Instituto Federal de Educação, Ciência e Tecnologia do RN
Campus Parnamirim

Prof. Dr. Bruno Emerson Gurgel Gomes (bruno.gurgel@ifrn.edu.br)

Primeira Aplicação com Angular

última atualização em 29/03/2018

Objetivos

- Demonstrar como usar o *Visual Studio Code* e *plugins* para desenvolver uma aplicação Angular;
- Instalar o pacote **@angular/cli** e através dos seus comandos criar uma aplicação simples e rodá-la em modo de desenvolvimento;
- Apresentar os conceitos de *template* e *componente* e a relação entre eles.

Pré-requisitos

- Instalar o **NodeJS** (`https://nodejs.org`)
- Instalar o pacote **TypeScript** de forma global. Na linha de comando, faça:
  ```
  npm install -g typescript
  ```
- Obter e instalar a ferramenta *Visual Studio Code (VS Code)*. Instalar o suporte do *VS Code* a *TypeScript* e *JavaScript*.

1 Instalação de plugins no Visual Studio Code

Antes de começar, sugiro a instalação de diversos *plugins* que auxiliam e melhorar a produtividade no desenvolvimento. É interessante instalar o *Angular Extension Pack* da desenvolvedora Loine Groner. Trata-se de um conjunto de *plugins* desenvolvidos por terceiros que fornecem autocompletar para *tags* HTML, diretivas e componentes do *Angular*, *auto imports* para classes e componentes, "destaque" para ícones dos diversos tipos de arquivo que integram o desenvolvimento, dentre outros.

Para instalar clique no ícone de extensões (um ícone quadrado no canto esquerdo da tela) ou aperte a combinação de teclas (**`Ctrl+Shift+x`**). Digite na busca *Angular Extension Pack* e clique em instalar (Figura 1).

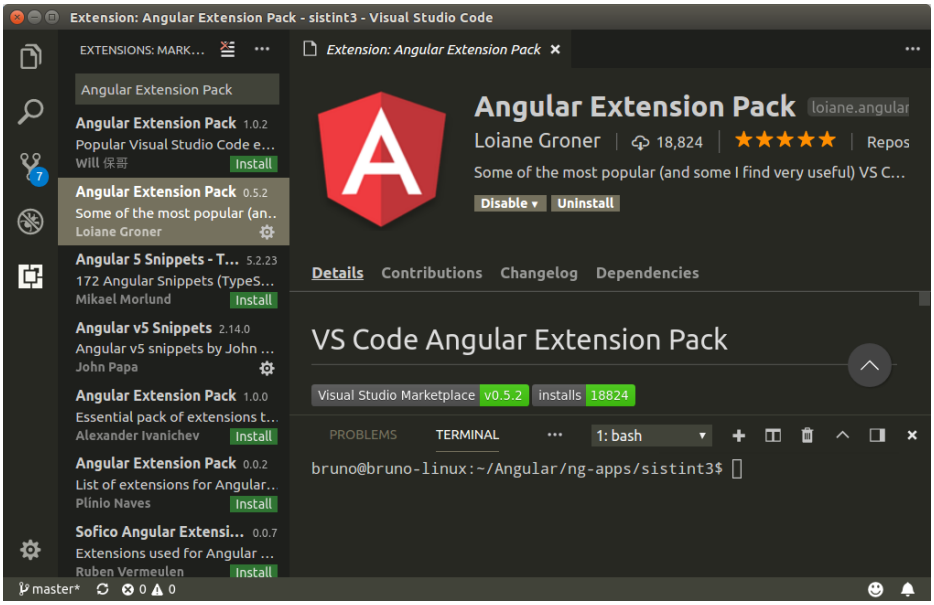

Figura 1: Pacote de plugins Angular Extension Pack. Fonte: autoria própria.

2 Instalação do @angular/cli

O **`@angular/cli`** é um pacote de ferramentas para criar e gerenciar um projeto Angular. Ele automatiza tarefas que seriam bastante tediosas e consumiriam bastante tempo do desenvolvedor, como criar e organizar os arquivos de configuração do projeto, rodar e construir a aplicação, criar componentes, serviços, dentre outras atividades.

