



# Introdução a Angular

---

Profa. Maria Janaina da Silva Ferreira



# Introdução

- Angular é um framework para design de aplicativos e uma plataforma de desenvolvimento para criação de aplicações single pages eficientes e sofisticadas.
- Informações sobre a plataforma estão disponíveis em

<https://angular.io/>

# Introdução

- Parceria Microsoft + Google
- Open Source, código está disponível no Github
- Utiliza padrões Web e WebComponents
- Orientado a componentes

# Blocos Principais

Componentes

Diretivas

Roteamento

Serviços

Template

Metadata

Databinds

Injeção de  
dependências

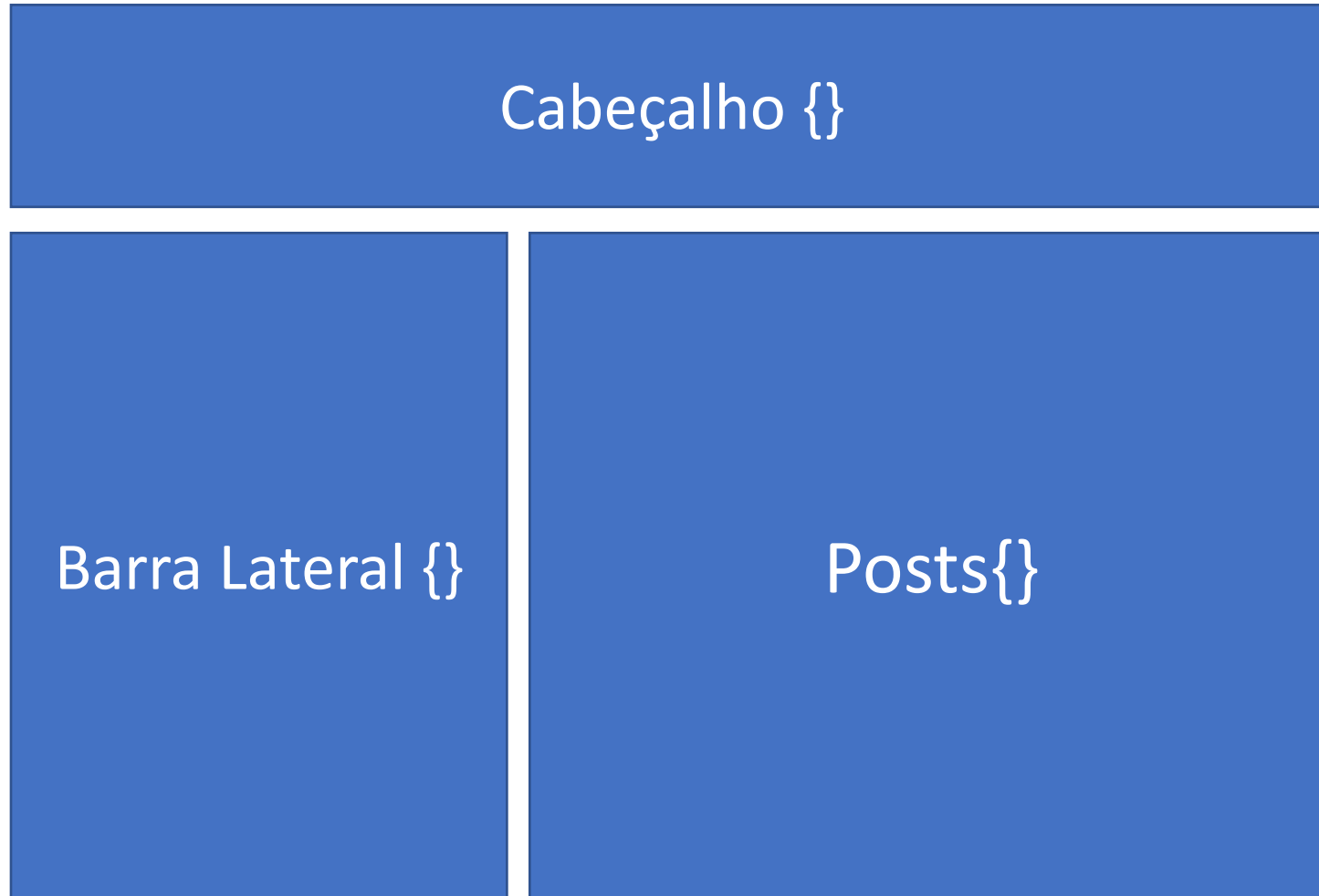
# Componente

## Encapsula:

- Template (código html)
- Metadata (são metadados, permitem ler e processar as classes do Angular)
- Dados que serão mostrados na tela (DataBinding, associação dos dados com o html)
- Comportamento da view (junção do template, do controller e do escopo do Angular)

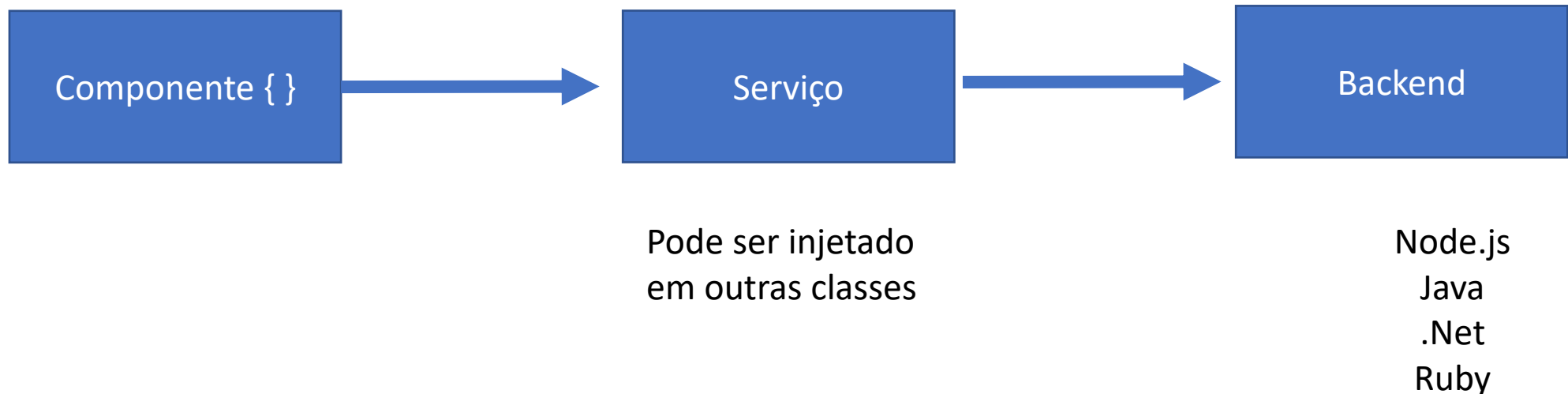
Componentes são marcadores ou extensões de elementos que compõem o DOM. estes marcadores informam ao Angular para inserir alguma funcionalidade específica a um elemento.

