





Angular Framework

### Introdução

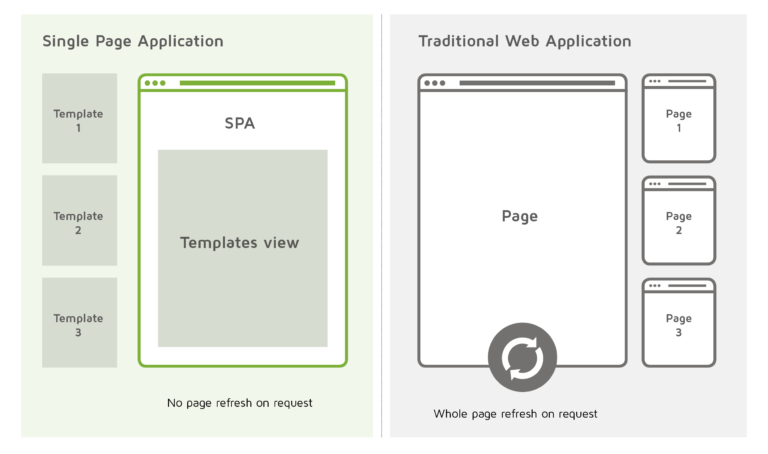
Angular é uma plataforma de aplicações web de código-fonte aberto e front-end baseado em TypeScript liderado pela Equipe Angular do Google e por uma comunidade de indivíduos e corporações. Angular é uma reescrita completa do AngularJS, feito pela mesma equipe que o construiu.

### SPA – Single Page Aplication

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 deve ser interpretado por um servidor de 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.

**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.

Sendo assim, por ter havido uma confusão de versões de componentes onde o componente Router já estava com a versão 3.5, a próxima versão do framework, foi denominada como 4 alterando todos os componentes para versão 4 também.

Hoje estamos com a versão 7 do Angular e esta versão trouxe muitas melhorias para o framework como podemos observar a seguir:

### 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.

#### Módulo de rotas automaticamente

|  |
| --- |
| O Angular CLI perguntará se desejamos utilizar o módulo de rotas do Angular. Se dissermos que sim, o CLI fará toda a configuração necessária e não precisaremos ter esse trabalho: |

#### Pré-processador CSS

Na segunda pergunta escolhemos o processador de CSS que queremos utilizar ou se desejamos utilizar CSS puro no nosso projeto. Para responder utilize as setas do teclado e aperte enter:



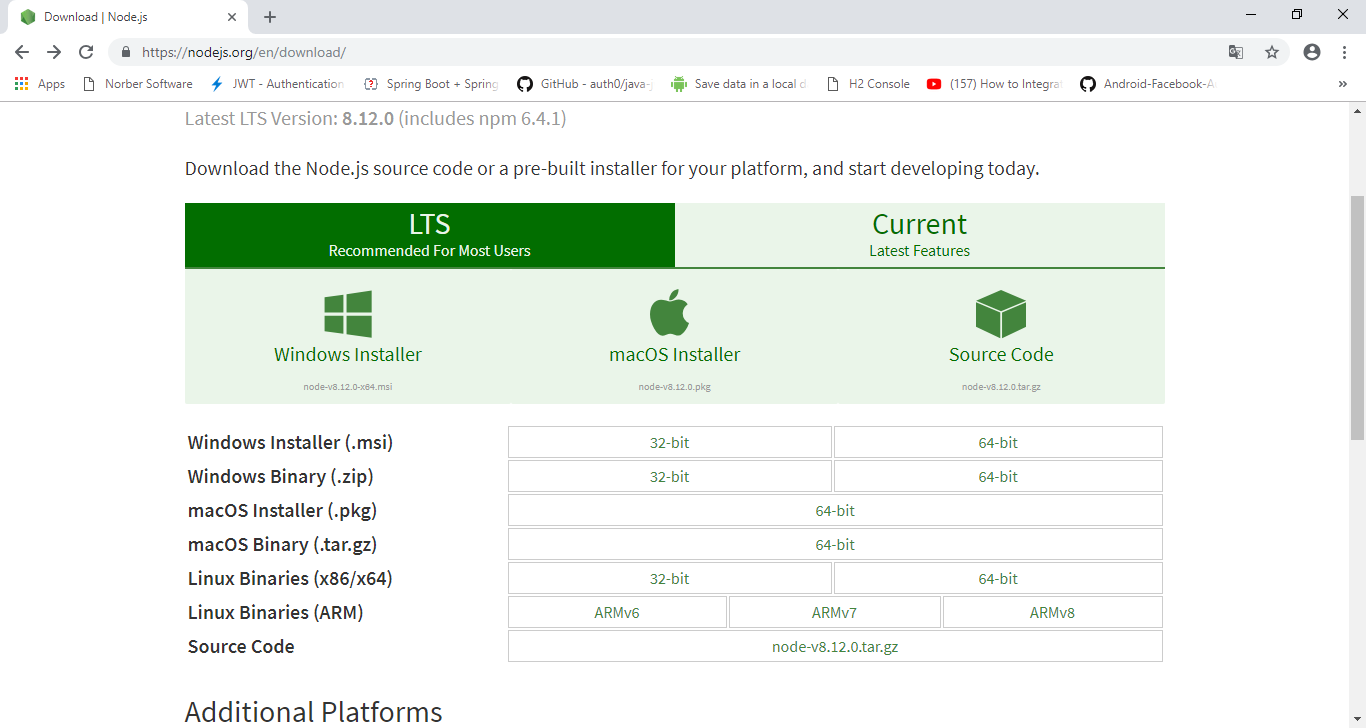
## CONFIGURANDO AMBIENTE

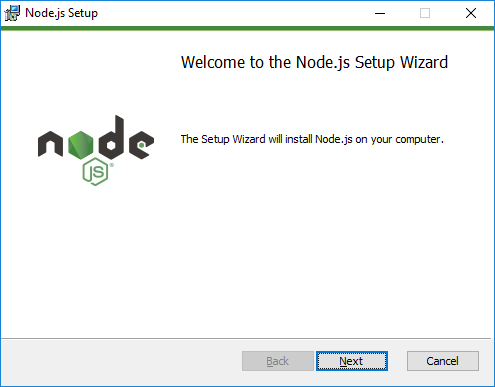
### NodeJS – Servidor de Aplicação

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

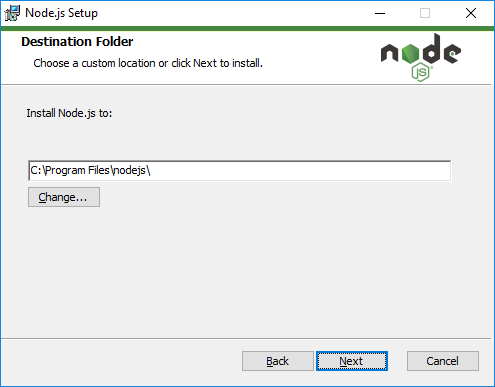
Acesse o link abaixo para realizar o download do Node:

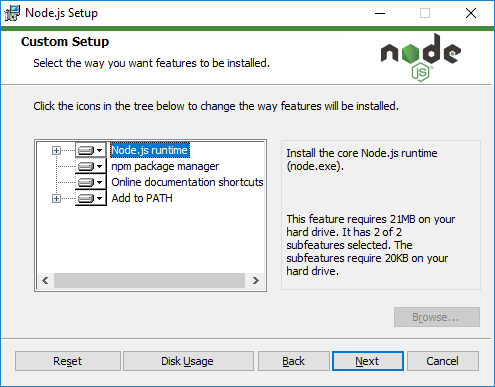
<https://nodejs.org/en/download>

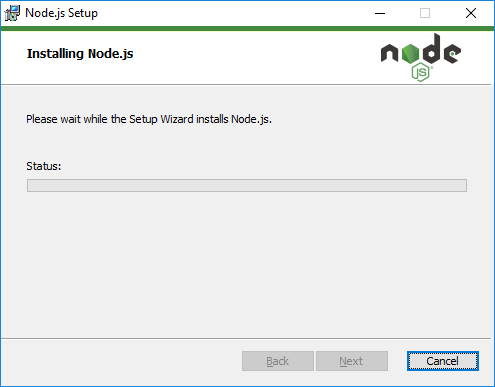


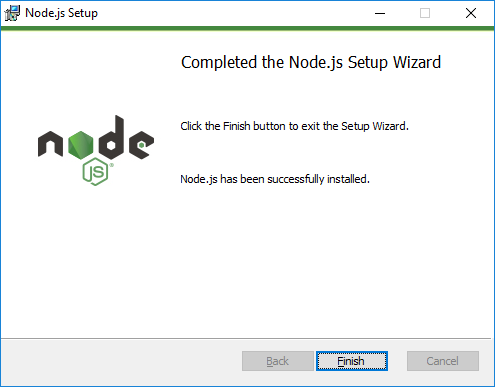




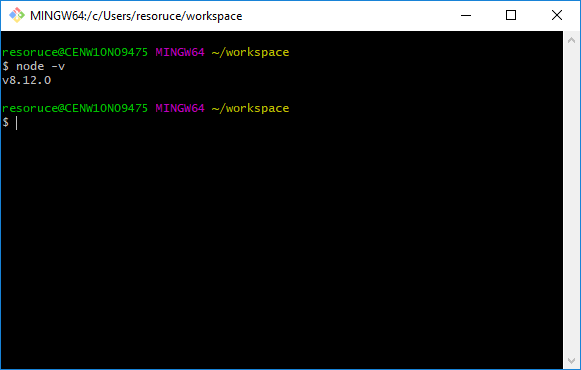


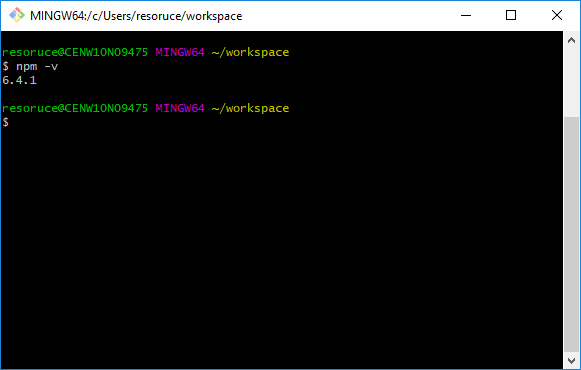






Após realizar a instalação, abre qualquer terminal e confirma se o Node e o npm está sendo reconhecido corretamente seguindo os comandos abaixo:



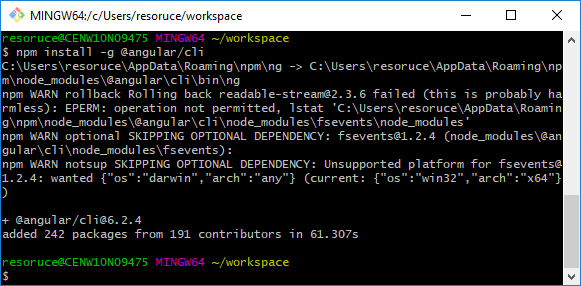


### Angular CLI

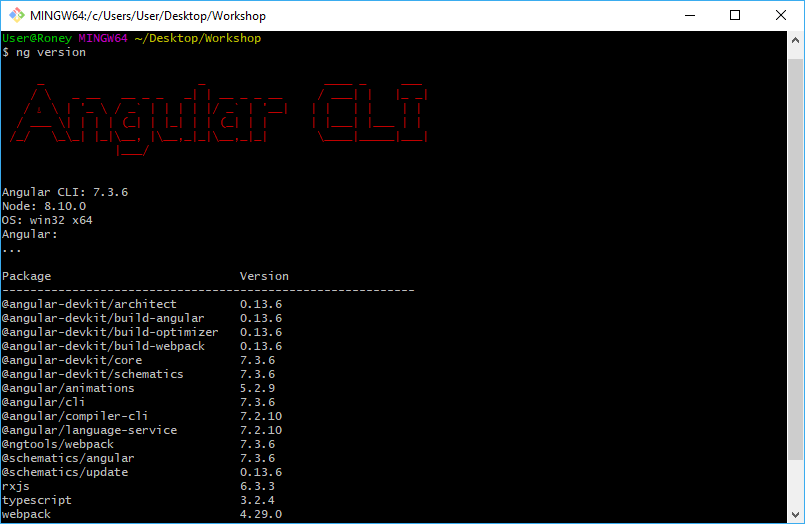
O Angular CLI é uma ferramenta para inicializar, desenvolver, sustentar e manter aplicações através de comandos básicos que podemos encontrar facilmente em sua documentação.

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

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



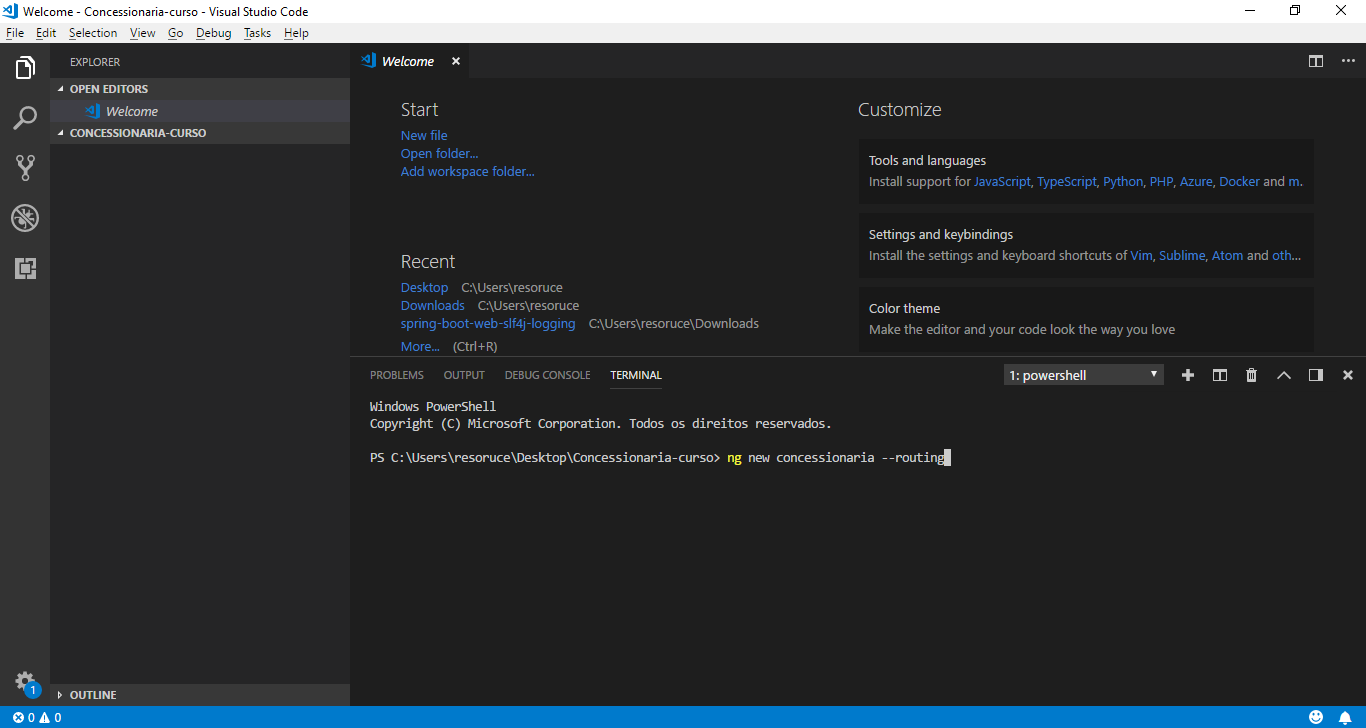
Confirma se o angular está sendo reconhecido através do comando ng version:

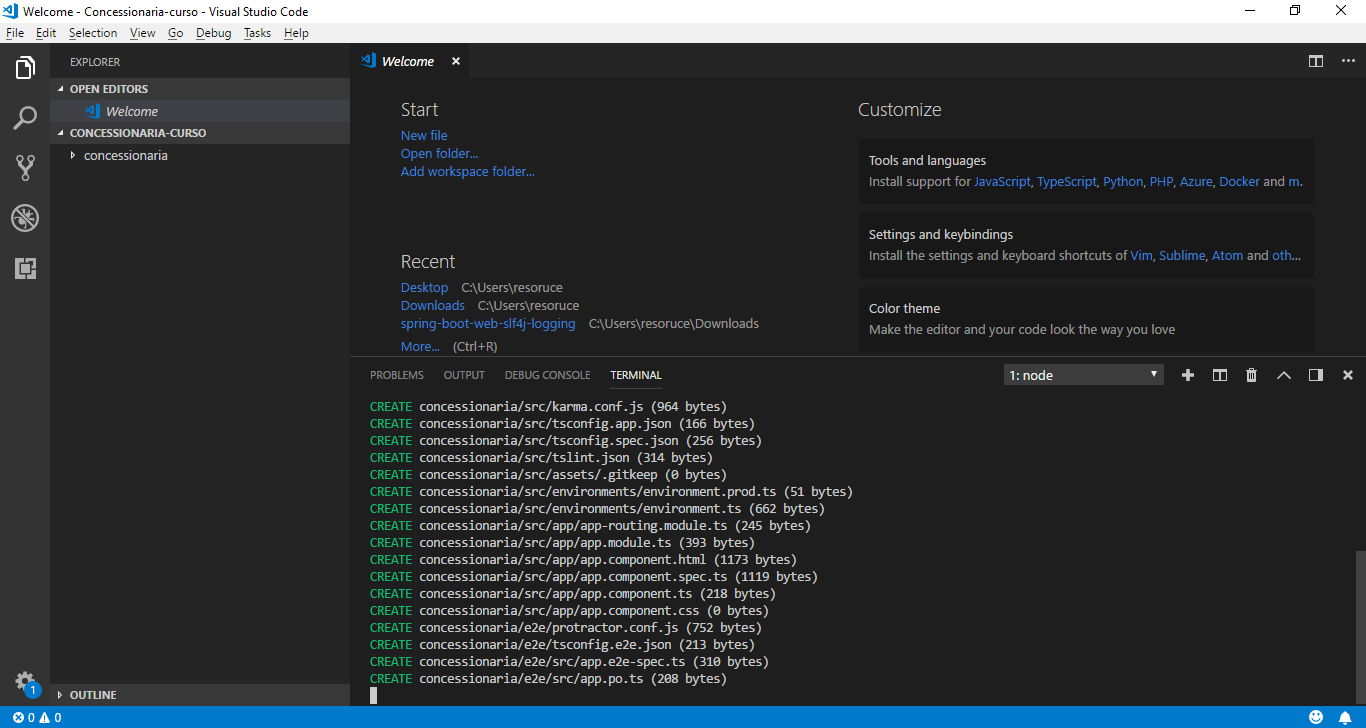


### Criar novo projeto

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 usuarios --routing

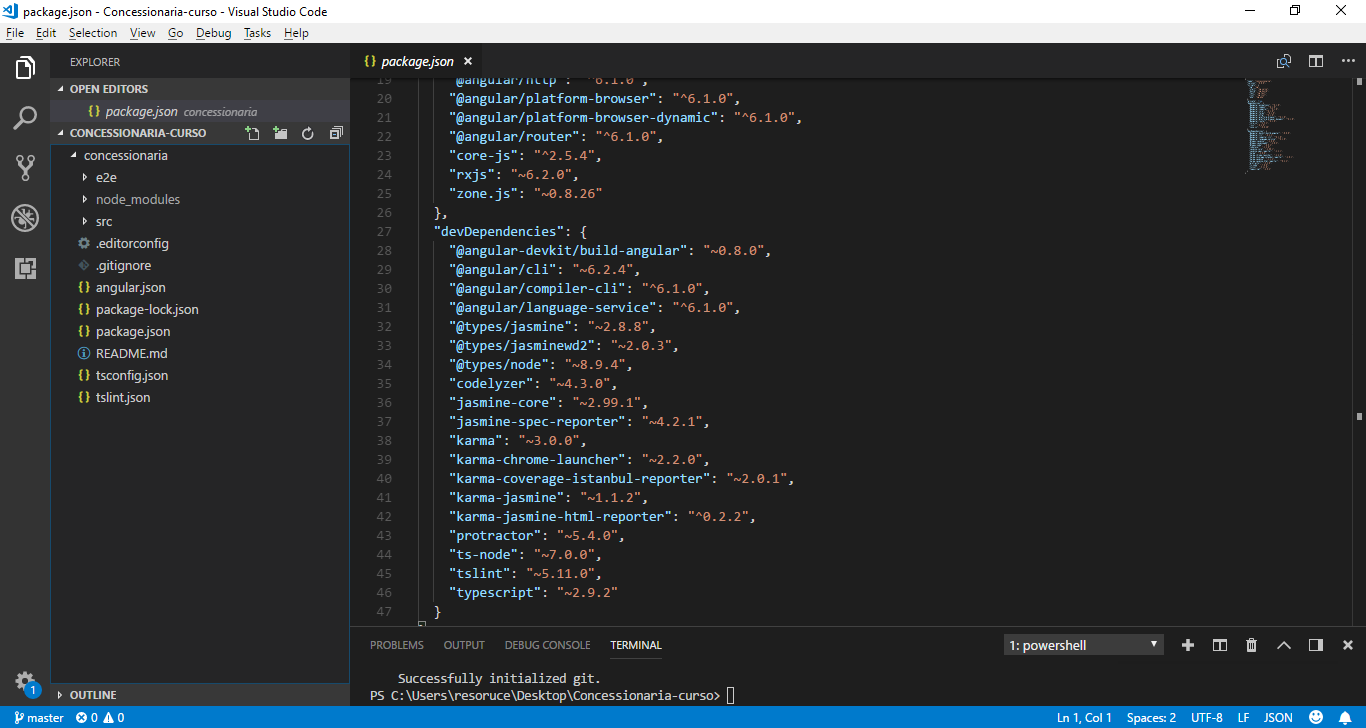
**Criar novos componentes do modulo**

  Ng g c usuarios /cadastro

**Criar novos serviço dos componentes**

  Ng g s usuarios /usuario

