

**WORKSHOP ANGULAR 7**

**WORKSHOP ANGULAR 7**

# INTRODUÇÃO

Angular é uma plataforma de aplicações web de código-fonte aberto e front-end baseado em TypeScript liderado pela Equipe Angular do Google e por uma comunidade de indivíduos e corporações. Angular é uma reescrita completa do AngularJS, feito pela mesma equipe que o construiu.

## **Informações:**

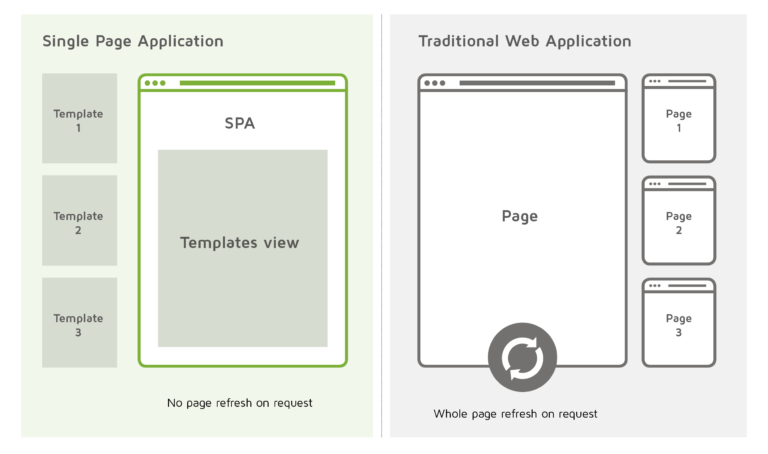
* Gravado em: TypeScript
* Plataforma: Web platform
* Lançamento: 14 setembro 2016; há 2 anos
* Versão estável: 7.1.4 (18 dezembro 2018; há 3 mês)
* Versão em teste: 7.2.0-beta.2 (11 dezembro 2018; há 4 meses)
* Desenvolvedor: Google/ Misko Hevery

# SPA – SINGLE PAGE APPLICATION

Antes de entendermos tudo sobre Angular, é muito importante entendermos o que significa uma Aplicação Single Page para o Angular.

Um dos melhores conceitos que o Angular oferece é o de “Single Page”, onde os recursos apropriados são dinamicamente carregados e adicionados à página, conforme necessário, geralmente em resposta a ações do usuário.

Para isto acontecer o framework oferece módulos que te possibilitam ter apenas uma página index, com outras páginas de conteúdo (views) sendo carregadas de acordo com uma específica rota (route).



# Diferença entre os Framworks Angular

## AngularJS :

AngularJS é um framework JavaScript onde ele estende atributos HTML com diretivas e vincula dados a HTML com expressões. A sua incorporação no projeto pode ser feita através de um import a uma página HTML com uma tag <script>.

Por se tratar de um framework é válido destacar que seu core é totalmente escrito em JavaScript.

## Angular:

Angular foi reescrito passando a ser um framework diferente e não uma evolução. A princípio, o propósito dessa reescrita foi adequar o framework Angular para uso de padrões web atuais, a utilização de web componentes e uma melhor integração com o HTML 5, CSS 3 e Javascript.

### Por que foi criado o Angular 2 e Quais as diferenças entre o AngularJS e o Angular 2?

O Angular 1.x, por se tratar de um framework incorporado, ele precisa esperar pelo DOM para “carregar”, e se incluir na aplicação. Já o Angular 2 é totalmente diferente, ele tem o poder total no controle dos templates e consegue fazer todas as mudanças necessárias e antes mesmo delas serem alcançadas pelo DOM.

Diferente do angularJS, é utilizado Typescript em sua utilização e por isso obrigatoriamente em Angular 2 e suas versões em diante devem ser interpretados por um servidor de aplicação.

# Principais melhorias das Evoluções

## Angular v4

**Por que mudaram do angular 2 para o 4?**

Internamente no Angular, existem diversos componentes e cada componente possui a sua versão por terem passados por pequenas atualizações sem alterar a estrutura do framework.

O componente Router por sua vez já estava na versão 3.5 devido as suas atualizações e por conta deste componente, a próxima versão do framework, foi denominada como 4 subindo todos os componentes para versão 4 também para não haver confusões, principalmente por compatibilidade.

* Novo website disponibilizando uma documentação rica do framework
* Inclusão do HttpClient

## Angular v5

* Bundle, menor e mais rápido
* Suporte oficial para PWA

## Angular v6

* + - Angular Elements
    - Juntamente com o CLI, novos comando como ng update e ng add
    - Update.angular.io – guia de update de versões (Versões de para, complexibilidade, npm ou yarn)

## Angular v7

### Melhor performance

Uma grande utilidade de uma aplicação SPA é o download completo da aplicação no primeiro acesso para reduzir as taxas de downloads por navegação. Isso pode ser um problema quando lembramos que grande parte do tráfego na internet vem de dispositivos móveis que, em geral, costumam acessar a internet em baixa velocidade de conexão.

|  |
| --- |
| Para mantermos o tamanho da nossa aplicação sobre controle,**o Angular 7 conta com a função de Bundle Budgets**. Isto é, quando construímos nossa aplicação, recebemos alertas quando o seu tamanho excede um limite pré-determinado. Este recurso já existia no Angular e agora se tornou padrão. Na construção de uma aplicação que ultrapasse 2 MB receberemos um alerta e em aplicações que ultrapassem 5 MB seremos impedidos de prosseguir com a construção. É claro que esses valores são personalizáveis, e podemos modificá-los no arquivo **angular.json** do nosso projeto:   * **maximumWarning:** Tamanho da aplicação que, se excedido, disparará um alerta. * **maximumError:** Tamanho da aplicação que, se excedido, causará um erro e interromperá a construção. |

### Dependências

Algumas dependências utilizadas pelo Angular também foram atualizadas:

* **TypeScript:** A versão do TypeScript, o Superset do JavaScript utilizado pelo Angular foi atualizada para a versão 3.1, que conta apenas com algumas pequenas mudanças em relação a declaração de propriedades em funções.
* **RxJS:** O RxJS, a biblioteca que provem ao Angular, dentre outras coisas, Observables, foi atualizada para a versão 6.3. Essa nova versão implementou apenas correções.
* NodeJS: Agora é possível utilizar o Angular e o Angular CLI na versão 10 do Node.js, embora ainda seja totalmente compatível com a versão 8.

