Instruções para a entrega: mostrar para o professor na aula do dia 15/out.

Objetivo: Fazer uma SPA (Single Page Application) para manter o cadastro de matrículas assim como é mostrado na Figura 1. Os dados devem ser salvos no Firebase Realtime Database – que é ambiente de Cloud da Google. Essa aplicação é uma continuação da Atividade 8, então faça apenas as alterações.

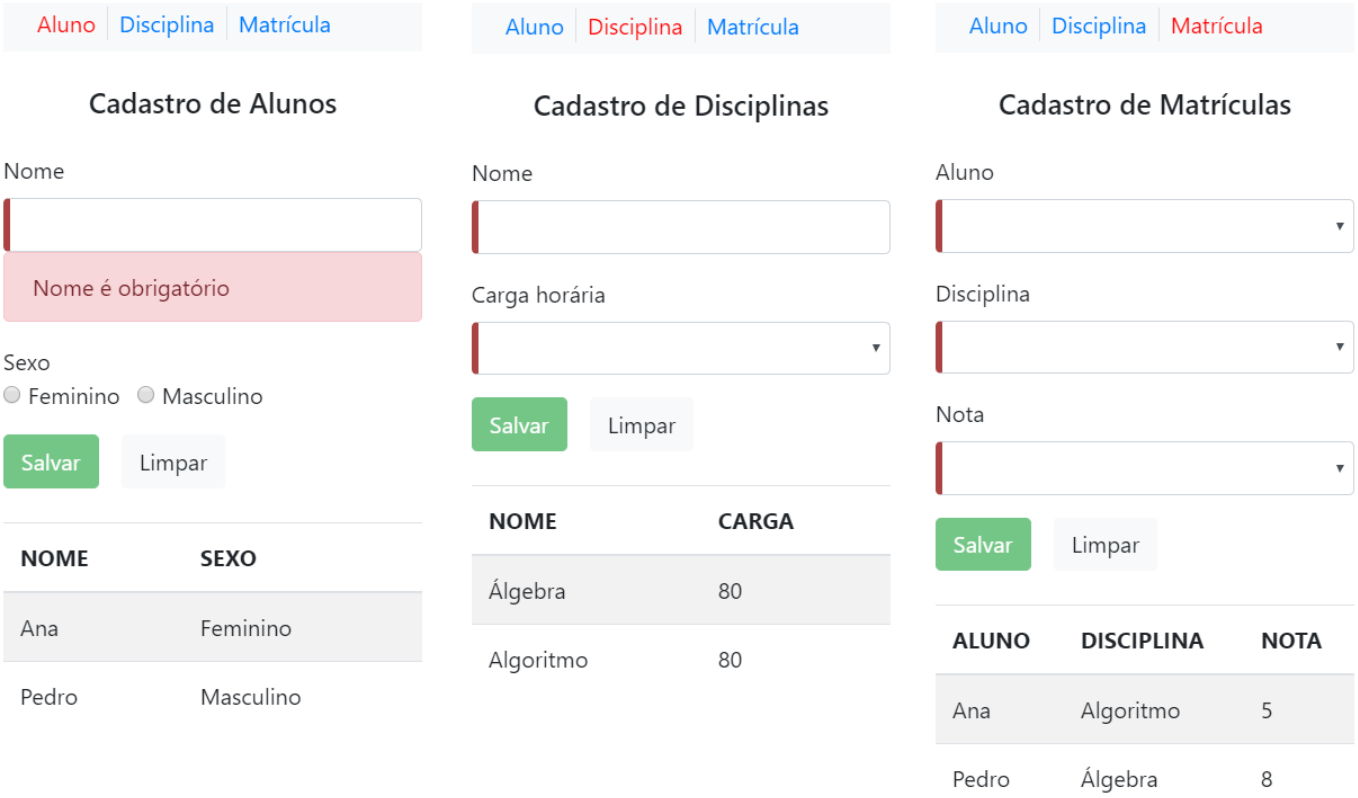

Figura 1 – Aplicação Angular para manter um cadastro de alunos.

