DESENVOLVIMENTO WEB – FRONT END

AULA 2

Prof. Mauricio Antonio Ferste

CONVERSA INICIAL

Primeiramente temos de entender onde estamos pisando, o que estamos usando para desenvolver. Concorda?

A infraestrutura e arquitetura são fundamentais para o desenvolvimento de softwares de qualidade que atendam às necessidades dos usuários. Uma boa infraestrutura e arquitetura garantem que o software seja escalável, seguro, confiável e fácil de manter e evoluir. A infraestrutura inclui todos os recursos necessários para suportar o software, como servidores, bancos de dados, redes, armazenamento, entre outros. Uma infraestrutura bem planejada e configurada pode melhorar a performance, aumentar a disponibilidade e reduzir custos.

A arquitetura, por sua vez, é a estrutura do software, incluindo suas camadas, componentes, interações e fluxos de dados. Uma arquitetura bem definida e projetada pode tornar o software mais modular, fácil de entender e evoluir e permitir a integração com outros sistemas. Além disso, contribui para a redução de erros, melhorar a segurança e permitir a implantação automatizada e rápida de novas funcionalidades.

Portanto, investir em infraestrutura e arquitetura adequadas é fundamental para garantir a qualidade do software e a satisfação dos usuários, além de permitir que a empresa esteja preparada para enfrentar desafios futuros e atender às demandas do mercado.

Nosso ambiente:

IDE: VsCode

Plataforma: Node.js + NPM

Linguagem: Javascript e Typescript, com HTML5 E CSS3.

Framework: Angular

Vamos agora entender nossa infraestrutura básica para executar nosso primeiro projeto.

Para concluir esta etapa com sucesso, tem de existir a meta de conseguir fazer a instalação da infraestrutura e criação de seu primeiro projeto. Vamos lá.

TEMA 1 – INSTALAÇÕES POR SISTEMA OPERACIONAL DO VS CODE

Neste tópico vamos desvendar como instalar, configurar nossa IDE, para que consigamos trabalhar.

Alguns exemplos de IDEs populares incluem Visual Studio Code, IntelliJ IDEA, Eclipse, PyCharm e Xcode. Cada uma dessas IDEs tem suas próprias vantagens e desvantagens, e a escolha de uma delas depende das necessidades específicas do projeto e das preferências do desenvolvedor.

Saiba mais

IDE: um ambiente de desenvolvimento integrado (IDE) é uma ferramenta essencial para desenvolvedores de software, que combina várias funcionalidades em uma única aplicação. Algumas das funcionalidades mais comuns que uma IDE pode oferecer incluem:

- **Editor de código**: permite escrever, editar e formatar o código de maneira eficiente, com recursos como destaque de sintaxe, sugestões de código e atalhos de teclado. Compilador para verificar a sintaxe do código e traduzi-lo em código executável.
- **Depurador**,: permite testar o código e encontrar e corrigir erros. Gerenciador de pacotes que facilita a instalação e gerenciamento de bibliotecas e dependências de código.
- **Ferramentas de análise de código:** para identificar e corrigir problemas de desempenho, segurança e qualidade do código. Integração com sistemas de controle de versão, que facilita o gerenciamento de código-fonte em equipe.

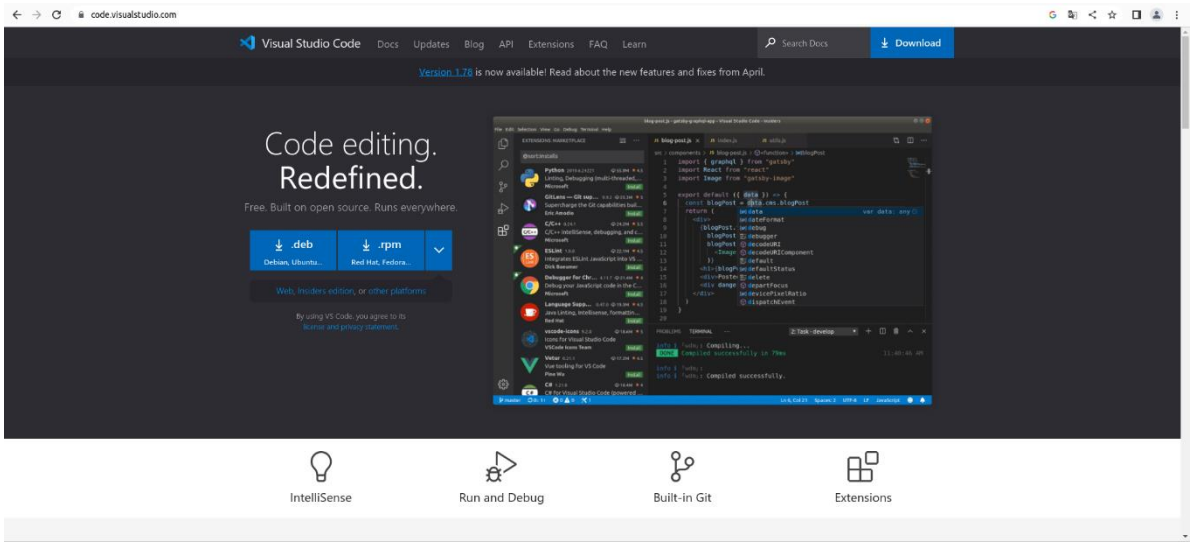
Importante saber que não existe o melhor ou a melhor IDE, não estamos aqui defendendo uma em especial, mas por questões acadêmicas temos de adotar uma plataforma para direcionar nossos códigos, e a plataforma escolhida foi VsCode ou Visual Studio Code (Disponível em: <https://code.visualstudio.com/>. Acesso em: 24 ago. 2023).

1.1 Onde encontrar o VsCode

O Visual Studio Code pode ser baixado gratuitamente no site oficial da Microsoft disponível em: <https://code.visualstudio.com/>. Acesso em: 24 ago. 2023. Lá você encontrará versões disponíveis para Windows, macOS e Linux.

Basta escolher a versão adequada ao seu sistema operacional e seguir as instruções de instalação.

Figura 1 – Tela do sítio do Visual Studio Code



Fonte: Visualstudio, 2023.

1.2 Instalação básica

Para instalar o Visual Studio Code (VSCode), siga as etapas abaixo:

1. Acesse o site oficial do VSCode disponível em: <https://code.visualstudio.com/>. Acesso em: 24 ago. 2023.
2. Clique no botão de download para o sistema operacional que você está usando (Windows, macOS ou Linux).
3. Abra o arquivo de instalação que foi baixado.
4. Siga as instruções do assistente para concluir a instalação.
5. Quando a instalação estiver concluída, abra o VSCode.

O VSCode também está disponível em gerenciadores de pacotes de algumas distribuições Linux, como o apt-get do Ubuntu, o dnf do Fedora e o pacman do Arch Linux. Se você estiver usando uma dessas distribuições, poderá instalar o VSCode usando o gerenciador de pacotes da sua distribuição.

Saiba mais

VsCode, ou Visual Studio Code, é uma plataforma diferente de Microsoft Visual Studio! É utilizado para Javascript, Python, Typescript... e vamos usar aqui.

Já o Microsoft Visual Studio é uma plataforma bem mais antiga usada para C++, C Sharp, Visual Basic .NET, J Sharp e é fechada, paga!

Não confunda! Não é a mesma plataforma.

1.3 Instalação básica do Node.js

O Node.js é uma plataforma de tempo de execução JavaScript que possibilita a criação de funcionalidades escaláveis de back-end para programação do lado do servidor. Com a familiaridade da linguagem JavaScript, já amplamente utilizada no desenvolvimento de aplicações baseadas em navegador, muitos desenvolvedores se sentem confortáveis para utilizar essa plataforma.

O Node.js pode ser baixado gratuitamente no site oficial em: https://nodejs.org/en.

Figura 2 – Tela do sítio do Node.js

.

Fonte: Node.js, 2023.

Saiba mais

"Node LTS Current" refere-se à versão mais recente da série LTS (Long-Term Support) do Node.js no momento. A versão LTS é geralmente recomendada para uso em ambientes de produção, pois recebe suporte a longo prazo e atualizações de segurança regulares. No caso de estudo, é esta que devemos usar!

No momento em que escrevo isso, a versão LTS Current mais recente do Node.js é a versão 18. A versão Current, atualmente a 20, deve ser evitada, pois ainda tem detalhes que estão evoluindo para deixá-la pronta!

É importante manter o Node.js atualizado para garantir que seu código esteja rodando em uma plataforma segura e atualizada. Você pode verificar a versão do Node.js instalada em sua máquina digitando "node –v" no terminal. Se você precisar atualizar sua versão do Node.js, pode fazer o download da versão mais recente no site oficial do Node.js.