# Componente



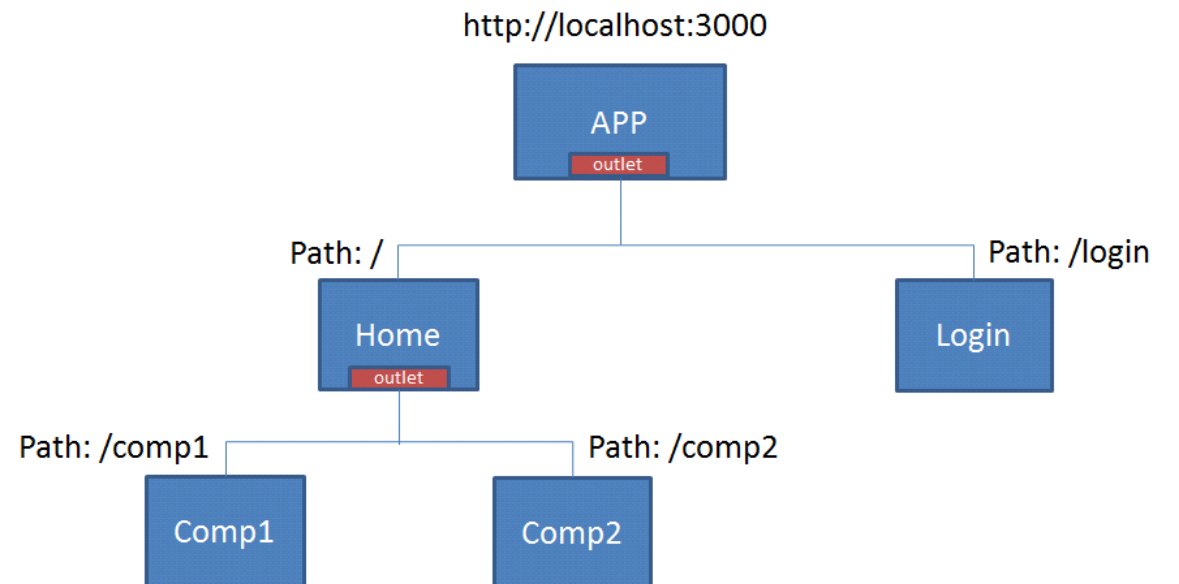
# Serviço

- Contém as regras de negócio da aplicação, métodos para acesso e manipulação de dados no banco de dados etc. Usado para integração do componente com uma api



# Roteamento

- Responsável pelo roteamento das páginas
  - Angular utiliza SPA
  - Responsável pela navegação entre as páginas
  - Caminhos relativos
  - Restrições de acesso etc





# Diretivas

- Responsáveis por modificar comportamento do DOM e/ou seu comportamento
- Componentes também são diretivas
- Exemplo: Diretivas que modificam o DOM (ao clicar em um input ele aumenta de tamanho)

# Pré-requisitos

- Para trabalhar com Angular é necessário algum conhecimento prévio em:
  - JavaScript
  - Html
  - CSS
  - TypeScript é recomendado mas não é necessário
  - Node.js
  - Npm package manager
  - Angular



Preparando o ambiente

# Node.js

- Node.js é uma plataforma construída sobre o motor JavaScript do Google Chrome para construção de aplicações de rede rápidas e escaláveis.
- Node.js usa o modelo de I/O direcionada a eventos que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos.



# Node.js

- Para utilizar o Angular, faça o download do node.js no link:
- <https://nodejs.org/en/>
- Baixe a última versão e execute em seu computador
- Para verificar a última versão instalada no seu computador, digite no prompt de comando:  
node -v

# npm package manager

- O Angular necessita das funcionalidades do npm package manager. Vamos utilizar o npm cliente, que instalado com o Node.js
- Saiba mais sobre npm no link: <https://docs.npmjs.com/>
- Para verificar sua instalação, digite no prompt de comando:  
`npm -v`
- Para atualizar sua versão para a última lançada, digite o comando:  
`npm install -g npm@latest`

# Instalação do Angular

- Vamos utilizar o Angular Cli, saiba mais no link:

<https://cli.angular.io/>

- Para instalar o Angular, abra o prompt de comando e digite o seguinte código:

`npm install -g @angular/cli`

```
npm install -g @angular/cli
```

# Angular cli

## Funcionalidades do Angular cli

- `ng new`
  - Criar uma nova aplicação
- `ng generate (ng g)`
  - Criar componentes, rotas, serviços e pipes
- `ng serve`
  - Testar a aplicação em desenvolvimento



# Criando o primeiro projeto

- Angular trabalha com o contexto de Workspace. Uma estrutura com projetos, arquivos e bibliotecas, que geralmente e por segurança ficam armazenadas em um repositório, como git por exemplo.
- Para criar o primeiro projeto, no prompt de comando ou terminal (Linux) digite o seguinte código:  
`ng new my-app`
- Onde my-app é o nome do seu Workspace
- Você também pode utilizar o site: <https://stackblitz.com/> para desenvolver seu projetos

# Executando o projeto criado

- No prompt de comando ou terminal, acesse a pasta do seu projeto  
`cd my-app`

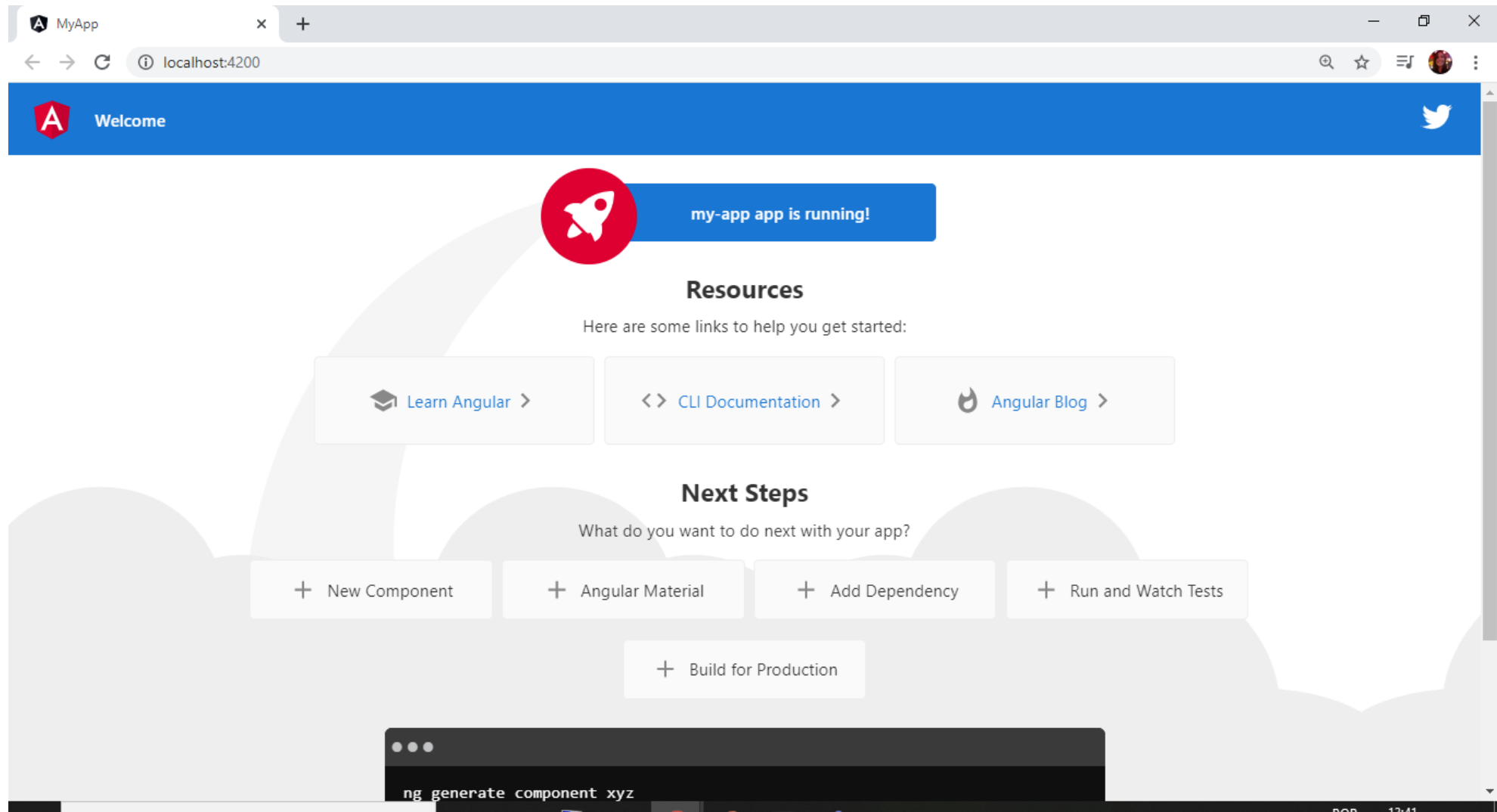
- Digite o comando abaixo para compilar e abrir o projeto:  
`ng serve --open`