Para instalar o **`@angular/cli`** digite o comando a seguir na linha de comandos do seu sistema operacional (*terminal*, *bash*, *cmd*, etc.):

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

A instalação irá demorar alguns segundos ou poucos minutos, a depender da configuração do seu computador.

3 Criação do projeto

Agora vamos começar a criar o projeto. Inicialmente, escolha um diretório de trabalho para armazenar os seus projetos. Se estiver em uma máquina corporativa esse diretório deve estar dentro de sua pasta de usuário para que você não venha a ter problemas com permissões de acesso a arquivos e pastas. Tenha em mente também que o *Angular* ocupa muito espaço durante o desenvolvimento. Uma pasta de projeto tipicamente possui dezenas de pastas, arquivos e pacotes de código e ocupa cerca de 300 MB de espaço em disco.

Suponha que esse diretório se chama **`angular`** e está dentro da pasta do usuário do sistema. Rode o *Visual Studio Code* e abra esse diretório (*open folder*). Em seguida, abra o terminal integrado ao *Visual Studio Code* (*menu View → Integrated Terminal*). Ao abrir o terminal integrado ele deve abrir na pasta que você selecionou. Digite o comando a seguir para criar o novo projeto que irá se chamar **`primeiro-app`**.

```
ng new primeiro-app
```

Seja paciente. A criação do esqueleto básico do projeto com todos os seus arquivos pode demorar cerca de 4 minutos. A maior parte deste tempo é gasta na instalação dos pacotes básicos por meio do *Node Package Manager* (**`npm`**). Esses arquivos estão sob uma pasta com o nome de **`primeiro-app`**, conforme nomeado no comando **`ng new`**. A pasta irá aparecer no menu lateral esquerdo do *Visual Studio Code*, conforme demonstra a Figura 2.

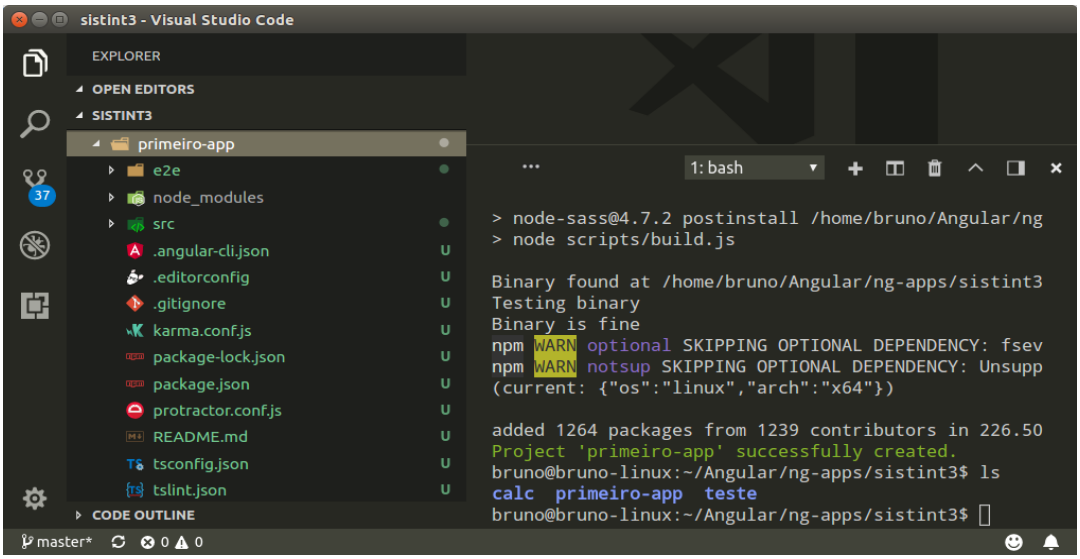

Figura 2: Pasta do projeto após a sua criação no VS Code. Fonte: autoria própria.

