Olá, tudo bem?

Nesta videoaula aula, você continuará seus estudos sobre o uso do Angular e aprenderá acerca da criação de componentes e interpolação ou como é conhecido o data binding. Por fim, entenderá como se dá o repasse de dados estáticos através de seu componente. Então, vamos iniciar sua jornada pelo Angular, um dos frameworks front-end mais famosos do mundo.

A web vem evoluindo radicalmente desde sua criação nos anos 90. Com esse progresso, alguns marcos ocorreram, além de trazer um modo de interação diferenciado para os usuários da época. Entre eles têm-se: a web 2.0 voltada a conteúdos dinâmicos, que tinha, como a finalidade, a criação de ambientes colaborativos; e a web 3.0 chamada de web semântica, que visa novos modelos de busca de informação e interação entre homem e máquina.

Diante disso, surgiram as primeiras soluções que tinham, como princípio, possibilitar a reutilização de códigos HTML e JavaScript, mas sem especificação, ou seja, um conjunto de padrões a serem adotados para solucionar o problema de reuso da web. Assim, surgiu a componentização, também conhecida como web components.

Agora, você deve estar se perguntando, “mas o que tem a ver a componentização com a história da web?” Bem, praticamente toda a arquitetura, por trás da componentização, foi criada a partir dos marcos históricos citados anteriormente. Portanto, em linhas gerais, a componentização é um termo utilizado para descrever um conjunto de regras e padrões para se criar componentes na web.

Dessa forma, a reutilização de código é uma técnica praticada em vários paradigmas de linguagem de programação. Quando isso é aplicado a web, observa-se que há um certo grau de dificuldade em criar marcações customizadas com estruturas complexas de HTML e JavaScript, que precisam ser renderizadas e muitas vezes repetidas em várias partes do código. Pensando nesse impedimento de reuso, dentro da web, foram criados os componentes (Web Components), ou componentização, que são conjuntos de tecnologias que possibilitam a criação de elementos (ou TAGs) personalizadas, reutilizáveis e direcionadas a aplicações webs, tendo suas funcionalidades encapsuladas. Para mais informações acesse o site https://www.webcomponents.org/ (https dois pontos barra barra www ponto webcomponents ponto org barra), lá você encontrará a base dos web components, tutoriais e exemplos.

Até aqui, você conheceu um pouco sobre web componentes. A partir de agora, você aprenderá a criar seu primeiro componente. Ele deverá ser composto pelo nome do filme e o ano de publicação.

Para isso, abra o programa Visual Studio Code (VS Code) e navegue até src/app (src barra app). Em primeiro lugar, você deve criar um diretório chamado filmes dentro da pasta app no projeto

catalogodefilme, pois ele irá conter o conteúdo direcionado ao componente. Agora, navegue até a pasta app e clique com o botão direito sobre ela, para que seja exibida uma caixa de diálogo. Em seguida, escolha a opção new file. Um campo de texto irá aparecer para que você informe o caminho juntamente com o nome do arquivo a ser criado. Nesse campo, adicione o valor filmes/filme.component.ts (filmes barra filme ponto component ponto ts). Isto indicará, ao VS Code, que ele precisa criar um diretório chamado filmes, e incluirá, nele, o arquivo filme.component.ts (filme ponto component ponto ts).

#Audiodescrição: A imagem mostra a tela do programa Visual Studio Code. Nela, estão sendo exibidas as pastas expandidas src/app. Dentro da pasta app, foi inserido o diretório “filmes”, que está sendo destacado pelo cursor.

Nesse momento, no arquivo filme.component.ts (filme ponto component ponto ts), você irá criar a classe FilmeComponent. Sua estrutura é semelhante à de uma classe da linguagem Java, referente à especificação ECMAScript 2015, também conhecida como ES6, que é um conjunto de padrões e regras para serem adotados em linguagem do tipo script. Nesse caso, um pacote de melhorias e novidades.

#Audiodescrição: A imagem mostra a tela do programa Visual Studio Code. Nela, estão sendo exibidas as pastas expandidas src/app/filmes. Dentro da pasta filmes, há o arquivo “filme.component.ts”, que está sendo destacado pelo cursor. À direita, há o código “class FilmeComponent {
}”

Agora, você precisa informar ao Angular que a classe FilmeComponent trata-se de um componente. Para isso, use um decorator ou anotação, que é bastante comum em linguagens Java e C#. Depois, é preciso importar a diretiva @Component (arroba component) do core do Angular, que se encontra no pacote @angular/core (arroba angular barra core). Feito isso, adicione, sobre a classe, o decorator @Component (arroba component).

