.com.tr/) , um fornecedor com anos de experiência no desenvolvimento de soluções de UI de código aberto.



Importando o PrimeNg em nosso Projeto

**Primeiro passo**

Primeiramente, é necessário baixar as dependências do PrimeNG onde as mesmas estão disponíveis no npm.

Para baixa-las basta executar os comandos abaixo:

npm install primeng font-awesome

npm install primeicons

npm install @angular/animations

npm install @angular/cdk

**Segundo passo**

Importarno as dependências do PrimeNG no package json e declarar nos assets os styles do PrimeNG.

Se caso em nosso package.json não ter sido gravadas as declarações das dependências do PrimeNG, devemos adiciona-las dentro de **dependencies**:

"dependencies": {

//...

"font-awesome": "^4.7.0",

"primeng": "^7.0.0",

"primeicons": "^1.0.0"

},

**Terceiro passo**

Para nossa aplicação reconhecer os estilos do PrimeNG, devemos adicionar os seguintes CSSs dentro de Styles do arquivo agular.json.

"styles": [

"src/styles.css",

"node\_modules/font-awesome/css/font-awesome.min.css",

"node\_modules/primeng/resources/primeng.min.css",

"node\_modules/primeicons/primeicons.css",

"node\_modules/primeng/resources/themes/bootstrap/theme.css"

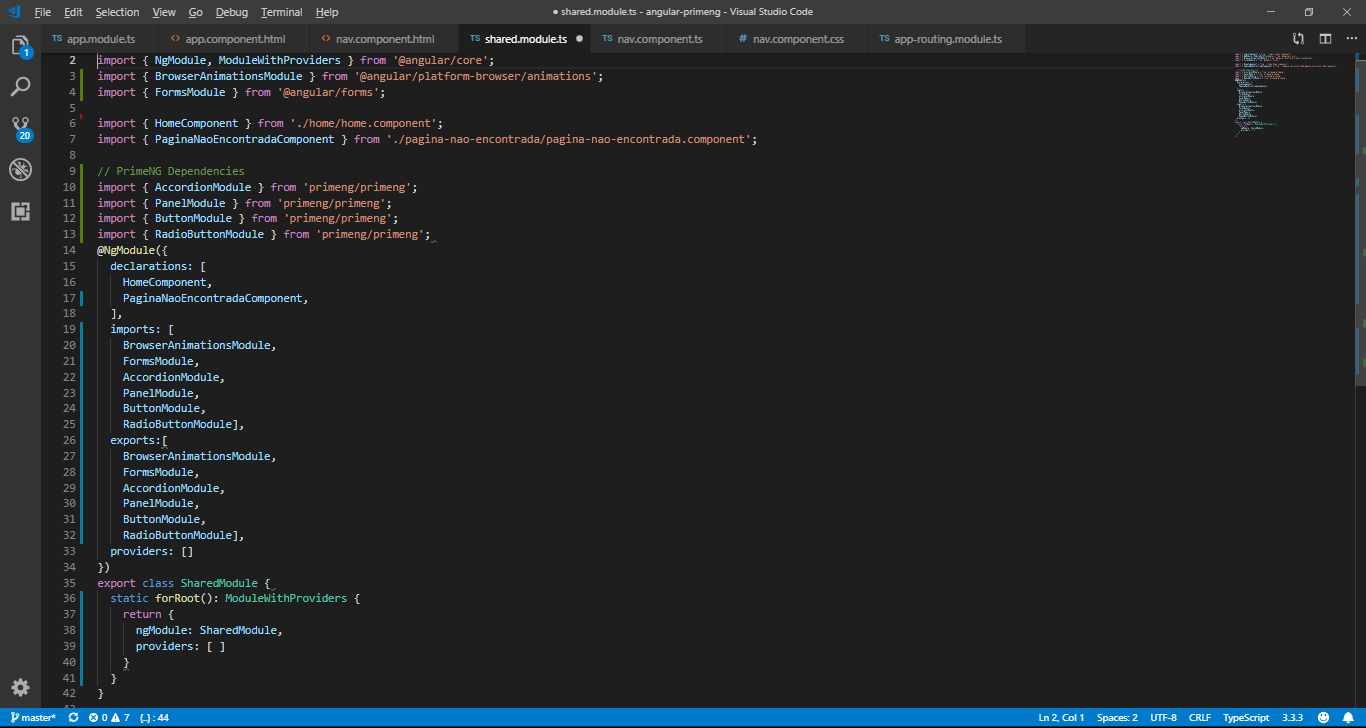
],

**Quarto passo**

A dependência animations é utilizada em diversos componentes do framework, logo ao realizar o download da biblioteca, o modulo **BrowserAnimationsModule**

do Animations deve ser importado em nossa aplicação.