Recomenda-se fazer um fork no projeto do Atividade 9 e fazer as alterações, pois o objetivo é apenas persistir os dados no Realtime Database do Firebase.

Passo 1: Para adicionar uma biblioteca no projeto precisamos acessar a pasta de dependencies. É necessário instalar as duas bibliotecas do Firebase:

- `firebase`
- `@angular/fire`

Passo 2 – Configurar uma aplicação no Firebase (https://firebase.google.com/docs/database/?authuser=0).

O Firebase Realtime Database é um banco de dados NoSQL hospedado na nuvem que permite armazenar e sincronizar dados entre seus usuários em tempo real. Na versão gratuita ele oferece (https://firebase.google.com/pricing):

- Conexões simultâneas: 100
- GB armazenados: 1 GB
- GB de download: 10 GB/mês
- Só pode usar 1 BD por projeto.

Para criar um projeto no Firebase você precisa estar logado na conta Google e acessar o console do Firebase:
https://console.firebase.google.com/

Primeiramente você precisa criar um projeto no Firebase, então faça os passos (a) a (c) da Figura 2. Na sequência você precisa configurar um aplicativo que possui acesso ao projeto. Ao final será gerado um script com os dados de acesso ao aplicativo.

Após configurar o aplicativo de acesso ao Realtime Database, temos de alterar as regras de acesso seguindo os passos da Figura 3.