Na Figura 2, o diretório de trabalho está abaixo do diretório do usuário “bruno” e se chama **`Angular/ng-apps/sistint3`**, o qual simplificamos neste tutorial apenas por **`angular`**. Abaixo dele é possível ver no lado direito da janela o resultado final da criação do projeto na mensagem “*Project ‘primeiro-app’ sucessfully created*” e outros projetos na pasta (comando **ls** no linux e **dir** no windows), a saber *calc* e *teste*. Nesse momento, por meio do terminal integrado, entre na pasta do projeto **`primeiro-app`** (comando **`cd primeiro-app`**). Nos próximos passos iremos rodar o projeto no modo de desenvolvimento e criar um componente. Sendo assim, é necessário estar na pasta do projeto para que possamos executar os devidos comandos aplicados a ele.

4 Rodar o projeto em modo de desenvolvimento

O comando **`ng serve`** constrói a aplicação e a executa em um servidor local temporário. Esse modo de execução é interessante durante o desenvolvimento. O comando também fica observando por mudanças nos arquivos e sempre que elas ocorrem é feito o *rebuild* (reconstrução) e a visualização da aplicação é atualizada. Estando no diretório da aplicação, rode o comando:

```
ng serve
```

Opcionalmente, é possível usar a opção **`--open`** ou simplesmente **`-o`** para que após a compilação do projeto o navegador padrão do usuário seja aberto e executado automaticamente.

```
ng serve -o
```

A aplicação pode ser acessada através da URL `http://localhost:4200`. O resultado padrão exibido é a página da Figura 3.

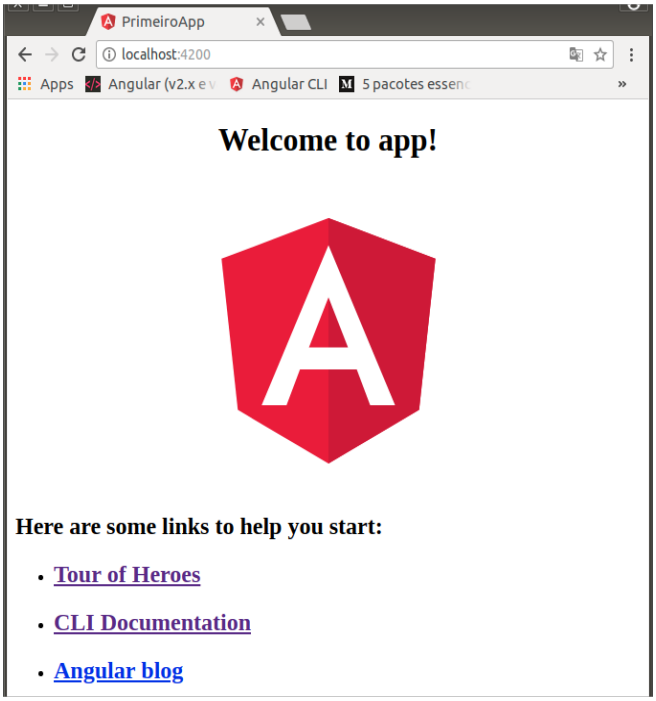

Figura 3: Aplicação rodando no navegador após o comando ng serve. Fonte: Autoria própria.

5 Módulo principal e componente principal da aplicação

Nesta seção iremos dar uma visão geral dos componentes principais de interação do desenvolvedor com o código criado para a aplicação. Uma aplicação *Angular* é composta em um nível mais alto por módulos e componentes. Pense no módulo como um pacote código maior que agrega diversos componentes. Por exemplo, em um sistema acadêmico poderíamos ter o módulo Aluno e dentro desse módulo diversos componentes, um para exibir o boletim, outro para os dados pessoais do aluno, outro para permitir a matrícula dele(a) em um período letivo e assim por diante. No *Angular*, o módulo principal encontra-se no arquivo **`app.module.ts`** do diretório **`src/app`** abaixo do diretório principal da aplicação (no caso, **`primeira-app`**). Ele consiste em uma classe *TypeScript* (**`AppModule`**) conforme pode ser visto no Código 1.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Código 1: Módulo *AppModule*. Fonte: autoria própria gerado com o @angular/cli.