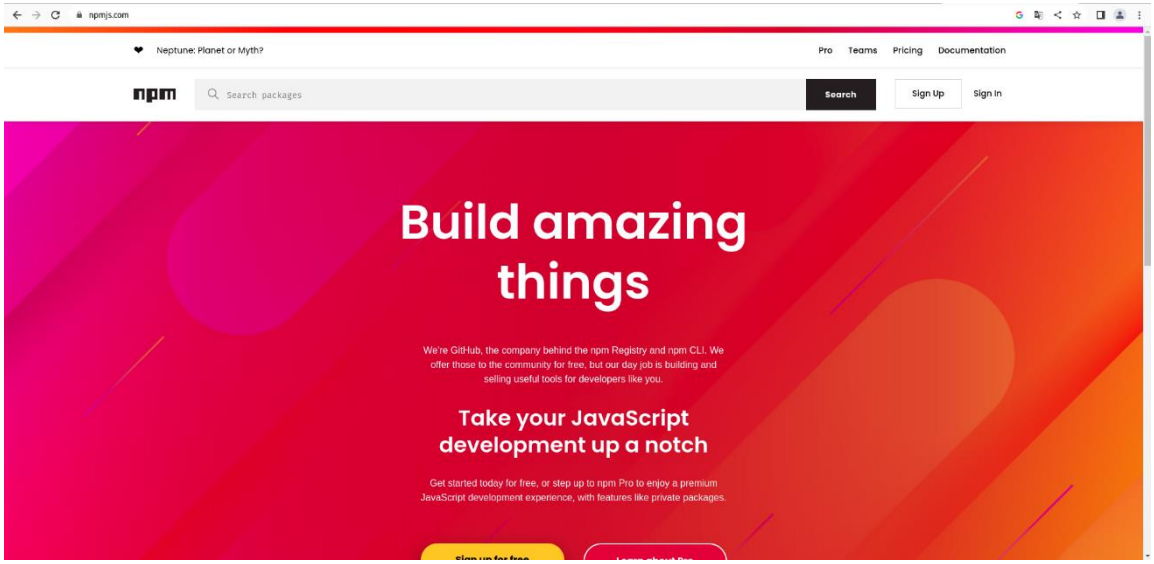
1.4 Instalação básica do NPM

Node é uma plataforma de desenvolvimento de software construída em cima do mecanismo JavaScript do Google Chrome. O Node.js permite que os desenvolvedores usem JavaScript no lado do servidor para construir aplicativos web escaláveis e de alto desempenho. Além disso, o Node.js vem com um gerenciador de pacotes de software conhecido como NPM (Node Package Manager). O NPM é uma ferramenta de linha de comando que facilita a instalação, a atualização e o gerenciamento de pacotes de software (bibliotecas) usados em projetos Node.js (NPM, 2023). Com o NPM, os desenvolvedores podem instalar pacotes de software por meio do repositório oficial do NPM bem como de repositórios externos, como o GitHub. O NPM também permite aos desenvolvedores criar seus próprios pacotes de software e publicá-los no repositório do NPM para que outros desenvolvedores possam usá-los. Além disso, o NPM é um dos maiores repositórios de código aberto do mundo, com mais de um milhão de pacotes de software disponíveis para os desenvolvedores. Isso significa que os desenvolvedores podem economizar tempo e esforço ao usar o NPM para encontrar pacotes de software já criados e testados pela comunidade. Outra vantagem do NPM é que ele facilita a gestão de dependências em projetos Node.js. Quando os desenvolvedores usam pacotes de software em seus projetos, esses pacotes podem ter dependências em outros pacotes de software. O NPM gerencia automaticamente essas dependências, garantindo que todas as bibliotecas sejam instaladas e atualizadas corretamente.

Os pacotes podem ser baixados do sítio do NPMJS disponível em: <https://www.npmjs.com/>. Acesso em: 24 ago. 2023.

Em resumo, o Node.js e o NPM são ferramentas poderosas para o desenvolvimento de aplicativos web escaláveis e de alto desempenho. Com o Node.js e o NPM, os desenvolvedores podem usar o JavaScript em todo o lado do servidor, gerenciar facilmente pacotes de software e suas dependências e acessar um vasto repositório de código aberto.

Figura 3 – Tela do sítio do NPM



Fonte: NPM, 2023.

1.5 Resumo de comandos básicos

Abra o terminal (no Windows, abra o Prompt de Comando ou PowerShell) e digite o seguinte comando para verificar se o Node.js e o NPM estão instalados e qual é a versão atual:

Figura 4 - Comandos de verificação da configuração, node –v npm -v

```
 web >> node -v
v18.16.0
 web >> npm -v
9.5.1
 web >>
```

Fonte: Ferste, 2023.

Saiba mais

A versão do Node, na minha máquina, por exemplo, é 18.16.0, e do NPM é 9.5.1. Elas **não são as mesmas numericamente falando**, o que é totalmente normal.

Normalmente a instalação do NodeJS e NPM não exige configurações especiais. Elas existem, mas são diversas. Na prática não utilizaremos nada no texto que exija algum tipo de conhecimento especial.

Entenda que não se faz diferenciação entre Windows, Linux, Mac, ou qualquer outra plataforma em termos de desenvolvimento. Como

desenvolvedores devemos utilizar a ferramenta que temos, ou seja, a melhor disponível, mas não transforme seu trabalho em uma discussão, resolva seus problemas, resolva as dores do seu usuário!

TEMA 2 – WSL E INSTALAÇÃO COMPLETA DO ANGULAR CLI

É possível utilizar o Visual Studio Code com a extensão WSL para desenvolver em um ambiente baseado em Linux e utilizar as ferramentas e os utilitários específicos desse sistema. O gerenciamento do controle de versão também é facilitado com o suporte interno do Git do VS Code, e é possível executar comandos e extensões do VS Code diretamente em projetos WSL. Além disso, podem-se editar arquivos em sistemas de arquivos Linux ou do Windows montados, sem se preocupar com problemas de caminho, compatibilidade binária ou outros desafios entre sistemas operacionais. Essa integração entre o Visual Studio Code e o WSL permite uma experiência de desenvolvimento mais flexível e produtiva para os usuários que desejam trabalhar em ambientes Linux por meio do Windows.

Saiba mais

Pode ser utilizada qualquer plataforma para desenvolvimento, Linux, Mac, Windows, infra local e em nuvem. Para os nossos exemplos utilizaremos a plataforma VsCode e Linux, entretanto é importante mencionar que não mudam os comandos, não muda a instalação, é padrão.

2.1 Configurando o WSL

Algumas distribuições do WSL Linux não possuem as bibliotecas necessárias para executar o VS Code Server. Você pode adicionar bibliotecas à sua distribuição Linux usando um gerenciador de pacotes. Por exemplo, para atualizar Debian ou Ubuntu, use:

2.2 Sudo apt-get update

A extensão WSL divide o código VS em uma arquitetura “cliente-servidor”, onde o cliente (front-end) é executado em uma máquina Windows e o servidor (seu código, Git, plug-ins etc.) é executado “remotamente” na distribuição WSL. Ao usar o plug-in WSL, selecionar a guia “Plugins” exibe uma lista de plug-ins compartilhados entre seu computador local e a distribuição WSL. A instalação

de um plug-in local como um tema só precisa ser instalada uma vez. O VS Code exibirá um ícone de aviso ⚠ e um botão verde “Instalar WSL” se você tiver um plug-in instalado localmente que não esteja instalado em sua distribuição WSL.

2.3 Instalar o Git no Windows

A instalação do Git no Windows é um procedimento simples e essencial para aproveitar os benefícios do controle de versão em projetos de desenvolvimento de software. O Git é uma ferramenta amplamente utilizada que permite rastrear alterações, colaborar com outros desenvolvedores e gerenciar o código-fonte de maneira eficiente. Para começar, você precisa acessar o site oficial do Git e encontrar a seção de downloads. Certifique-se de selecionar a versão correta do Git para o Windows.

Durante a instalação, você poderá escolher o diretório onde o Git será instalado. Normalmente, o diretório padrão é adequado para a maioria dos usuários. Além disso, você poderá selecionar o editor de texto padrão para commits do Git. Se não tiver preferências específicas, você pode manter a opção padrão selecionada. Durante o processo de instalação, você também terá a opção de adicionar o Git ao PATH do sistema. Isso permitirá que você acesse o Git por meio do prompt de comando ou de qualquer local no sistema. Uma vez que todas as configurações tenham sido selecionadas, clique em “Instalar” ou em um botão similar para iniciar a instalação.