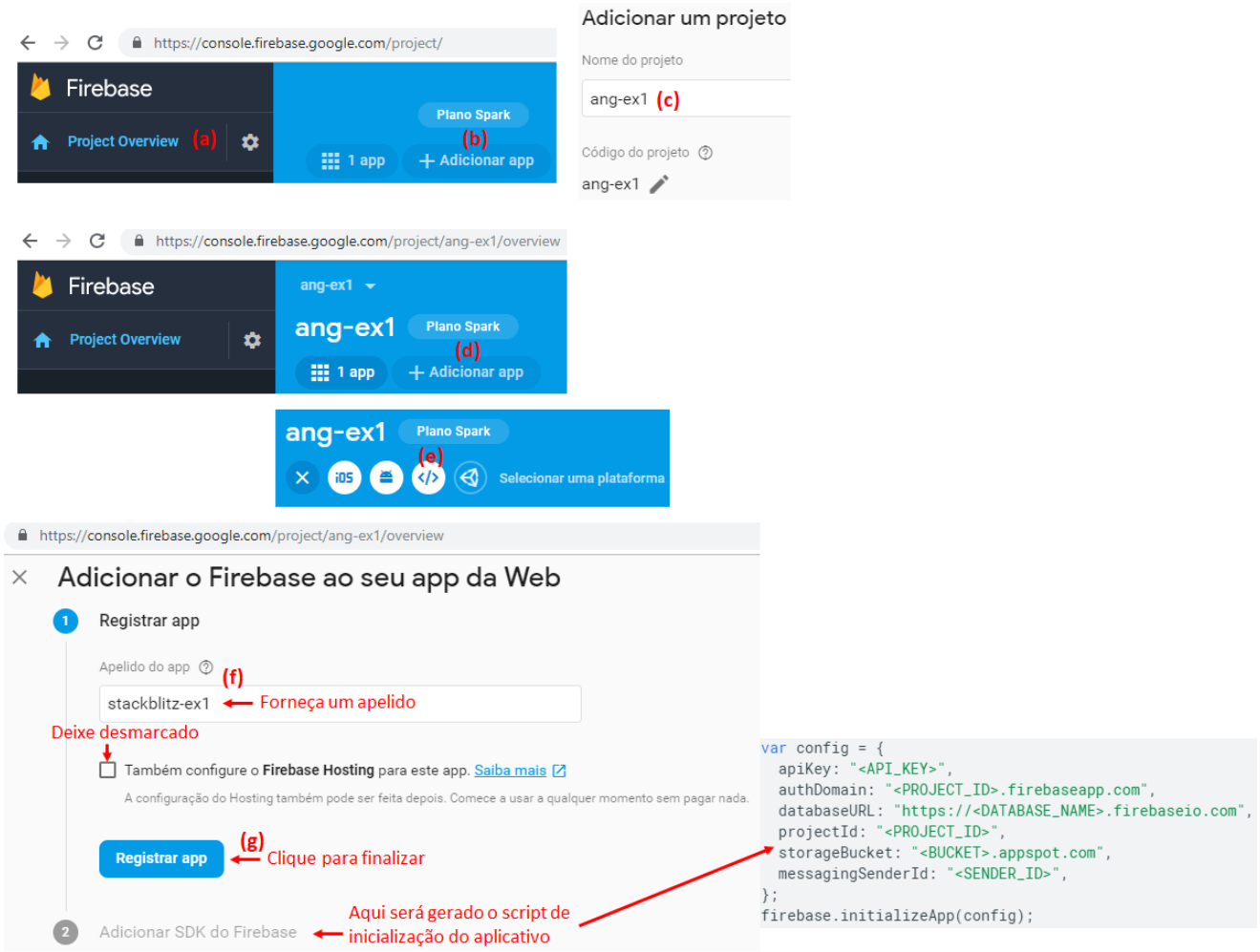

Figura 2 – Passos para criar um projeto e configurar um aplicativo de acesso ao projeto.

Passo 3 – Importar as bibliotecas do Firebase e do Realtime Database no arquivo `app.module.ts`, assim como está destacado em amarelo na Figura 4.

Passo 4 – Configurar o acesso a aplicação Firebase no seu projeto Angular.

Copie a variável `firebaseConfig` e cole ela no arquivo `app.module.ts`, assim como está destacado em ciano na Figura 4. Lembre-se que cada aplicação possui uma chave diferente, então não adianta copiar de outro projeto. Para obter esses dados acesse a aba mostrada na Figura 5.

Além disso, será necessário inicializar as configurações, assim como está destacado na cor verde na Figura 4.