`ng serve` compila seu projeto

`--open` abre o projeto após a compilação

Pressione `ctrl+c` para finalizar o servidor

# Meu primeiro projeto!

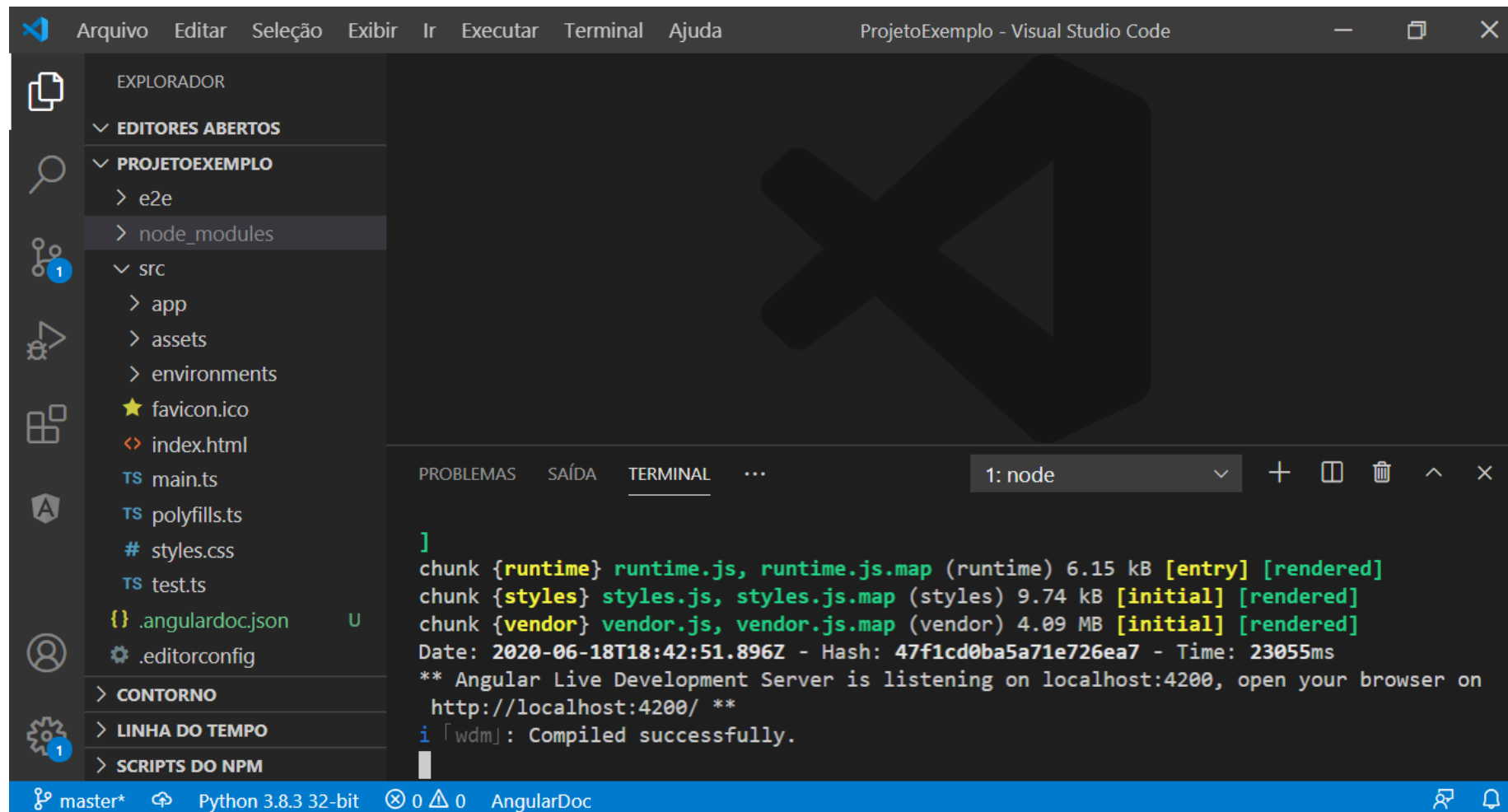


# Editor de texto para Angular

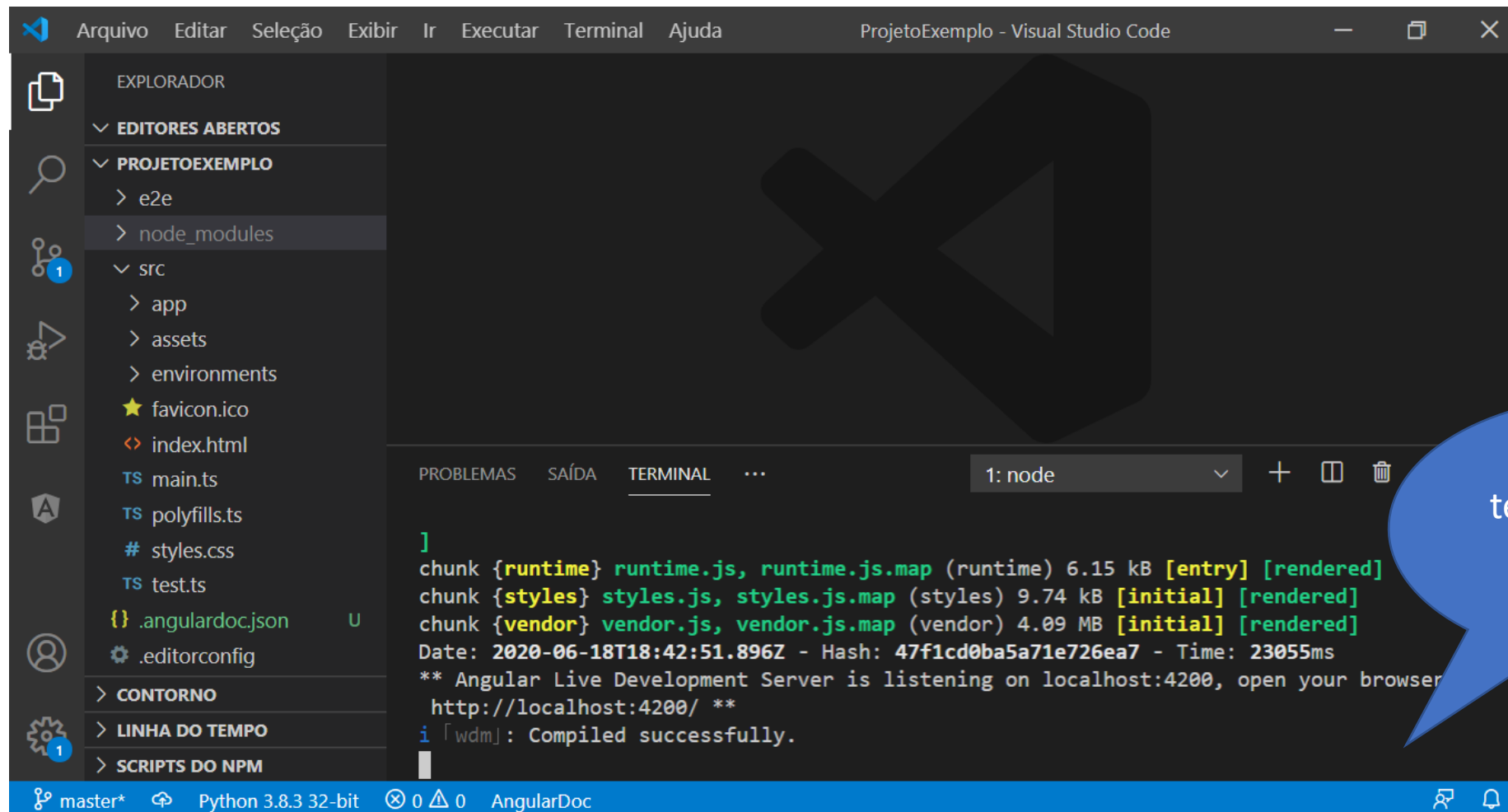
- Recomendo instalar o Visual Studio Code, disponível no link:  
[code.visualstudio.com](https://code.visualstudio.com)
- Após instalação, adicione a extensão 'Angular Extension Pack' (Lorraine Groner) que vai facilitar o desenvolvimento
- No prompt, na pasta do seu projeto, para abrir o projeto no Visual Studio code, digite  
code .

```
PWEB\Aula1Angular\ProjetoExemplo>code .
```