Aguarde até que o processo de instalação seja concluído. Em seguida, você receberá uma confirmação de que a instalação do Git foi bem-sucedida. Agora, o Git está instalado no seu sistema Windows e você pode começar a usá-lo. A versão do Git será exibida, confirmando que a instalação foi concluída com sucesso. A partir de agora, você está pronto para aproveitar os recursos e benefícios do Git no Windows. O Git oferece uma maneira eficiente de gerenciar o controle de versão de projetos de software, facilitando a colaboração e o gerenciamento de alterações.

2.4 Instalar o Git WSL (opcional por enquanto, mas é bom conhecer)

Como estudante de programação, é fundamental aprender e utilizar ferramentas que possam ajudar a aprimorar as habilidades e o desenvolvimento de projetos. O Git é uma ferramenta de controle de versão que permite aos usuários rastrear as mudanças feitas em um código ao longo do tempo. Isso é particularmente útil para os estudantes de programação, pois permite que eles acompanhem as mudanças que fizeram em seus projetos, além de ajudá-los a identificar e corrigir erros.

Além disso, o Git é uma ferramenta colaborativa, o que significa que várias pessoas podem trabalhar no mesmo projeto simultaneamente evitando conflitos de versão. Outra razão pela qual é importante usar o Git é que ele fornece um registro completo de todas as alterações feitas em um projeto. Isso é particularmente útil para projetos de longo prazo, pois permite verificar como o projeto evoluiu durante o seu desenvolvimento. Além disso, o Git é uma ferramenta amplamente utilizada na indústria de desenvolvimento de software, o que significa que saber usar o Git é um requisito importante para muitas posições de desenvolvimento de software, e os estudantes que podem demonstrar essa habilidade têm uma vantagem sobre outros candidatos.

Para instalar o Git no WSL (https://git-scm.com/), você pode seguir estes passos:

1. Abra o terminal do WSL.
2. Atualize a lista de pacotes disponíveis para instalação usando o comando:

sudo apt-get update

3. Instale o Git usando o comando:

sudo apt-get install git

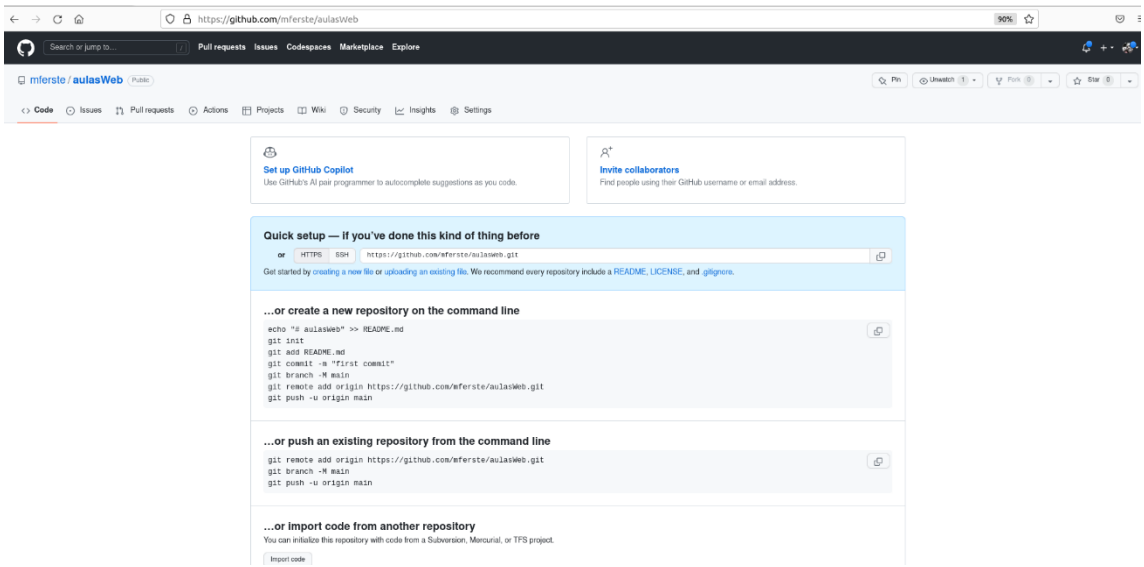
4. Aguarde enquanto o Git é baixado e instalado em sua distribuição do WSL.
5. Depois que a instalação estiver concluída, verifique se o Git foi instalado corretamente executando o comando:

git –version

Se o Git estiver instalado corretamente, você verá a versão do Git que foi instalada em sua distribuição do WSL. Com o Git instalado, você pode começar a usá-lo para gerenciar seus projetos de código fonte e integrá-lo com o VS Code para ter uma experiência de desenvolvimento ainda mais suave.

Não é obrigatório usar o Git por meio dos plugins, mas com o Git poderá acessar o repositório on-line com alguns exemplos.

Figura 5 – Repositório Git



Fonte: Ferste, 2023.

Normalmente a instalação do Git exige alguns conhecimentos para configuração correta, principalmente para configurar usuário e para integrar com o VsCode. Na Figura 5 mostramos passos dessa configuração que envolvem criar uma conta, gerar um token, dar ao VsCode acesso para o Git.

Atualmente precisamos cada vez mais de segurança, por isso, quem desejar configurar o Git para trabalho necessita desse estudo.

Estes detalhes não são necessários para baixar código, somente para quem deseja trabalhar com código, baixar, subir código, enfim, trabalhar plenamente no Git, que é hoje uma ferramenta fundamental para quem quer ser desenvolvedor.

TEMA 3 – ESTRUTURA DE UM PROJETO ANGULAR

A partir de agora trataremos de desenvolvimento. Temos a infraestrutura instalada e vamos para algo mais específico: entender de framework.

Figura 6 – Comparando frameworks, qual usar?

Fonte: Ferste, 2023.

Sempre vão existir divergências entre o melhor e o pior framework para se utilizar. O fato é que não existe o melhor ou pior, todos os frameworks de mercado têm méritos, e muitos. Portanto, aprenda e experimente.

3.1 Primeiramente necessitamos de uma pasta

Vamos, a partir de agora, criar um projeto chamado **Faculdade**, que trata alguns dados de uma faculdade genérica. Precisaremos então de um diretório “faculdade”.

Dentro desse diretório teremos nosso projeto web, que ficará em uma pasta chamada “web” (front-end).

Outra pasta importante será a pasta api que vai conter dados da api ou estrutura que vai nos fornecer os dados (back-end).

Nossa estrutura ficará assim:

faculdade -> raiz

web -> NOSSO DIRETÓRIO DE ESTUDO

api -> nossa api para consumo de dados

Pergunta: Pode ser outro diretório? Pode! Mas saiba que nossas configurações em nossos exemplos refletirão o diretório acima e sua estrutura.

Em tempo: não use espaços, acentos..., pois inviabilizam o funcionamento do sistema, quer ao criar o diretório quer nas pastas anteriores, ou seja, algo como:

João Da Silva / faculdade é bem desaconselhável, pois vai gerar problemas dados os espaços e a acentuação.

3.2 Primeiramente Angular

Primeiramente, verifique se você tem o Node.js instalado em seu computador. Angular CLI é baseado em Node.js e precisa dele para funcionar. Vimos isso no item 1.2 em diante. Em caso de necessidade, revise-o.

Primeiramente precisamos criar um diretório, que chamamos de projetoweb, dividido em dois subdiretórios: api, que conterá depois o projeto de backend (que não faremos, só utilizaremos), e o projeto web, com nosso front-end (Angular, 2023).

Acessando o nosso projeto, que estamos propondo como sendo:

```
projetoweb/api
projetoweb/web
```

Saiba mais

Desenvolver significa utilizar comandos de prompt e ações mais técnicas com as quais devemos nos acostumar para configurar o sistema.

Figura 7 – Comandos para listar diretório de trabalho e acesso a ele

Fonte: Ferste, 2023.

