# C:\Users\User\Desktop\Workshop\angular-project\Apresentação\Imagens\angular2.png

**WORKSHOP ANGULAR 7**

**WORKSHOP ANGULAR 7**

No módulo 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 e configuramos as rotas de cada componente da aplicação.

# RXJS - OBSERVABLES E PROMISES

Ambos são encapsuladores e emissores de dados e quem os assinam podem manipular métodos internos de cada um.

## Algumas Diferenças

|  |  |
| --- | --- |
| **Observables** | **Promises** |
| Emite mais de um valor. | Emite um único valor. |
| Evento só é acionado quando há um assinante (Lazy). | Evento é acionado independente se existe um assinante (Eager). |
| Cada subscribe é independente de outro. | Cada then é compartilhado como outro. |

## Maneiras de definir um atributo:

|  |  |  |
| --- | --- | --- |
|  | **SYNCHRONOUS** | **ASYNCHRONOUS** |
| **MÚLTIPLOS** | Array | Observable |
| **ÚNICOS** | Objeto | Promise |

## Observação:

Métodos e serviços baseados em Observables e Promises são asynchronos e para resolver qualquer situação de forma synchrona, podemos utilizar os assessores async e await nos métodos.

# TRABALHANDO COM SHARED.MODULE

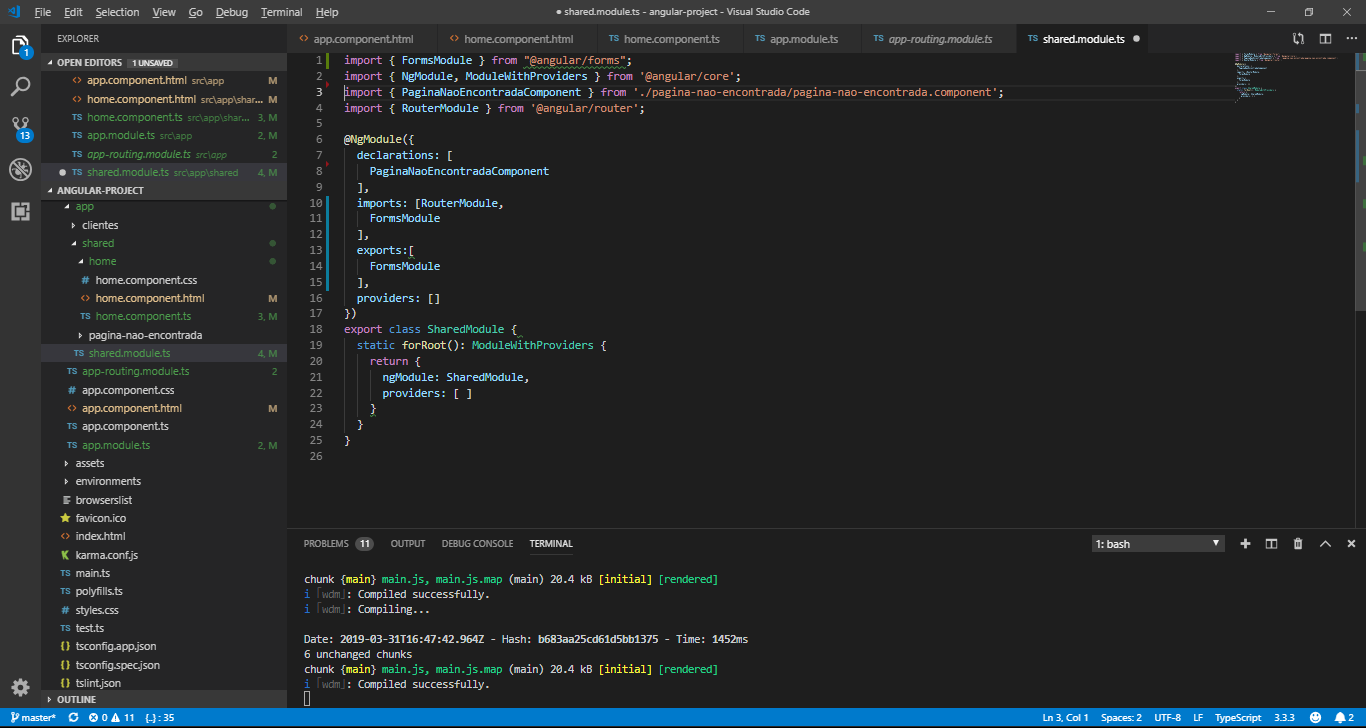
Por questões de organi­zação de código, é comum a prática de criar um Módulo apenas para declarar, importar, prover e exportar componentes, pipes, diretivas, módulos para serem compartilhados para toda a aplicação. Geralmente damos um nome para este módulo: shared.module ou core.module.

