Instruções para a entrega: mostrar para o professor na aula do dia 12/set.

Objetivo: Fazer uma SPA (Single Page Application) para fazer o cadastro de usuários, 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;
- Somente o campo nome é obrigatório;
- 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 alunos.

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.

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 a biblioteca Bootstrap 4 e suas dependências.

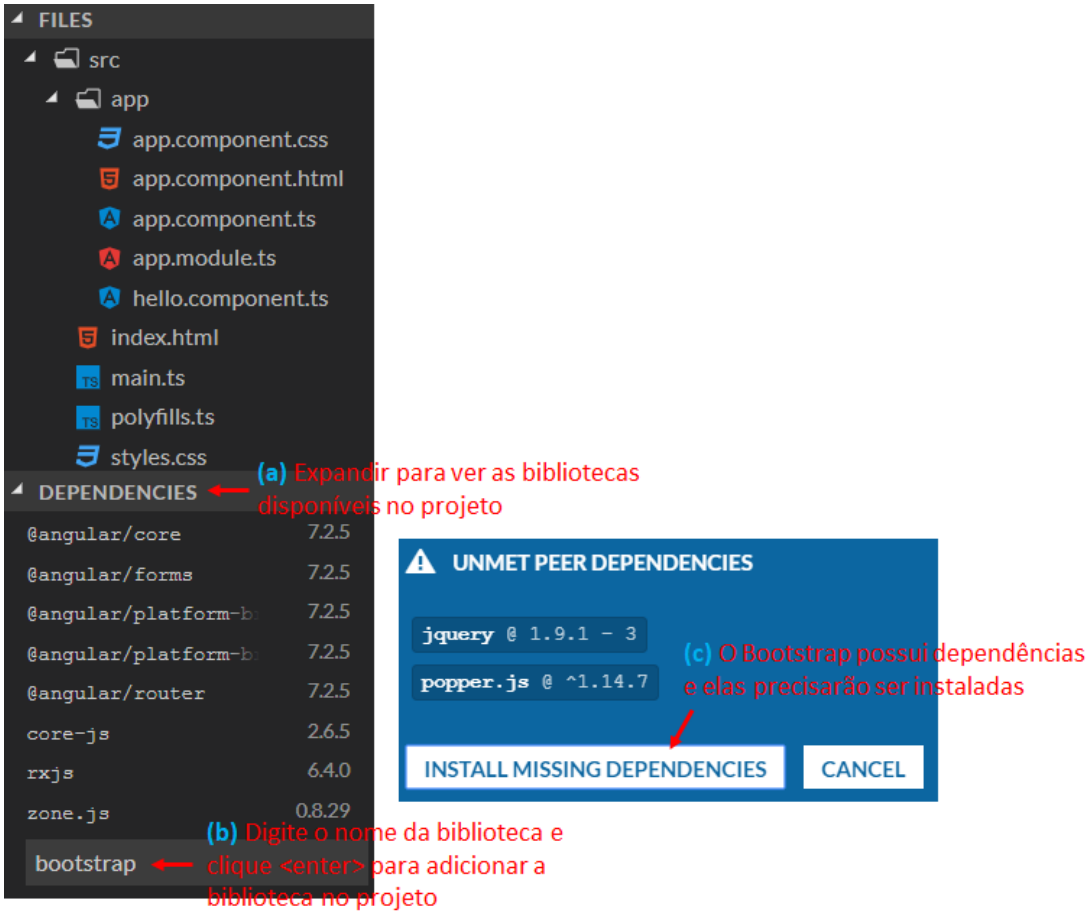

Figura 2 – Adicionar a biblioteca Bootstrap 4 e suas dependências na nossa aplicação.

Passo 3: Para usar os estilos Bootstrap na nossa aplicação precisamos importar o arquivo CSS na nossa aplicação, então coloque a instrução a seguir no arquivo `src/styles.css`:

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

Passo 4: Para adicionar um componente na aplicação é necessário clicar com o botão direito sobre o a pasta **app** e selecionar `Angular Generator > Component`. Repita esse procedimento duas vezes para criar os componentes `form` e `tabela`.

Passo 5: Os serviços são usados para fazer a comunicação entre os componentes. Para adicionar um serviço na aplicação é necessário clicar com o botão direito sobre o a pasta **app** e selecionar `Angular Generator > Service`. O serviço deverá ter o nome de `servico`.