Basicamente agora temos de criar nosso projeto, com o comando abaixo:

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

Onde:

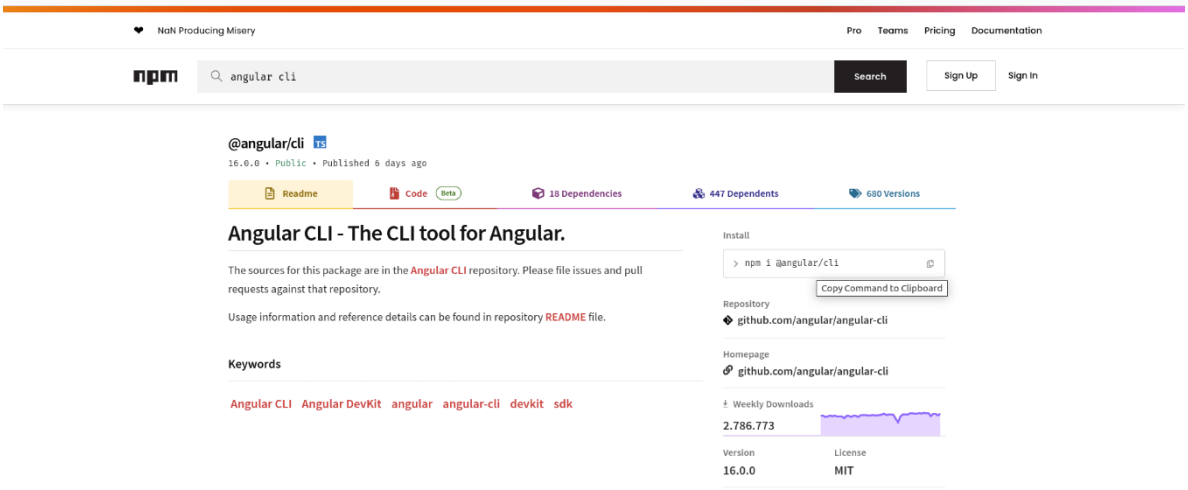
npm deve estar disponível

-g indica a instalação em nível global para todos os projetos

@angular/cli -> indica o cliente Angular, é o nosso framework

Se tem dúvidas, verifique no sítio do npmjs, o pacote e mais configurações em https://www.npmjs.com/.

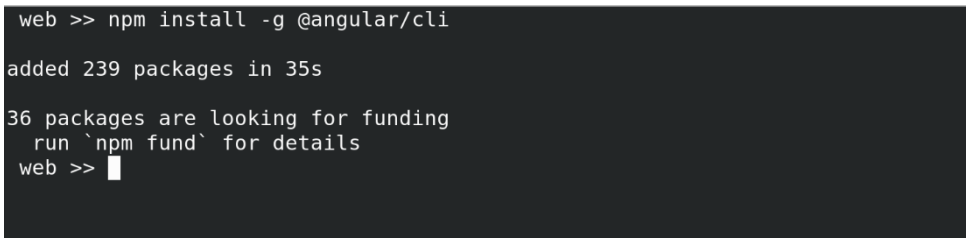
Figura 8 – Sítio do npm mostrando pesquisa do Angular Cli e seu uso



Fonte: Ferste, 2023.

Ok, então após executar o comando temos uma saída próxima da mostrada abaixo:

Figura 9 – Instalação do cliente angular



Fonte: Ferste, 2023.

Aqui podemos dizer que está instalado. Temos sempre alguns comandos para testar e a partir de agora podemos executar os comandos ng. Tudo o que faremos de agora em diante de uma forma ou outra vai tratar com algum comando Angular. Esse comando é o ng. Dois exemplos iniciais seriam o de versão e ajuda, respectivamente:

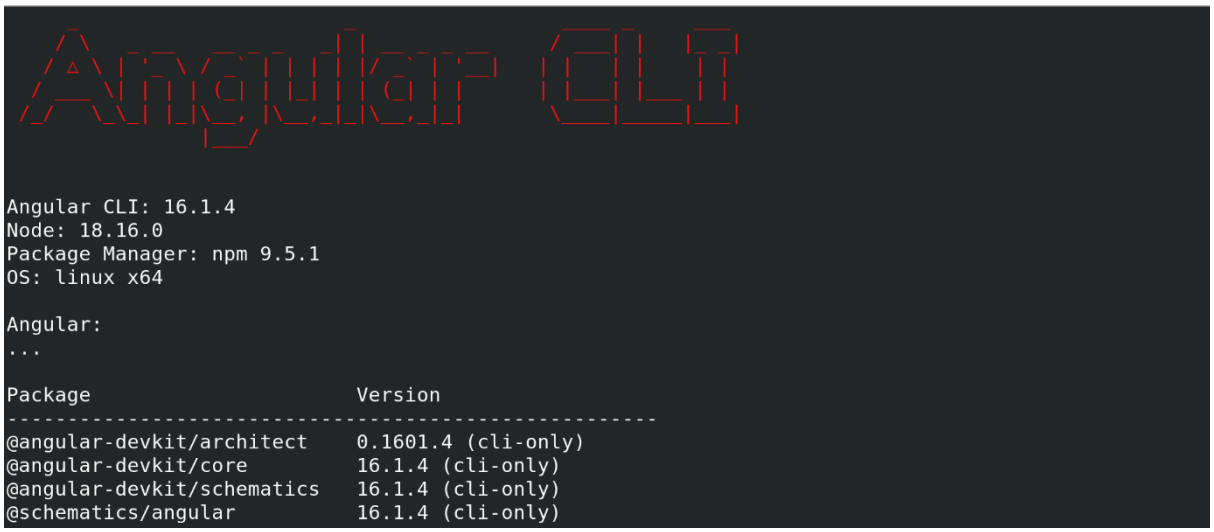
```
ng version
```

e

```
ng help
```

Na Figura 10 vemos então a saída do comando ng version.

Figura 10 – Teste do @angular cli, com ng version



Fonte: Ferste, 2023.

Agora está configurada a infraestrutura. Se chegou até aqui, tem o Angular instalado e estamos prontos para iniciar nosso projeto. Falta iniciar o entendimento da estrutura do projeto. Como falamos, nosso estudo versa sobre a estrutura web. O projeto api será um pouco detalhado nas unidades seguintes, mas não o construiremos. Vamos então para o projeto web:

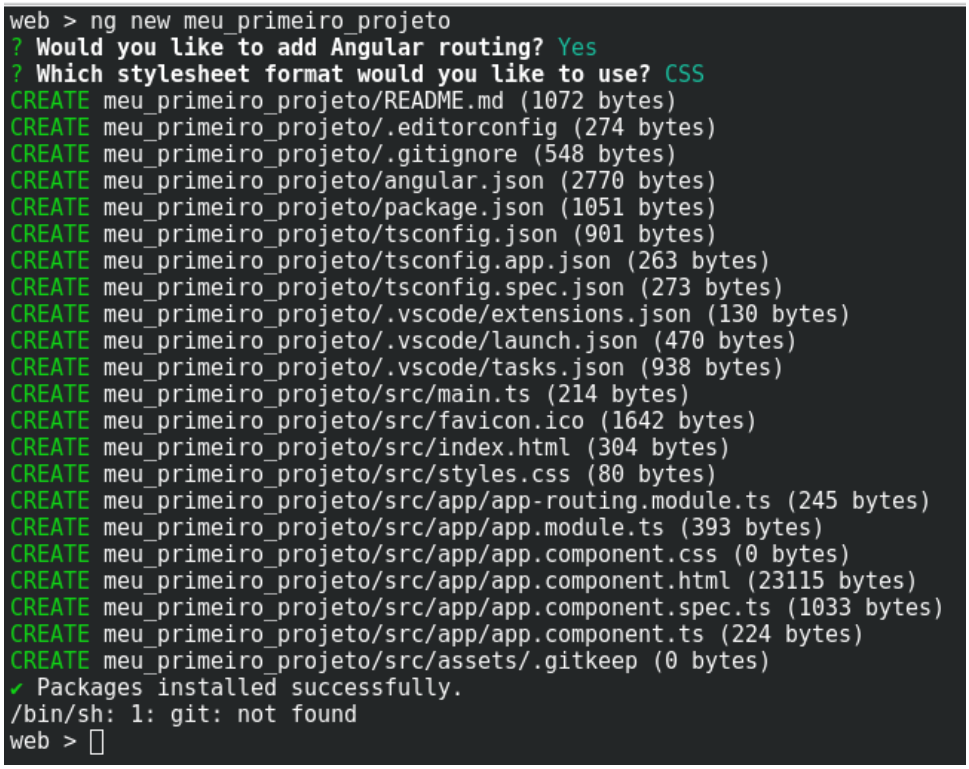
TEMA 4 – CRIAÇÃO DE UM COMPONENTE E EXECUÇÃO DE UM OLÁ MUNDO – ESTRUTURA DE PROJETO

Agora vamos ver uma visão geral da estrutura básica de um projeto Angular. Dependendo das necessidades do projeto, essa estrutura pode ser personalizada e expandida, mas basicamente podemos dizer que existe um padrão que sempre vai existir (Angular, 2023).

4.1 ng new

Com ng new criamos um projeto. O primeiro passo para nosso projeto Angular segue na Figura 11.

Figura 11 – Criação do primeiro projeto com comando ng new meu_primeiro_projeto



Fonte: Ferste, 2023.