## Utilizando o método estático .forRoot()

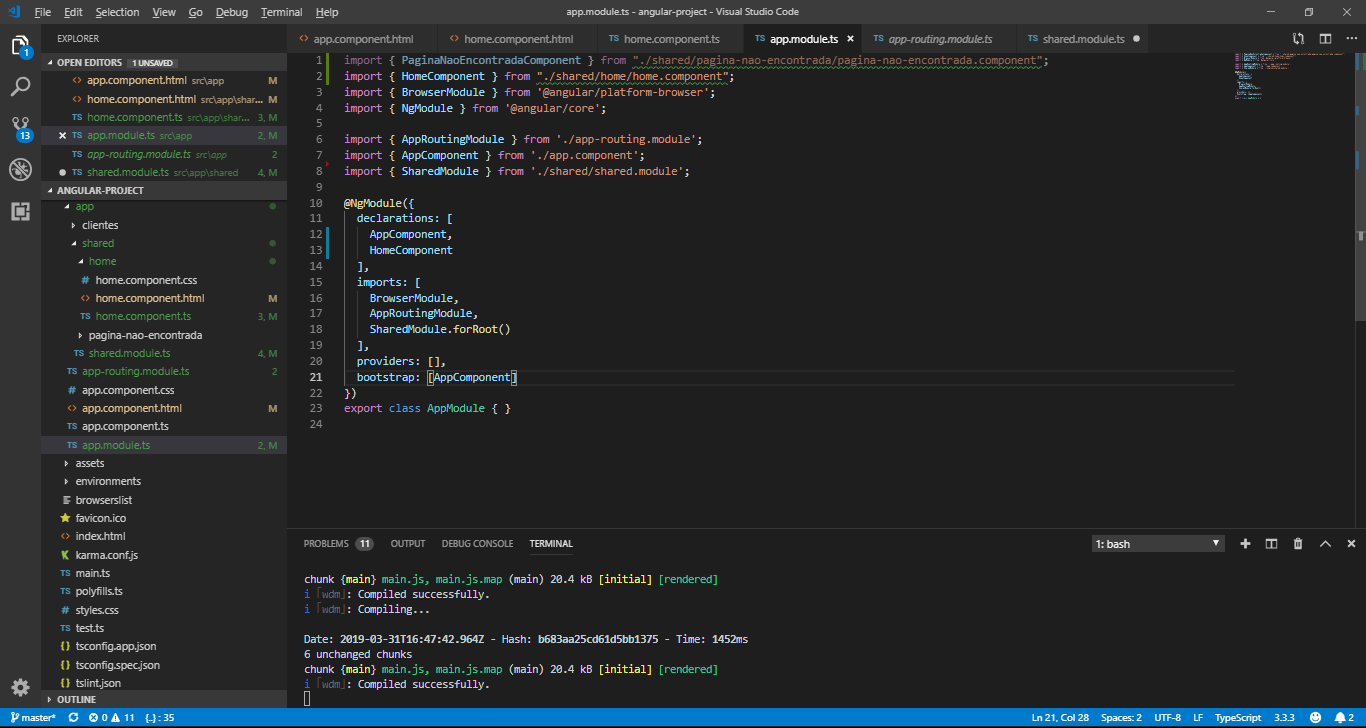
Algumas vezes necessitamos compartilhar serviços para que toda a aplicação possa utilizar, mas nem sempre há a necessidade de expor um serviço com múltiplas instancias a cada vez que é solicitado.

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.

