Instruções para a entrega: mostrar para o professor na aula do dia 19/set.

Objetivo: Fazer uma SPA (Single Page Application) para manter o cadastro de matrículas. Cada matrícula é formada por aluno, disciplina e nota, assim como é mostrado na Figura 1 será necessário construir um formulário para cada cadastro. Considere os seguintes requisitos:

- A aplicação deverá ser formada por 3 componentes Angular, onde o formulário e tabela estará no mesmo componente;
- Os componentes deverão estar centralizados horizontalmente;
- Cada cadastro deverá ser mantido num array e exibidos na tabela do componente correspondente;
- No topo da página deverá ter uma barra de navegação com os links para os componentes. Use Angular Router (https://angular.io/guide/router) para direcionar os componentes para serem exibidos no elemento `<router-outlet>`;
- Somente o campo sexo não 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.

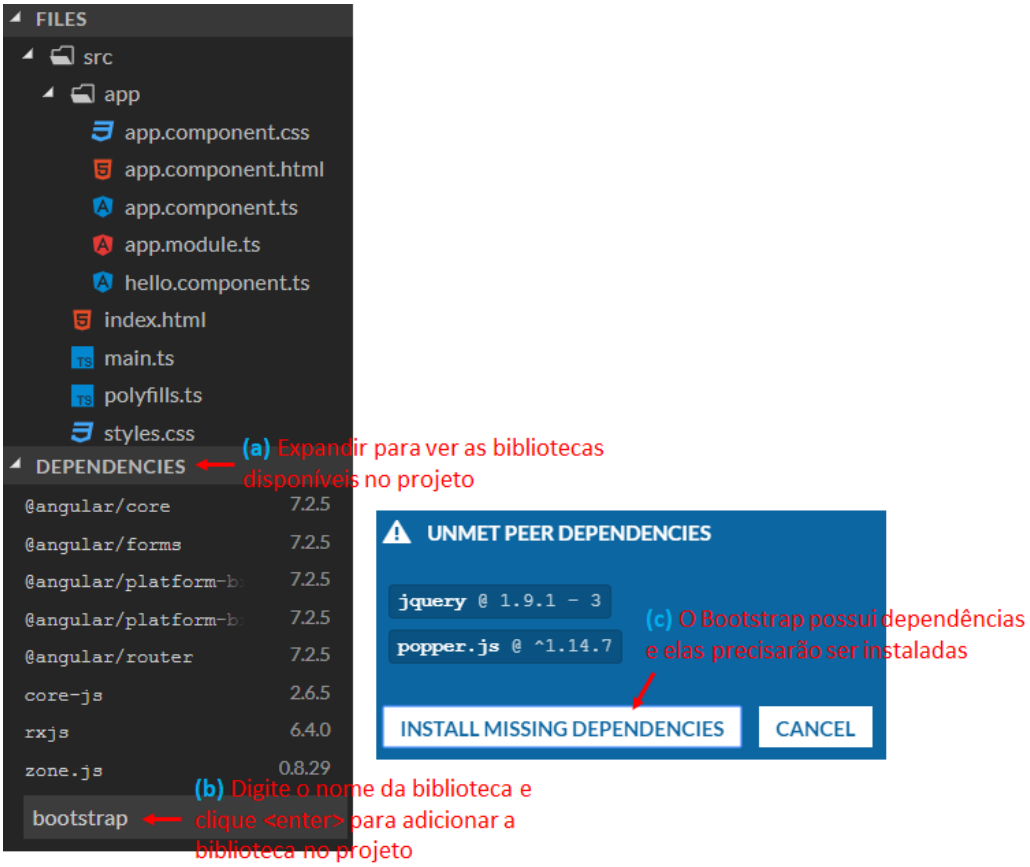

Figura 2 – Adicionar a biblioteca Bootstrap 4 e suas dependências na nossa aplicação.

Passo 3: Para usar os estilos Bootstrap na nossa aplicação precisamos importar o arquivo CSS na nossa aplicação, assim como é mostrado na Figura 3. Além disso, precisamos definir os estilos CSS para os campos de entrada e para a barra de navegação.

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

/* campos de entrada */
.ng-valid[required], .ng-valid.required  {
  border-left: 5px solid #42A948; /* green */
}
.ng-invalid:not(form)  {
  border-left: 5px solid #a94442; /* red */
}

/* estilo para a barra de navegação */
.ativo {color:red !important}
nav a {text-decoration: none !important}
```

Figura 3 – Conteúdo do arquivo `src/styles.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 3 vezes para criar os componentes `form-aluno`, `form-disciplina` e `form-matricula`.

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 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 as classes `Aluno`, `Disciplina` e `Matricula`. Para adicionar uma classe na aplicação é necessário clicar com o botão direito sobre o a pasta **app** e selecionar `Angular Generator > Class`. O arquivo de classes deverá ter o nome de `dados`.
Copie o código da Figura 4 para o arquivo `src/app/dados.ts`. A interrogação indica que o atributo `sexo` não precisa ser fornecido. Observe que os atributos `aluno` e `disciplina` da classe `Matricula` são, respectivamente, dos tipos de dados `Aluno` e `Disciplina`.