A notação **`@NgModule`** é um decorador (*decorator*) aplicado a um módulo e que permite configurá-lo. Na parte *declarations* são inseridos os componentes e diretivas (discutiremos mais adiante no curso) declarados e oferecidos pelo módulo. Por enquanto, há apenas o componente principal, denominado de **`AppComponent`**, que está definido no arquivo **`app.component.ts`**. O código HTML exibido no navegador ao rodar a aplicação na seção 4 é obtido a partir desse componente. Em *imports* são inseridos outros módulos utilizados pelo módulo **`AppModule`**.

O módulo **`BrowserModule`** provê serviços essenciais para permitir que a aplicação rode no navegador. Em *providers* podem ser inseridos os *serviços* oferecidos pelo módulo. Serviços são classes que fornecem funcionalidades que podem ser compartilhadas por componentes. Por exemplo, se quisermos obter informações de uma base de dados devemos criar uma classes de serviço para realizar as operações de CRUD. Ou no exemplo do sistema acadêmico, poderíamos ter um serviço para criar um objeto com o boletim do aluno. No caso, ainda não há nenhum serviço declarado.

Por fim, em *bootstrap* é inserido o nome do componente principal que irá ser executado após a inicialização do módulo. Nesse caso o componente **`AppComponent`**.

Os componentes no Angular definem as “visões” da aplicação. Ou seja, os pedaços de telas ou páginas que serão exibidas para o usuário. Um componente é composto basicamente por 4 arquivos. Tomemos como exemplo o componente principal **`AppComponent`**. É importante observar que qualquer outro componente criado pelo usuário terá uma estrutura similar:

- Uma página HTML denominada de **`app.component.html`**;
- Um arquivo CSS denominado de **`app.component.css`**;

- Um arquivo *TypeScript* denominado de **app.component.ts**;
- Um arquivo de testes para o componente **app.component.spec.ts**.

A página **app.component.html** é denominada de *template* e é composta por marcações HTML juntamente com marcações próprias do *Angular* para exibir informações advindas dos componentes, adicionar CSS com base em alguma condição, exibir ou não exibir determinadas partes da aplicação, dentre outras funcionalidades. O *template* possui ligação direta com seu componente sendo possível a interação entre eles. O arquivo **app.component.css** permite a inserção de CSS específico para o componente e que pode ser exibido no *template*. Por fim, o arquivo **app.component.ts** contém a classe **AppComponent** que define as configurações do componente, os atributos e métodos que podem ser chamados diretamente no *template*. O arquivo **app.component.spec.ts** é opcional e pode ser usada para realizar testes.

5.1 A classe **AppComponent** (**app.component.ts**)

Uma classe de componente é uma classe *TypeScript* comum com o decorador **@Component** adicionado, como pode ser visto para a classe **AppComponent** no Código 2.

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  title = 'primeira aplicação';
}
```

Código 2: classe *AppComponent*. Fonte: autoria própria gerado com o @angular/cli.

Na classe **AppComponent** do Código 2 há apenas o atributo **title** que armazena o título a ser exibido no seu *template*. Esse título foi modificado para ‘primeira aplicação’.

No decorador **@Component** há 3 (três) partes de declaração, a saber, *selector*, *templateURL* e *styleUrls*. Em *selector* é fornecido um nome para que se possa selecionar o componente como uma *tag HTML*. No caso do componente **AppComponent** a sua seleção é feita na página **index.html** (Código 3) encontrada no diretório principal da aplicação (**primeira-app/index.html**).

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Primeira aplicação</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Código 3: Página *index.html*. Fonte: autoria própria.

Normalmente, a página **index.html** será muito pouco alterada, pois o código HTML estará contido na maior parte nos *templates* dos componentes da aplicação. A *tag* **<app-root></app-root>** irá inserir no *body* da página o código do *template* para o componente **AppComponent.** Esse

conteúdo é o que aparece na janela do navegador na Figura 3. A descrição do arquivo onde está localizado esse *template* é feita em *templateUrl* no decorador **`@Component`**. No caso, ‘./app.component.html’. O ./ (*ponto-barra*) indica que o arquivo encontra-se no mesmo diretório do componente. Por fim, *styleUrls* indica o arquivo de estilos que para o componente principal é **`app.component.css`**.

Vamos examinar agora o código do *template* para o componente **`AppComponent`** (Código 4). No exemplo, ele é o responsável pela geração do conteúdo que aparece no *body* da janela principal da aplicação (Figura 3).