## SharedModule



## AppModule

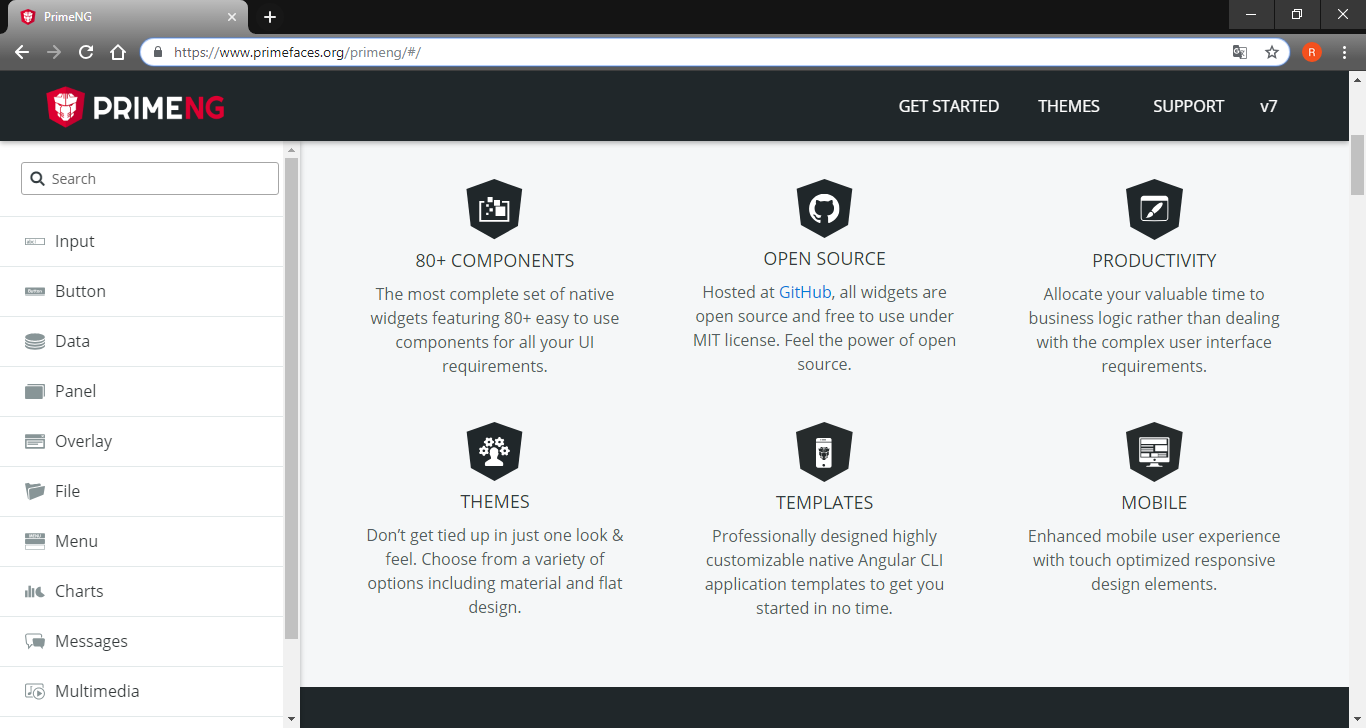
Com o nosso novo modulo de componentes compartilhados, em nosso app.module podemos importar nosso shared.module chamando diretamente o método forRoot().

# REALIZANDO UM CRUD

## Realizar Exercício 3 da lista de exercícios

# 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

## 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",

"bootstrap": "^4.3.1",

"jquery": "^3.3.1",

"primeicons": "^1.0.0",

"primeng": "^7.1.0"