**Entendendo a 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.

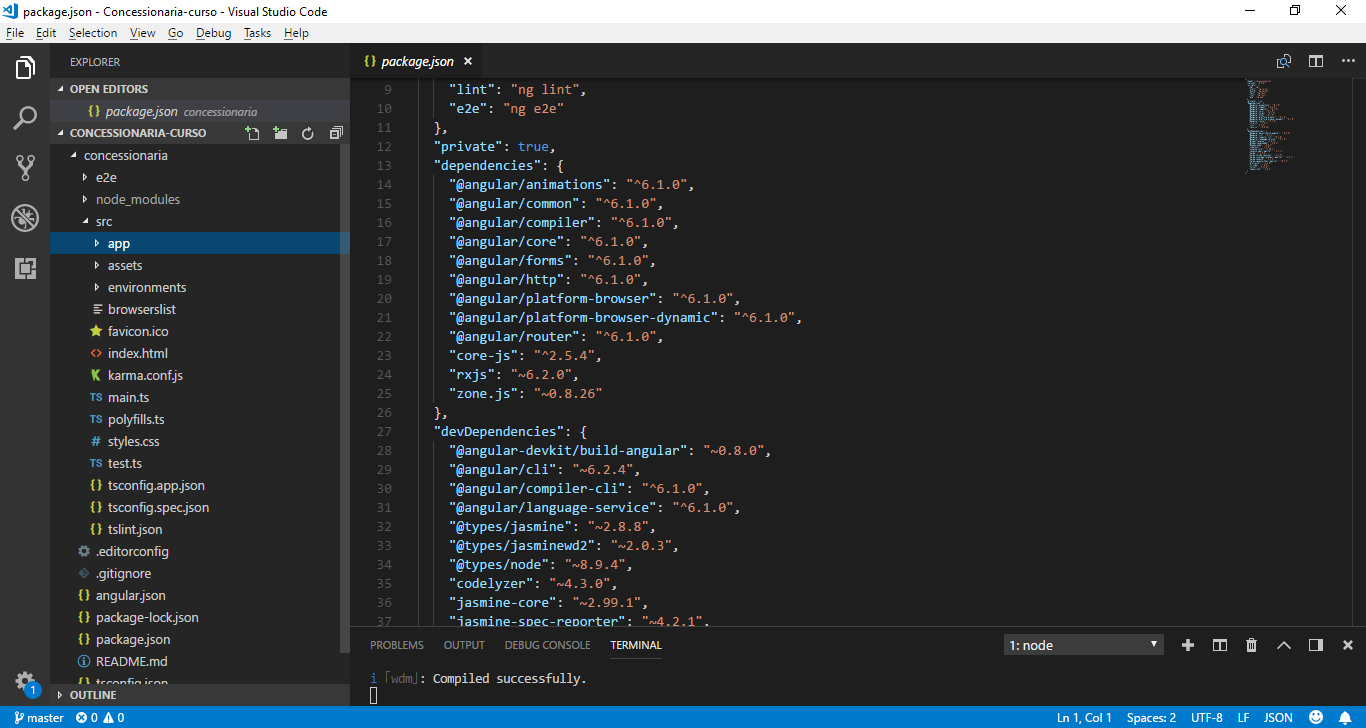
Exemplificando, nele é possível declarar todos os styles e scriptis globais a serem utilizados nas páginas seguindo a estrutura e declarando-os em assets e styles.