```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Bem vindo à sua {{ title }}!
  </h1>
  <img width="300" alt="Angular Logo" src="data:image/svg+xml;base64">
</div>
<h2>Here are some links to help you start: </h2>
<ul>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://angular.io/tutorial">Tour of
Heroes</a></h2>
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://github.com/angular/angular-
cli/wiki">CLI Documentation</a></h2>
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://blog.angular.io/">Angular blog</a></h2>
  </li>
</ul>
```

Código 4: *Template* para o *AppComponent*. Obs.: O conteúdo de “src” da *tag img* foi abreviado por ser muito longo.
Fonte: autoria própria gerado com o @angular/cli.

No *template* da Figura 4 é possível observar o texto “*Bem vindo à sua {{title}}*” dentro da *tag h1*. O código composto por **`{{ }}`** é denominado de interpolação e permite a impressão de valores do componente em alguma *tag* do *template*, tais como atributos, resultados de chamadas de métodos ou expressões válidas em *TypeScript* compostas por constantes, variáveis, atributos ou chamadas de métodos. No exemplo, é impresso o texto do atributo **`title`** de **`AppComponent`**. A interpolação é apenas um recurso de comunicação *componente-template*, veremos outros mais adiante.

Recapitulando, *template* é o que será “desenhado” na tela para um determinado componente. É o seu código HTML podendo ser composto também por elementos e *tags* do *Angular*. O *template* pode enviar informações para o componente e receber informações dele. Ao rodar a aplicação, o módulo **`AppModule`** é executado. Ele define que o componente **`AppComponent`** é o ponto de entrada da aplicação. Esse componente por sua vez define o seu seletor (**`<app-root></app-root>`**) e quais são seus arquivos de *template* e *css*. O conteúdo do *template* do componente principal é inserido no corpo da página **`index.html`** através do seletor **`<app-root></app-root>`** e o seu conteúdo é exibido ao usuário ao se rodar a página **`index.html`** por meio da chamada à URL http://localhost:4200.

6 Criação de um novo componente

Nesta seção vamos demonstrar como criar um novo componente “nosso” e inseri-lo na página através do *template* do componente principal. Esse componente terá por objetivo

inicialmente apenas exibir uma mensagem de boas vindas para o usuário. Posteriormente iremos incrementá-lo para obter o nome do usuário por meio de um campo *input* de formulário e inserir um pouco de CSS.

O nome do componente é **exibir-mensagem**. Antes de executar o comando de criação, abra do terminal de comando e verifique se está na página do projeto (pasta **primeiro-app**). Nessa pasta, digite o seguinte comando:

```
ng g c exibir-mensagem
```

No comando, o "g" quer dizer *generate* e "c" *component*. Ou seja, "gerar componente". O comando também pode ser escrito com as palavras em inglês de forma completa, se preferir:

```
ng generate component exibir-mensagem
```

A execução do comando irá criar no diretório **src/app** uma pasta com o nome do componente (**exibir-mensagem**) e dentro dela os 4 (quatro) arquivos padrão para um componente, a saber, um *HTML*, um *CSS*, um *TypeScript* e um arquivo para testes conforme explicação na secção 5 para o componente principal. O módulo principal **AppModule** é modificado automaticamente para incluir o componente recém criado na seção **declarations** do *decorator* **@NgModule**.

```
bruno@bruno-linux:~/angular/primeiro-app$ ng g c exibir-mensagem
  create src/app/exibir-mensagem/exibir-mensagem.component.css (0 bytes)
  create src/app/exibir-mensagem/exibir-mensagem.component.html (34 bytes)
  create src/app/exibir-mensagem/exibir-mensagem.component.spec.ts (685 bytes)
  create src/app/exibir-mensagem/exibir-mensagem.component.ts (304 bytes)
  update src/app/app.module.ts (430 bytes)
```

O seletor do componente é **app-exibir-mensagem**. Assim, para exibirmos o seu *template* na página principal iremos colocar esse seletor como uma *tag* no *template* do componente principal. Para tanto, abra o arquivo **app.component.html**, apague o seu conteúdo que vem por padrão e insira a *tag* **<app-exibir-mensagem></app-exibir-mensagem>** conforme o Código 5.

