Instruções para a entrega: mostrar para o professor na aula do dia 03/set. A entrega pode ser em dupla.

Objetivo: Fazer uma SPA (Single Page Application) para fazer o cadastro de veículos, assim como é mostrado na Figura 1. Considere os seguintes requisitos:

- A aplicação deverá ser formada por 2 componentes Angular (form e tabela);
- Os componentes deverão estar centralizados horizontalmente e um abaixo do outro;
- Os dados deverão ser mantidos num array e exibidos no componente tabela;
- A interface deverá manter as características mostradas na Figura 1;
- Esta atividade deverá ser feita e entregue no site https://stackblitz.com/. Essa ferramenta deixa o projeto salvo e poderá ser disponibilizado no GitHub, porém é necessário efetuar login. Recomenda-se usar a sua conta no GitHub para efetuar o cadastro no StackBlitz.

Figura 1 – Aplicação Angular para manter um cadastro de carros e caminhões.

Siga os passos a seguir após criar a conta e efetuar o login no StackBlitz.

Passo 1: Crie um projeto Angular. O nome do projeto e sua URL é gerada automaticamente pela ferramenta StackBlitz. O projeto inicia-se com a estrutura mostrada na Figura 2.

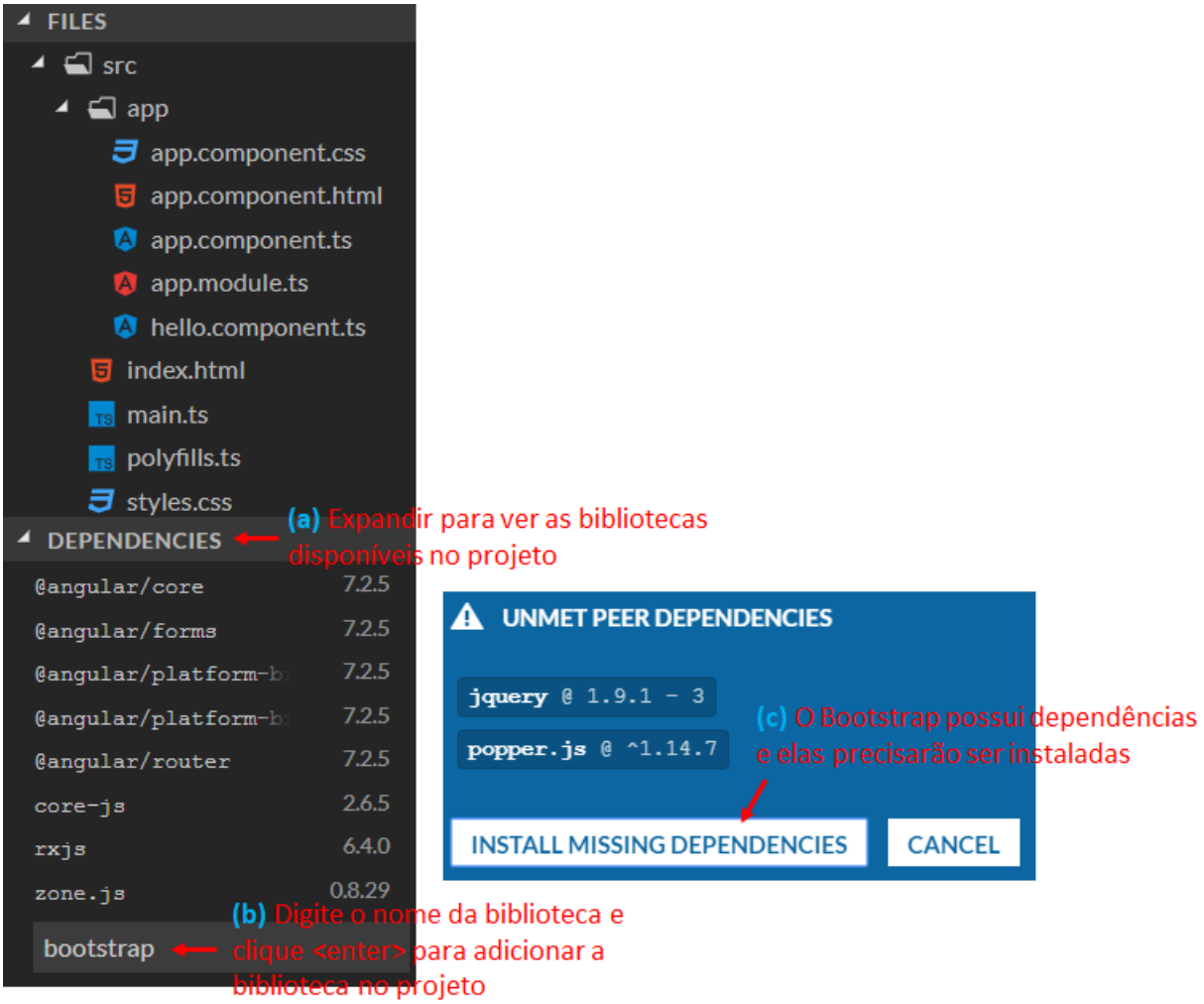

Figura 2 – Adicionar o framework Bootstrap 4 e suas dependências no projeto.

Passo 2: Para adicionar uma biblioteca no projeto precisamos acessar a interface de dependências do projeto - sinalizado por (a) na Figura 2. Siga os passos da Figura 2 para adicionar o framework Bootstrap 4 no seu projeto, mas para importar o código do framework em todo o projeto precisamos incluir a instrução de importação

```
@import '~bootstrap/dist/css/bootstrap.min.css';
```

no arquivo `src/styles.css` (Figura 3).

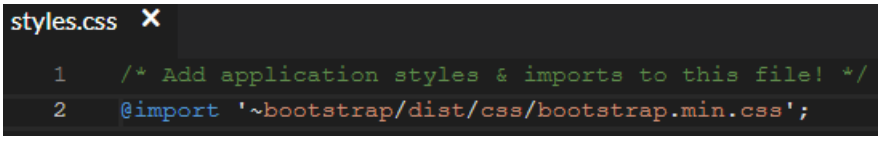

Figura 3 – Importar o framework Bootstrap 4 no projeto.

Passo 3: Para adicionar um componente na aplicação é necessário clicar com o botão direito sobre o a pasta `app` e selecionar `Angular Generator > Component` (Figura 4). Repita esse procedimento duas vezes para criar os componentes `form` e `tabela`. Esse procedimento faz com que os seguintes comandos sejam executados:

```
ng generate component form
```

```
ng generate component tabela
```

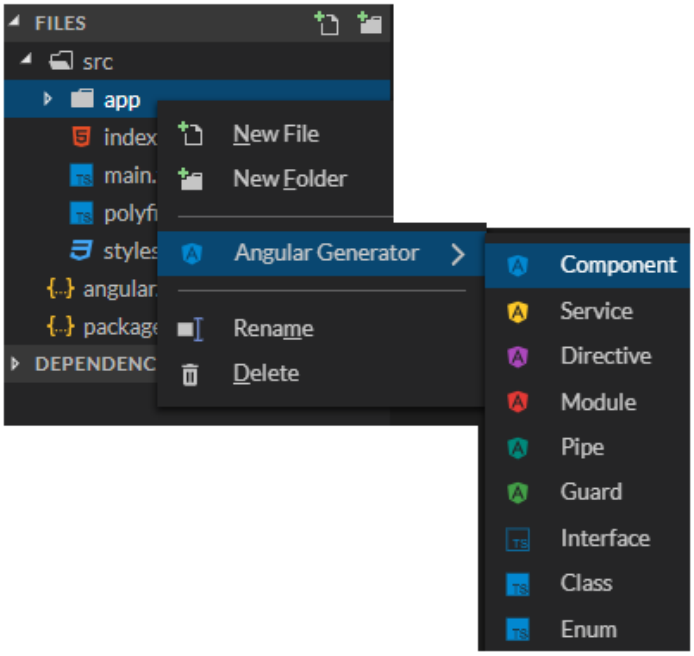

Figura 4 – Adicionar os componentes e serviço na aplicação.