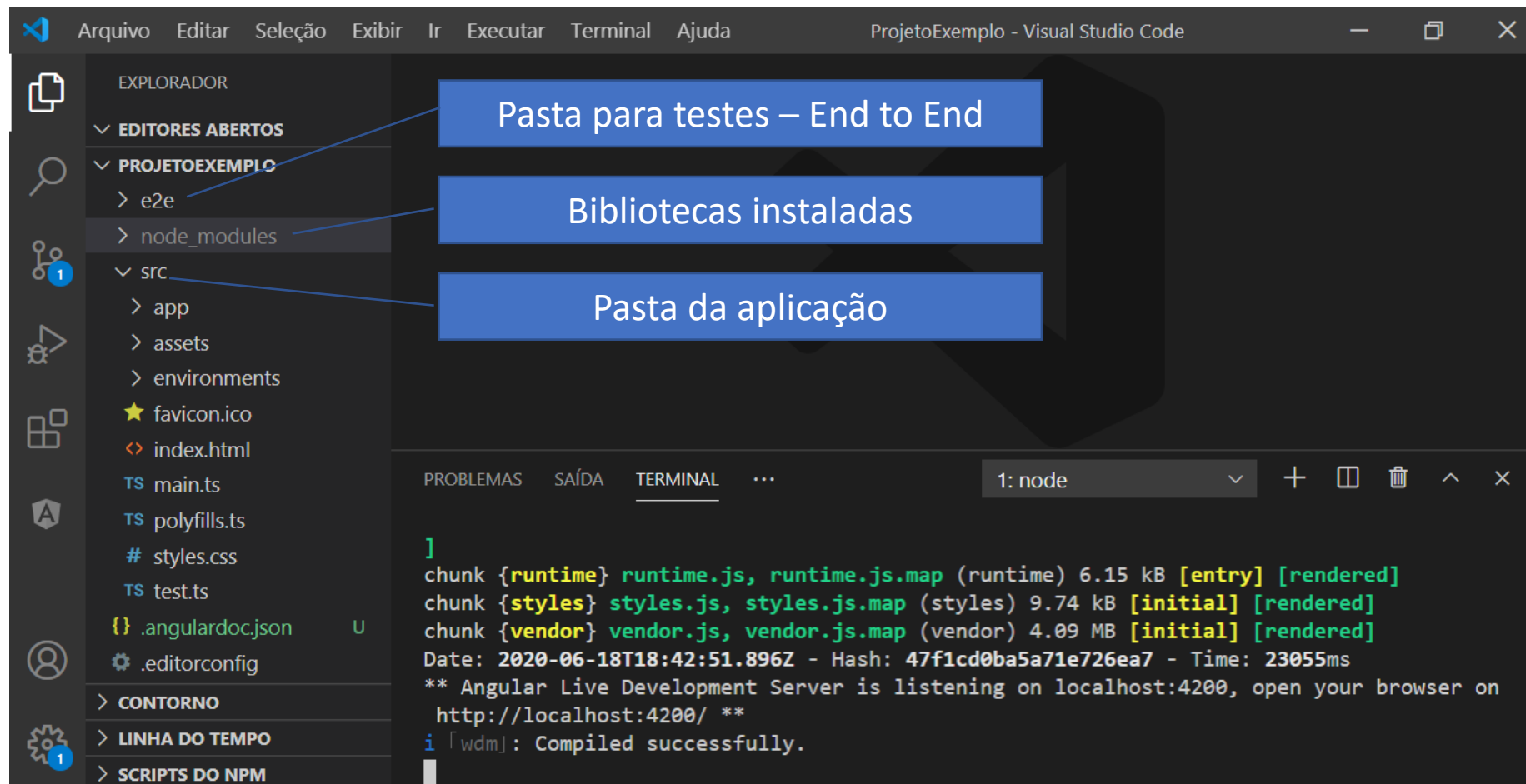
# Estrutura do Projeto criado



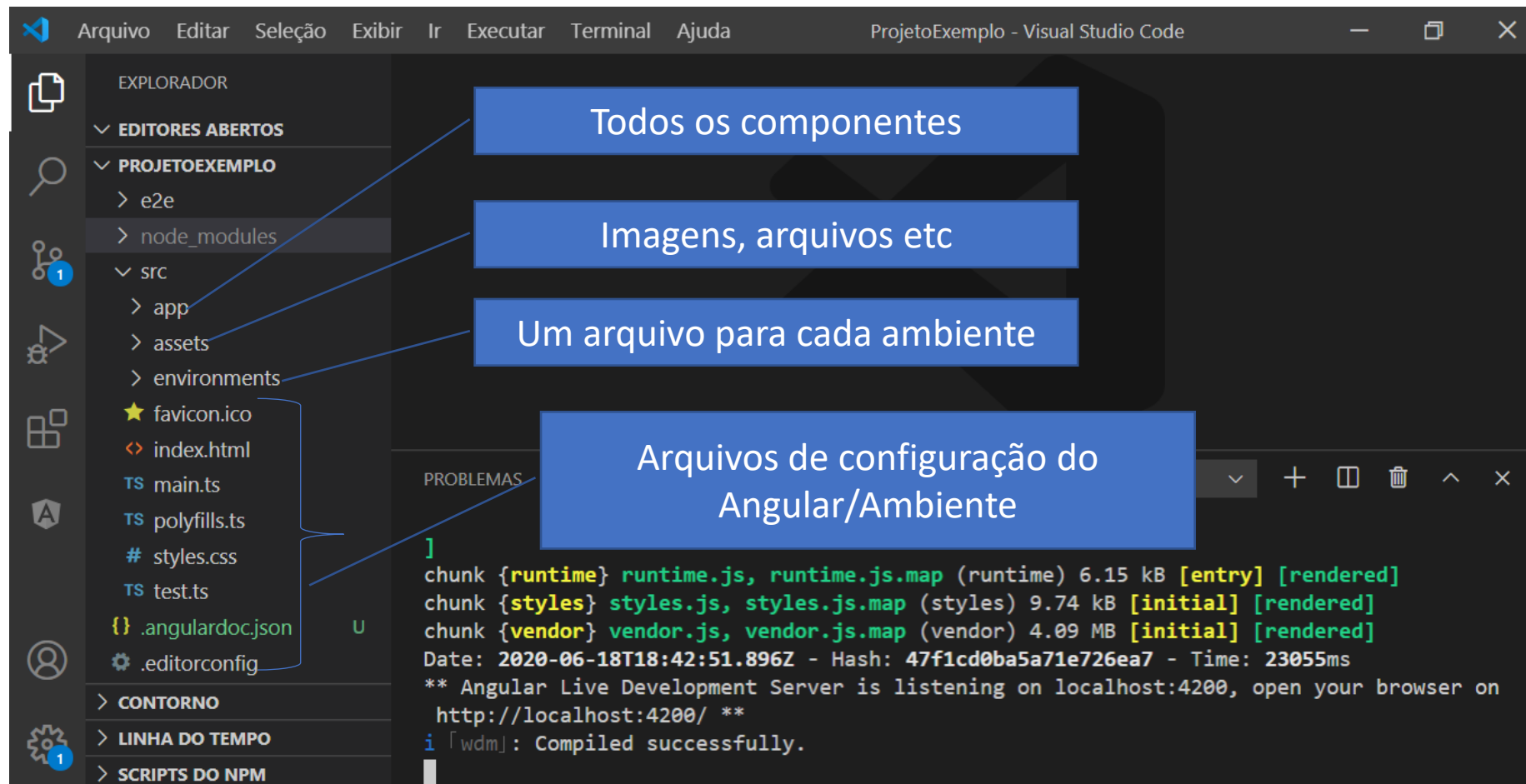
# Estrutura do Projeto criado



# Estrutura do Projeto criado



# Estrutura do Projeto criado



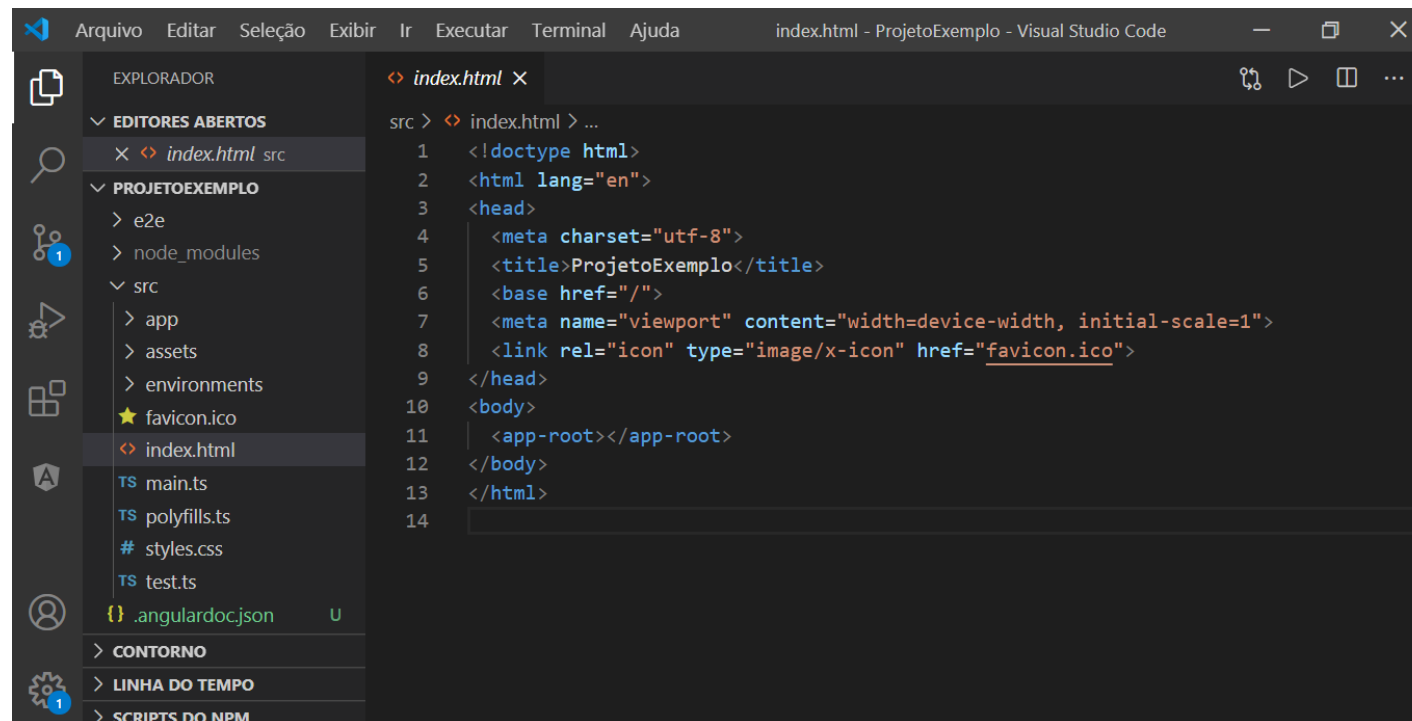


# Conhecendo as extensões

- `app.component.ts`— Classe do componente, escrito em TypeScript
- `app.component.html`— o Template do componente, escrito em HTML.
- `app.component.css`— Os estilos privados do componente, CSS exclusivo para aquele componente.

# Arquivo index.html

- Arquivo principal. Contém uma chamada para o componente inicial para o `<app-root>` que redireciona para a página `app.component.ts`