```
<app-exibir-mensagem></app-exibir-mensagem>
```

Código 5. Seletor do componente ExibirMensagemComponent inserido em app.component.html. Fonte: autoria própria.

Agora verifique o resultado através da URL http://localhost:4200 no seu navegador. O resultado deve ser a mensagem "*app-mensagem works!*". Essa mensagem é conteúdo HTML do arquivo de *template* **exibir-mensagem.component.html**. Caso a aplicação não esteja rodando e o resultado não esteja aparecendo, abra uma nova aba de terminal, vá para o diretório da aplicação **primeiro-app** e digite o comando **ng serve**, conforme demonstrado na seção 4. Depois disso abra a URL.

Na pasta **src/app/exibir-mensagem** abra o arquivo de *template* **exibir-mensagem.component.html** e altere o texto no parágrafo para "Seja benvindo" e o seu nome, por exemplo, "Seja benvindo, Bruno". Vá ao navegador e veja o resultado da alteração. A seção 7 a seguir apresenta como obter o nome do usuário no *template* por meio de um formulário.

7 Obtendo o nome do usuário a partir de uma entrada de formulário

Nesta seção inserimos mais elementos para interação do componente com o *template*. Já vimos a marcação de interpolação (**{{}}**) para obter um informação do componente e imprimir em uma *tag* do *template* ou simplesmente escrever o resultado de uma expressão. Agora, vamos incluir o conceito de *evento*, que é uma forma de enviar uma informação em sentido inverso, ou seja, do *template* para o componente. No caso, o que se quer é passar o nome da pessoa a ser digitado em um campo *input* de formulário no *template* para uma método no componente que irá alterar a saudação de boas-vindas incluindo esse nome.

Inicialmente, como iremos trabalhar com formulário, vamos incluir um módulo específico para isso, denominado de **FormsModule**, com um *import* no início do arquivo **exibir-mensagem.component.ts** e na seção **imports** do decorador **@NgModule** da classe **AppModule**, conforme destacado nas linhas 3 e 13 no Código 6 a seguir. Neste exemplo, o formulário é bem simples e os elementos de **FormsModule** ainda não serão explorados. Porém, tendo em vista que precisaremos deste módulo mais adiante no curso, é interessante fazer essa importação.

```
1. import { BrowserModule } from '@angular/platform-browser';
2. import { NgModule } from '@angular/core';
3. import { FormsModule } from '@angular/forms';

4. import { AppComponent } from './app.component';
5. import { ExibirMensagemComponent } from './exibir-mensagem/exibir-
mensagem.component';

6. @NgModule({
7.  declarations: [
8.    AppComponent,
9.    ExibirMensagemComponent
10.  ],
11.  imports: [
12.    BrowserModule,
13.    FormsModule
14.  ],
15.  providers: [],
16.  bootstrap: [AppComponent]
17. })
18. export class AppModule { }
```

Agora abra o arquivo do *template* **exibir-mensagem.component.html** e altere-o conforme o Código 7.

```
<form>
<label for="nome">Digite seu nome</label>
<input id="nome" name="nome" type="text"
       #nome (change)="alterarMensagem(nome.value)">
</form>
```

Código 7: Formulário para obter o nome do usuário.

No formulário do código 7, os elementos específicos do *Angular* são os códigos **#nome** e **(change)**. No primeiro caso, **#nome** é uma variável local ao *template* que foi inserida para se obter uma referência ao DOM (*document object model*) do objeto *input* no qual ela está definida. Essa

variável é necessária no exemplo para se obter o atributo *value* da *tag input* (*nome.value*), ou seja, o texto digitado pelo usuário.

O nome entre parênteses representa um evento no *Angular*. No caso, o evento "*change*". Há diversos tipos de eventos, como o evento de clique, de pressionamento de teclas, de perda de foco em um elemento, dentre outros. No *Angular*, um evento é o gatilho para uma ação a ser realizada no componente. O evento *change* realiza uma ação sempre que a sua *tag* relacionada for modificada. No caso do exemplo, será chamado um método do componente **`ExibirMensagemComponent`** denominado **`alterarMensagem`** que recebe o valor digitado pelo usuário **`alterarMensagem(nome.value)`**. Como este método ainda não está incluído no componente, abra o arquivo **`exibir.mensagem.component.ts`** para adicioná-lo à classe **`ExibirMensagemComponent`** conforme o Código 8.

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

@Component({
  selector: 'app-exibir-mensagem',
  templateUrl: './exibir-mensagem.component.html',
  styleUrls: ['./exibir-mensagem.component.css']
})
export class ExibirMensagemComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

  alterarMensagem(n: string) {
    console.log(`Seja benvido, ${n}!`);
  }
}
```