```
export class Aluno {
  nome: string;
  sexo?: string;
}

export class Disciplina {
  nome: string;
  carga: number;
}

export class Matricula {
  aluno: Aluno;
  disciplina: Disciplina;
  nota: number;
}
```

Figura 4 – Conteúdo do arquivo `src/app/dados.ts`.

Passo 7: No arquivo `app.module.ts` precisamos definir um array com as rotas. Cada rota é formada pela URL que dá acesso a ela e o componente que a URL mapeia, por exemplo, a URL `www.dominio.com/projeto/aluno` será direcionada para o componente `FormAlunoComponent`.

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

import { AppComponent } from './app.component';
import { FormAlunoComponent } from './form-aluno/form-aluno.component';
import { ServicoService } from './servico.service';
import { FormDisciplinaComponent } from './form-disciplina/form-disciplina.component';
```

```
import { FormMatriculaComponent } from './form-matricula/form-matricula.component';

import { RouterModule, Routes } from '@angular/router';

/* definição das rotas */
const rotas: Routes = [
  /* será chamado o componente FormAluno quando a URL endereçar /aluno */
  {path: 'aluno', component: FormAlunoComponent},
  /* será chamado o componente FormDisciplina quando a URL endereçar /disciplina */
  {path: 'disciplina', component: FormDisciplinaComponent},
  /* será chamado o componente FormMatricula quando a URL endereçar /matricula */
  {path: 'matricula', component: FormMatriculaComponent},
  /* será redirecionado para a URL /aluno quando a URL terminar na raiz / */
  {path: '', redirectTo: '/aluno', pathMatch: 'full'},
  /* será redirecionado para a URL /matricula quando a URL for desconhecida, por exemplo, /teste */
  {path: '**', redirectTo: '/matricula' }
];

@NgModule({
  imports:      [
    BrowserModule,
    FormsModule,
    /* é necessário registrar as rotas usando RouterModule.forRoot()
    Ao usar forRoot o serviço Router estará disponível em toda a aplicação */
    RouterModule.forRoot(rotas)
     ],
  declarations: [ AppComponent, FormAlunoComponent,
                  FormDisciplinaComponent, FormMatriculaComponent ],
  bootstrap:    [ AppComponent ],
  providers: [ServicoService]
})
export class AppModule { }
```

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

Passo 8: O componente root da aplicação está definido nos arquivos `app.component.*`. Copie o código da Figura 6 para o arquivo `src/app/app.component.html`, veja que os componentes são mapeados através de links que possuem a propriedade `routerLink` e serão exibidos no elemento `<router-outlet>`.

```
<div class="container-fluid">
  <nav class="navbar navbar-expand navbar-light bg-light">
    <ul class="navbar-nav">
      <li class="nav-item pl-2 pr-2 border-right">
        <a routerLink="/aluno" routerLinkActive="ativo">Aluno</a>
      </li>
      <li class="nav-item pl-2 pr-2 border-right">
        <a routerLink="/disciplina" routerLinkActive="ativo">Disciplina</a>
      </li>
      <li class="nav-item pl-2 pr-2">
        <a routerLink="/matricula" routerLinkActive="ativo">Matrícula</a>
      </li>
```

```
    </ul>
  </nav>
</div>
<div class="container">
  <div class="row justify-content-center">
    <div class='col-12 col-sm-10 col-md-8 col-lg-6'>
      <!-- A diretiva RouterOutlet é usada para receber os componentes mapeados -->
      <router-outlet></router-outlet>
  </div>
  </div>
</div>
```