### Angular CLI

A ferramenta de linha de comando também mudou. Agora, sempre que criamos um novo projeto, o CLI pergunta se desejamos adicionar o módulo de rotas e se desejamos utilizar algum processador CSS, como SASS ou LESS. Veja a seguir o processo de criação de uma aplicação.

## Resumindo a evolução

Após a entrada do angular 2 no mercado e a sua boa reputação, a google não perdeu tempo e foi lançando atualizações do framework corrigindo bugs com a versão 4 principalmente.

Desde o Angular 2 era utilizado uma biblioteca HTTP para que seja possível a aplicação realizar requisições externas via protocolo http. Porém nas novas versões, foi incluído o HttpClient que herda da biblioteca Http mas possui alguns atributos a mais que facilitam o desenvolvimento de um cliente, um exemplo é o suporte para tipos JSON de forma nativa em sua implementação no corpo da resposta de uma requisição.

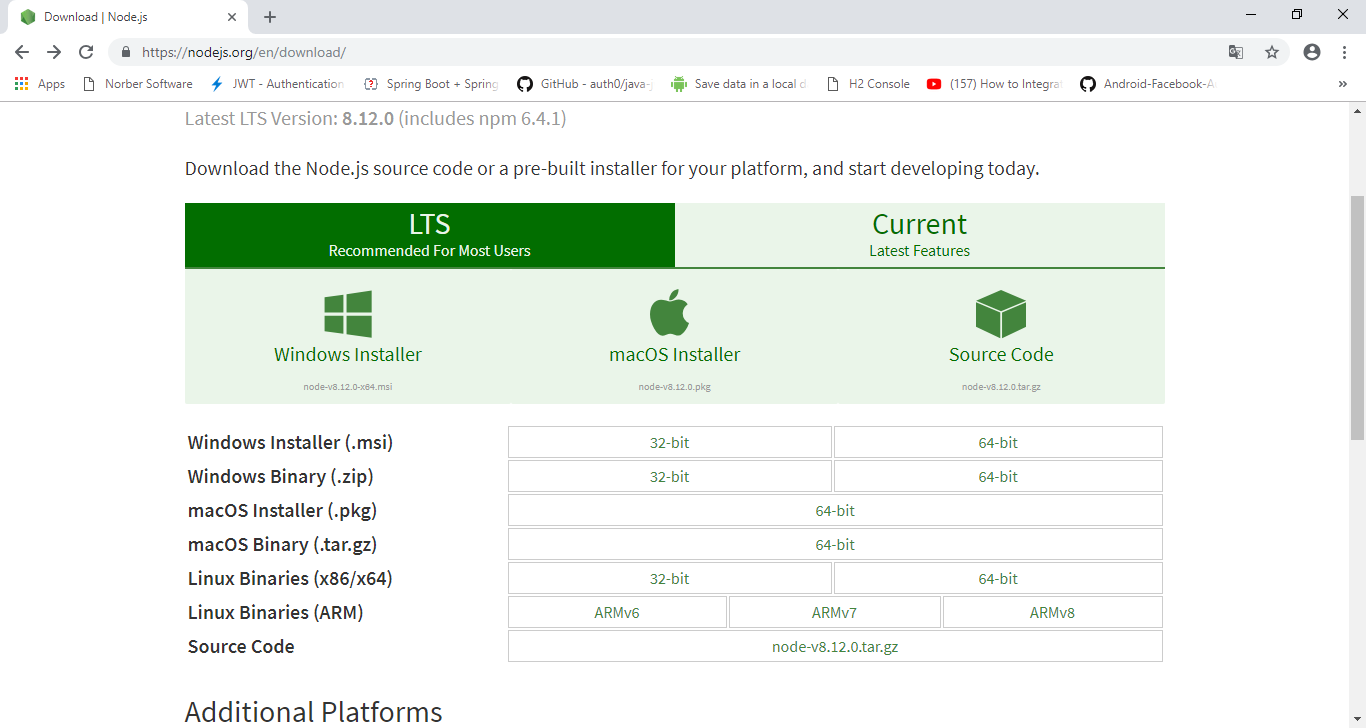
# CONFIGURANDO AMBIENTE

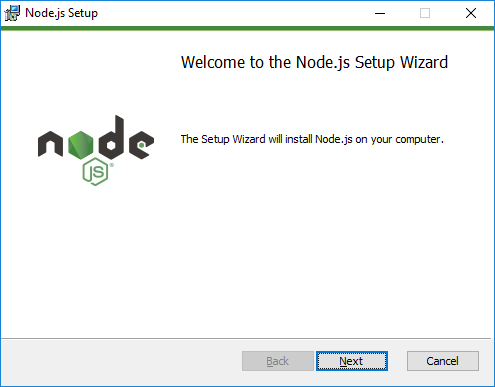
## NodeJS – Servidor de Aplicação

O NodeJS é um interpretador de código JavaScript com o código aberto, focado em migrar o Javascript do lado do cliente para servidores.

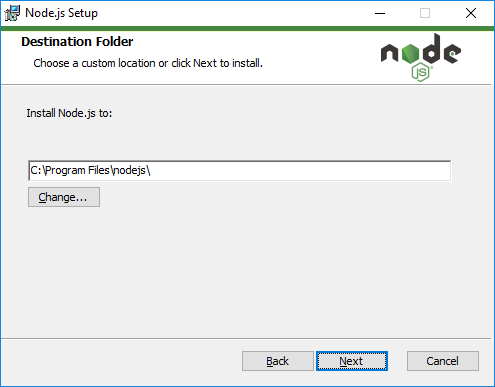
Acesse o link abaixo para realizar o download do Node:

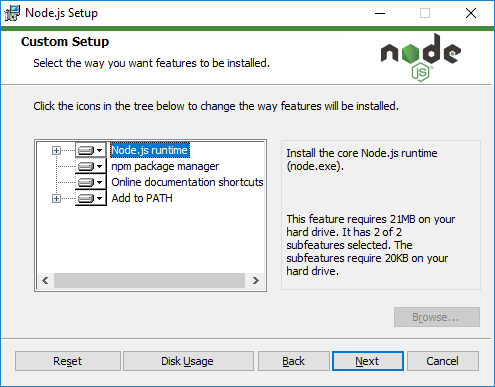
<https://nodejs.org/en/download>

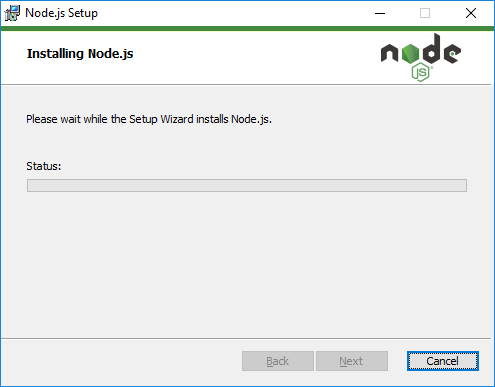


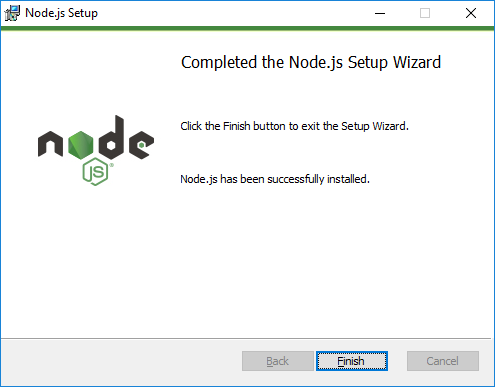






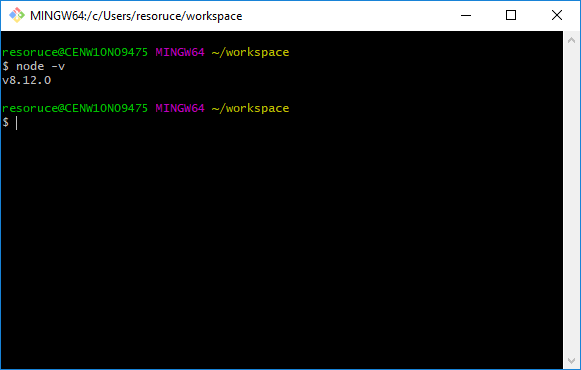


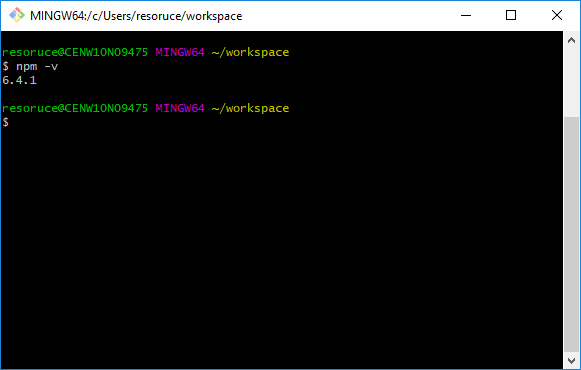




Feito a instalação, incluir nas variáveis de ambiente para sua conta o o caminho do npm na variável path.

Após realizar a instalação, abre qualquer terminal e confirma se o Node e o npm está sendo reconhecido corretamente seguindo os comandos abaixo:



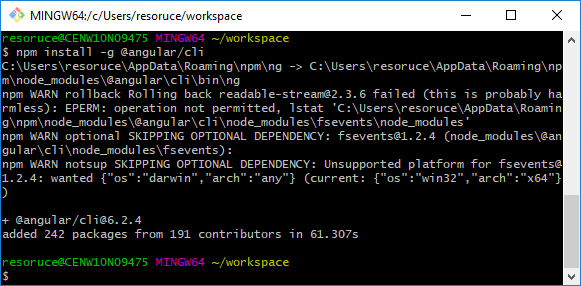


# Angular CLI

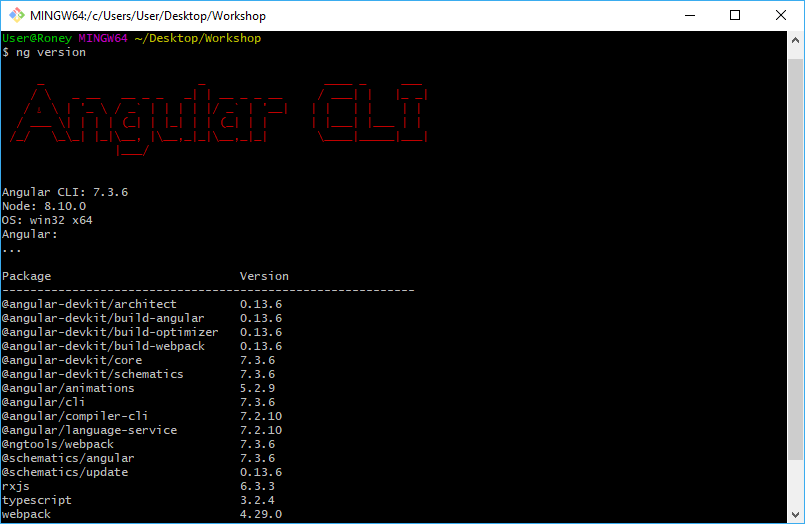
O Angular CLI é uma ferramenta para inicializar, desenvolver, sustentar e manter aplicações através de comandos básicos que podemos encontrar facilmente em sua documentação.