- **app-root** é um seletor existente no arquivo `app.component.ts`

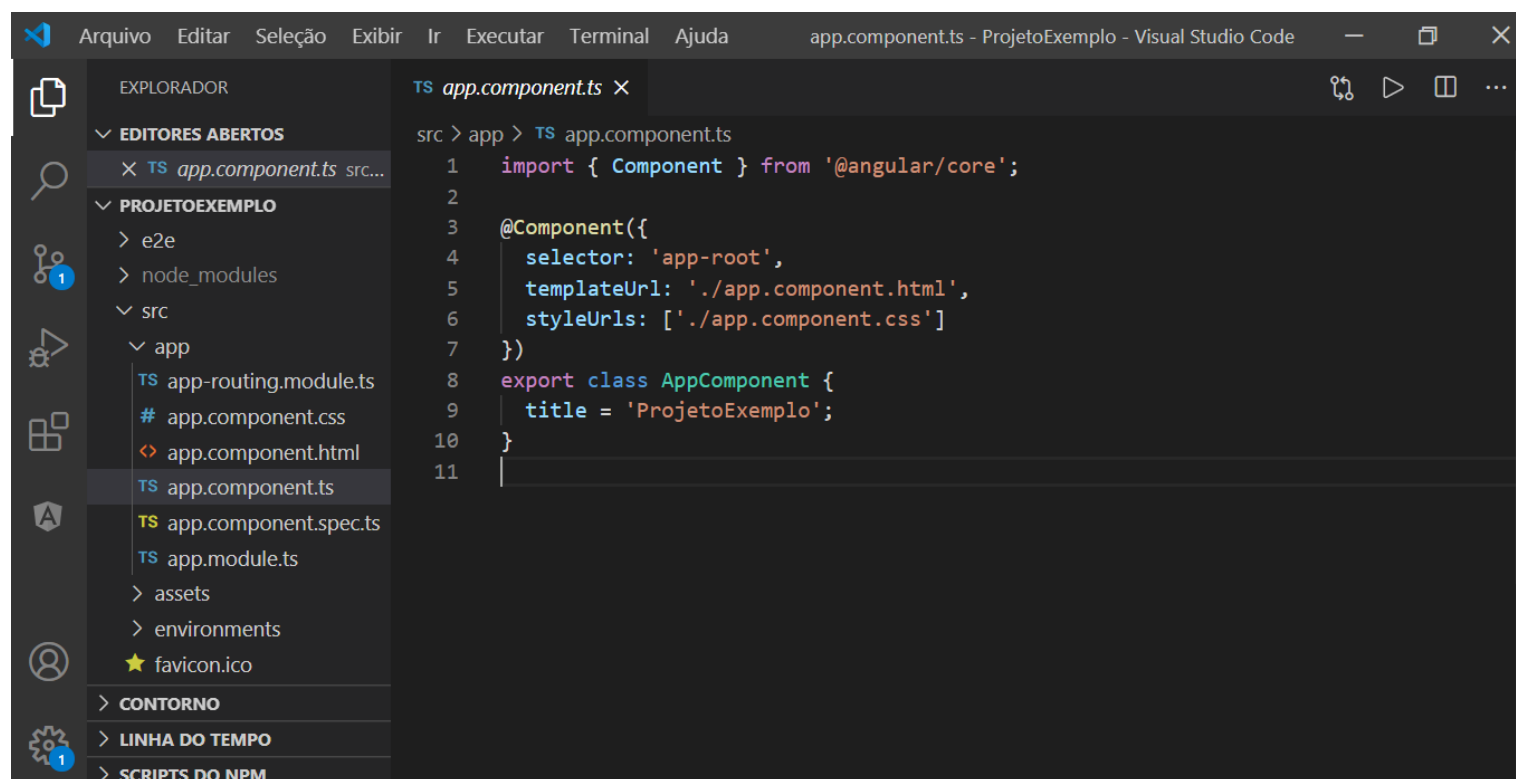


The screenshot shows the Visual Studio Code interface with the 'index.html' file open in the editor. The file explorer on the left shows the project structure, including the 'src' directory and the 'index.html' file. The editor displays the following HTML code:

```
src > index.html > ...
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>ProjetoExemplo</title>
6   <base href="/">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <link rel="icon" type="image/x-icon" href="favicon.ico">
9 </head>
10 <body>
11   <app-root></app-root>
12 </body>
13 </html>
14
```