}

## 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"

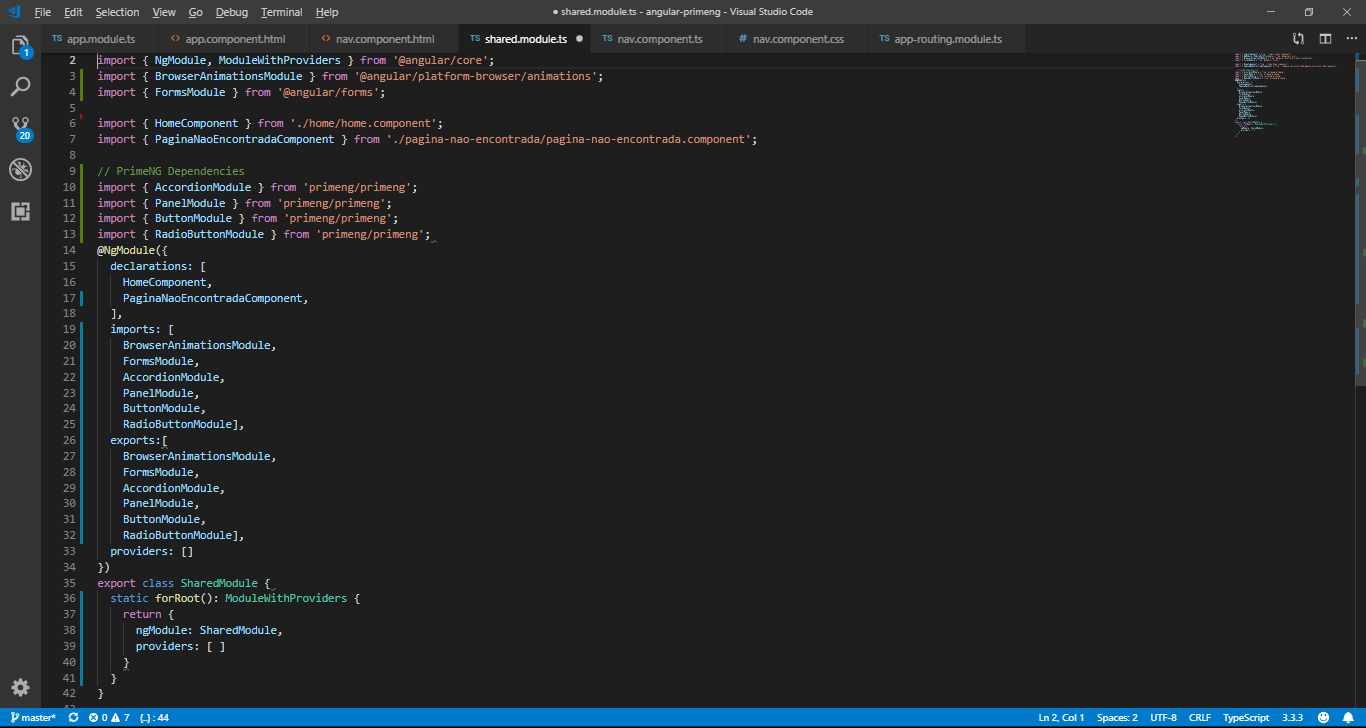
],

## 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.

# CRIANDO UM MENU, TELA DE LOGIN E CONTROLE DE EXIBIÇÃO DE MENU

Dentro da pasta shared, criaremos o componente nav e o componente login para trabalharmos com o sistema de Login mocado e o controle de exibição do menu ao realizar o login. Para criar os componentes basta executar os comandos:

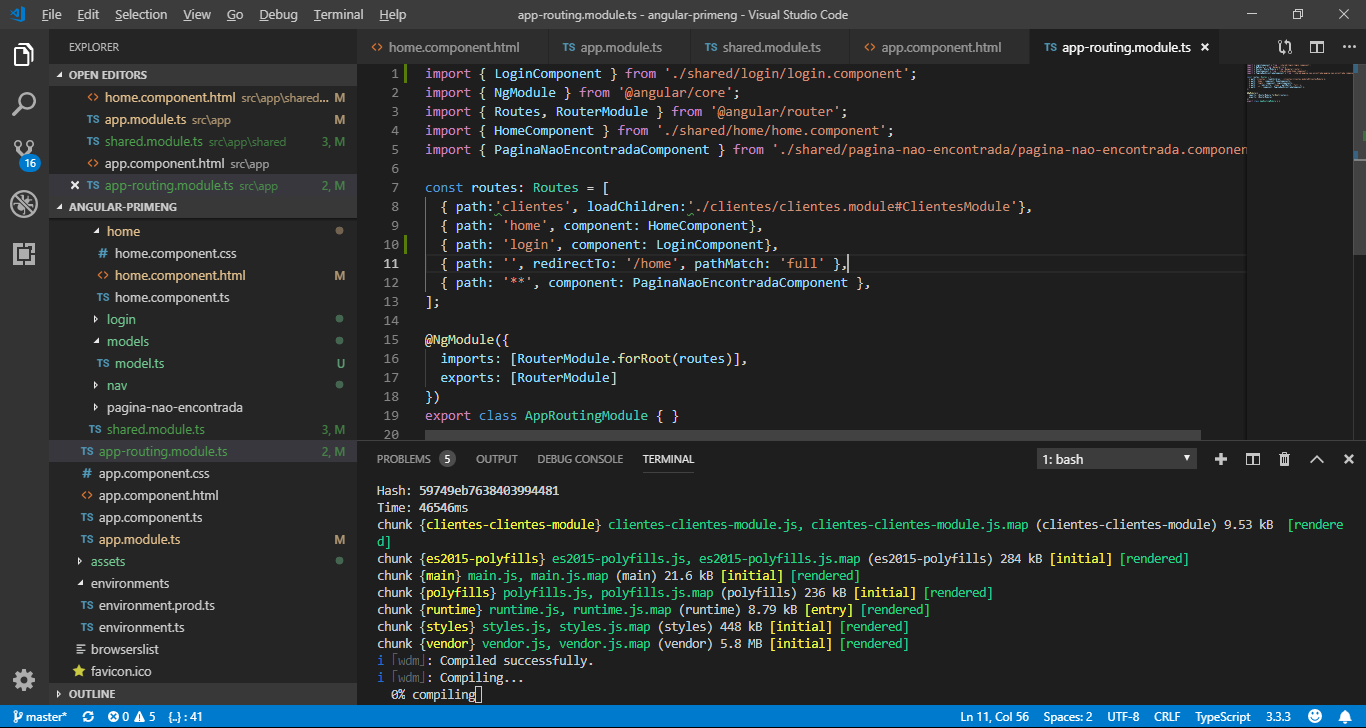
**ng g c shared/nav**

**ng g c shared/login**

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

<app-nav></app-nav>

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



## Configurando Style com Bootstrap e JQuery

Mesmo utilizando um framework como PrimeNg é possível mesclar com bibliotecas de CSS e JS sem nenhum problema de compatibilidade.