Com o npm reconhecido, basta executar o comando abaixo para instalar o Angular CLI globalmente:

Execute no terminal ***npm install –g @angular/cli***

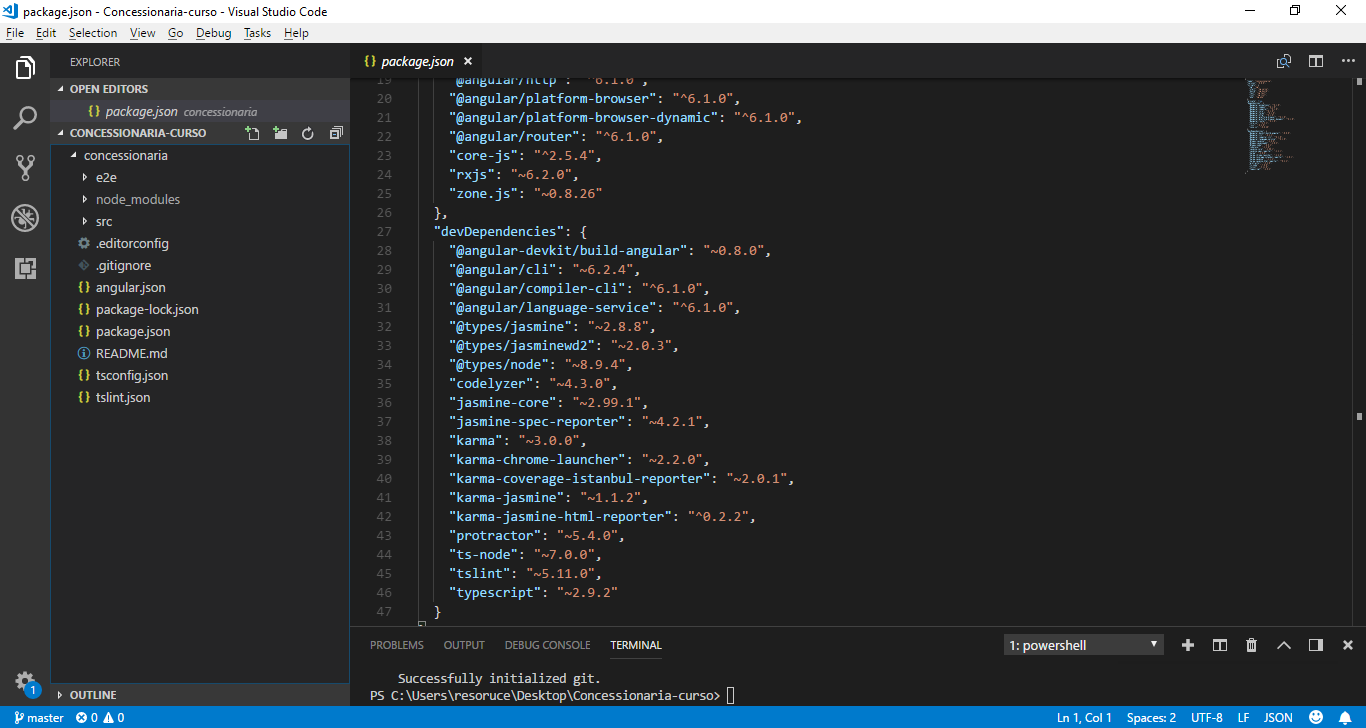


Confirma se o angular está sendo reconhecido através do comando ng version:



# ENTENDENDO O FRAMEWORK ANGULAR

## Estrutura de um projeto Angular com Angular CLI



### package.json

O arquivo package.json é um gerenciador de pacotes npm intalados localmente.

Com ele, é possível listar os pacotes dos quais o projeto depende, permite que você especifique as versões de uma dependência que seu projeto pode usar e torna sua compilação reproduzível e, portanto, muito mais fácil de compartilhar com outros desenvolvedores.

### angular.json ou angular-cli.json

O mais provável é que todos nós encontremos o arquivo angular-cli.json ao gerar um projeto usando o Angular CLI.

É bem conhecido que esse arquivo é usado como o esquema de configuração para todo o projeto manipulado pela CLI, incluindo o gerenciamento de diferentes ambientes, testes, proxy, recursos de terceiros e [muitas ferramentas integradas e recursos](https://github.com/angular/angular-cli/wiki/1-x-stories) para o desenvolvimento de uma aplicação.

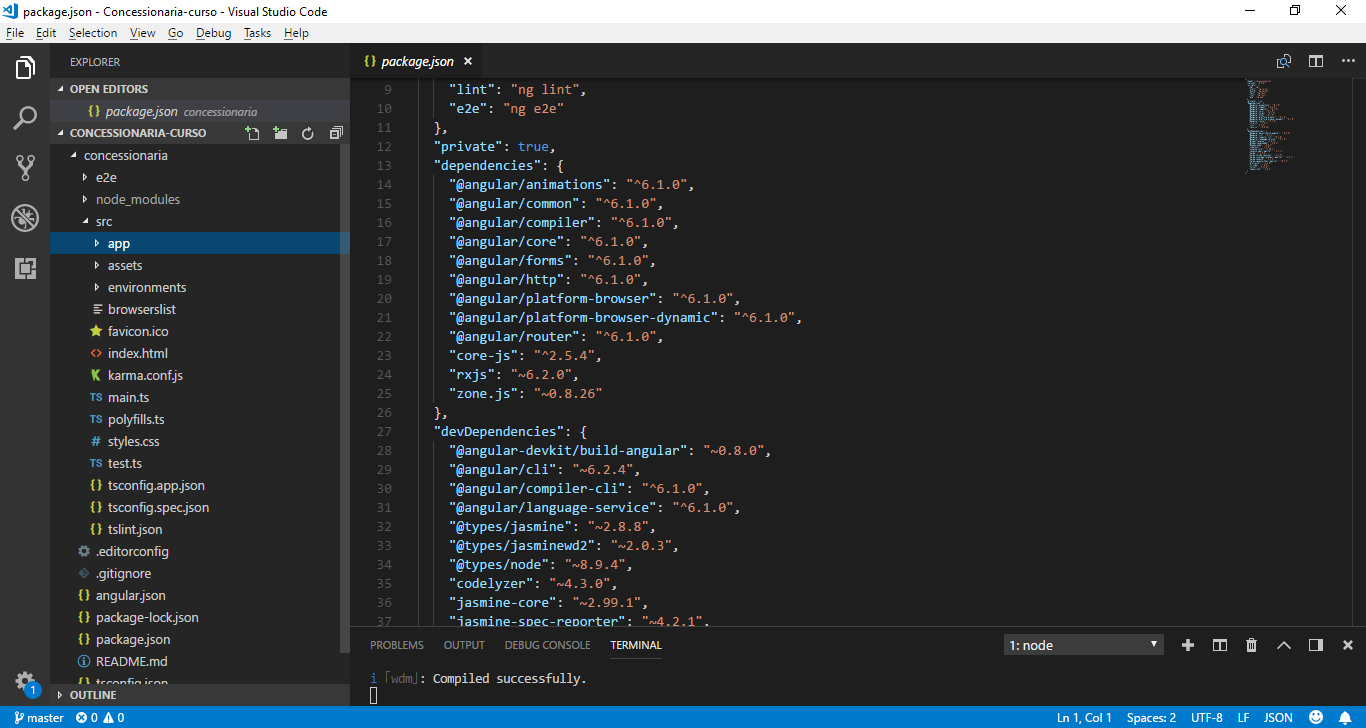
Exemplificando, nele é possível declarar todos os styles e scriptis globais a serem utilizados nas páginas seguindo a estrutura e declarando-os em assets e styles.