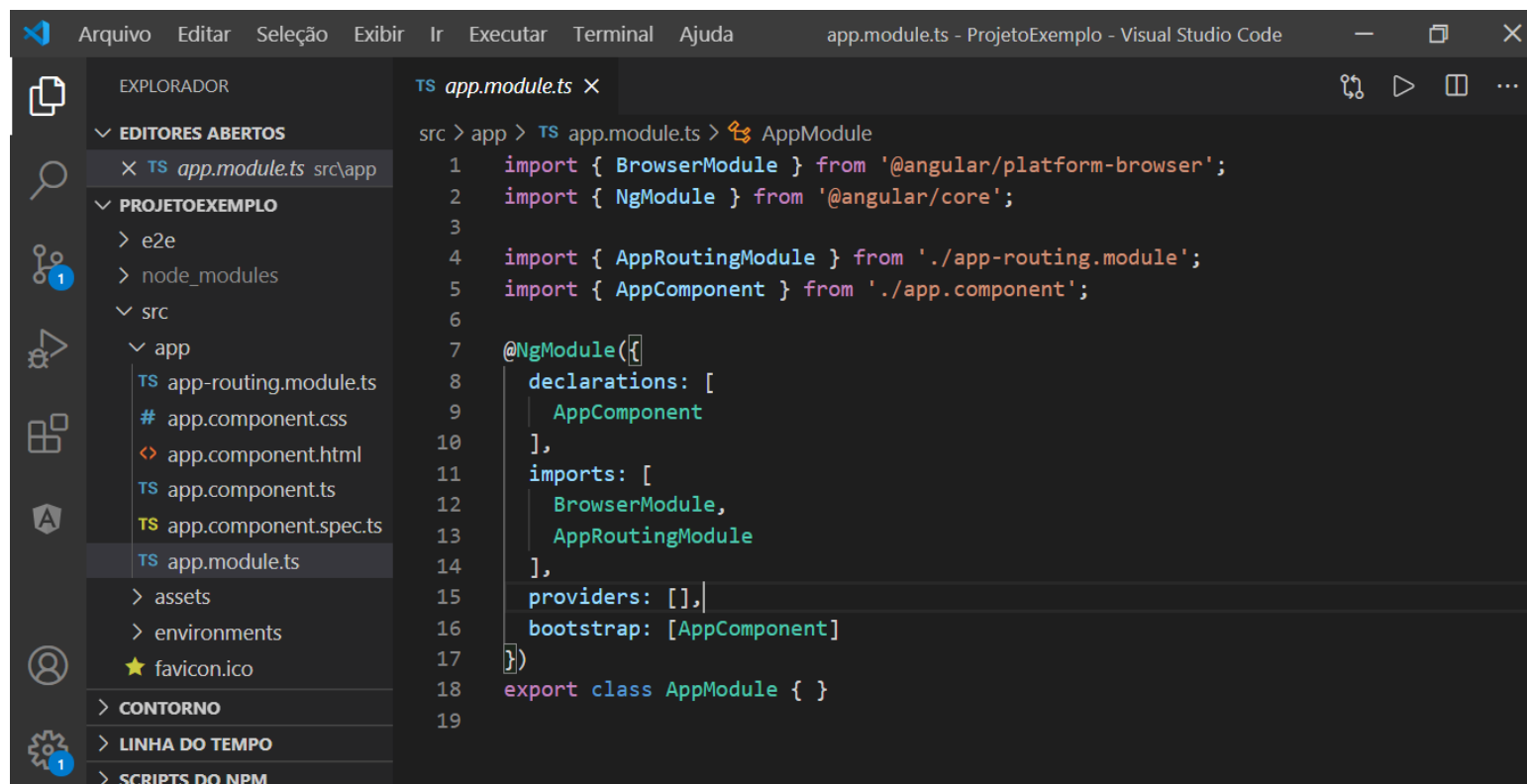
# Principais arquivos app/app.component.ts

- Definição do componente raiz, que será uma árvore de componentes aninhados conforme a aplicação for evoluindo



# Principais arquivos app/app.module.ts

- Módulo raiz onde serão importados todos os módulos e componentes



The screenshot shows the Visual Studio Code interface with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the following files and folders:

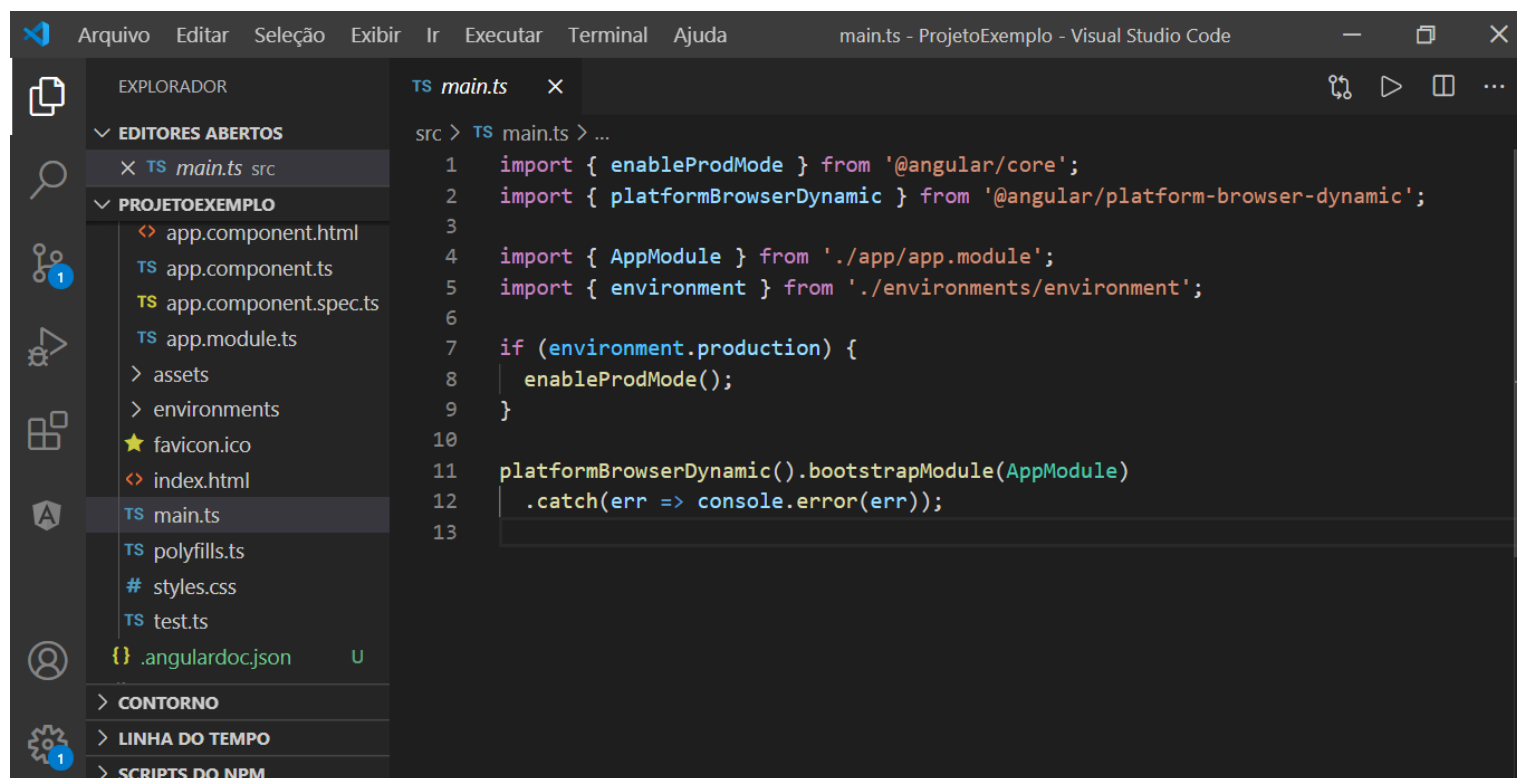
- EDITORES ABERTOS
  - TS app.module.ts src\app
- PROJETOEXEMPLO
  - e2e
  - node\_modules
  - src
    - app
      - TS app-routing.module.ts
      - # app.component.css
      - <> app.component.html
      - TS app.component.ts
      - TS app.component.spec.ts
      - TS app.module.ts
    - assets
    - environments
    - ★ favicon.ico
  - CONTORNO
  - LINHA DO TEMPO
  - SCRIPTS DO NPM