### Instalando:

Bootstrap e oJjquery

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>

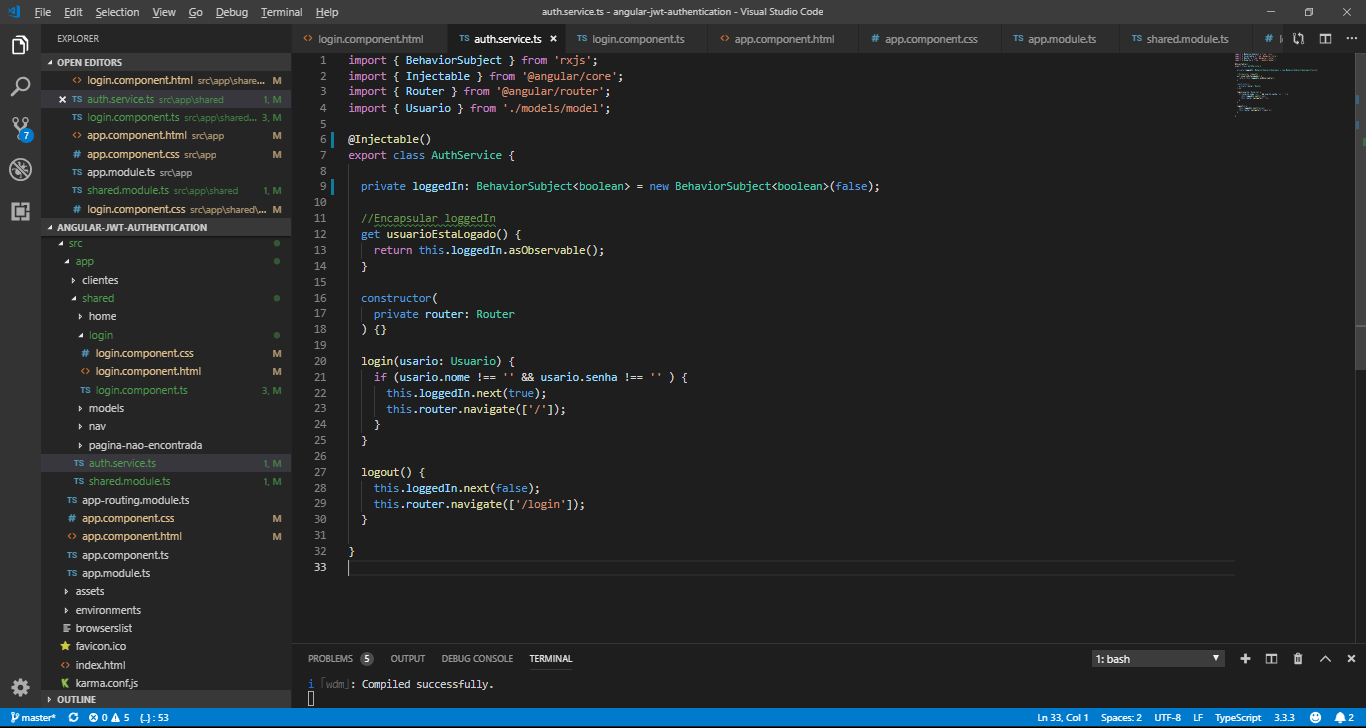
<script src="../../../../node\_modules/jquery/dist/jquery.min.js"></script>