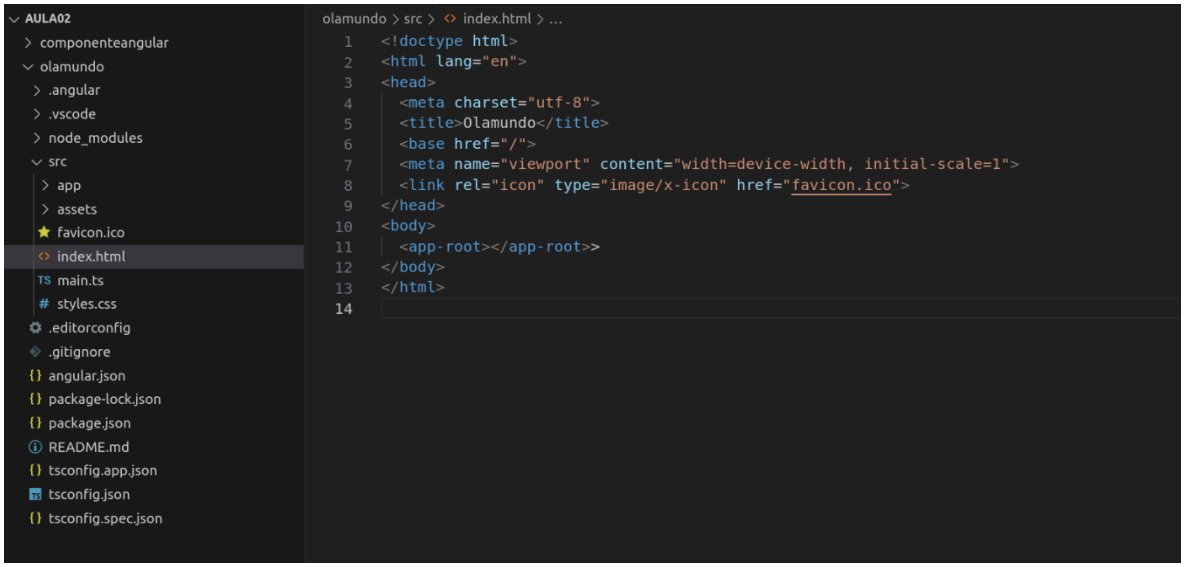
Analisando a Figura 11, temos esta configuração de diretórios ou arquivos:

- **node_modules/**: esta pasta contém todas as dependências instaladas do projeto.
- **src/: esta pasta é o diretório raiz do código-fonte do projeto.**
  - **app/**: esta pasta contém todos os componentes, módulos, serviços, diretivas e pipes da aplicação.
    - **app.component.ts**: o componente principal da aplicação.
    - **app.component.html**: o template HTML do componente principal da aplicação.
    - **app.component.css**: o arquivo de estilo CSS do componente principal da aplicação.

        - **app.module.ts**: o módulo principal da aplicação que importa e exporta os componentes, módulos e serviços.
    - **assets/**: esta pasta contém arquivos estáticos como imagens, fontes e arquivos JSON.
    - **environments/**: esta pasta contém arquivos de configuração de ambiente para diferentes ambientes de compilação, como desenvolvimento e produção.
    - **index.html**: o arquivo HTML principal que é carregado quando a aplicação é iniciada.
    - **main.ts**: o arquivo TypeScript que inicia a aplicação.
    - **polyfills.ts**: o arquivo TypeScript que carrega os polyfills necessários para executar a aplicação em diferentes navegadores.
- **angular.json:** o arquivo de configuração do Angular que define as configurações do projeto, como o diretório de saída, as configurações de compilação, entre outras.
- **package.json:** o arquivo de configuração do projeto que define as dependências do projeto e as configurações de script.
- **tsconfig.json:** o arquivo de configuração do TypeScript que define as opções do compilador TypeScript.
- **tslint.json:** o arquivo de configuração do TSLint que define as regras do TypeScript.

Detalhando alguns pontos bem específicos, veja a figura 12:

Figura 12 – Estrutura de um projeto Angular vista no VsCode



Fonte: Ferste, 2023.

pasta scr: onde todo o código do aplicativo Angular é mantido. A pasta app contém a maior parte do código do aplicativo, incluindo componentes, serviços, modelos, diretivas, pipes, interfaces, enums e páginas. A pasta assets contém arquivos como imagens, ícones e outros recursos usados pelo aplicativo. A pasta environments contém os arquivos de configuração para diferentes ambientes (desenvolvimento, produção etc.). O arquivo index.html é o ponto de entrada do aplicativo.

O arquivo package.json contém as dependências do projeto e scripts para executar o projeto.

package.json é um arquivo usado em projetos Node.js para definir as informações do projeto, as dependências e scripts necessários para executar o projeto. O arquivo está localizado na raiz do diretório do projeto Node.js e é usado pelo gerenciador de pacotes npm para instalar as dependências do projeto e executar os scripts definidos.

Aqui estão algumas das principais seções e propriedades definidas no arquivo.

package.json:

- **name**: o nome do projeto.
- **version**: a versão do projeto.
- **description**: uma descrição breve do projeto.
- **main**: o arquivo principal que será executado quando o projeto for iniciado.
- **dependencies**: as dependências necessárias para executar o projeto.
- **devDependencies**: as dependências necessárias apenas para o desenvolvimento do projeto.
- **scripts**: os scripts que podem ser executados para executar tarefas específicas, como iniciar o servidor, executar testes ou construir o projeto.
- **author**: o nome do autor do projeto.
- **license**: a licença do projeto.
- **repository**: a URL do repositório do projeto.

Parte importante de qualquer projeto Node.js e é usado para definir as informações e dependências necessárias para executar e desenvolver o projeto.

Ele também permite que outras pessoas baixem e instalem as dependências do projeto de forma fácil e rápida.

Saiba mais

A configuração dos arquivos de Typescript veremos em conteúdos posteriores. Por enquanto vamos usar sem mudar a configuração.

angular.json: é um arquivo de configuração usado em aplicações Angular para definir várias configurações relacionadas ao projeto, como opções de build, ativos de aplicação e outras opções de configuração.

O arquivo está localizado na raiz do diretório do projeto Angular e é usado pelo Angular CLI para construir e servir a aplicação.

Aqui estão algumas das principais seções e propriedades definidas no arquivo **angular.json**:

- **projects**: esta seção define os projetos no workspace e suas configurações. Por exemplo, pode definir o nome do projeto, sua pasta raiz, as opções do arquiteto e as opções de build.
- **architect**: esta seção define os vários comandos que podem ser executados usando o Angular CLI. Por exemplo, pode definir o comando de build, o comando de serve e o comando de test, juntamente com suas respectivas configurações.
- **build**: esta seção define as opções para construir o projeto. Por exemplo, pode definir o caminho de saída, as opções de mapa de origem e as substituições de arquivo.
- **serve**: esta seção define as opções para servir o projeto. Por exemplo, pode definir o número da porta, as opções SSL e as configurações de proxy.
- **test**: esta seção define as opções para executar testes no projeto. Por exemplo, pode definir as opções de cobertura de código, as opções do framework de teste e os padrões de arquivo.
- **lint**: esta seção define as opções para executar ferramentas de análise de código no projeto. Por exemplo, pode definir as regras de linting, os padrões de arquivo de linting e os arquivos de configuração de linting.

Em geral, o arquivo **angular.json** fornece um local central para configurar todas as opções relacionadas a um projeto Angular, tornando mais fácil gerenciar e manter o projeto.

app.component é um componente principal em uma aplicação Angular. Ele é criado automaticamente pelo Angular CLI ao gerar um novo projeto.

Esse componente é responsável por definir a estrutura básica da aplicação e é geralmente composto dos seguintes elementos:

- **template**: o template HTML que define o layout da aplicação.
- **style**: o arquivo CSS que define o estilo da aplicação.
- **class**: a classe do componente que define o comportamento da aplicação.

O **app.component** pode ser usado como o ponto de entrada da aplicação, controlando a navegação entre as diferentes páginas e componentes. Ele pode ser personalizado de acordo com as necessidades da aplicação, adicionando elementos como menus de navegação, barras de pesquisa e muito mais.

Além disso, o **app.component** também pode ser usado para gerenciar dados da aplicação, como a autenticação do usuário e informações de sessão, bem como para se comunicar com outros componentes da aplicação por meio do serviço de comunicação de eventos do Angular (Angular, 2023).

TEMA 5 – CRIAÇÃO DE UM COMPONENTE E EXECUÇÃO DE UM OLÁ MUNDO – ESTRUTURA DE COMPONENTE

O termo *componente* tem sido amplamente utilizado em diversas áreas da programação e pode ter significados ligeiramente diferentes dependendo do contexto. No entanto, em geral, um componente pode ser definido como uma unidade modular e independente de código que pode ser reutilizada e combinada com outros componentes para criar sistemas mais complexos (Goulão, 2008).