Passo 4: Para adicionar um serviço na aplicação é necessário clicar com o botão direito sobre o a pasta `app` e selecionar `Angular Generator > Service` (Figura 4). O serviço deverá ter o nome de `servico`. Esse procedimento faz com que o seguinte comando seja executado:

```
ng generate service servico
```

Passo 5: Para adicionar uma classe na aplicação é necessário clicar com o botão direito sobre o a pasta `app` e selecionar `Angular Generator > Class` (Figura 4). A classe deverá ter o nome de `veiculo`. Esse procedimento faz com que o seguinte comando seja executado:

```
ng generate class veiculo
```

Copie o código da Figura 5 no arquivo `src/app/veiculo.ts`, aqui estamos definindo a classe `Veiculo`.

```
export class Veiculo {
  marca: string;
  modelo: string;
  valor: number;
}
```

Figura 5 – Conteúdo do arquivo `src/app/veiculo.ts`.

Passo 6: Os componentes precisam ser colocados no componente root (Figura 6), então copie o código da Figura 6 para o arquivo `src/app/app.component.html`.

A classe Bootstrap `justify-content-center` é usada para centralizar os componentes na horizontal (https://getbootstrap.com/docs/4.0/utilities/flex/). Lembre-se que a classe `mt` é `margin-top`.

```
<div class="container">
  <div class="row justify-content-center mt-4">
  <app-form></app-form>
  </div>
   <div class="row justify-content-center mt-4">
  <app-tabela></app-tabela>
  </div>
</div>
```

Figura 6 – Conteúdo do arquivo `src/app/app.component.html`.

Passo 7: Copie o código da Figura 7 no arquivo `src/app/servico.service.ts`, aqui estamos definindo o array que mantém os veículos cadastrados.

```
import { Injectable } from '@angular/core';
import { Veiculo } from './veiculo';

@Injectable()
export class ServicoService {
  public lista:Veiculo[] = [];

  constructor() { }

  add(veiculo: Veiculo): void {
    let aux: Veiculo = {
      marca: veiculo.marca,
      modelo: veiculo.modelo,
      valor: veiculo.valor
    };
    this.lista.push(veiculo);
  }
}
```

Figura 7 – Conteúdo do arquivo `src/app/servico.service.ts`.

Passo 8: Copie o código da Figura 8 no arquivo `src/app/form/form.component.ts`, aqui estamos definindo a classe `FormComponent`.

```
import { Component, OnInit } from '@angular/core';
import { ServicoService } from '../servico.service';
import { Veiculo } from '../veiculo';

@Component({
  selector: 'app-form',
  templateUrl: './form.component.html',
  styleUrls: ['./form.component.css']
})
export class FormComponent implements OnInit {
  private marcas: string[] = ['Fiat', 'Ford', 'GM', 'Volkswagen'];
```

```
  private veiculo: Veiculo;

  constructor(private servico: ServicoService) { }

  ngOnInit() {
    this.reset();
  }

  salvar(): void {
    this.servico.add(this.veiculo);
    this.reset();
  }

  reset(): void {
    this.veiculo = {
      marca: undefined,
      modelo: undefined,
      valor: undefined
    };
  }
}
```

Figura 8 – Conteúdo do arquivo `src/app/form/form.component.ts`.

Passo 9: Você precisa fazer o código HTML do arquivo `src/app/form/form.component.html`.

Passo 10: Você precisa fazer o código HTML do arquivo `src/app/tabela/tabela.component.ts`.

Passo 11: Você precisa fazer o código HTML do arquivo `src/app/tabela/tabela.component.html`.