## Implementando o sistema de Login

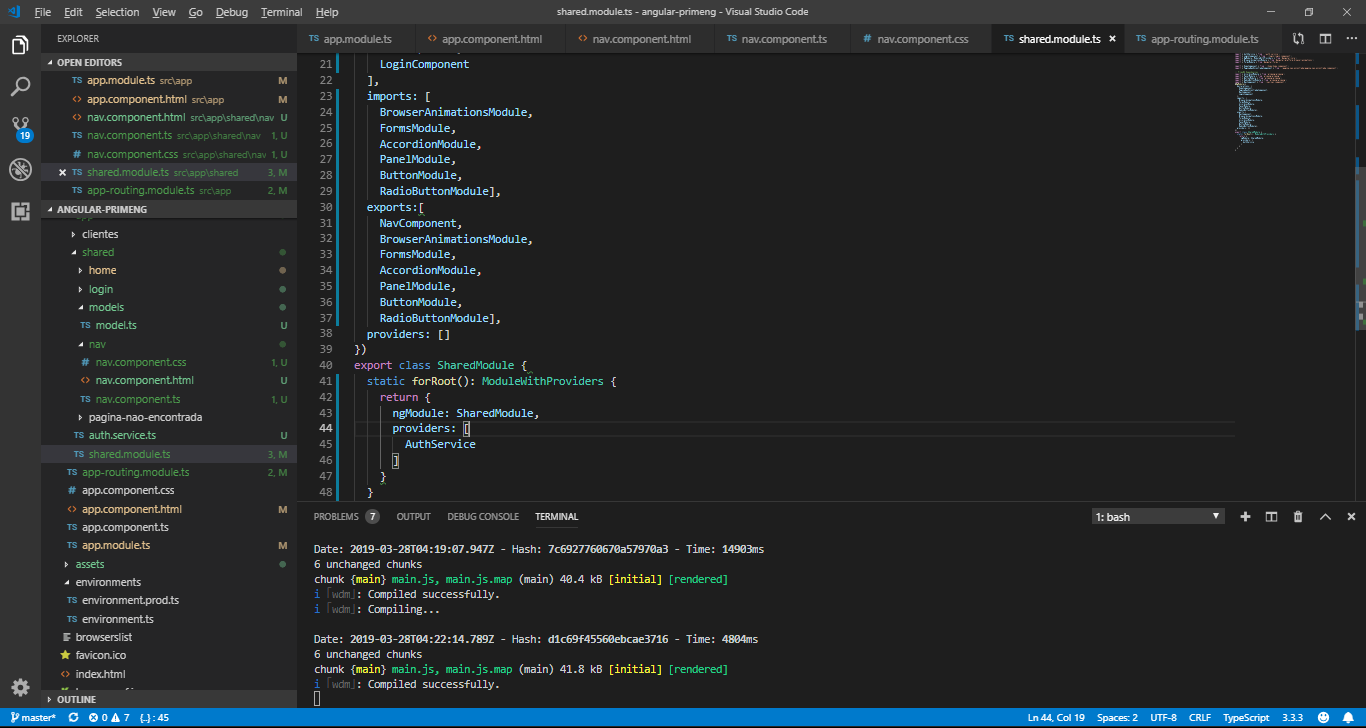
Após criar o menu e a tela de login percebemos que o menu independente da rota sempre irá aparecer. Não é muito amigável exibir um menu na tela de login.