Em programação orientada a objetos, um componente pode ser uma classe que encapsula um conjunto específico de funcionalidades e propriedades, permitindo que ele seja reutilizado em diferentes partes de um sistema. Em outras áreas, como no desenvolvimento web, um componente pode se referir a um conjunto de arquivos HTML, CSS e JavaScript que definem um elemento de interface do usuário, como um botão ou um formulário.

Independentemente do contexto, a ideia fundamental por trás do conceito de componente é a modularidade e a reutilização de código, permitindo que os

desenvolvedores criem sistemas mais complexos e sofisticados com mais facilidade.

5.1 Componente em Angular

Em Angular, um componente é uma das principais estruturas para construir interfaces de usuário. Ele é uma classe que representa um elemento da interface do usuário e inclui informações sobre como esse elemento deve ser renderizado e comportar-se em resposta a eventos.

Os componentes em Angular são compostos de três partes principais: uma classe que define o comportamento do componente, um template que define como o componente deve ser renderizado e um arquivo de metadados que fornece informações adicionais sobre o componente, como seu seletor e seus estilos.

Os componentes em Angular permitem que os desenvolvedores criem interfaces de usuário modulares e reutilizáveis, o que significa que os componentes podem ser usados em diferentes partes de um aplicativo ou até mesmo em diferentes aplicativos. Eles também fornecem recursos avançados, como injeção de dependência, ciclos de vida do componente e comunicação entre componentes, permitindo que os desenvolvedores criem aplicativos mais sofisticados e escaláveis.

Saiba mais

Temos de iniciar por algum ponto. Na criação de projeto, o entendimento da arquitetura e da estrutura vem antes de alguns detalhamentos sobre arquivos ts (Typescript), eventos (onclick) que serão mais bem detalhados em conteúdos posteriores, ou seja, o objetivo aqui é praticar os comandos do angular e não alterar o código além do mostrado.

Primeiramente criamos um projeto, em um diretório fazemos o comando:

```
ng new olamundo
```

Então entramos no diretório “olamundo” para criar um componente.

Para criar um componente em Angular, siga estes passos:

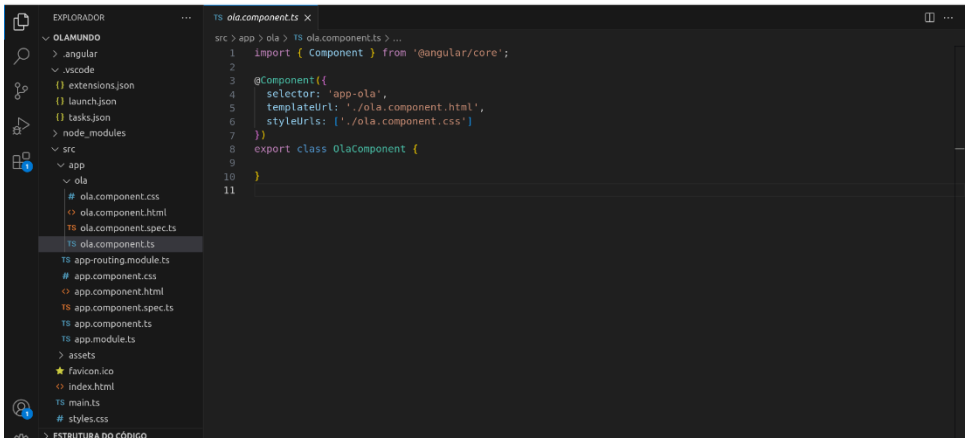
1. Abra o terminal e navegue até o diretório onde deseja criar o componente.

2. Execute o comando **ng generate component nome-do-componente** (ou **ng g c nome-do-componente** para abreviar) para gerar automaticamente os arquivos necessários para o componente. No caso, nosso componente chama-se “ola” e fizemos:

```
ng generate component ola
```

Com isso criamos um componente conforme é mostrado na figura 13:

Figura 13 – Estrutura de um projeto Angular vista no VsCode



Fonte: Ferste, 2023.

3. Conforme visto foi criado um novo diretório com o nome do componente dentro da pasta “app” do seu projeto Angular. Dentro desse diretório, você encontrará os arquivos do componente, incluindo um arquivo TypeScript **.ts**, um arquivo HTML **.html**, um arquivo CSS **.css** e um arquivo de metadados**. Spec.ts**.
4. Abra o **ola.componet.ts** do componente e defina a lógica do componente, incluindo as propriedades, os métodos e eventos necessários. Veja o exemplo do componente que foi criado automaticamente:

```
import { Component } from '@angular/core';


@Component({
selector: 'app-ola',
templateUrl: './ola.component.html',
styleUrls: ['./ola.component.css']
})
export class OlaComponent {
}
```

Abra o ola.component**.html** do componente e defina o template do componente, como abaixo:

```
<p>Ola mundo !!!!</p>
```

Opcionalmente, você também pode definir estilos para o componente no arquivo **.css**.

Finalmente, use o seletor do componente **app-nome-do-componente** em outros componentes ou no arquivo **app.component.html** para incluir o componente na sua aplicação. Por exemplo, temos o seguinte:

```
<app-nome-do-componente> </app-nome-do-componente>
```

Isso é tudo! Agora você criou um componente em Angular e pode usá-lo em sua aplicação.

Veremos então na próxima seção a criação de um componente Olá Mundo passo a passo.

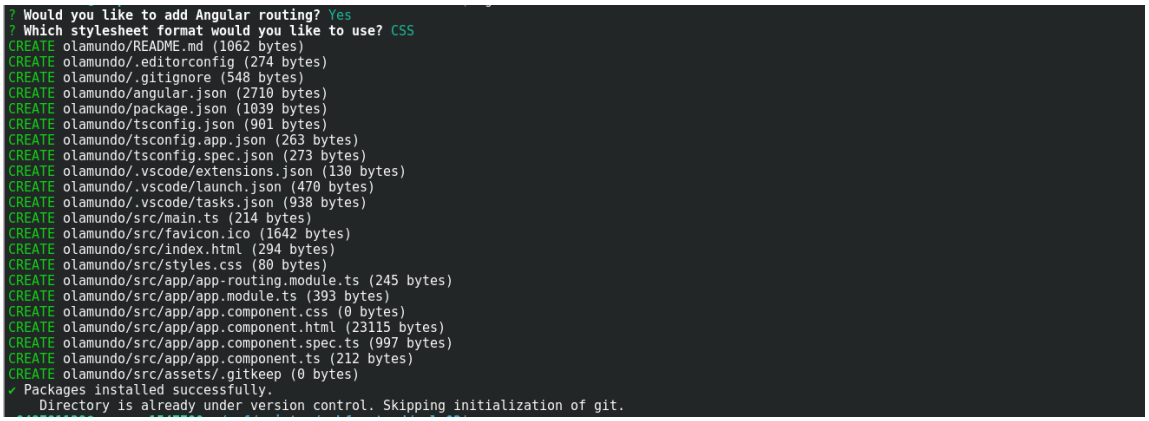
5.2 Criando o nosso Olá Mundo

Vamos agora reafirmar passo a passo nosso projeto.

```
ng new olamundo
```

- Responda com Y para Angular routing
- Responda com <enter> para assumir a primeira opção CSS para estilo.
- Note no fim do comando que se existe o git configurado. O comando npm já indica que está “under version control” ou seja, seu código vai estar protegido em seu Git, não é necessário para executar o projeto.

Figura 14 – Outro exemplo de criação de projeto



Fonte: Ferste, 2023.