Assim como no animation, podemos importar mais alguns módulos que utilizaremos durante o desenvolvimento.



É válido lembrar que os itens que são importados como módulos, pipes, components, etc, devem ser exportados se forem utilizados de forma compartilhada.

**Exercício PrimeNg**

**Criando um menu, tela de login e controle de exibição de menu**

**Instalar o bootstrap e o jquery**

npm install bootstrap@4 jquery --save

**Inserir imports em nav.component.css**

@import "~bootstrap/dist/css/bootstrap.min.css";

**Inserir imports em nav.component.html**

<script src="../../../../node\_modules/bootstrap/dist/js/bootstrap.min.js"></script>

<script src="../../../../node\_modules/jquery/dist/jquery.min.js"></script>

Dentro da pasta shared, criaremos o componente nav e o componente login

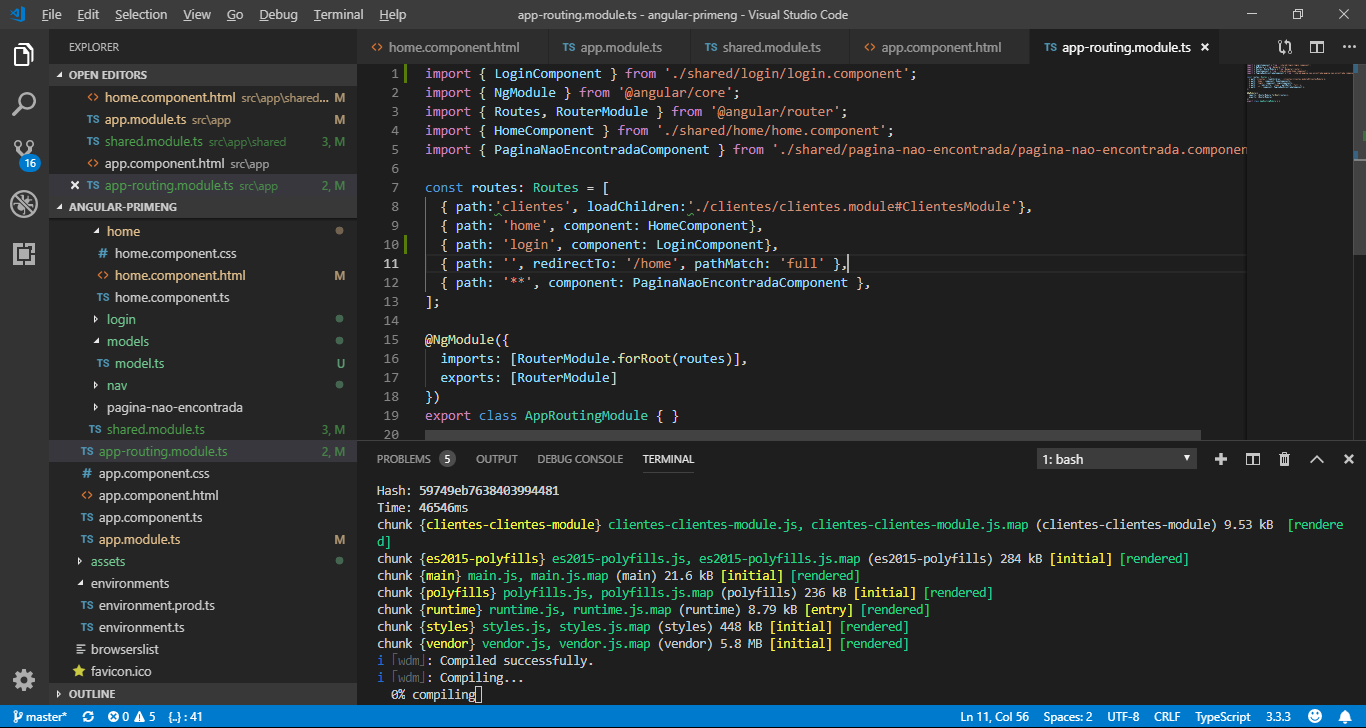
**ng g c shared/nav**

**ng g c shared/login**

**Em app.component.html, devemos incluir nosso componente de menu.**

<app-nav></app-nav>

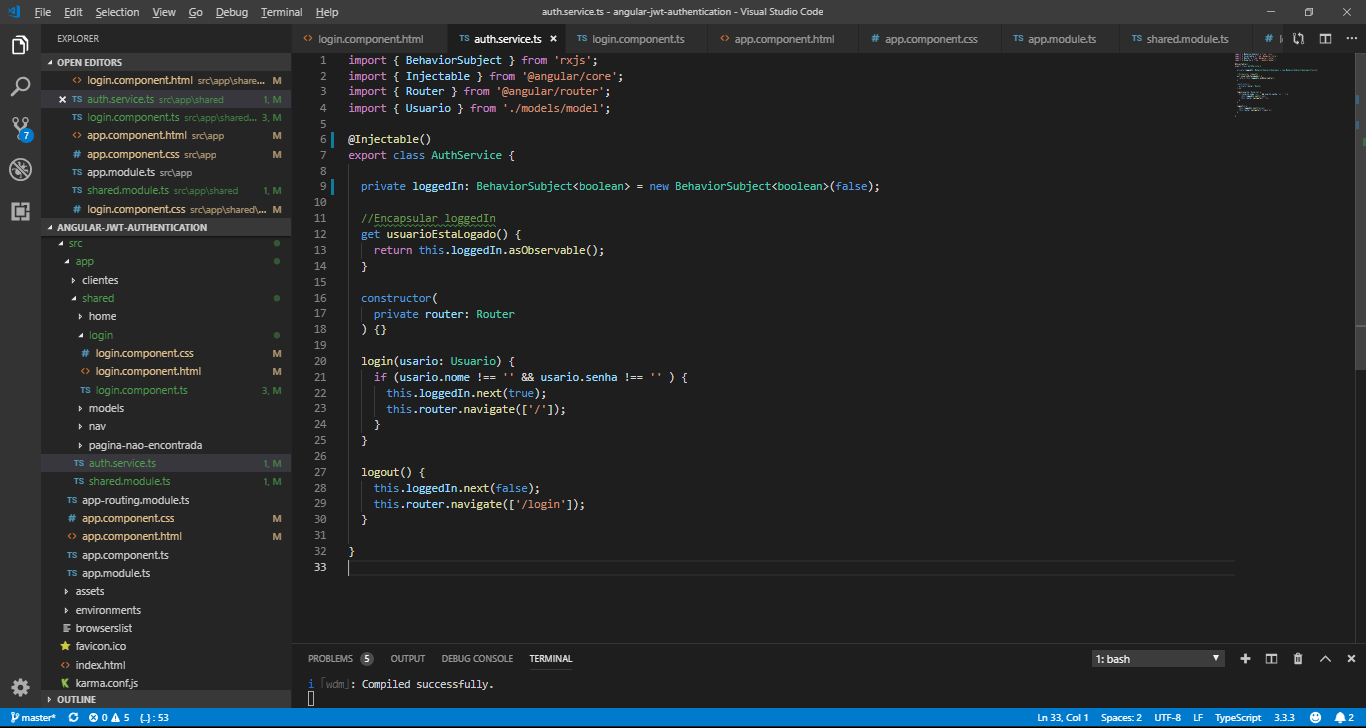
Em app-routing.module, devemos adicionar uma rota específica para nosso componente login