Criar o serviço de autenticação auth.service

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

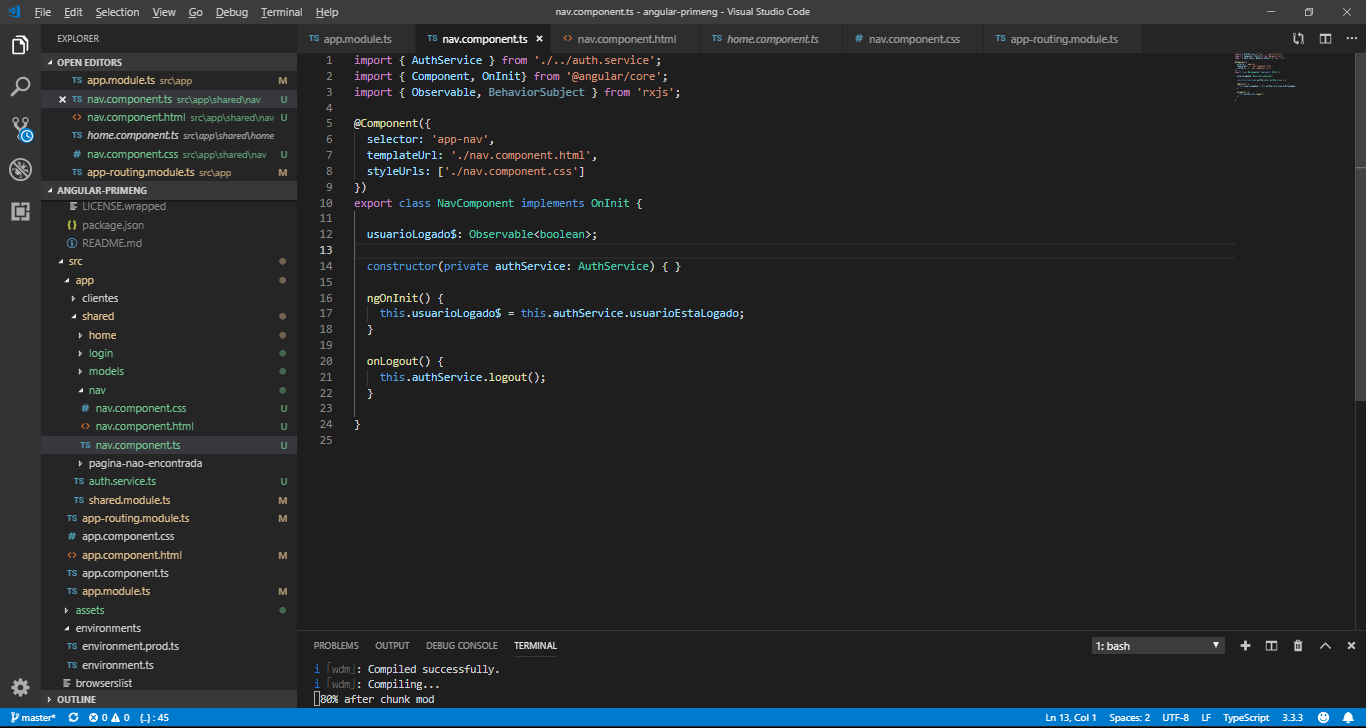


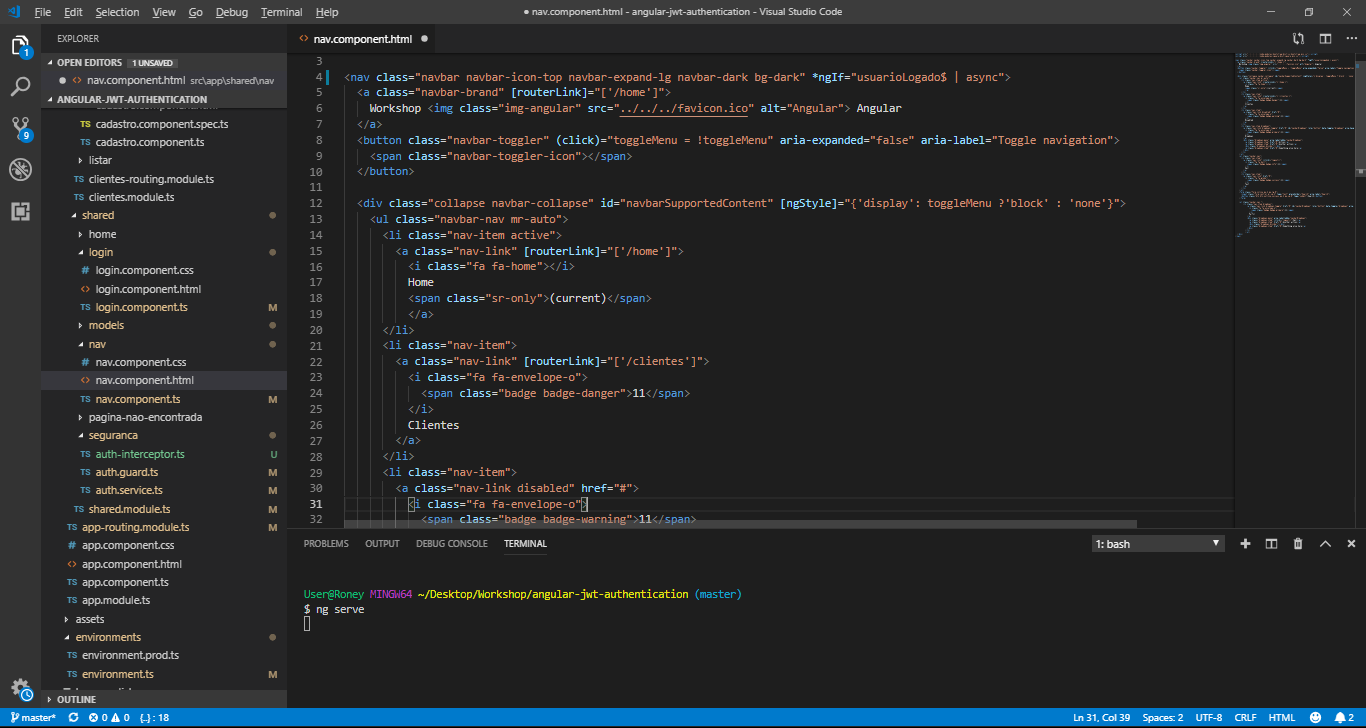
Prover o auth.service com o SharedModule.forRoot()