**Para saber mais acesse:** https://nitayneeman.com/posts/understanding-the-angular-cli-workspace-file/

**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 dependência se tornam módulos internos do projeto prontas para serem utilizadas.

**src**



**Index, Style e main**

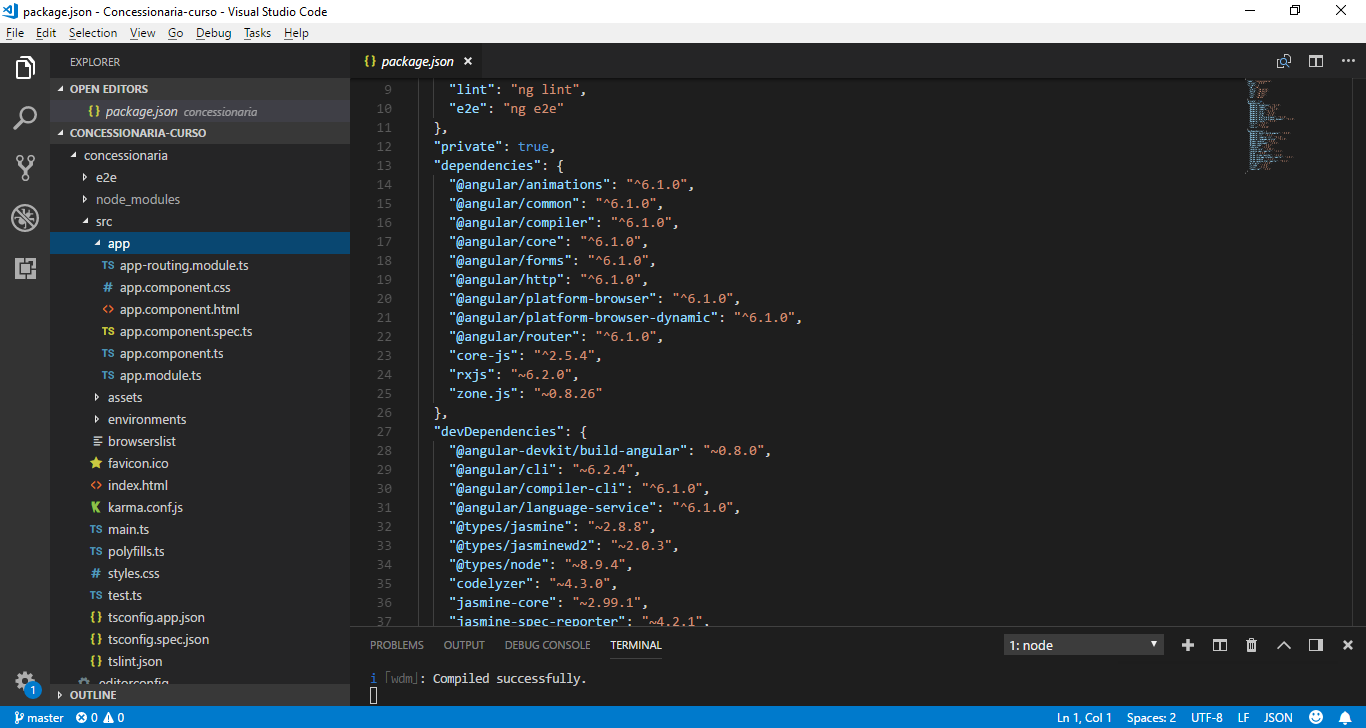
**Arquivos básicos iniciais (html, css e js)**