Código 8: Componente *ExibirMensagemComponent* com método *alterarMensagem* adicionado. Fonte: autoria própria.

No caso do Código 8 o método **`alterarMensagem(n: string)`** irá receber o nome digitado no campo *input* do *template* em seu parâmetro (`n: string`) e exibir a mensagem no console do navegador. Para verificar a saída, vá para o seu navegador, abra a janela de ferramentas com um clique no botão **`F12`** e vá para a aba "console" da janela (*testado no Chrome e Firefox*). Posteriormente digite o seu nome no campo *input* do formulário e saia dele com um clique na janela do navegador ou uma tabulação (tecla *tab*). O resultado é exibido na Figura 4.

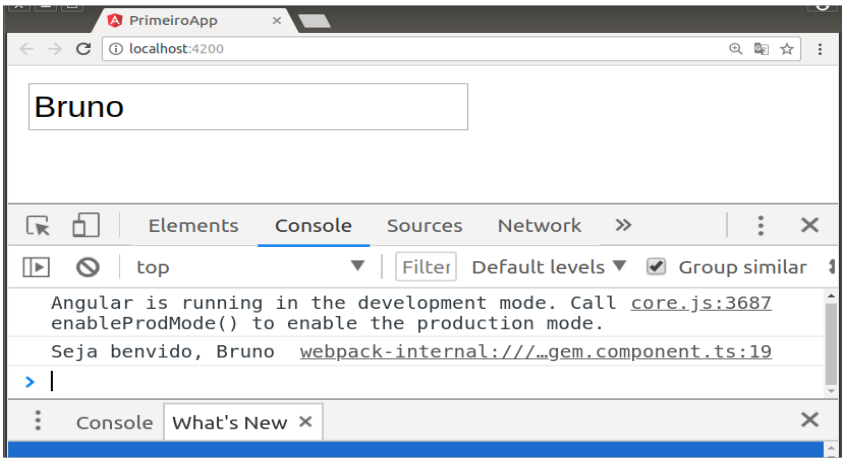

Figura 4: Componente ExibirMensagemComponent rodando no navegador com a mensagem de boas vindas sendo exibida no console. Fonte: autoria própria.

Agora, ao invés de exibir o resultado no *console*, vamos exibir no próprio *template*. Para tanto, modifique o código do componente e adicione um atributo "**mensagem**" responsável por guardar a mensagem que será exibida. Tendo em vista que no momento em que a página for exibida o usuário o campo *input* estará vazio, o atributo mensagem deve ser inicializado com o texto vazio. Isso deve ser feito no método construtor (*constructor*) do componente. A mensagem será alterada sempre que o método **alterarMensagem** for chamado, conforme pode ser visto no Código 9. Desse modo, para exibir a mensagem no *template* basta inserir o nome do atributo **mensagem** em uma interpolação.

```
//...trecho de código anterior omitido
export class ExibirMensagemComponent implements OnInit {
  mensagem: string;

  constructor() {
    this.mensagem = '';
  }

  ngOnInit() {
  }

  alterarMensagem(n: string) {
    this.mensagem = `Seja benvido, ${n}!`;
  }
}
```

Código 9. Componente *ExibirMensagemComponent* alterado para armazenar a mensagem em um atributo. Fonte: autoria própria.