The code editor shows the content of the file `app.module.ts`:

```
src > app > TS app.module.ts > AppModule
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6
7  @NgModule({
8    declarations: [
9      AppComponent
10   ],
11   imports: [
12     BrowserModule,
13     AppRoutingModule
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule { }
19
```

# Principais arquivos app/main.ts

- É o vínculo que combina o componente (app.component.ts) com a página (index.html)



# Processo de inicialização

- Começa a partir do arquivo `main.ts` que é o ponto de entrada da aplicação
- O `AppModule` importado em '`main.ts`' funciona como o módulo root
- O módulo root é configurado para utilizar o arquivo '`app.component.ts`' como o componente principal e ser inicializado e este será chamado para renderizar toda a tag `<app-root>` no arquivo `index.html`, quando solicitamos ao browser que abra o arquivo `index.html`, através da translação do arquivo '`main.ts`', a função *`platformBrowserDynamic().bootstrapModule(AppModule)`*, se encarrega de realizar o start do processo de inicialização da aplicação

# Vamos praticar



- Abra o projeto criado anteriormente
- Apague todo o conteúdo do arquivo app.component.html
- Crie uma variável chamada mensagem no arquivo app.component.ts

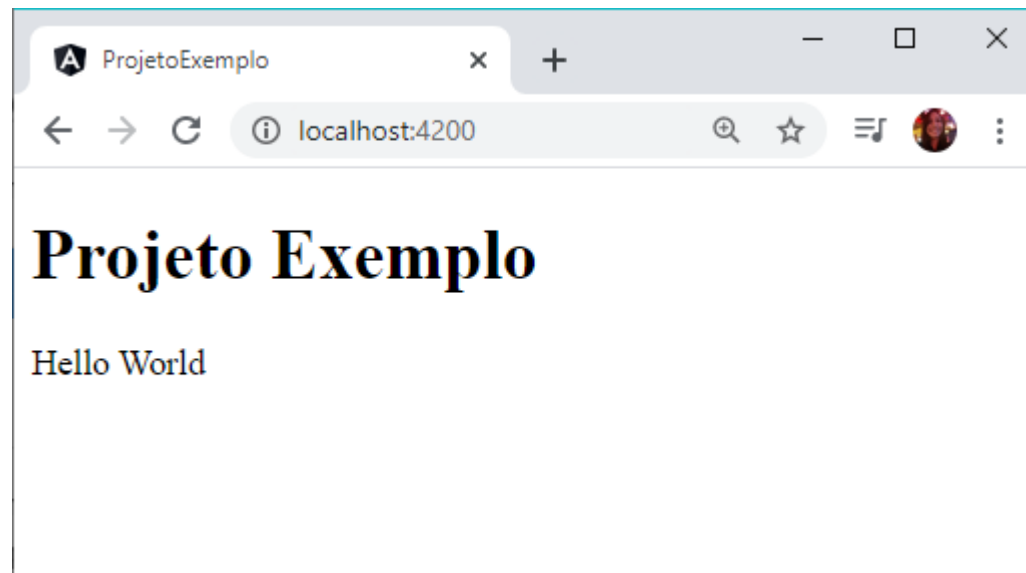
```
export class AppComponent {  
  title = 'Projeto Exemplo';  
  mensagem = 'Hello World';  
}
```

- Inclua o título e a mensagem criada no Html

```
<> app.component.html X TS app.component.ts  
src > app > <> app.component.html > ...  
1 | <h1>{{title}}</h1>  
2 | <p>{{mensagem}}</p>
```

# Vamos praticar

- No terminal, digite `ng serve`
- Abra o navegador e digite `localhost:4200`
- Verifique se os dados foram exibidos em sua aplicação





# Vamos praticar

- Vamos aplicar um estilo (css) ao nosso arquivo:
- Abra o arquivo app.component.css e configure a fonte, cor e tamanho

```
src > app > # app.component.css > p
1  h1 {
2      color: #369;
3      font-family: Arial, Helvetica, sans-serif;
4      font-size: 250%;
5  }
6
7  p{
8      color: #333;
9      font-family: Cambria;
10     font-weight: lighter;
11 }
12
```

# Exercícios

1) Acesse o projeto disponível no Figma

<https://www.figma.com/proto/jFDd5LPUTQuNS6lsgRr4V9/Aula-PWEB-II?type=design&node-id=1-51&t=he1ik4ltJtyDFJ5x-1&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A51>

2) **Entre na pasta do projeto criado anteriormente.** Crie o componente principal main. Ele será responsável por agrupar todos os outros componentes

# Adicionando um novo componente

- Para adicionar um novo componente(página) em nosso projeto, vamos usar o comando

ng g **component** pasta/nomedoarquivo

generate

Item a ser criado

- Exemplo:

ng g componente main

# Adicionando um novo componente

- Por padrão, se não for especificada a pasta para criação dos arquivos, eles serão criados na pasta src/app
- Observe que ao executar o comando, foram criados quatro arquivos com extensões distintas:
  - O .html é arquivo onde iremos criar o layout do nosso componente
  - O .css é o arquivo onde iremos incluir toda a formatação e definir a aparência do nosso componente
  - O .ts é o arquivo onde iremos incluir o código fonte do nosso componente
  - O .spec.ts é o arquivo onde iremos incluir os testes unitários do componente

```
CREATE src/app/main/main.component.html (20 bytes)
CREATE src/app/main/main.component.spec.ts (605 bytes)
CREATE src/app/main/main.component.ts (238 bytes)
CREATE src/app/main/main.component.css (0 bytes)
```

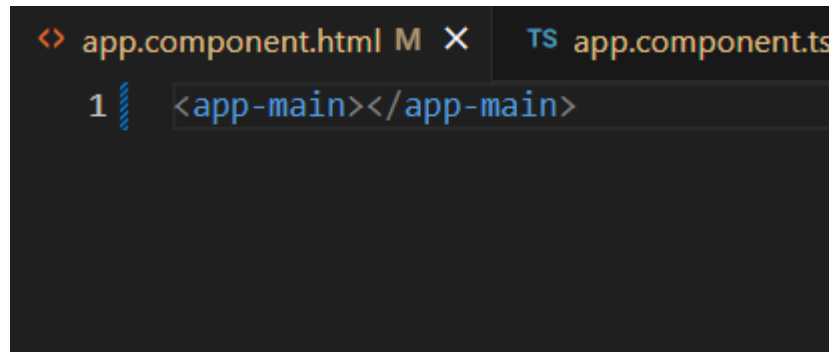
# Adicionando um novo componente

- Vamos editar o arquivo main.component.ts para que seu acesso fique disponível a outros componentes da aplicação. Para isso, vamos verificar se o parâmetro standalone está definido para true.

```
TS main.component.ts U X
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-main',
5    standalone: true,
6    imports: [],
7    templateUrl: './main.component.html',
8    styleUrls: ['./main.component.css']
9  })
10 export class MainComponent {
11
12 }
```

# Alterando a página inicial do projeto

- Nosso projeto por padrão aponta para a página app.component.ts
- Para redirecionar para outra página, abra o arquivo app.component.ts e coloque o seletor da página que deseja chamar.
- Exemplo:



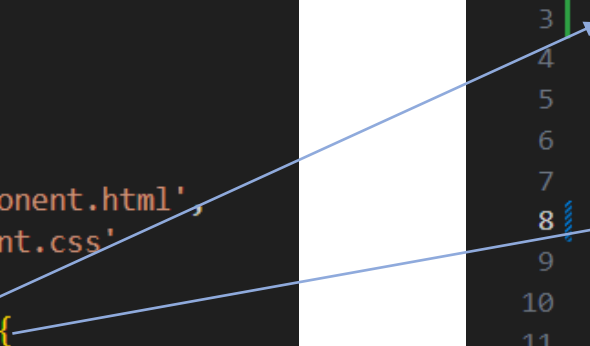
```
<> app.component.html M X TS app.component.ts
1 <app-main></app-main>
```

# Alterando a página inicial do projeto

- Sempre que incluirmos um componente em outro componente, precisamos importar a classe desse componente, no arquivo no qual ele está sendo inserido.