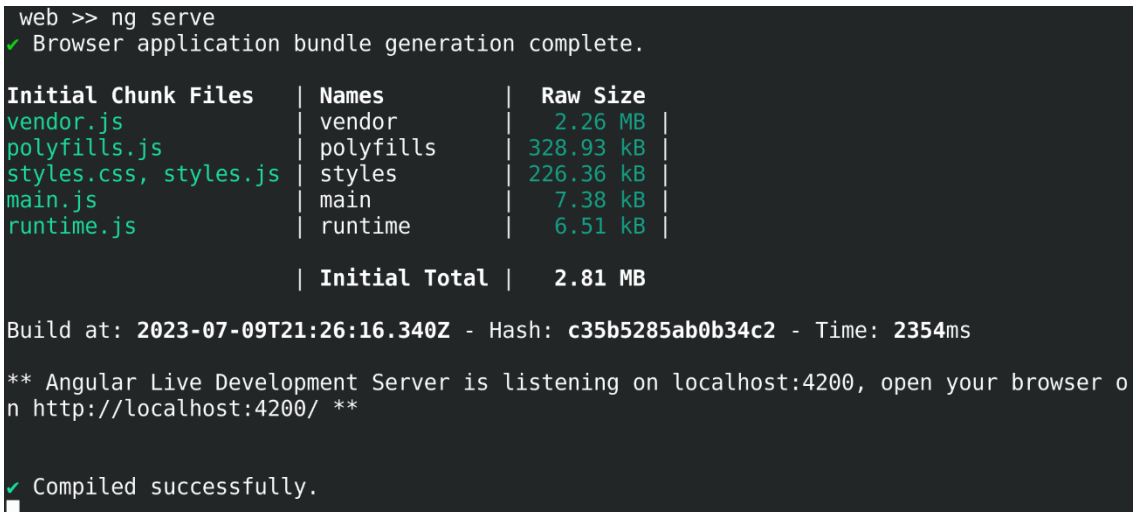
Depois fazer:

```
cd olamundo
```

```
ng serve
```

Este comando é importantíssimo, pois cria uma instância do node que sobe o seu projeto Angular. Para executar temos a saída na Figura 16.

Figura 15 –Execução do ng serve

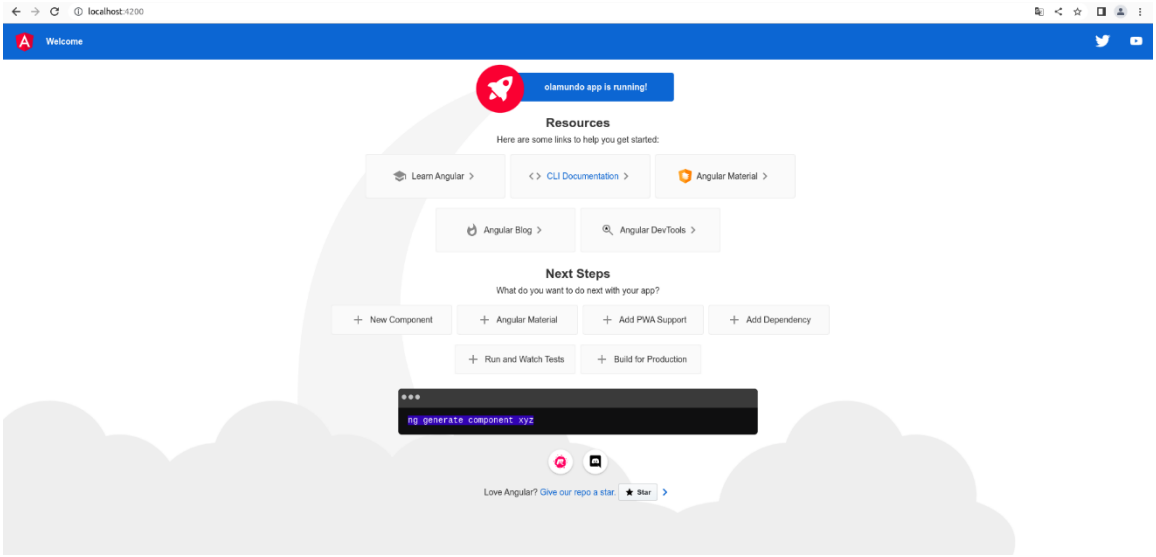


Fonte: Ferste, 2023.

Agora o projeto pode ser visualizado no seu browser, digitando a url l**ocalhost:4200,** importantíssimo, pois podemos ver nosso projeto. O template inicial do Angular cria uma página similar à página a seguir. Essa página significa um objetivo atingido, pois sua configuração está correta e seu projeto funcionou.

Ela muda muito entre as diferentes versões do Angular, mas sempre vai trazer alguns links interessantes para navegar.

Figura 16 – Execução de um projeto angular, tela inicial



Fonte: Ferste, 2023.

Dois pontos importantes: o prompt que foi usado (cmd), ou seja, o terminal fica travado e, se fechado ou se executar **<Ctrl> + <C>,** ele encerra automaticamente a execução do servidor. Interessante manter esse terminal para manter a execução de sua instância do Angular. Existem formas alternativas que mostraremos depois.

Segundo ponto: sempre quando atualizado, o código automaticamente gera uma nova versão de sítio, portanto código pela metade implica que o sítio fique quebrado.

Relembrando a criação de nosso componente, conforme a figura18:

```
ng generate component ola
```

Figura 17 – Criação de um componente

```
CREATE src/app/ola/ola.component.css (0 bytes)
CREATE src/app/ola/ola.component.html (18 bytes)
CREATE src/app/ola/ola.component.spec.ts (538 bytes)
CREATE src/app/ola/ola.component.ts (190 bytes)
UPDATE src/app/app.module.ts (463 bytes)
```

Fonte: Ferste, 2023.

Vamos trocar então o código da tela inicial pelo nosso novo componente.

Importante ressaltar que três novos arquivos foram criados dentro de um diretório chamado “ola”:

- ola.component.css : usado para estilo
- ola.component.html: nossa view onde construímos nosso componente; lá simplesmente já existe um simples comando

ola works!

Vamos deixar desta forma.

- Ola.component.ts, conforme abaixo:

```
import { Component } from '@angular/core';
@Component({
selector: 'app-ola',
templateUrl: './ola.component.html',
styleUrls: ['./ola.component.css']
})
export class OlaComponent {


}
```

Importante mencionar que:

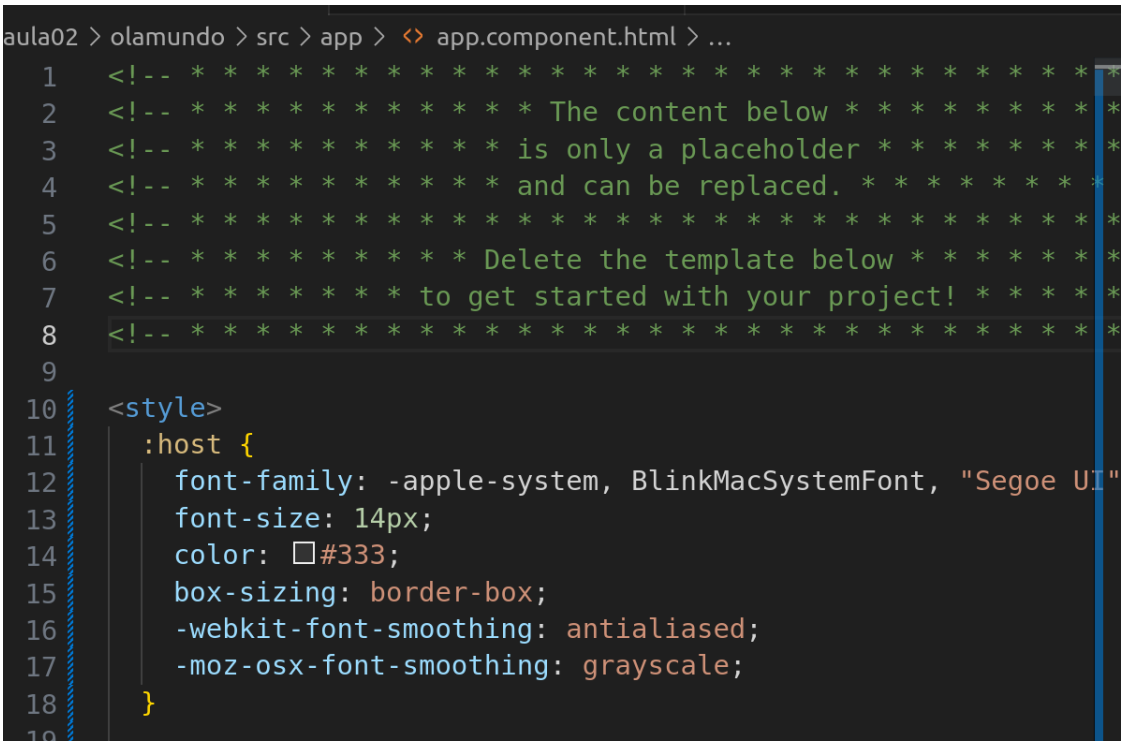
- selector: indica o nome pelo qual o componente será chamado
- TemplateUrl: é a amarração com nosso arquivo HTML
- StyleUrls: é o nosso arquivo de estilo, mencionado acima

Ainda temos a cláusula de export, utilizada para declarar que o componente é passível de ser usado por outro componente. Basta fazer import quando necessário, o que veremos mais para a frente.

Agora vamos colocar nosso componente para funcionar!