Para destacar a mensagem acrescente no arquivo **exibir-mensagem.component.css** (Código 10) uma classe CSS a ser aplicada na divisão (*div*) que irá exibir a mensagem. Nessa classe, coloque uma cor de fundo (*background-color*) para a *div*.

```
div.msg {
  background-color: #CED8F6;
}
```

Código 10. Classe a ser aplicada em uma *div* para destacar o texto da mensagem.

O próximo passo é exibir o resultado direto no *template* **exibir-mensagem.component.html** e para isso crie uma divisão (**div**) com a referência à classe definida anteriormente e adicione o atributo mensagem entre as marcações de interpolação (**{{}}**) em um parágrafo (**p**), conforme a exibido no Código 11.

```
<form>
  <label for="nome">Digite seu nome</label>
  <input id="nome" name="nome" type="text"
         #nome (change)="alterarMensagem(nome.value)">
</form>

<div class="msg">
  <p>{{mensagem}}</p>
</div>
```

Código 11. Método *alterarMensagem* modificado para retornar a mensagem de boas vindas. Fonte: autoria própria.

A Figura 5 exibe o resultado final da execução da aplicação no navegador.

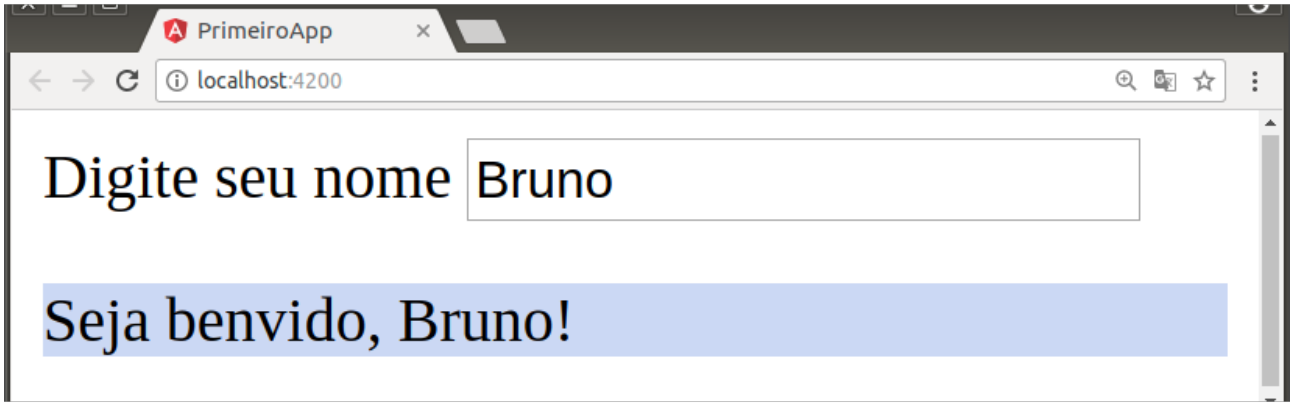

Figura 5: Resultado da alteração do componente ExibirMensagemComponent para que a mensagem seja mostrada no template. Fonte: autoria própria.

8 Considerações Finais

Neste tutorial vimos como instalar as ferramentas básicas do Angular para desenvolver uma aplicação composta de um componente simples para obter uma entrada e exibir uma mensagem para o usuário. Mesmo neste exemplo pequeno introduzimos comandos e conceitos importantes do *framework Angular*, como a compreensão dos conceitos de módulo e componente, a comunicação de *template* e componente por meio de eventos e interpolação, bem como as tarefas de criação e execução do desenvolvimento usando os comandos do @angular/cli.

Sugiro uma olhada nas documentações do framework nas referências deste documento para que você possa se aprofundar melhor nesses conceitos iniciais. E, apesar de parecer à complexo primeira vista, com a prática o desenvolvedor percebe que o aprendizado do *framework* não é tão complicada assim e que seus benefícios superam as dificuldades iniciais.

Referências

ANGULAR. **Documentação oficial do Angular**. Disponível em: https://angular.io/docs. Acesso em: 29/03/2018.

ANGULAR. **Angular CLI**. Disponível em: https://cli.angular.io/. Acesso em: 29/03/2018.

PAPA, John. **Introducing Angular Modules**. Disponível em: https://johnpapa.net/introducing-angular-modules-root-module/. Acesso em: 29/03/2018.