### node\_modules

Local onde ficam alocadas todas as dependências do projeto. A pasta node\_modules é uma cópia das dependências instaladas globalmente no ambiente fazendo com que as dependências se tornem módulos internos do projeto prontas para serem utilizadas.

Caso o projeto não possua um node\_module, basta executar o comando **npm install** na raiz do projeto.

### src



### Index, Style e main

Arquivos básicos iniciais (html, css e js)

### environment.json

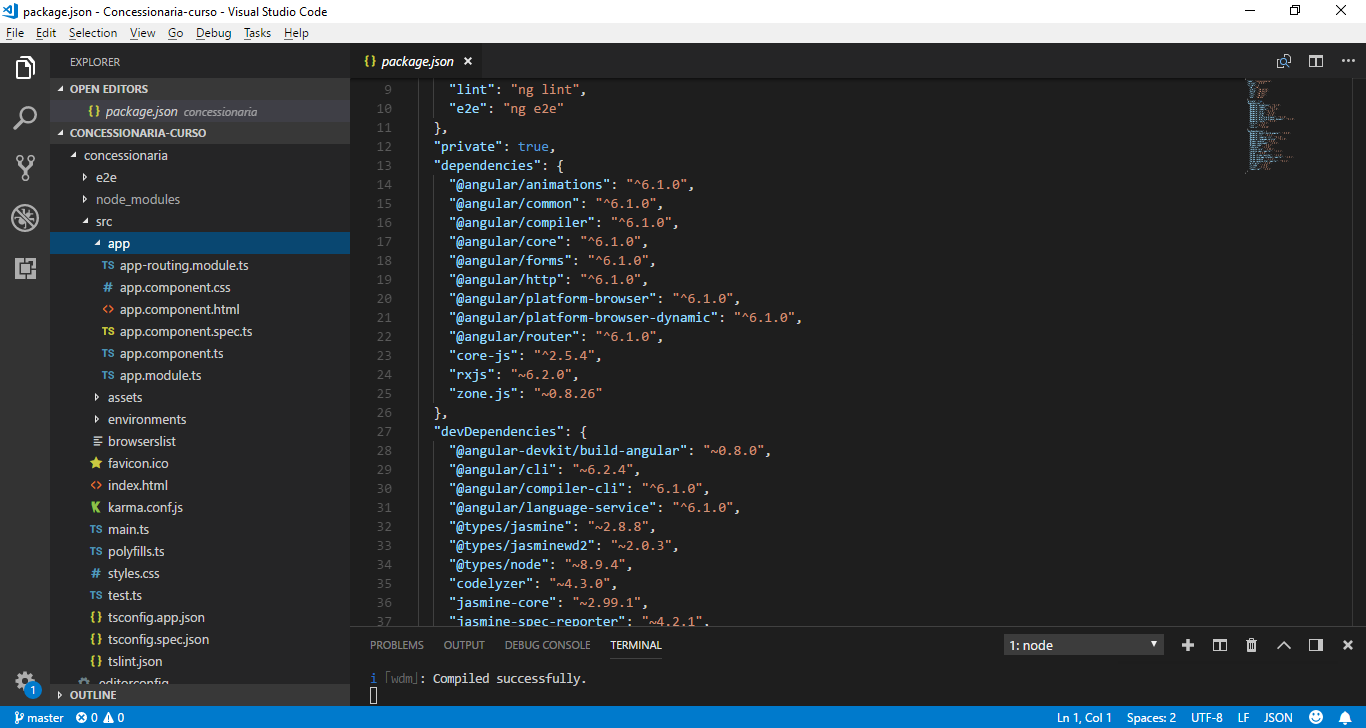
Arquivo no qual encontra-se informações de acesso a API do projeto como URL e porta.

### Assets

Local onde incluímos nossas galeria de imagens, gifs, arquivos em geral da aplicação.

### App

A pasta app é onde se encontra toda nossa estrutura da aplicação desde o modulo principal (app.modulo) até os componentes mais simples.



## Modules

Módulos tem a função de configurar o injetor e o compilador do módulo e auxiliar a organização de todos os componentes relacionados ao mesmo.

Dentro do decorator NgModule, podemos identificar alguns itens importantes na construção de um módulo.

### Imports

Imports são responsáveis por tornar as declarações exportadas de outros módulos disponíveis para o módulo atual.

### Declarations

Em declarations, incluímos todos os Componentes, diretivas, pipes que serão usados pelo módulo atual.

### Providers

Qualquer tipo de serviço, incluímos em providers para que assim possa ser injetado e utilizado pelos componentes do módulo atual através da injeção de dependência.

## Decorators

Decoradores são marcadores que são responsáveis por denominar se uma classe é um módulo, um componente ou até mesmo se um atributo é um dado de entrada ou de saída.

Os decoradores são na verdade apenas funções, são tão simples quanto isso e são chamados com o que quer que estejam decorando. Um decorador de método será chamado com o valor do método que está decorando, e um decorador de classe será chamado com a classe a ser decorada.

## Diretives

Você pode definir suas próprias diretivas para anexar um comportamento personalizado aos elementos no DOM. As opções fornecem metadados de configuração que determinam como a diretiva deve ser processada, instanciada e usada no tempo de execução.

## Components

fornece metadados de configuração que determinam como o componente deve ser processado, instanciado e usado em tempo de execução.