**environment.json**

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

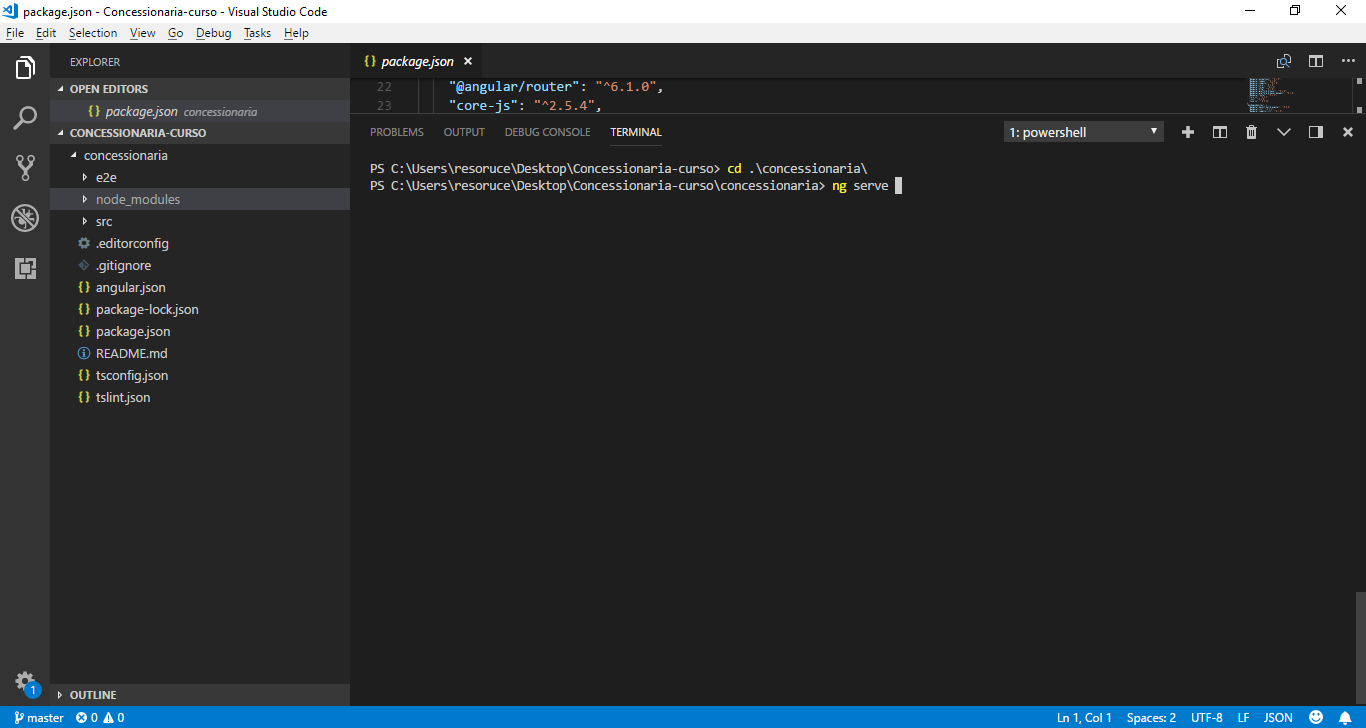
**assets**

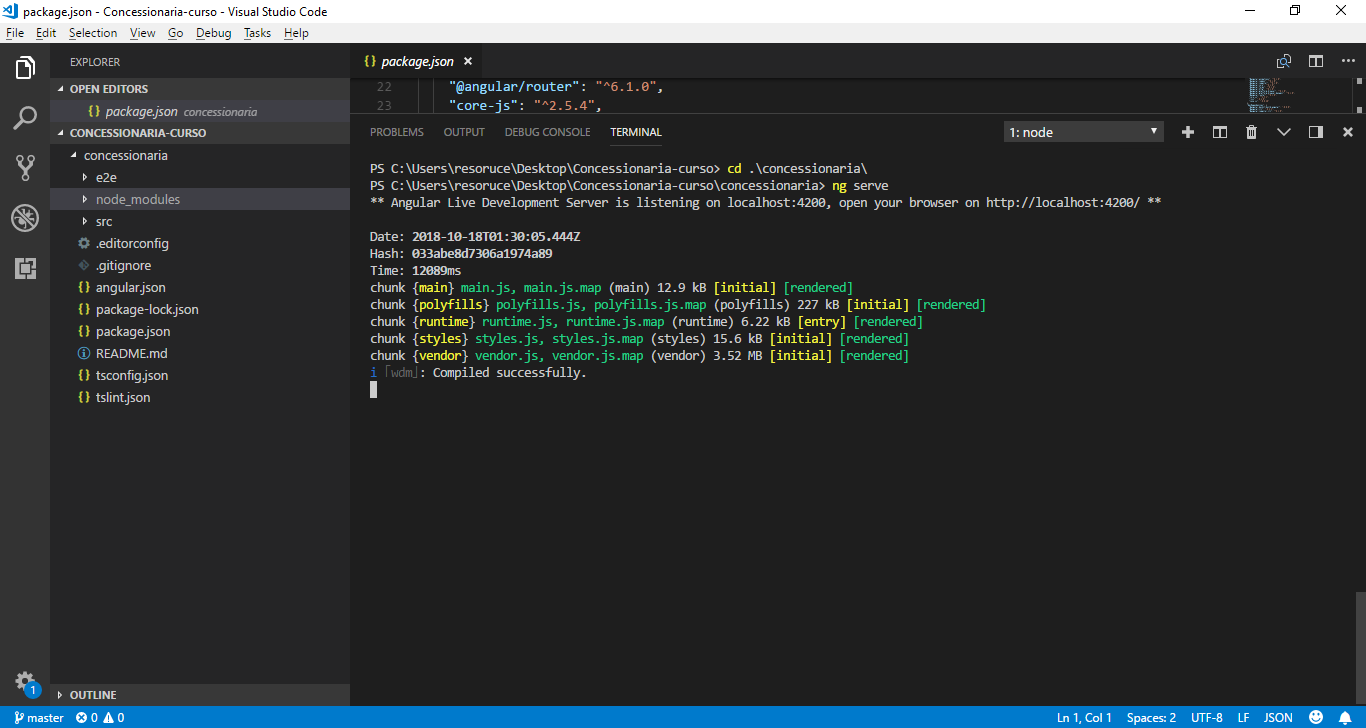
**app**

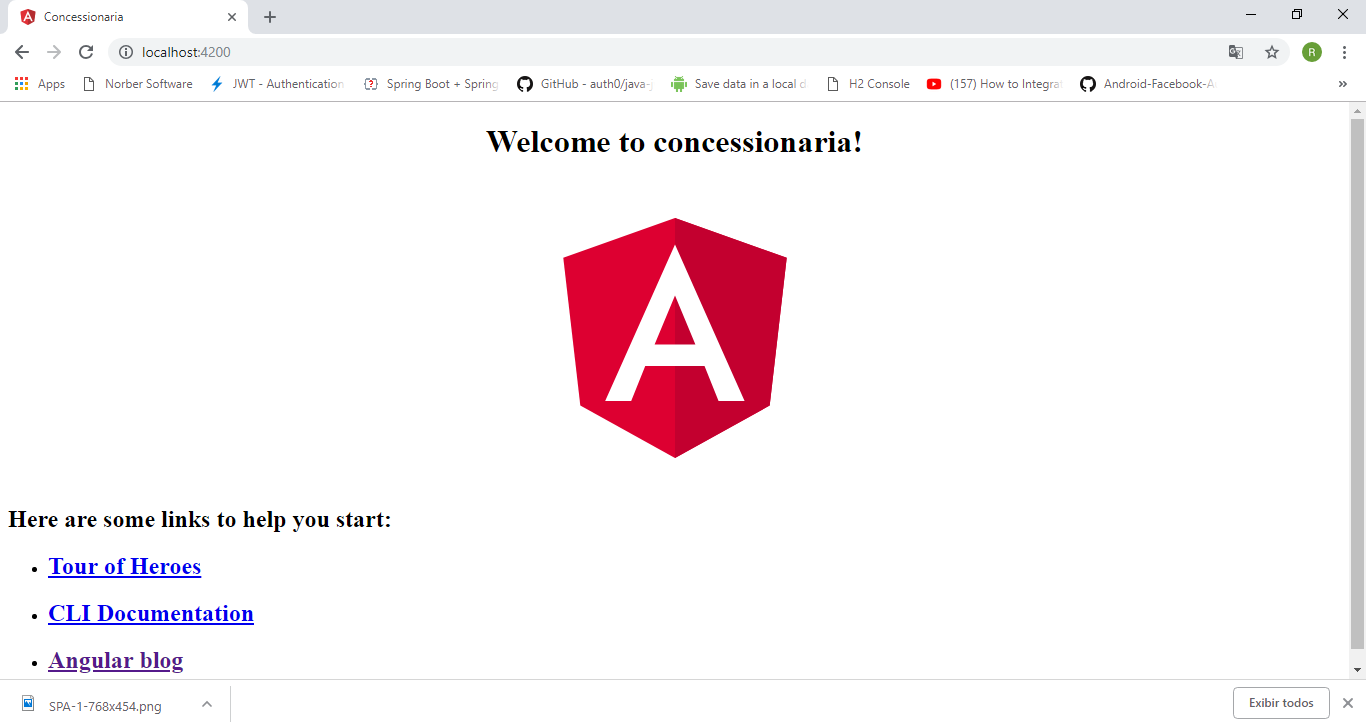


### Inicializando o Projeto

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/**

**Criando rotas**

**Criando um menu**

**Trabalhando com Core.module.forRoot()**

**Realizando um CRUD**

 - Observables e Promises

Criando métodos orientados a eventos e emiter onde assinantes escutam as mensagens

Usando o Primeng e explorando-o

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.

npm install primeng --save

npm install primeicons –save

npm install @angular/animations --save

Trabalhando com Guards

Segregando nível de acesso por Roles

Usando auth e jwt

Token de acesso

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3IiOiJyYW1vcmluIiwibmFtZSI6IlJPTkVZIEFVR1VTVE8gQU1PUklNIiwiZnVuY2lvbmFsIjoiMDMzMjIwOTkzIiwiaWF0IjoxNTE2MjM5MDIyLCJleHAiOiIyMDE5LTEwLTAxIDAwOjAwOjAwIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9HUlVQT19DQURBU1RSQVIiLCJST0xFX0dSVVBPX0VESVRBUiIsIlJPTEVfR1JVUE9fQ09OU1VMVEFSIiwiUk9MRV9HUlVQT19FWENMVUlSIl19.xyrCM336Qx-Xw4i-zsSOg2YhAUsC-ftKHdTbpvbJjQU

https://www.youtube.com/watch?v=z4JUm0Bq9AM

<https://stackoverflow.com/questions/37364973/promise-vs-observable>

Usando o async await

Referencias

<https://www.devmedia.com.br/novidades-do-angular-7/40194>

<https://angular.io/cli>

<https://blog.angular.io/version-6-of-angular-now-available-cc56b0efa7a4>

Pré Requisitos

Git Bash

Visual Studio Code

Possíveis Problemas

Proxy e comunicação por IP