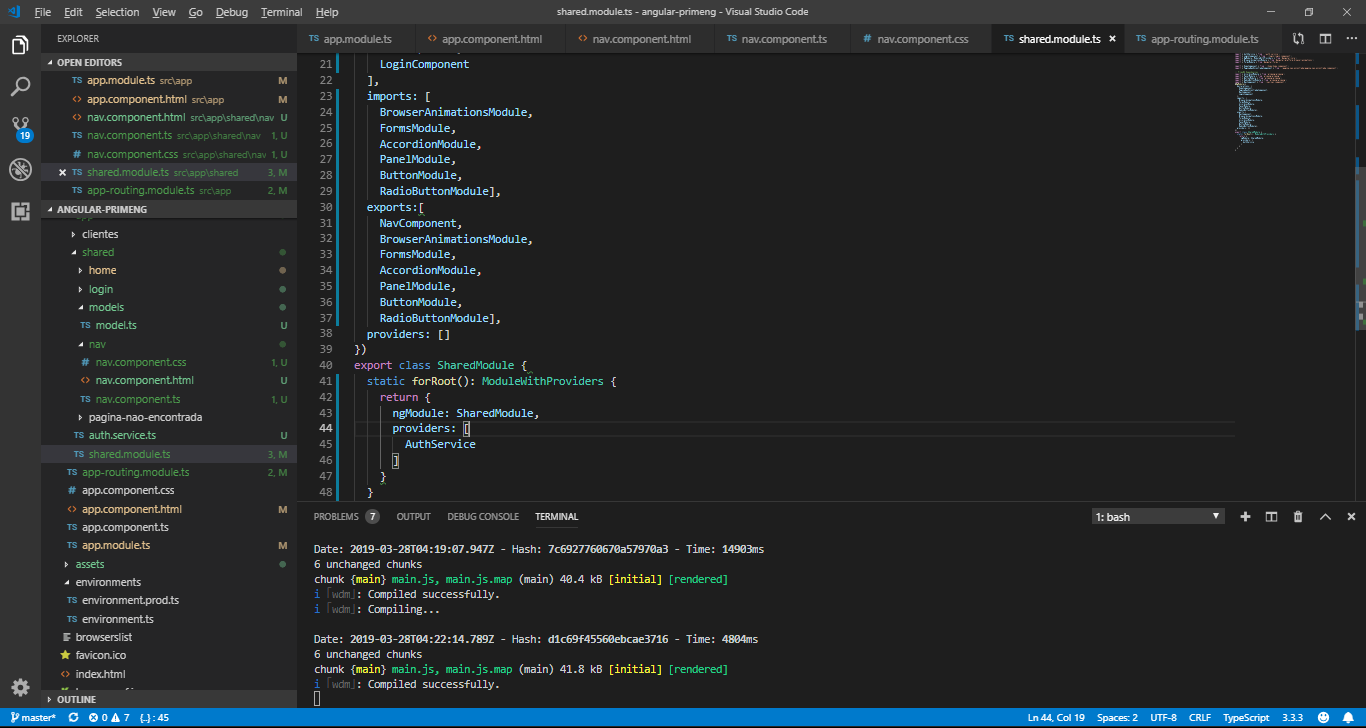
Após criar o menu e a tela de login percebemos que o menu independente da rota sempre irá aparecer. Não é muito amigável exibir um menu na tela de login.

Criar o serviço de autenticação auth.service

Metodo de login e de logout e Variável de usuário logado.