## Router

Define o objeto Route que mapeia um caminho de URL para um componente e a diretiva RouterOutlet que você usa para colocar uma exibição roteada em um modelo, bem como uma API completa para configurar, consultar e controlar o estado do roteador.

## 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:

|  |  |  |
| --- | --- | --- |
| **Múltiplos** | Array | Observable |
| **Únicos** | Objeto | Promise |
|  | **Synchronous** | **Asynchronous** |

**Observação:**

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

<https://www.youtube.com/watch?v=QGA0ApPtgWo>

## Lidando com o two-way data binding

No two-way data binding, alterações na view são refletidas na fonte de dados e atualizações na fonte refletem na view sem a necessidade de manipulação explícita do DOM. Vejamos um exemplo:

### Implementando two-way data binding no Angular

Antes de realizarmos um componente utilizando um two-way data binding, devemos importar em nosso app.module o Módulo FormsModule.

import { FormsModule } from '@angular/forms';

FormsModule é o módulo principal que manipula as diretivas do Angular.

O two-way data binding no Angular é implementado com o ngModel em um input conforme o exemplo a seguir:



|  |
| --- |
|  |

Quando o evento input é disparado, $event.target representa quem disparou o evento. Sendo um elemento do DOM, podemos acessar seu valor com $event.target.value. É este valor que é atualizado na propriedade palestrante do nosso componente.

Contudo, a equipe do Angular criou uma diretiva que funciona como uma espécie de atalho para esses dois tipos de associação.