#Audiodescrição: A imagem mostra a tela do programa Visual Studio Code. Nela, estão sendo exibidas as pastas expandidas src/app/filmes. Dentro da pasta filmes, há o arquivo “filme.component.ts”, que está sendo destacado pelo cursor. À direita, há o código a seguir:
“import { Component } from ‘@angular/core’;

@Component()

class FilmeComponent {
}”

O decorator @Component (arroba component) possui algumas propriedades ou metadados que definem o comportamento do componente. Dentre eles têm-se o selector, template, styles e vários outros. Entretanto, você irá utilizar somente esses dois primeiros para criação do componente, porque o metadado selector é responsável por definir qual o seletor associado ao componente FilmeComponent, enquanto o template descreve o HTML a ser renderizado pelo Angular. Tudo isso é repassado para o decorator @Component (arroba component), como um objeto JSON.

O nome do seletor do componente FilmeComponent será filme e o template conterá um título e dois textos estáticos que informam o nome do filme e seu ano de publicação. Ah, lembre-se de que o HTML vinculado ao componente deve vir entre crases, já que estas são utilizadas pelo Angular, para indicar que se trata de um template a ser renderizando na view.

#Audiodescrição: A imagem mostra a tela do programa Visual Studio Code. Nela, estão sendo exibidas as pastas expandidas src/app/filmes. Dentro da pasta filmes, há o arquivo “filme.component.ts”, que está sendo destacado pelo cursor. À direita, há o código a seguir:

```
 import { Component } from '@angular/core';
@Component({
   selector: 'filme',
   template: `
            <h2> Catálogo de Filmes </h2>
            <label> Filme: {{nomeFilme}} </label>
            <br/>
            <label> Publicação: {{anoPublicacao}}  </label>
   `
})
export class FilmeComponent {
```

Com o componente FilmeComponent já criado, você irá aplicá-lo ao seu projeto, mas antes é necessário exportar o componente para que as demais estruturas, como componentes, módulos, templates e serviços, possam fazer uso.

Para tornar o componente acessível, deve-se exportá-lo. Para isso, inclua, antes da palavra reservada class, a instrução export. Isso tornará o componente FilmeComponent público e acessível a partir de qualquer componente, módulo, serviço ou template do Angular.

#Audiodescrição: A imagem mostra a tela do programa Visual Studio Code. Nela, estão sendo exibidas as pastas expandidas src/app/filmes. Dentro da pasta filmes, há o arquivo “filme.component.ts”, que está sendo destacado pelo cursor. À direita, há o código a seguir:

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

@Component({
    selector: 'filme',
    template: `
            <h2> Catálogo de Filmes </h2>
            <label> Filme: {{nomeFilme}} </label>
            <br/>
            <label> Publicação: {{anoPublicacao}}  </label>
    `
})
export class FilmeComponent {
}
```

Para verificar se o seu componente está em funcionamento, faça alguns ajustes nos arquivos que foram incluídos por padrão na criação do projeto. Qualquer componente, módulo, template ou serviço criado no Angular precisa ser declarado em um módulo. Por padrão, quando você cria um projeto no Angular, ele lhe disponibilizará o módulo app localizado em src/app/app.module.ts (src barra app barra app ponto module ponto ts). Esse módulo é o responsável por inicializar a sua aplicação e carregar todas as dependências necessárias.

No módulo app.module.ts (app ponto module ponto ts), especificamente no decorator @NgModule (arroba ngmodule), anotação que indica a declaração de um módulo, a propriedade declarations receberá mais um valor: o componente FilmeComponent criado anteriormente. A propriedade declarations recebe, como valor, um array de componentes, indicando que o módulo AppModule pode fazer uso deles em qualquer um dos componentes que o constitua.

#Audiodescrição: A imagem mostra a tela do programa Visual Studio Code. Nela, estão sendo exibidas as pastas expandidas src/app. Dentro da pasta app, há o arquivo “app.module.ts”, que está sendo destacado pelo cursor. À direita, há o código a seguir:

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

import { NgModule } from '@angular/core';

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

import { FilmeComponent } from './filmes/filme.component';

import { AutorComponent } from './autor/autor.component';

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

Para concluir essa etapa, navegue até o arquivo app.component.html (app ponto component ponto html), que se encontra em src/app (src barra app) e apague todo o conteúdo que ele possui. Em seguida, adicione somente o seletor vinculado ao componente FilmeComponent que é filme.

Código
<filme></filme>

Agora, você irá verificar se as configurações deram certo. Para isso, inicialize o Angular a partir do comando ng serve e acesse a URL http://localhost:4200/ (http dois ponto barra barra localhost dois ponto 4200 barra) a partir do seu navegador. Lá, você verá o componente FilmeComponent renderizado, com o título Catálogo de Filmes e algumas informações do filme Avatar, como nome e ano de publicação. Caso você não visualize o conteúdo da página, verifique, no console do navegador, os logs de erros lançados e reinicie o processo de configuração.

#Audiodescrição: A imagem mostra a barra de endereço preenchida por “localhost:4200”. Logo abaixo, é exibido o texto: “Catálogo de Filmes

Filme: Avatar
Publicação: 2009”.

Até esse momento, você pôde observar que todas as etapas para a criação de um componente no Angular foram realizadas de forma manual. Contudo, tais ações podem ser realizadas de forma automática, a partir do angular cli, através do comando ng generate component, seguido do nome do componente, conforme estudado na aula 13. Esse comando irá gerar arquivos HTML, folhas de estilo e de teste, assim como o do próprio componente, que define sua estrutura e comportamento. Além disso, atualizará o módulo de inicialização da aplicação o app.module.ts com a do novo componente criado.

Nesse ponto da videoaula, será recapitulado o conceito de databinding, também conhecido como interpolação no Angular.

Você deve preparar seu componente para enviar os dados contidos nele para o template, ou seja, o HTML vinculado ao seu componente. Essa técnica é chamada de interpolação e é um tipo de data binding. Na prática, ele é um procedimento de associação de dados entre o componente e o template. Na videoaula 12, sobre API REST, você estudou sobre data binding. Lá, o contexto era de criação de

instâncias ou objetos Java a partir de JSON. Contudo, nesta aula, você aplicará o data binding para realizar a transferência de dados entre o componente e a view.

Inicialize o VS Code e localize o componente FilmeComponent em app/filmes/filme.component.ts (app barra filmes barra filme ponto component ponto ts), após isso crie dois atributos nomeFilme do tipo string e anoPublicacao do tipo number no componente. O JavaScript, por padrão, não é uma linguagem fortemente tipada. Então, aqui, você irá fazer uso do TypeScript para declarar e incluir os tipos das variáveis nomeFilme e anoPublicacao. Portanto, elas serão encarregadas de repassar para o template do componente os dados a serem renderizados pelo Angular.

#Audiodescrição: A imagem mostra a tela do programa Visual Studio Code. Nela, estão sendo exibidas as pastas expandidas src/app/filmes. Dentro da pasta filmes, há o arquivo “filme.component.ts”, que está sendo destacado pelo cursor. À direita, há o código a seguir:

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

@Component({
   selector: 'filme',
   template: `
            <h2> Catálogo de Filmes </h2>
            <label> Filme: {{nomeFilme}} </label>
            <br/>
            <label> Publicação: {{anoPublicacao}}  </label>
   `
})
export class FilmeComponent {
   nomeFilme: string;
   anoPublicacao: number;
}
```

Com os atributos nomeFilme e anoPublicacao já declarados, você deve criar um construtor no componente FilmeComponent para que as variáveis nomeFilme e anoPublicacao venham inicializadas, ou seja, com valores pré-definidos. Feito isso, estará preparado o componente para a exemplificação da

técnica de data binding. Assim, o construtor do componente FilmeComponent não terá nenhum parâmetro. Por isso, no seu bloco de declaração, atribua, ao nomeFilme, o valor Matrix; e, a anoPublicacao, o valor 1999. Isso fará com que o componente apresente, na página em HTML, os dados referentes ao filme Matrix. Lembre-se de que, para se fazer referência às variáveis criadas, basta utilizar a palavra reservada this antes do atributo.

#Audiodescrição: A imagem mostra a tela do programa Visual Studio Code. Nela, estão sendo exibidas as pastas expandidas src/app/filmes. Dentro da pasta filmes, há o arquivo “filme.component.ts”, que está sendo destacado pelo cursor. À direita, há o código a seguir:

```
import { Component } from '@angular/core';
@Component({
   selector: 'filme',
   template: `
            <h2> Catálogo de Filmes </h2>
            <label> Filme: {{nomeFilme}} </label>
            <br/>
            <label> Publicação: {{anoPublicacao}}  </label>
   `
})
export class FilmeComponent {

   nomeFilme: string;
   anoPublicacao: number;

   constructor() {
      this.nomeFilme = 'Matrix';
      this.anoPublicacao = 1999
   }

}
```

Para que o template renderize os valores vindos do componente FilmeComponent, adicione duas diretivas a ele, que são {{nomeFilme}} (abre chave chave novamente nomeFilme chave novamente fecha chave) e {{anoPublicacao}} (abre chave chave novamente anoPublicacao chave novamente fecha chave), no lugar dos valores Avatar e 2009, respectivamente. Dentro das chaves, deve estar o atributo do componente a ser exibido no template. As diretivas devem conter os mesmos nomes dos atributos do componente FilmeComponent, pois, assim, o Angular irá conseguir casar e realizar a transferência de dados.