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

Passo 9: Copie o código da Figura 7 no arquivo `src/app/servico.service.ts`. No serviço estamos definindo os arrays que mantêm os cadastros de alunos, disciplinas e matrículas, e os métodos para adicionar e remover elementos desses arrays.

```
import { Injectable } from '@angular/core';
import { Aluno, Disciplina, Matricula } from './dados';

@Injectable()
export class ServicoService {
  public alunos: Aluno[] = [{nome:'Ana',sexo:'Feminino'},{nome:'Pedro',sexo:'Masculino'}];
  public disciplinas: Disciplina[] = [{nome:'Álgebra',carga:80},{nome:'Algoritmo',carga:80}];
  public matriculas: Matricula[] = [];

  constructor() { }

  addAluno(aluno: Aluno): void {
    this.alunos.push(aluno);
  }

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

  addDisciplina(disciplina: Disciplina): void {
    this.disciplinas.push(disciplina);
  }

  removeDisciplina(disciplina: Disciplina) {
    let indice = this.disciplinas.indexOf(disciplina, 0);
    if (indice > -1) {
      this.disciplinas.splice(indice, 1);
    }
  }
```

```
  addMatricula(matricula: Matricula): void {
    this.matriculas.push(matricula);
  }

  removeMatricula(matricula: Matricula) {
    let indice = this.matriculas.indexOf(matricula, 0);
    if (indice > -1) {
      this.matriculas.splice(indice, 1);
    }
  }
}
```

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

Passo 10: A Figura 8 possui o código da classe `FormAlunoComponent`. Na classe estão os métodos `salvar` e `excluir` que serão invocados pelo view do componente (HTML) e a definição do objeto `Aluno` que mantém os dados do aluno que está sendo cadastrado. Repita o mesmo procedimento para codificar as classes `FormDisciplinaComponent` e `FormMatriculaComponent`.

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

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

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

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

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

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

Passo 11: A Figura 9 possui o código da view do `FormAlunoComponent`. Repita o mesmo procedimento para codificar as views dos componentes `FormDisciplinaComponent` e `FormMatriculaComponent`.

A Figura 10 mostra o código do campo de seleção para fornecer a carga horário no componente `FormDisciplinaComponent` e a Figura 11 mostra o código do campo de seleção do aluno para cadastrar uma matrícula.

```
<h5 class="text-center mt-4">Cadastro de Alunos</h5>
<form (ngSubmit)="salvar(); formulario.resetForm()" #formulario="ngForm">
  <div class="form-group mt-4">
    <label>Nome</label>
    <input [(ngModel)]="aluno.nome" class="form-control" name="nome" #nome="ngModel" required>
    <div [hidden]="nome.valid || nome.pristine" class="alert alert-danger">
      Nome é obrigatório
    </div>
  </div>
  <div class="form-group mt-3">
    <div>Sexo</div>
    <div class="form-check form-check-inline">
      <input class="form-check-input" type="radio" [(ngModel)]="aluno.sexo" name="sexo"
id="sexof" value="Feminino">
      <label class="form-check-label" for="sexof">
        Feminino
      </label>
    </div>
    <div class="form-check form-check-inline">
      <input class="form-check-input" type="radio" [(ngModel)]="aluno.sexo" name="sexo"
id="sexom" value="Masculino">
      <label class="form-check-label" for="sexom">
        Masculino
      </label>
    </div>
  </div>

  <div>
    <button type="submit" [disabled]="!formulario.form.valid" class="btn btn-success mr-
3">Salvar</button>
    <button type="reset" class="btn btn-light">Limpar</button>
  </div>
</form>

<div class="table-responsive-sm mt-4" *ngIf="servico.alunos.length > 0">
  <table class="table table-striped table-hover">
    <thead>
      <tr>
        <th scope="col">NOME</th>
        <th scope="col">SEXO</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let item of servico.alunos" (contextmenu)="excluir(item)">
        <td>{{item.nome}}</td>
```

```
        <td>{{item.sexo}}</td>
      </tr>
    </tbody>
  </table>
</div>
```

Figura 9 – Conteúdo do arquivo `src/app/form-aluno/form-aluno.component.html`.

```
  <div class="form-group mt-3">
    <label>Carga horária</label>
    <select [(ngModel)]="disciplina.carga" class="form-control" name="carga" #carga="ngModel"
required>
      <option></option>
      <option>40</option>
      <option>80</option>
    </select>
    <div [hidden]="carga.valid || carga.pristine" class="alert alert-danger">
      Carga horária é obrigatória
    </div>
  </div>
```

Figura 10 – Código HTML do campo de seleção da carga horária do arquivo `src/app/form-disciplina/form-disciplina.component.html`.

```
  <div class="form-group mt-4">
    <label>Aluno</label>
    <select [(ngModel)]="matricula.aluno" class="form-control" name="aluno" #aluno="ngModel"
required>
      <!-- como são objetos, então o property binding não pode ser [value], mas [ngValue] -->
      <option *ngFor="let item of servico.alunos" [ngValue]="item">{{item.nome}}</option>
    </select>
    <div [hidden]="aluno.valid || aluno.pristine" class="alert alert-danger">
      É obrigatório selecionar o aluno
    </div>
  </div>
```

Figura 11 – Código HTML do campo de seleção de aluno do arquivo `src/app/form-matricula/form-matricula.component.html`.