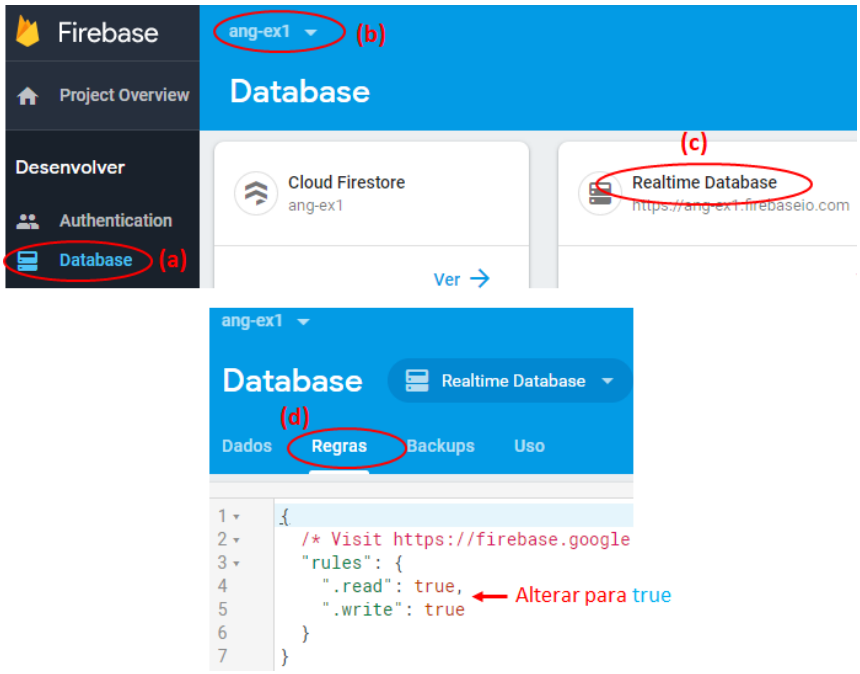

Figura 3 – Passos para alterar as regras de acesso ao banco de dados.

```
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';

/* faz a inicialização do App */
import { AngularFireModule } from '@angular/fire';
/* módulo do Real Time Database */
import { AngularFireDatabaseModule } from '@angular/fire/database';

/* 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' }
```

```
];

const firebaseConfig = {
  apiKey: "colar aqui",
  authDomain: "colar aqui",
  databaseURL: "colar aqui",
  projectId: "colar aqui",
  storageBucket: "colar aqui",
  messagingSenderId: "colar aqui",
  appId: "colar aqui"
};

@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),
    AngularFireModule.initializeApp(firebaseConfig),
    AngularFireDatabaseModule
     ],
  declarations: [ AppComponent, FormAlunoComponent,  FormDisciplinaComponent, FormMatriculaComponent
],
  bootstrap:    [ AppComponent ],
  providers: [ServicoService]
})
export class AppModule { }
```

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

Figura 5 – Aba de acesso aos dados de acesso a aplicação. No final da página está o script com os dados de acesso ao BD.

Passo 5 – Assim como é sugerido no script de configuração do Firebase - que está no final da página da Figura 5 - inclua a importação da biblioteca do Firebase no arquivo `index.html`, assim como está destacado em amarelo na Figura 6.

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/6.0.2/firebase-app.js"></script>

<my-app>loading</my-app>
```

Figura 6 – Código do arquivo `src/index.html`.

Os passos a seguir são referentes a conexão do Angular com o Realtime Database. Então recomenda-se assistir ao seguinte vídeo https://www.youtube.com/watch?v=8FDwTjgLN48

Passo 6 – A conexão com o Realtime Database deverá estar no serviço, então o melhor é criarmos 3 serviços de nome: `aluno`, `disciplina` e `matricula`. Atualmente já temos o serviço `servico`, remove ele após terminar o projeto, pois ele não será necessário.

Copie o código dos serviços que estão na Figura 7, Figura 8 e Figura 9.

```
import { Injectable } from '@angular/core';
import { Aluno } from './dados';
import { AngularFireDatabase } from '@angular/fire/database';
import { BehaviorSubject } from 'rxjs';
import { map } from 'rxjs/operators';

@Injectable()
export class AlunoService {
  private alunoSource = new BehaviorSubject({aluno: null, key: ''});
  public currentAluno = this.alunoSource.asObservable();
  public alunos: Aluno[] = [];

  constructor(private bd: AngularFireDatabase) { }

  insertAluno(aluno: Aluno): void {
    this.bd.list('atv9/aluno').push(aluno)
      .then((result: any) => {
        console.log(result.key);
      })
      .catch((error: any) => {
        console.error(error);
      });
  }

  updateAluno(aluno: Aluno, key: string): void {
    this.bd.list('atv9/aluno').update(key, aluno)
      .catch((error: any) => {
        console.error(error);
      });
  }

  deleteAluno(key: string): void {
    this.bd.object(`atv9/aluno/${key}`).remove();
  }

  selectAluno() {
    return this.bd.list('atv9/aluno',
      ref => ref.orderByChild('nome'))
      .snapshotChanges() /* pegar as mudanças */
```

```
      .pipe(
        /* mapeamento das mudanças */
        map(changes => {
          /* ... são todas as demais propriedades do objeto JSON que está no BD */
          return changes.map(c => ({ key: c.payload.key, ...c.payload.val() }));
        })
      );
  }

  changeAluno(aluno: Aluno, key: string) {
    this.alunoSource.next({aluno: aluno, key: key});
  }
}
```

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