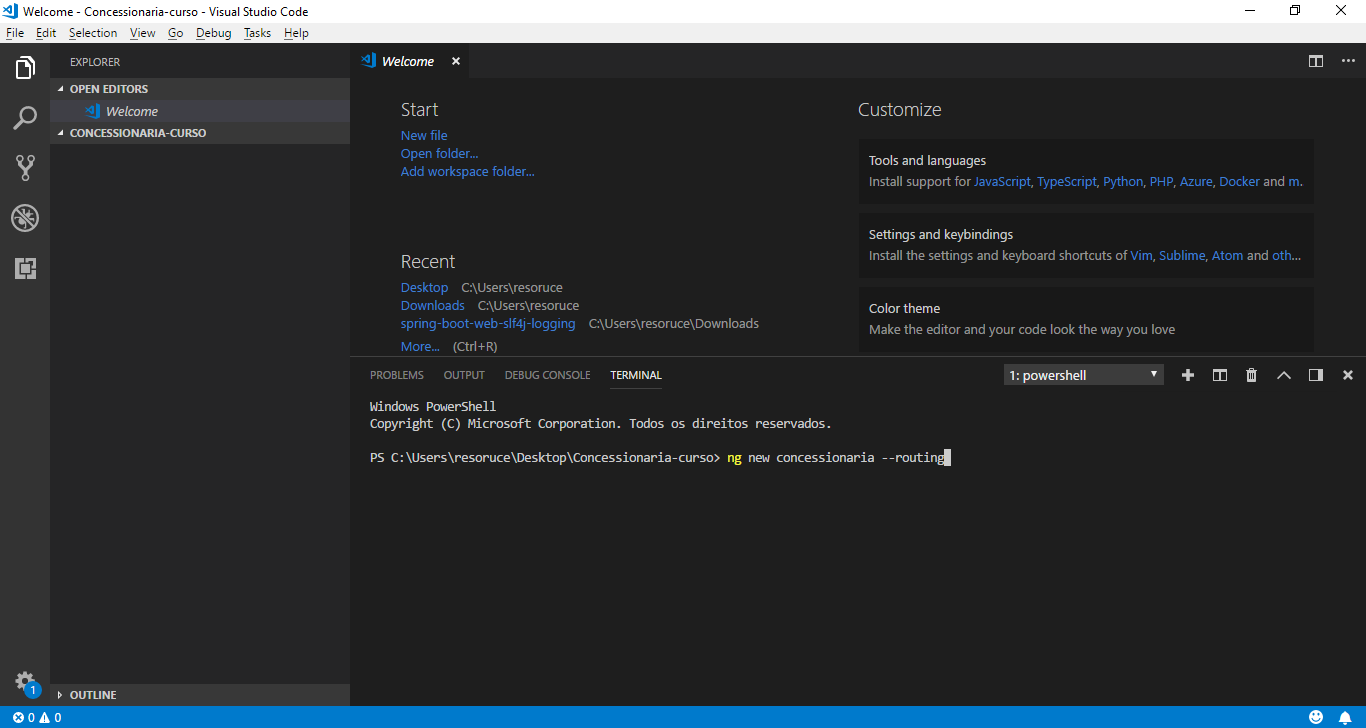
|  |
| --- |
|  |

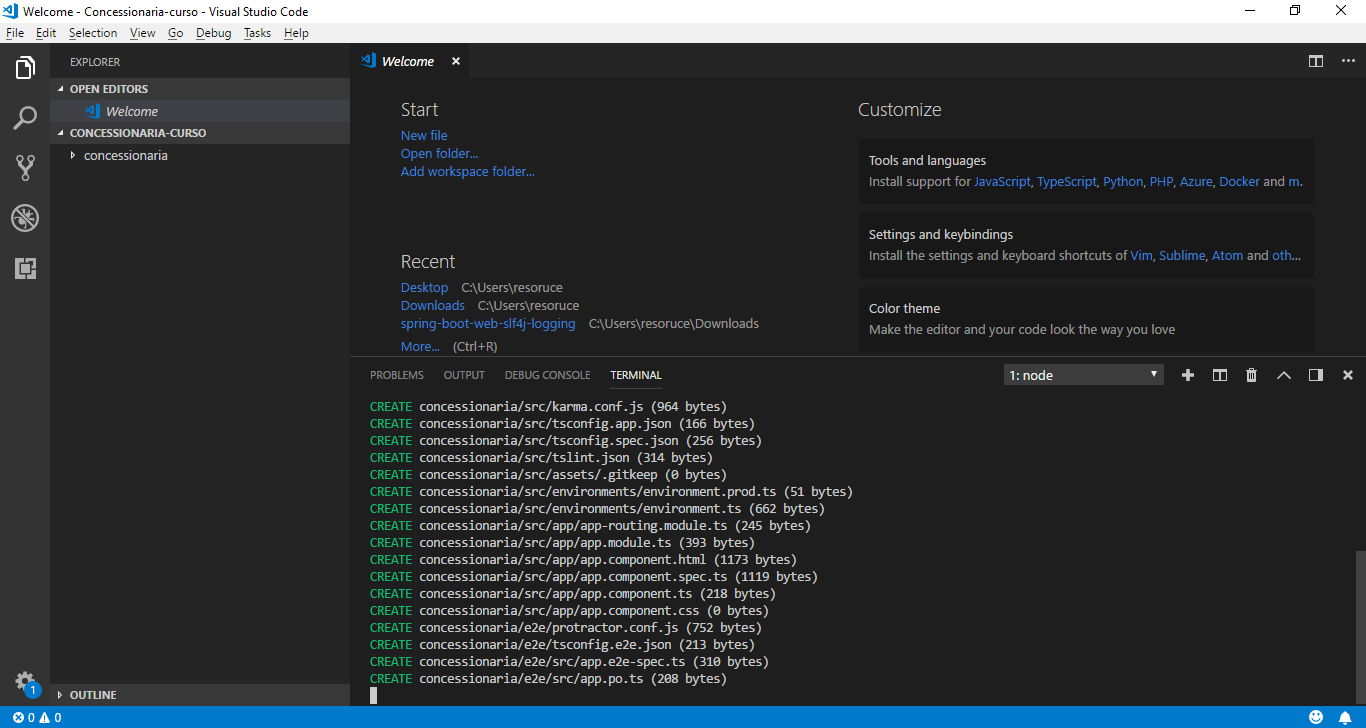
Veja que a diretiva está envolvida por [()], ou seja, temos os dois tipos de associações unidirecionais ao mesmo tempo, mas com direções opostas no fluxo de atualização.

## Criando novo projeto

Para criar um novo projeto vamos abrir nosso Visual Studio Code e digitar o seguinte comando no terminal interno:

Ng new my-app --routing





## Criar novo modulo

  Ng g m clientes --routing

## Criar novos componentes do modulo

  Ng g c clientes /cadastro-clientes

  Ng g c clientes /consulta-clientes

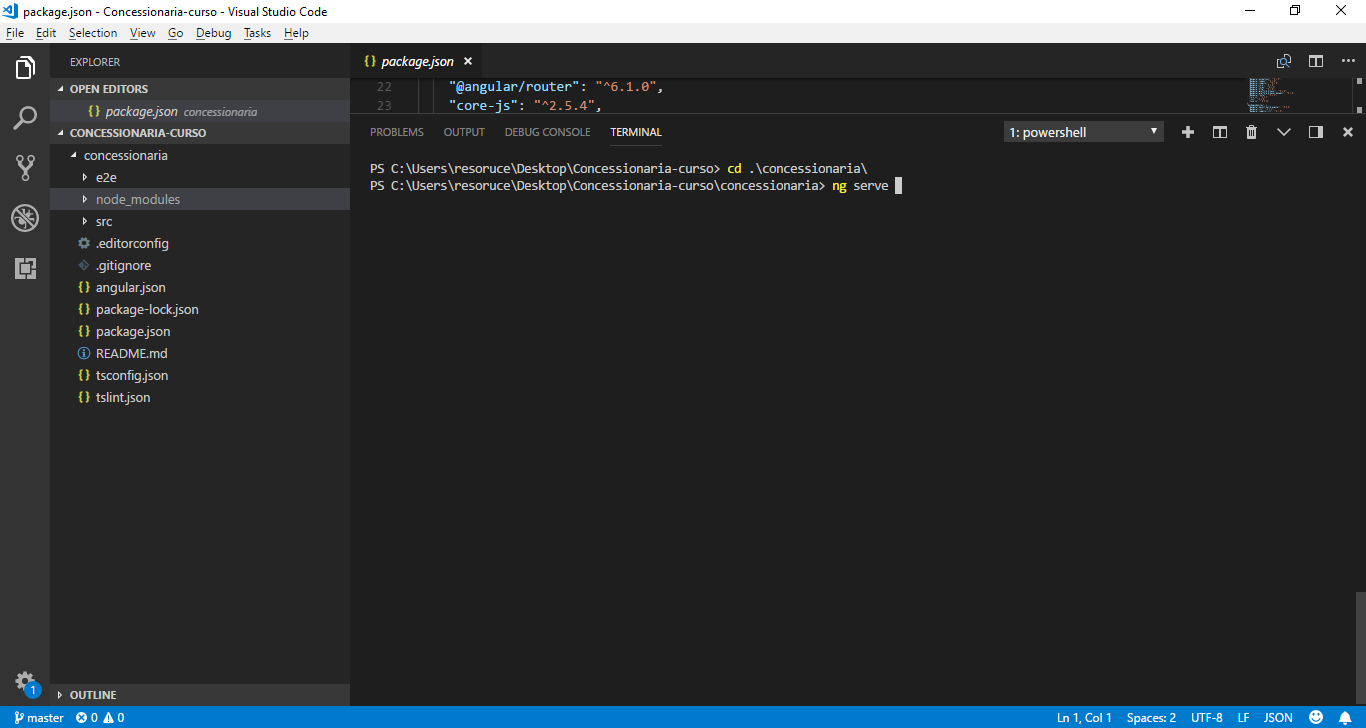
ng g c home --module app

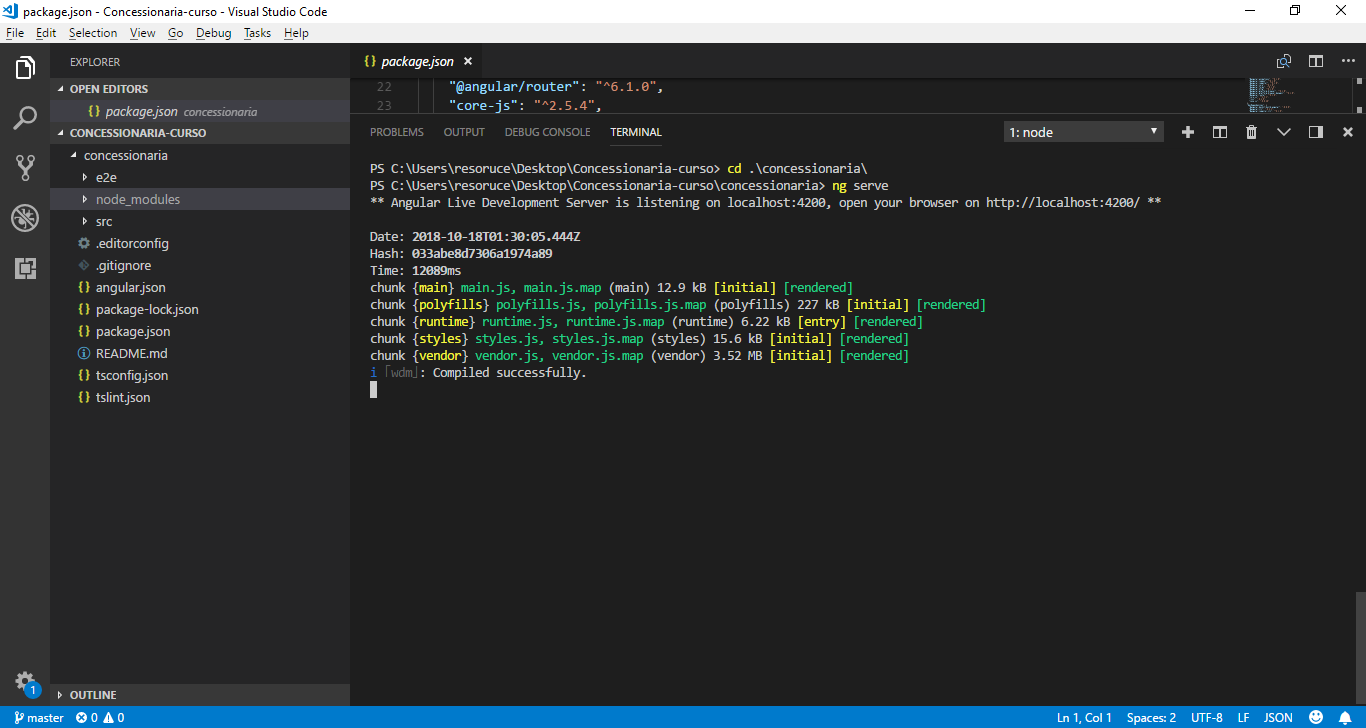
## Criar novos serviço dos componentes

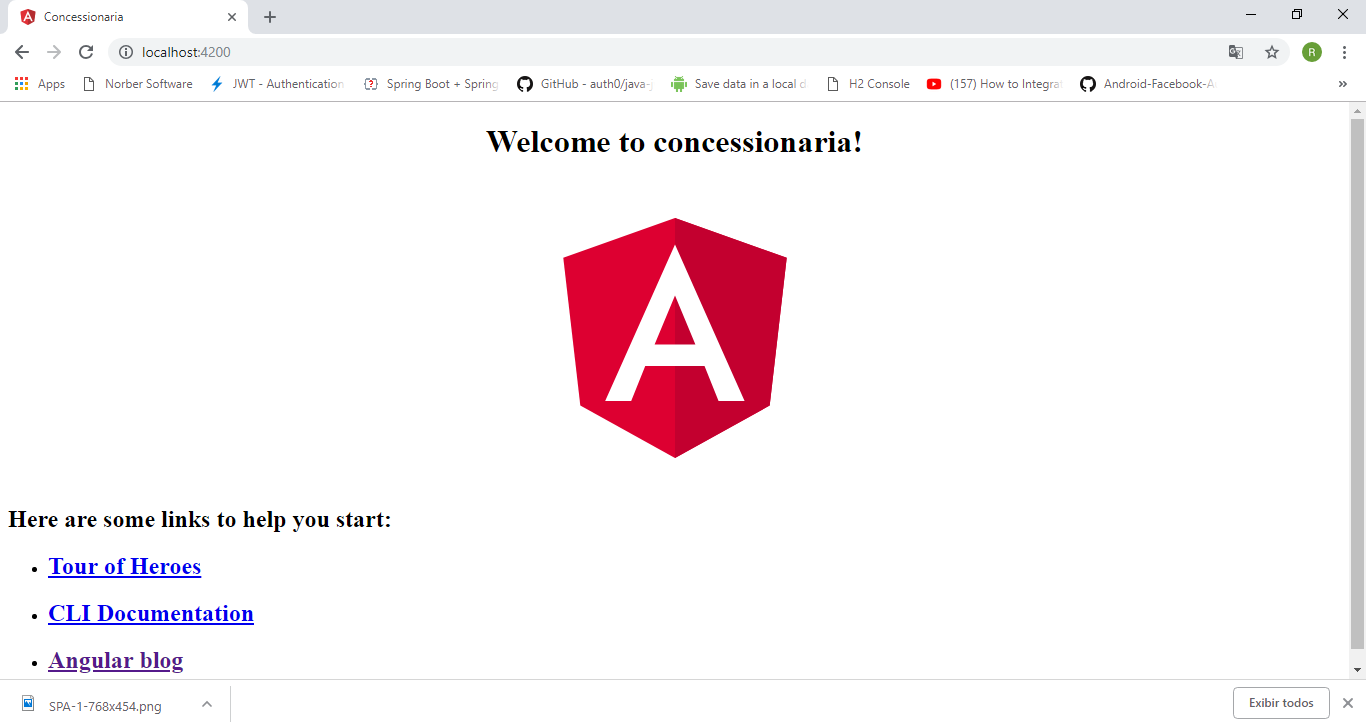
  Ng g s clientes / clientes

## Inicializando o Projeto

Para inicializar a aplicação, basta executar o comando **ng serve** no terminal conforme abaixo:





Por default, o Angular, sobe na porta 4200 quando inicializado. Para visualizar a página, acesse a rota [**http://localhost:4200/**](http://localhost:4200/)

# Exercício do Dia

**Realizar o exercício 01 da lista de exercícios**

# Próximo módulo

* Entender como funciona as rotas em uma aplicação angular e aplicar
* Aplicação de Designe Pattern com shared.module
* Realizar o Crud de clientes sem requisição para api
* Introdução ao PrimeNg