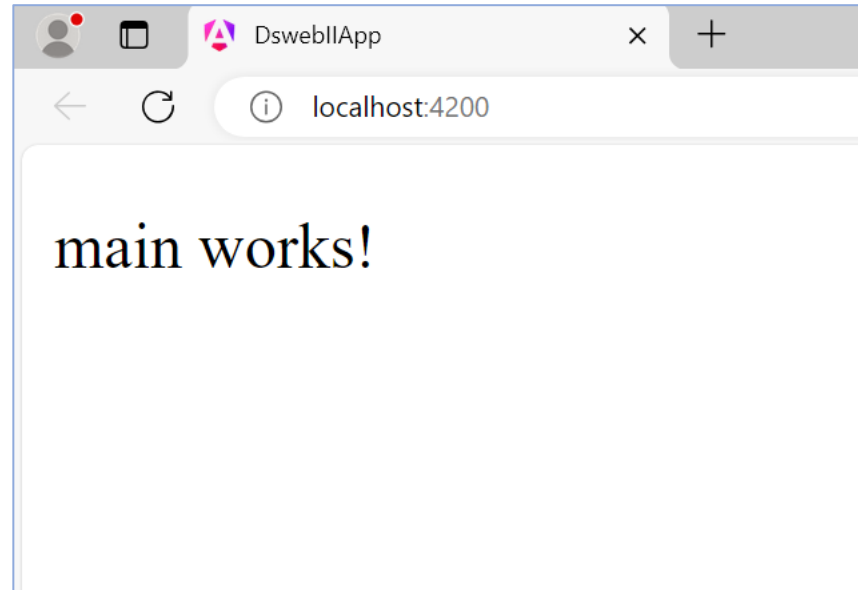
```
TS main.component.ts U X
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-main',
5    standalone: true,
6    imports: [],
7    templateUrl: './main.component.html',
8    styleUrls: ['./main.component.css']
9  })
10 export class MainComponent {
11
12
13
```

```
TS app.component.ts M X
1  import { Component } from '@angular/core';
2  import { RouterOutlet } from '@angular/router';
3  import { MainComponent } from './main/main.component';
4
5  @Component({
6    selector: 'app-root',
7    standalone: true,
8    imports: [RouterOutlet, MainComponent],
9    templateUrl: './app.component.html',
10   styleUrls: ['./app.component.css']
11 })
12 export class AppComponent {
13   title = 'dswebII-app';
14 }
15
```



# Adicionando um novo componente

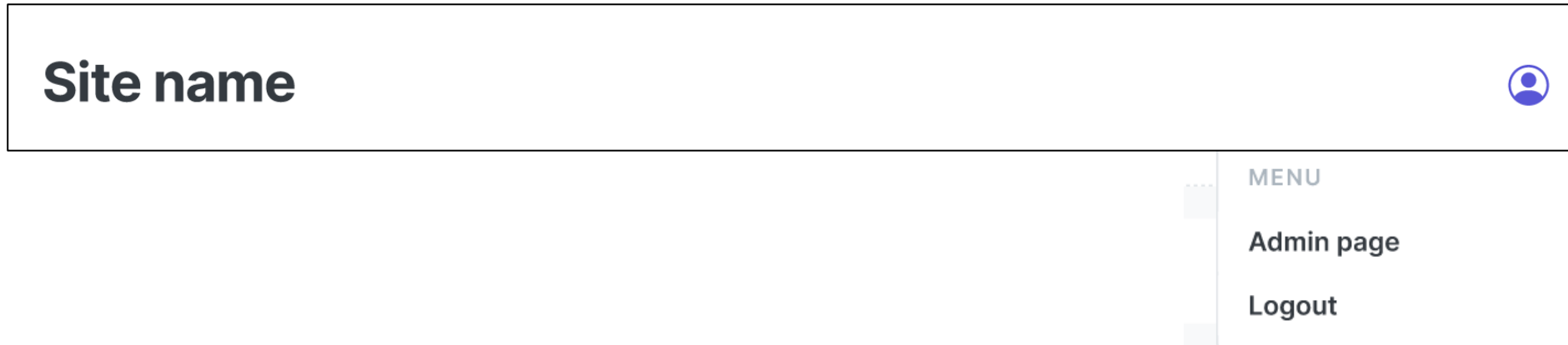
- Execute o projeto criado, digite no terminal:
  - `ng serve --open`
- Verifique no navegador se a aplicação foi executada corretamente





# Exercícios

3) Crie o componente principal header, conforme o template apresentado no figma.

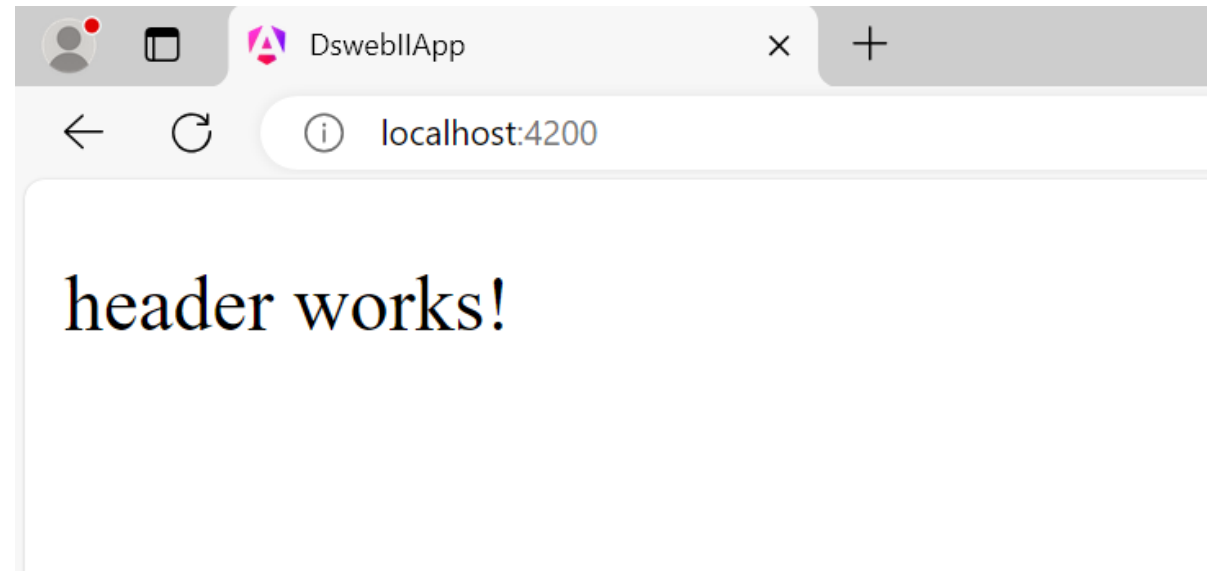


# Exercícios

4) Abra o componente main.html e inclua o seletor do component header. Não esqueça de incluir os imports.

# Exercícios

- Verifique no navegador se o componente header está sendo exibido corretamente.



# Exercícios

## 5) Publique o projeto criado no GitHub

- a) Clique no ícone Source Control do Visual Studio
- b) Adicione uma mensagem para seu commit e clique em commit
- c) A seguir, clique em Publish branch e escolha o repositório desejado