```
import { Injectable } from '@angular/core';
import { Disciplina } from './dados';
import { AngularFireDatabase } from '@angular/fire/database';
import { BehaviorSubject } from 'rxjs';
import { map } from 'rxjs/operators';

@Injectable()
export class DisciplinaService {
  private disciplinaSource = new BehaviorSubject({ disciplina: null, key: '' });
  public currentDisciplina = this.disciplinaSource.asObservable();
  public disciplinas: Disciplina[] = [];

  constructor(private bd: AngularFireDatabase) { }

  insertDisciplina(disciplina: Disciplina): void {
    this.bd.list('atv9/disciplina').push(disciplina)
      .then((result: any) => {
        console.log(result.key);
      })
      .catch((error: any) => {
        console.error(error);
      });
  }

  updateDisciplina(disciplina: Disciplina, key: string): void {
    this.bd.list('atv9/disciplina').update(key, disciplina)
      .catch((error: any) => {
        console.error(error);
      });
  }

  deleteDisciplina(key: string): void {
    this.bd.object(`atv9/disciplina/${key}`).remove();
  }

  selectDisciplina() {
```

```
    return this.bd.list('atv9/disciplina',
      ref => ref.orderByChild('nome'))
      .snapshotChanges() /* pegar as mudanças */
      .pipe(
        /* mapeamento das mudanças */
        map(changes => {
          /* ... são todas as demais propriedades do objeto JSON que está no BD */
          return changes.map(c => ({ key: c.payload.key, ...c.payload.val() }));
        })
      );
  }

  changeDisciplina(disciplina: Disciplina, key: string) {
    this.disciplinaSource.next({ disciplina: disciplina, key: key });
  }
}
```

Figura 8 – Conteúdo do arquivo `src/app/disciplina.service.ts`.

```
import { Injectable } from '@angular/core';
import { Matricula } from './dados';
import { AngularFireDatabase } from '@angular/fire/database';
import { BehaviorSubject } from 'rxjs';
import { map } from 'rxjs/operators';

@Injectable()
export class MatriculaService {
  private matriculaSource = new BehaviorSubject({ matricula: null, key: '' });
  public currentMatricula = this.matriculaSource.asObservable();
  public matriculas: Matricula[] = [];

  constructor(private bd: AngularFireDatabase) { }

  insertMatricula(matricula: Matricula): void {
    this.bd.list('atv9/matricula').push(matricula)
      .then((result: any) => {
        console.log(result.key);
      })
      .catch((error: any) => {
        console.error(error);
      });
  }

  updateMatricula(matricula: Matricula, key: string): void {
    this.bd.list('atv9/matricula').update(key, matricula)
      .catch((error: any) => {
        console.error(error);
      });
  }

  deleteMatricula(key: string): void {
    this.bd.object(`atv9/matricula/${key}`).remove();
```

```
  }

  selectMatricula() {
    return this.bd.list('atv9/matricula',
        ref => ref.orderByChild('aluno/nome')
      )
      .snapshotChanges() /* pegar as mudanças */
      .pipe(
        /* mapeamento das mudanças */
        map(changes => {
          /* ... são todas as demais propriedades do objeto JSON que está no BD */
          return changes.map(c => ({ key: c.payload.key, ...c.payload.val() }));
        })
      );
  }