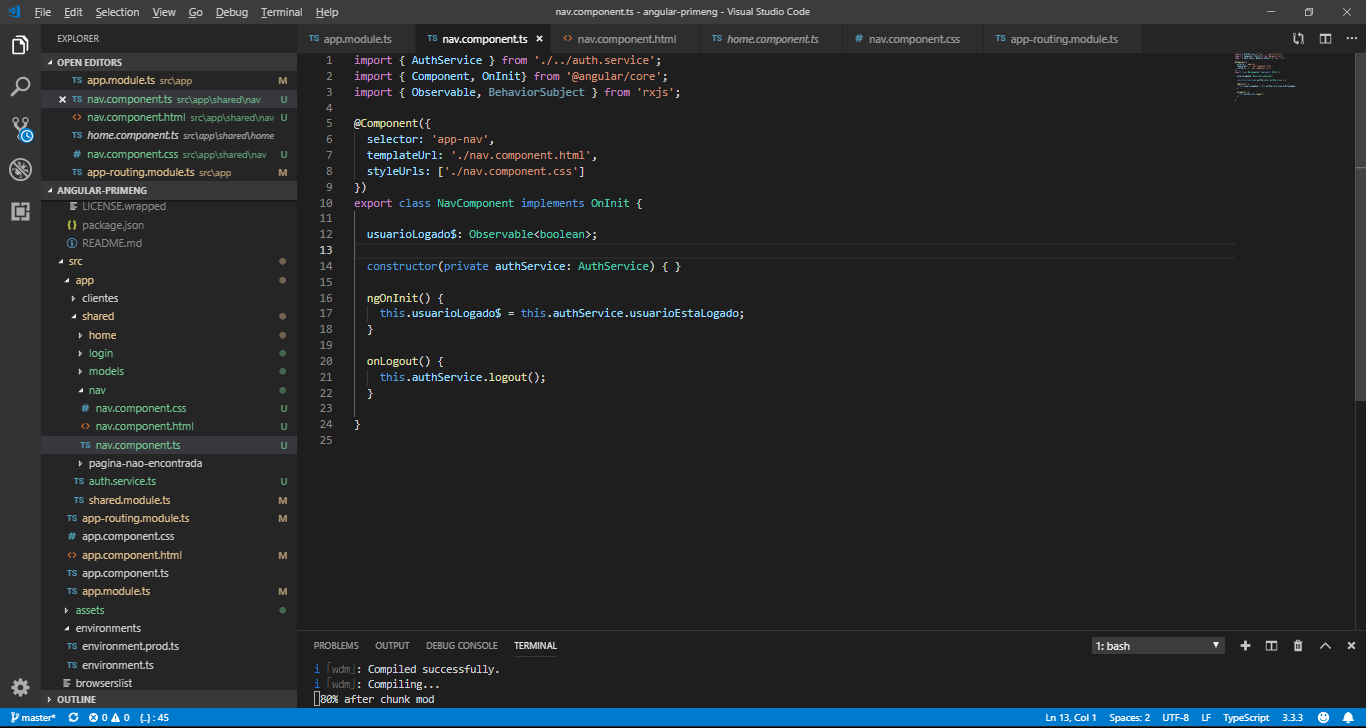


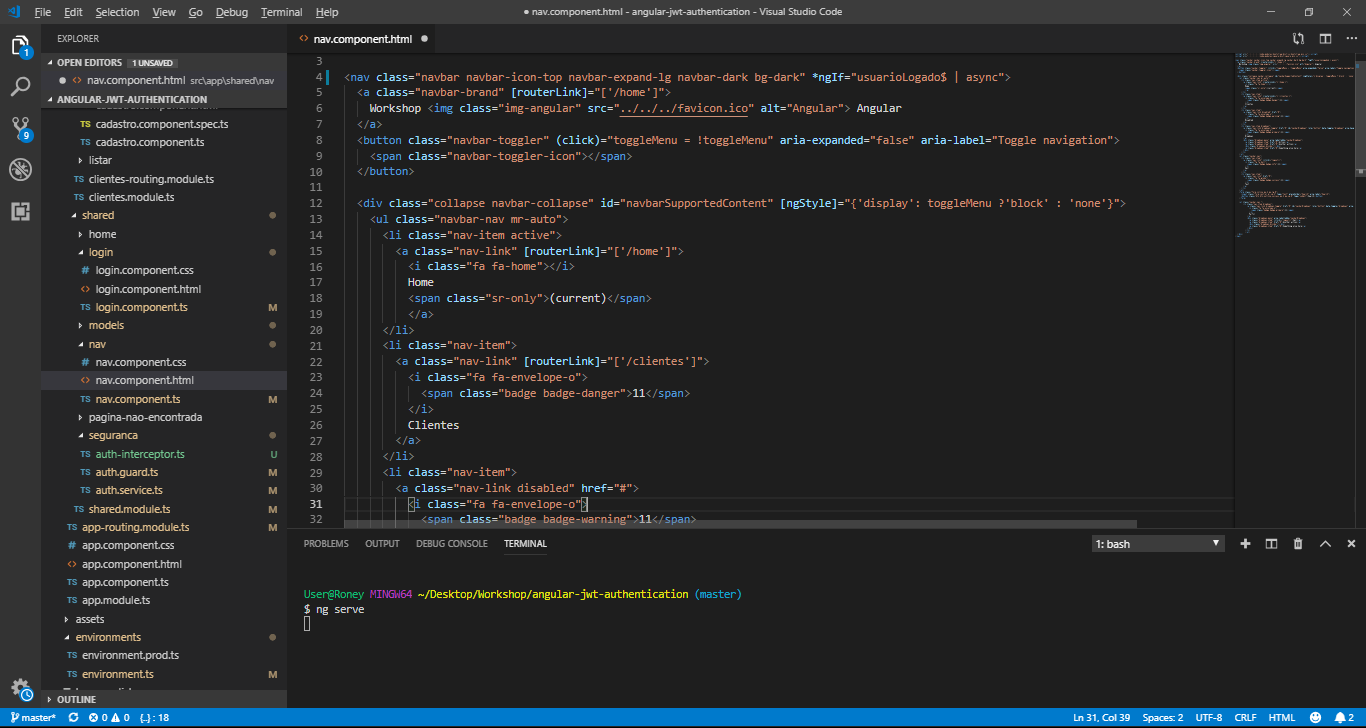
Prover o auth.service com o SharedModule.forRoot()