Como já foi criado, basta utilizá-lo. É sugerido pelo próprio Angular mudar a página inicial como exercício, ou seja, a página inicial vista na Figura 18 é montada por meio do arquivo app.component.html:

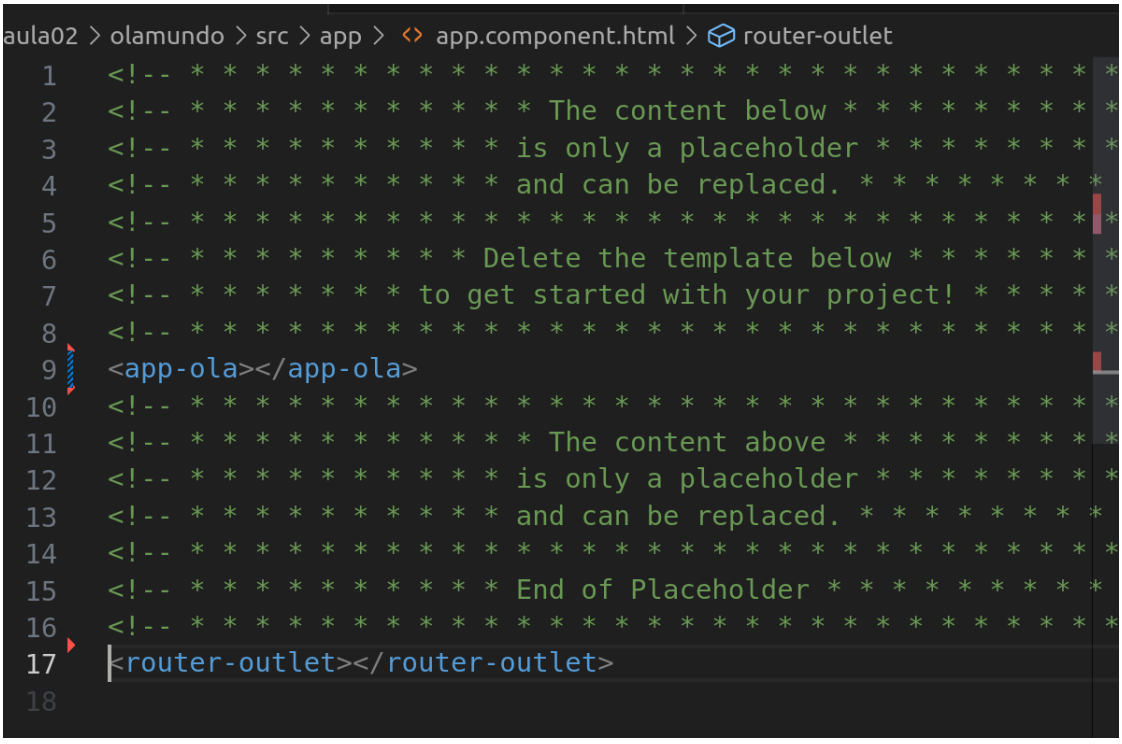
Figura 18 – Tela original do app.component.html que deve ser modificada



Fonte: Ferste, 2023.

Basta trocar o conteúdo para o novo componente, conforme abaixo, na figura 19:

Figura 19 – app.component modificado



Fonte: Ferste, 2023.

Deve ser mantido o **router-outlet,** que é usado no template do componente raiz ou em um componente pai que serve como contêiner para as rotas. Quando a rota é ativada, o componente correspondente é renderizado dentro do **router-outlet**. Isso permite que o Angular exiba diferentes componentes com base nas rotas definidas na aplicação. Deve então ficar a tela de sua aplicação como na figura 20 abaixo, lembrando que é somente um teste para seu primeiro componente.

Figura 20 – Execução do primeiro programa

Fonte: Ferste, 2023.

FINALIZANDO

Aprender a infraestrutura do Angular é fundamental para desenvolvedores, pois permite aproveitar ao máximo os recursos e as funcionalidades do framework. Isso resulta em um desenvolvimento mais eficiente, com código organizado e modular, facilitando a manutenção e evolução dos aplicativos. Além disso, o conhecimento da infraestrutura do Angular possibilita otimizar o desempenho dos aplicativos, integrar bibliotecas e ferramentas populares e fazer parte de uma comunidade ativa que oferece suporte e recursos adicionais. Em resumo, dominar a infraestrutura do Angular é essencial para criar aplicativos de alta qualidade e aproveitar todas as vantagens que o framework tem a oferecer.

O primeiro componente em um projeto Angular é crucial para a estruturação básica do aplicativo. Ele define a estrutura inicial do projeto e serve como ponto de partida para o desenvolvimento de outros componentes. O primeiro componente lida com as rotas iniciais, define a exibição inicial do

aplicativo e permite a injeção de dependências. Além disso, ajuda a estabelecer a estrutura e os padrões de desenvolvimento do projeto e pode ser usado como ponto de partida para escrever testes automatizados. Em suma, o primeiro componente é fundamental para iniciar o desenvolvimento de um aplicativo Angular bem estruturado e funcional.

REFERÊNCIAS

ANGULAR. Disponível em: <https://www.angular.io>. Acesso em: 12 maio 2023.

APACHE. Disponível em: <https://struts.apache.org/>. Acesso em: 10 maio 2023.

BALSAMIQ. Disponível em: <https://balsamiq.com/>. Acesso em: 10 maio 2023.

BETRYBE. Disponível em: <https://blog.betrybe.com/framework-de-programacao/angular/>. Acesso em: 10 maio 2023.

BIANCH, L. **O que é aplicação, framework e linguagem de programação?** Qual a diferença? O que e qual devo escolher quando estou iniciando minha carreira? Disponível em: <https://pt.linkedin.com/pulse/o-que-%C3%A9-aplica%C3%A7%C3%A3o-framework-e-linguagem-de-qual-diferen%C3%A7a-bianch>. Acesso em: 18 abr. 2023.

BOOTSTRAP. Disponível em: <https://getbootstrap.com/>. Acesso em: 10 maio 2023.

DJANGO. Disponível em: <https://www.djangoproject.com/>. Acesso em: 10 maio 2023.

FRAIN, B. **Responsive web design with HTML5 and CSS**. Build future-proof responsive websites using the latest HTML5 and CSS techniques. 4.ed. Birmingham: Packt, 2022.

FIGMA. Disponível em: <https://www.figma.com/>. Acesso em: 10 maio 2023.

GENEXUS. Disponível em : <https://training.genexus.com/pt/aprendizagem/pdf/arquitetura-de-aplicativo-angular-pdf>. Acesso em: 10 maio 2023.

GIT. Disponível em: <https://git-scm.com/>. Acesso em: 10 maio 2023.

GOULÃO, M. **Component-based software engineering**: a quantitative approach. Lisboa: Universidade Nova de Lisboa, 2008.

INSOMNIA. Disponível em: <https://insomnia.rest/>. Acesso em: 10 maio 2023.

LARAVEL. Disponível em: <https://laravel.com/>. Acesso em: 10 maio 2023.

LEARN. Disponível em: <https://learn.microsoft.com/pt-br/windows/wsl/tutorials/wsl-vscode>. Acesso em: 11 maio 2023.

NODEJS. Disponível em: <https://nodejs.org/en>. Acesso em: 10 maio 2023.

NORMAN, D. A. **Design for a better world**: meaningful, sustainable, humanity centered. Cambridge, Massachusetts: The MIT Press, 2023.

NPM. Node Package Manager. Disponível em: <https://www.npmjs.com/>. Acesso em: 12 maio 2023.

REACT. Disponível em: <https://www.react.dev/>. Acesso em: 12 maio 2023.

SPRING. Disponível em: <https://spring.io/>. Acesso em: 10 maio 2023.

SKETCH. Disponível em: <https://www.sketch.com/>. Acesso em: 10 maio 2023.

SCHMIDT, D. C.; WALLNAU, K. **Component-based software engineering**: putting the pieces together. Londres: Addison Wesley, 2001.

TSCONFIG. Disponível em: <https://www.typescriptlang.org/docs/handbook/tsconfig-json.html>. Acesso em: 13 maio 2023.

TYPESCRIPT. Disponível em: <https://www.typescriptlang.org/> Acesso em: 14 maio 2023.

VUEJS. Disponível em: <https://vuejs.org/>. Acesso em: 10 maio 2023.

VSC. Disponível em: <https://code.visualstudio.com/>. Acesso em: 10 maio 2023.

WEBFOUNDATION. World Wide Web Foundation. Disponível em: <https://webfoundation.org/>. Acesso em: 12 maio 2023.

W3C. Disponível em: <https://www.w3schools.in/mvc-architecture>. Acesso em: 10 maio 2023.