#Audiodescrição: A imagem mostra a tela do programa Visual Studio Code. Nela, estão sendo exibidas as pastas expandidas src/app/filmes. Dentro da pasta filmes, há o arquivo “filme.component.ts”, que está sendo destacado pelo cursor. À direita, há o código a seguir:

```
import { Component } from '@angular/core';
@Component({
   selector: 'filme',
   template: `
            <h2> Catálogo de Filmes </h2>
            <label> Filme: {{nomeFilme}} </label>
            <br/>
            <label> Publicação: {{anoPublicacao}}  </label>
   `
})
export class FilmeComponent {

   nomeFilme: string;
   anoPublicacao: number;

   constructor() {
      this.nomeFilme = 'Matrix';
      this.anoPublicacao = 1999
   }

}
```

Seguido as orientações anteriores. Agora é hora de testar seu projeto. Para isso, inicialize o projeto catálogo de filmes a partir do terminal do VS Code com o comando ng serve. Isso fará com que sua

aplicação seja executada. Portanto, abra seu navegador e acesse a URL http://localhost:4200/ (http dois ponto barra barra localhost dois ponto 4200 barra). A tela exibirá o componente FilmeComponent renderizado com o título Catálogo de Filmes e algumas informações acerca do novo filme, que você definiu como valor padrão no construtor, que foi Matrix de 1999.

#Audiodescrição: A imagem mostra a barra de endereço preenchida por “localhost:4200”. Logo abaixo, é exibido o texto: “Catálogo de Filmes

Filme: Matrix
Publicação: 1999”.

Até aqui, você conheceu o conceito de componentização e os motivos deles terem surgidos. Também aprendeu a criar componentes no Angular e fez uso da técnica de interpolação para repassar dados do componente para os templates. Na próxima videoaula, você irá conhecer o services, um dos recursos disponíveis no Angular para agrupar as regras de negócio dos componentes.

Até a próxima aula e bons estudos!

Referências

ECMAScript. Disponível em: <https://pt.wikipedia.org/wiki/ECMAScript>. Acesso em: 18 jun 2020.

FERNANDES, Diego. **TypeScript:** *Vantagens, mitos, dicas e conceitos fundamentais*. Disponível em: <https://blog.rocketseat.com.br/typescript-vantagens-mitos-conceitos/>. Acesso em: 18 jun 2020.

LIMA, Matheus. **O Guia do ES6:** *TUDO que você precisa saber*. Disponível em: <https://medium.com/@matheusml/o-guia-do-es6-tudo-que-voc%C3%AA-precisa-saber-8c287876325f>. Acesso em: 18 jun 2020.

MARIANO, Matheus. **O mínimo que você precisa saber sobre TypeScript**. Disponível em: <https://medium.com/@matheusmariano/o-m%C3%ADnimo-que-voc%C3%AA-precisa-saber-sobre-typescript-58d1b418f78b>. Acesso em: 18 jun 2020.

MORAIS, Rafael. **Angular 5 para iniciantes:** *Componentes*. Disponível em: <https://medium.com/@rafaelmoraisdev/angular-5-para-iniciantes-componentes-e6d3c3a8a791>. Acesso em: 18 jun 2020.

PEDRO, Adilson de Almeida. **Bindings em Angular**. Disponível em: <http://blog.almeidapedro.com.br/post/2017/12/13/bindings-em-angular>. Acesso em: 18 jun 2020.

PEDRO, Adilson de Almeida. **Criando componentes em Angular**. Disponível em: <http://blog.almeidapedro.com.br/post/2017/11/23/components>. Acesso em: 18 jun 2020.

PORTO, Patrick. **Guia geral sobre Web Components**. Disponível em: <https://medium.com/@patrickporto/guia-geral-sobre-web-components-1974c7e5b1d9>. Acesso em: 18 jun 2020.

Qual é a diferença entre os tipos de binding no Angular? Disponível em: <https://pt.stackoverflow.com/questions/239600/qual-%C3%A9-a-diferen%C3%A7a-entre-os-tipos-de-binding-no-angular>. Acesso em: 18 jun 2020.

Web Components. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/Web_Components>. Acesso em: 18 jun 2020.

What are web components?. Disponível em: <https://www.webcomponents.org/introduction>. Acesso em: 18 jun 2020.