Em NavComponent

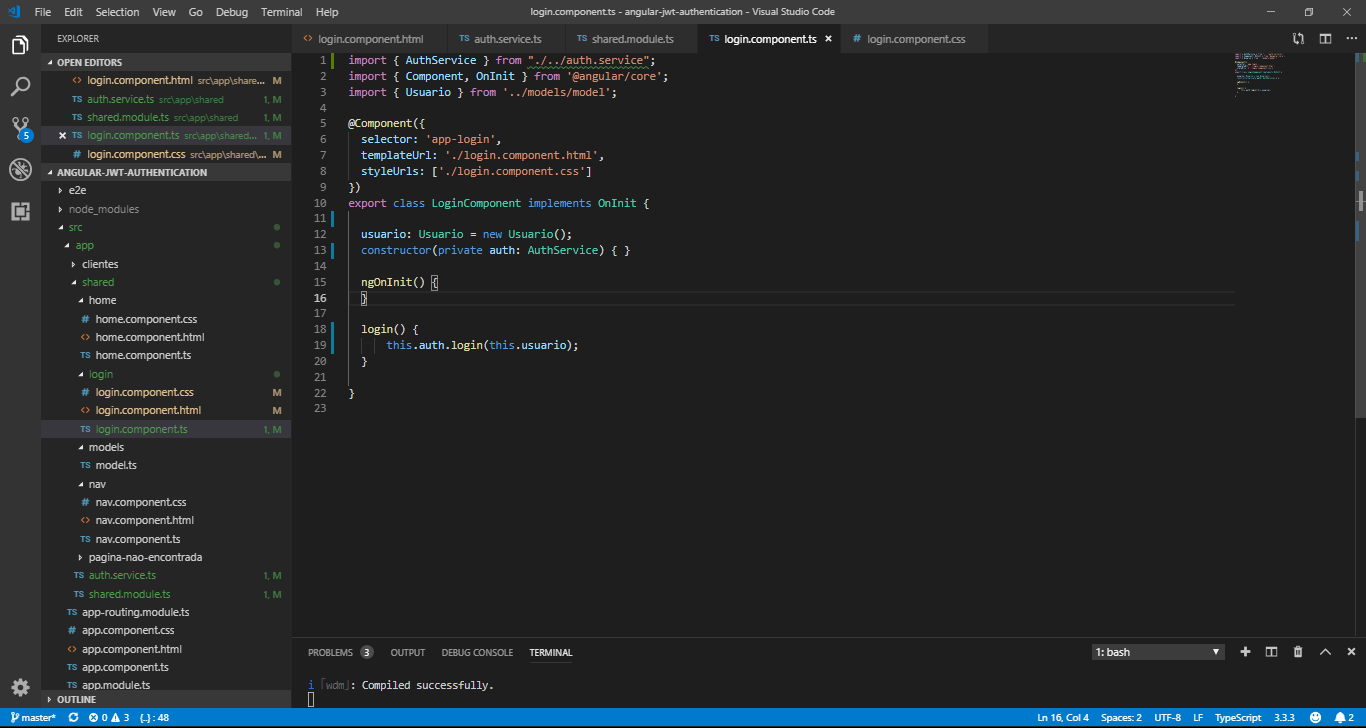
Controlar a exibição do menu com a variável do usuário logado





Login Component

Criar método de login eu solicitará a autenticação que fará a alteração do status do usuário logado.



Componente de Mensagem

Para um componente de mensagem, podemos utilizar o componente Toast do primeng.

Implementar com o ngx-toatr

1. Importar TostModule e compartilha-lo
2. Prover MessageService
3. Incluir a tag <p-toast [style]="{marginTop: '80px'}"></p-toast> no app.component.html