Passo 6: As classes são usadas para definir tipos de dados. Aqui iremos criar a classe `Aluno` para definir um tipo de dado. Para adicionar uma classe na aplicação é necessário clicar com o botão direito sobre o a pasta **app** e selecionar `Angular Generator > Class`. A classe deverá ter o nome de `aluno`.

Copie o código da Figura 3 para o arquivo `src/app/aluno.ts`, aqui estamos definindo a classe `Aluno`. A interrogação indica que o atributo `sexo` não precisa ser fornecido.

```
export class Aluno {
  nome: string;
```

```
  sexo?: string;
}
```

Figura 3 – Conteúdo do arquivo `src/app/aluno.ts`.

Passo 7: O componente root da aplicação está definido nos arquivos `app.component.*`. Copie o código da Figura 4 para o arquivo `src/app/app.component.html`, veja que os componentes `<app-form>` e `<app-tabela>` são invocados a partir desse arquivo. As classes Bootstrap são usadas para limitar a largura horizontal dos elementos na página.

```
<h5 class="text-center mt-4">Cadastro de Alunos</h5>
<div class="container mt-4">
  <div class="row justify-content-center">
    <app-form class='col-12 col-sm-10 col-md-8 col-lg-6'></app-form>
  </div>
  <div class="row justify-content-center mt-4">
    <app-tabela class='col-12 col-sm-10 col-md-8 col-lg-6'></app-tabela>
  </div>
</div>
```

Figura 4 – Conteúdo do arquivo `src/app/app.component.html`.

Passo 8: Copie o código da Figura 5 no arquivo `src/app/servico.service.ts`. No serviço estamos definindo o atributo `lista` para manter o array de alunos cadastrados e as operações (métodos) para adicionar e remover elementos do array.

```
import { Injectable } from '@angular/core';
import { Aluno } from './aluno';

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

  constructor() { }

  add(aluno: Aluno): void {
    this.lista.push(aluno);
  }

  remove(aluno: Aluno): void {
    // procura o objeto aluno na lista
    let indice = this.lista.indexOf(aluno, 0);
    if (indice > -1) {
      this.lista.splice(indice, 1);
    }
  }
}
```

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

Passo 9: Copie os códigos da Figura 6 e Figura 7, respectivamente, para os arquivos `src/app/form/form.component.ts`. e `src/app/tabela/tabela.component.ts`. Essas classes provêm a

interface entre a view (HTML) dos componentes e as operações que estão no serviço. Os métodos `salvar` e `excluir` serão invocados na view dos componentes.

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

@Component({
  selector: 'app-form',
  templateUrl: './form.component.html',
  styleUrls: ['./form.component.css']
})
export class FormComponent implements OnInit {
  private aluno: Aluno;
  constructor(private servico: ServicoService) { }

  ngOnInit() {
    this.aluno = new Aluno(); /* cria um novo aluno */
  }

  salvar() {
    this.servico.add(this.aluno);
    this.aluno = new Aluno(); /* cria um novo aluno */
  }
}
```

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

```
import { Component, OnInit } from '@angular/core';
import { ServicoService } from '../servico.service';
import { Aluno } from '../aluno';
@Component({
  selector: 'app-tabela',
  templateUrl: './tabela.component.html',
  styleUrls: ['./tabela.component.css']
})
export class TabelaComponent implements OnInit {
  constructor(private servico:ServicoService) { }

  ngOnInit() { }

  excluir(aluno:Aluno){
    this.servico.remove(aluno);
    return false; /* para evitar o popup menu */
  }
}
```

Figura 7 – Conteúdo do arquivo `src/app/tabela/tabela.component.ts`.

Passo 10: Copie os estilos da Figura 8 para o arquivo `src/app/form/form.component.css`. Aqui estamos definindo os estilos para o campo de entrada, quando o campo nome for inválido a cor da borda esquerda será vermelha.

```
.ng-valid[required]  {
  border-left: 5px solid #42A948; /* green */
}

.ng-invalid:not(form)  {
  border-left: 5px solid #a94442; /* red */
}
```

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

Passo 11: Para fazer o código HTML do arquivo `src/app/form/form.component.html` sugere-se consultar o exemplo https://angular.io/generated/live-examples/forms/stackblitz.html e a explicação sobre Template-driven Forms (https://angular.io/guide/forms).

Passo 12: Para fazer o código HTML do arquivo `src/app/tabela/tabela.component.html` sugere-se consultar https://getbootstrap.com/docs/4.0/content/tables/. Lembrando que cada elemento do array `lista` – que está na classe `ServicoService` – deverá ser uma linha da tabela, então use a diretiva `*ngFor` para percorrer a `lista`.