  changeMatricula(matricula: Matricula, key: string) {
    this.matriculaSource.next({ matricula: matricula, key: key });
  }
}
```

Figura 9 – Conteúdo do arquivo `src/app/matricula.service.ts`.

Passo 7 – A Figura 10 e Figura 11 possui, respectivamente, o código do modelo (classe TypeScript) e view (DOM) do componente `form-aluno`. O mesmo código pode ser adaptado para os componente `form-disciplina` e `form-matricula`. A Figura 12 possui o código do modelo do componente `form-matricula`, observe que este formulário não possui a opção de editar, apenas de salvar e deletar.

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

import { Observable } from 'rxjs';
import { map, startWith } from 'rxjs/operators';

@Component({
  selector: 'app-form-aluno',
  templateUrl: './form-aluno.component.html',
  styleUrls: ['./form-aluno.component.css']
})
export class FormAlunoComponent implements OnInit {
  private aluno: Aluno;
  private key: string = '';
  private alunos: Observable<any>;
  constructor(private servico: AlunoService) { }

  ngOnInit() {
    this.aluno = new Aluno();
    this.alunos = this.servico.selectAluno();
    this.servico.currentAluno.subscribe(data => {
      if (data.aluno && data.key) {
```

```
        this.aluno = new Aluno();
        this.aluno.nome = data.aluno.nome;
        this.aluno.sexo = data.aluno.sexo;
        this.key = data.key;
      }
    });
  }

  salvar(): void {
    if (this.key) {
      this.aluno.nome = this.aluno.nome.trim();
      this.servico.updateAluno(this.aluno, this.key);
    }
    else {
      this.aluno.nome = this.aluno.nome.trim();
      this.servico.insertAluno(this.aluno);
    }
    this.reset();
  }

  reset(): void {
    this.aluno = new Aluno();
    this.key = '';
  }

  delete(key: string) {
    this.servico.deleteAluno(key);
    return false; /* para evitar que o menu popup seja exibido */
  }

  edit(aluno: Aluno, key: string) {
    this.servico.changeAluno(aluno, key);
  }
}
```

Figura 10 – Conteúdo do arquivo `src/app/form-aluno/form-aluno.component.ts`.

```
<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" (click)="reset()" class="btn btn-light">Limpar</button>
  </div>
</form>

<div class="table-responsive-sm mt-4 table-sm">
  <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 alunos | async" (click)="edit(item, item.key)"
(contextmenu)="delete(item.key)">
        <td>{{item.nome}}</td>
        <td>{{item.sexo}}</td>
      </tr>
    </tbody>
  </table>
</div>
```

Figura 11 – Conteúdo do arquivo `src/app/form-aluno/form-aluno.component.html`.

```
import { Component, OnInit } from '@angular/core';
import { MatriculaService } from '../matricula.service';
import { AlunoService } from '../aluno.service';
import { DisciplinaService } from '../disciplina.service';
import { Matricula } from '../dados';

import { Observable } from 'rxjs';
import { map, startWith } from 'rxjs/operators';

@Component({
  selector: 'app-form-matricula',
  templateUrl: './form-matricula.component.html',
  styleUrls: ['./form-matricula.component.css']
```

```
})
export class FormMatriculaComponent implements OnInit {
  private matricula: Matricula;
  private key: string = '';
  private matriculas: Observable<any>;
  private alunos: Observable<any>;
  private disciplinas: Observable<any>;
  constructor(private servico: MatriculaService, private alunoService: AlunoService, private
disciplinaService: DisciplinaService) { }

  ngOnInit() {
    this.matricula = new Matricula();
    this.matriculas = this.servico.selectMatricula();
    this.alunos = this.alunoService.selectAluno();
    this.disciplinas = this.disciplinaService.selectDisciplina();
    this.servico.currentMatricula.subscribe(data => {
      if (data.matricula && data.key) {
        this.matricula = new Matricula();
        this.matricula.aluno = data.matricula.aluno;
        this.matricula.disciplina = data.matricula.disciplina;
        this.matricula.nota = data.matricula.nota;
        this.key = data.key;
      }
    });
  }

  salvar(): void {
    this.servico.insertMatricula(this.matricula);
    this.reset();
  }

  reset(): void {
    this.matricula = new Matricula();
    this.key = '';
  }

  delete(key: string) {
    this.servico.deleteMatricula(key);
    return false; /* para evitar que o menu popup seja exibido */
  }
}
```

Figura 12 – Conteúdo do arquivo `src/app/form-matricula/form-matricula.component.ts`.