Em NavComponent

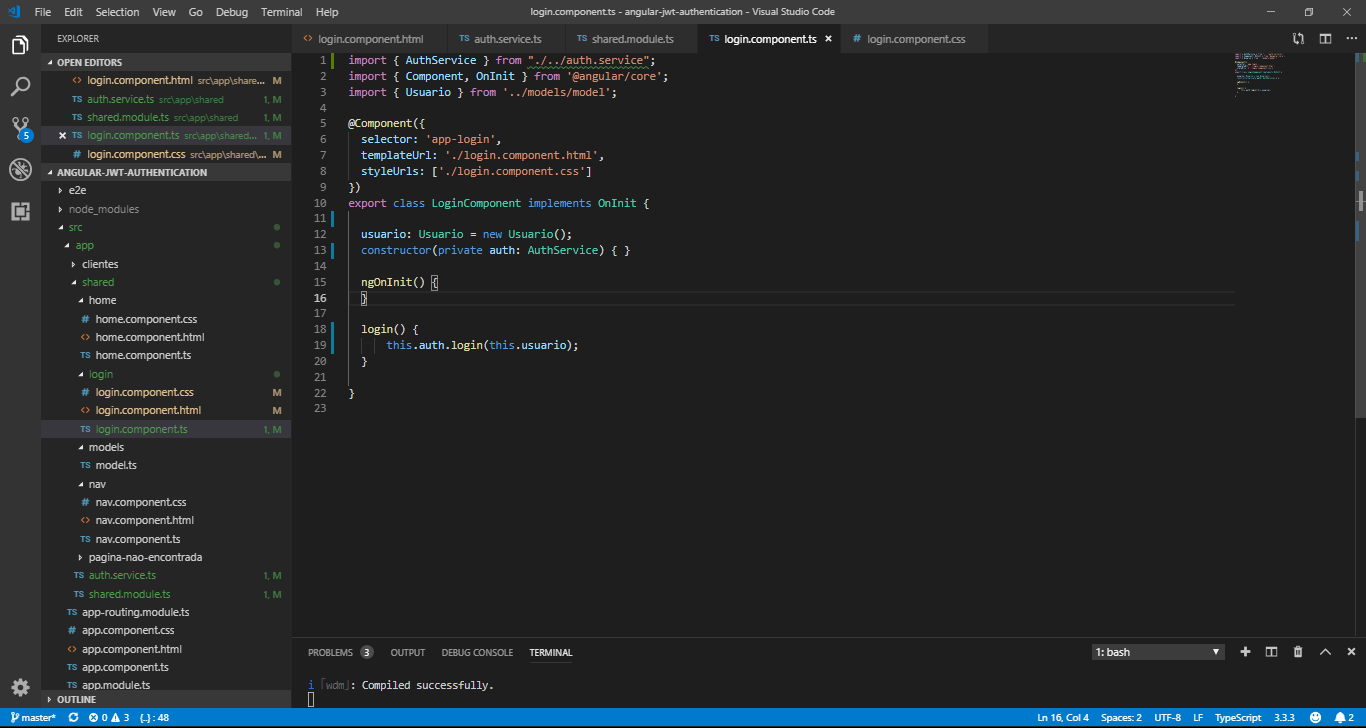
Controlar a exibição do menu com a variável do usuário logado





Login Component

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



# CRIANDO UM SERVIÇO DE MENSAGEM COM GROWL

Em toda aplicação, sempre há momentos onde devemos apresentar mensagens de sucesso, erro ou alerta para o usuário e com o PrimeNg é muito simples implementar um componente que gerencia a exibição de mensagens na tela.

Para um componente de mensagem, podemos utilizar o componente Growl do Primeng seguindo os itens abaixo:

1. Importar e exportar no SharedModule o GrowlModule para que possamos utilizar-lo

import { MessageService, GrowlModule } from "primeng/primeng";

1. Prover MessageService no método forRoot do SharedModule

export class SharedModule {

static forRoot(): ModuleWithProviders {

return {

ngModule: SharedModule,

providers: [

AuthService,

MessageService

]

}

}

}

1. Incluir no app.component.html o componente do Growl para que ele possa ser apresentado e controlado

<div class="content">

<router-outlet></router-outlet>

<p-growl [style]="{marginTop: '5%'}"></p-growl>

</div>

1. Chamar métodos de exibição de mensagem do MessageService passando parâmetros de definição de tipo de mensagem, label, titel, etc.

this.message.add({severity:'error', summary:'Erro', detail:'Usuário ou senha inválido!'});

# EXERCÍCIO PRIMENG

**Criando componentes compartilhados com PrimeNG.**

# PRÓXIMO MÓDULO

* Entendendo como funciona o HttpClient
* Componentes Guards em um projeto Angular
* Autenticação com Padrão JWT
* Nível de acesso com Roles