

Instituto Tecnológico de Aeronáutica

Divisão de Ciência da Computação

Exame de CES-26



Relatório de Tarefa – Exame Final

Tarefa:
Aplicação SPA em Angular 4
<https://github.com/ShellyLeal/Movie-List-Angular>

Aluna: Shelly Leal
Professor: Edgar Yano

Dezembro 2017

Listagem de Filmes em Angular4

Introdução

Com base nos conhecimentos de Angular4 e o tutorial Tour Of Heroes estudado passo a passo descrevendo-se os conceitos fundamentais, de criação de aplicações SPA com listas, serviços, roteamento, HTTP e animações, foi desenvolvida uma aplicação para visualização e edição de um ranking de filmes, procurando utilizar-se do máximo de ferramentas possíveis disponibilizadas particularmente pela versão do Angular4.

Resultados

Para se executar o projeto, apenas é necessário baixar o repositório do github referenciado na capa do relatório, utilizar o comando **npm install** na pasta root do projeto e **ng serve --open** para abrir o servidor local.

```
C:\Users\shell\Desktop\movies>npm install
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.0.0 (node_modules\
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents
} (current: {"os":"win32","arch":"x64"})

C:\Users\shell\Desktop\movies>ng serve --open
* NG Live Development Server is listening on localhost:4200, open your browser
11% building modules 16/16 modules 0 activewebpack: wait until bundle finisDat
Hash: 43609a3a850876bfd598
Time: 12314ms
chunk {inline} inline.bundle.js (inline) 5.79 kB [entry] [rendered]
chunk {main} main.bundle.js (main) 120 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 676 kB [initial] [rendered]
chunk {styles} styles.bundle.js (styles) 36.6 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js (vendor) 11.5 MB [initial] [rendered]

webpack: Compiled successfully.
```

A tela inicial do programa pode ser visualizada abaixo na Figura 1, onde se encontram a seção dos filmes superiores no ranking e logo abaixo os trailers de cada filme. Ao selecionar um dos filmes, obtém-se a informação como se apresenta na figura 2, além de haver o display de mensagens que indicam as ações do cliente.

Seleção de Filmes



Figura 1: Página inicial com os filmes superiores no ranking.

O MÁGICO DE OZ (1939) Details

id: 12

name:

O Mágico de Oz (1939)

go back

save

Messages

clear

Movie: selected movie id=12

Figura 2: Selecionando-se um dos filmes

Ranking

Filmes

Movie List

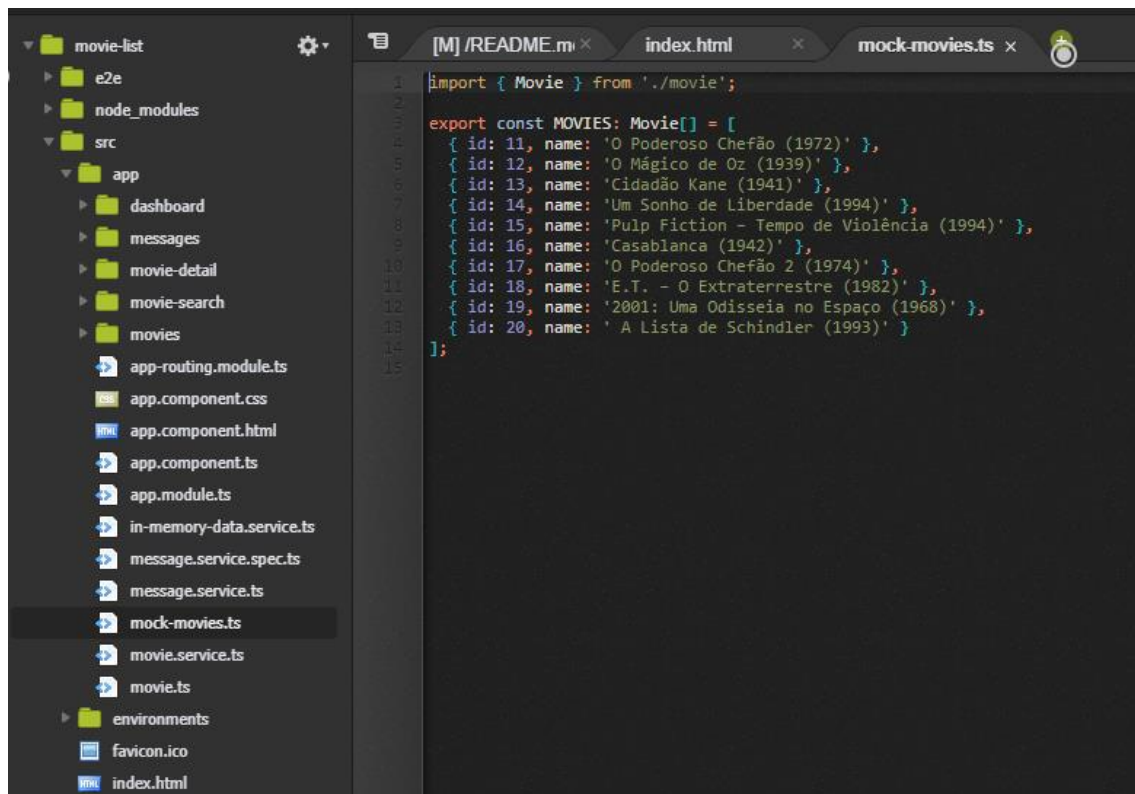
Movie name:



11	O Poderoso Chefão (1972)	x
12	O Mágico de Oz (1939)	x
13	Cidadão Kane (1941)	x
14	Um Sonho de Liberdade (1994)	x
15	Pulp Fiction – Tempo de Violência (1994)	x
16	Casablanca (1942)	x
17	O Poderoso Chefão 2 (1974)	x
18	E.T. – O Extraterrestre (1982)	x
19	2001: Uma Odisseia no Espaço (1968)	x
20	A Lista de Schindler (1993)	x

Figura 3: lista dos filmes ao clicar na aba Filmes

Essa lista introdutória de filmes na página inicial pela aba Filmes foi original de um mock inicial da lista de filmes:



```
1 import { Movie } from './movie';
2
3
4 export const MOVIES: Movie[] = [
5   { id: 11, name: 'O Poderoso Chefão (1972)' },
6   { id: 12, name: 'O Mágico de Oz (1939)' },
7   { id: 13, name: 'Cidadão Kane (1941)' },
8   { id: 14, name: 'Um Sonho de Liberdade (1994)' },
9   { id: 15, name: 'Pulp Fiction - Tempo de Violência (1994)' },
10  { id: 16, name: 'Casablanca (1942)' },
11  { id: 17, name: 'O Poderoso Chefão 2 (1974)' },
12  { id: 18, name: 'E.T. - O Extraterrestre (1982)' },
13  { id: 19, name: '2001: Uma Odisseia no Espaço (1968)' },
14  { id: 20, name: 'A Lista de Schindler (1993)' }
15 ];
```

Figura 3: criação de filmes mock

Serviços

Com base no aprendizado do Tour of Heroes, foi criado um MovieService que permitiu a injeção de dependências para o construtor MoviesComponent.

```
movie.service.ts x
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable } from 'rxjs/Observable';
import { of } from 'rxjs/observable/of';
import { catchError, map, tap } from 'rxjs/operators';

import { Movie } from './movie';
import { MessageService } from './message.service';

const httpOptions = {
  headers: new HttpHeaders({ 'Content-Type': 'application/json' })
};

@Injectable()
export class MovieService {

  private moviesUrl = 'api/movies'; // URL to web api

  constructor(
    private http: HttpClient,
    private messageService: MessageService) { }

  /** GET movies from the server */
  getMovies (): Observable<Movie[]> {
    return this.http.get<Movie[]>(this.moviesUrl)
      .pipe(
        tap(movies => this.log(`selected movies`)),
        catchError(this.handleError(`getMovies`, []))
      );
  }

  /** GET movie by id. Return `undefined` when id not found */
  getMovieNo404<Data>(id: number): Observable<Movie> {
    const url = `${this.moviesUrl}/?id=${id}`;
    return this.http.get<Movie[]>(url)
      .pipe(
        map(movies => movies[0]), // returns a {0/1} element array
        tap(h => {
          const outcome = h ? `selected` : `did not find`;
          this.log(`${outcome} movie id=${id}`);
        }),
        catchError(this.handleError<Movie>(`getMovie id=${id}`))
      );
  }

  /** GET movie by id. Will 404 if id not found */
  getMovie(id: number): Observable<Movie> {

```

Figura 4: Uso do decorador @Injectable() para dependências injetadas

Da mesma forma foi gerada a classe MessageService:

```

message.service x
import { Injectable } from '@angular/core';

@Injectable()
export class MessageService {
  messages: string[] = [];

  add(message: string) {
    this.messages.push(message);
  }

  clear() {
    this.messages = [];
  }
}

```

Figura 5: manipulação do cache das mensagens com os métodos add() e clear().

Roteamento

Foi adicionado um Angular router para navegar entre os componentes. Este foi configurado em um AppRoutingModuleModule e foram definidas rotas simples, uma de redirecionamento e outra parametrizada.

```

app-routing.module x
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3
4 import { DashboardComponent } from './dashboard/dashboard.component';
5 import { MoviesComponent } from './movies/movies.component';
6 import { MovieDetailComponent } from './movie-detail/movie-detail.component';
7
8 const routes: Routes = [
9   { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
10  { path: 'dashboard', component: DashboardComponent, data: { page: 'dashboard' } },
11  { path: 'detail/:id', component: MovieDetailComponent, data: { page: 'detail/:id' } },
12  { path: 'movies', component: MoviesComponent, data: { page: 'movies' } }
13 ];
14
15 @NgModule({
16   imports: [ RouterModule.forRoot(routes) ],
17   exports: [ RouterModule ]
18 })
19 export class AppRoutingModuleModule {}
20

```

Figura 6: direcionamento das rotas.

```

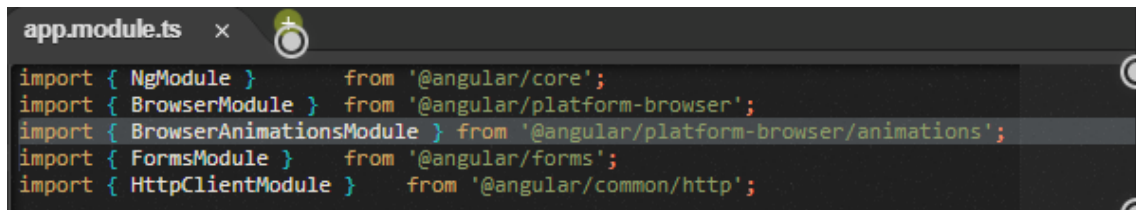
app.component.html x
1 <h1>{{title}}</h1>
2 <nav>
3   <a routerLink="/dashboard">Ranking</a>
4   <a routerLink="/movies">Filmes</a>
5 </nav>
6 <div class="main" [animRoutes]="getPage(appOutlet)">
7   <router-outlet #appOutlet="outlet"></router-outlet>
8 </div>
9 <app-messages></app-messages>
10

```

Figura 7: adicionado um link de navegação do dashboard ao template.

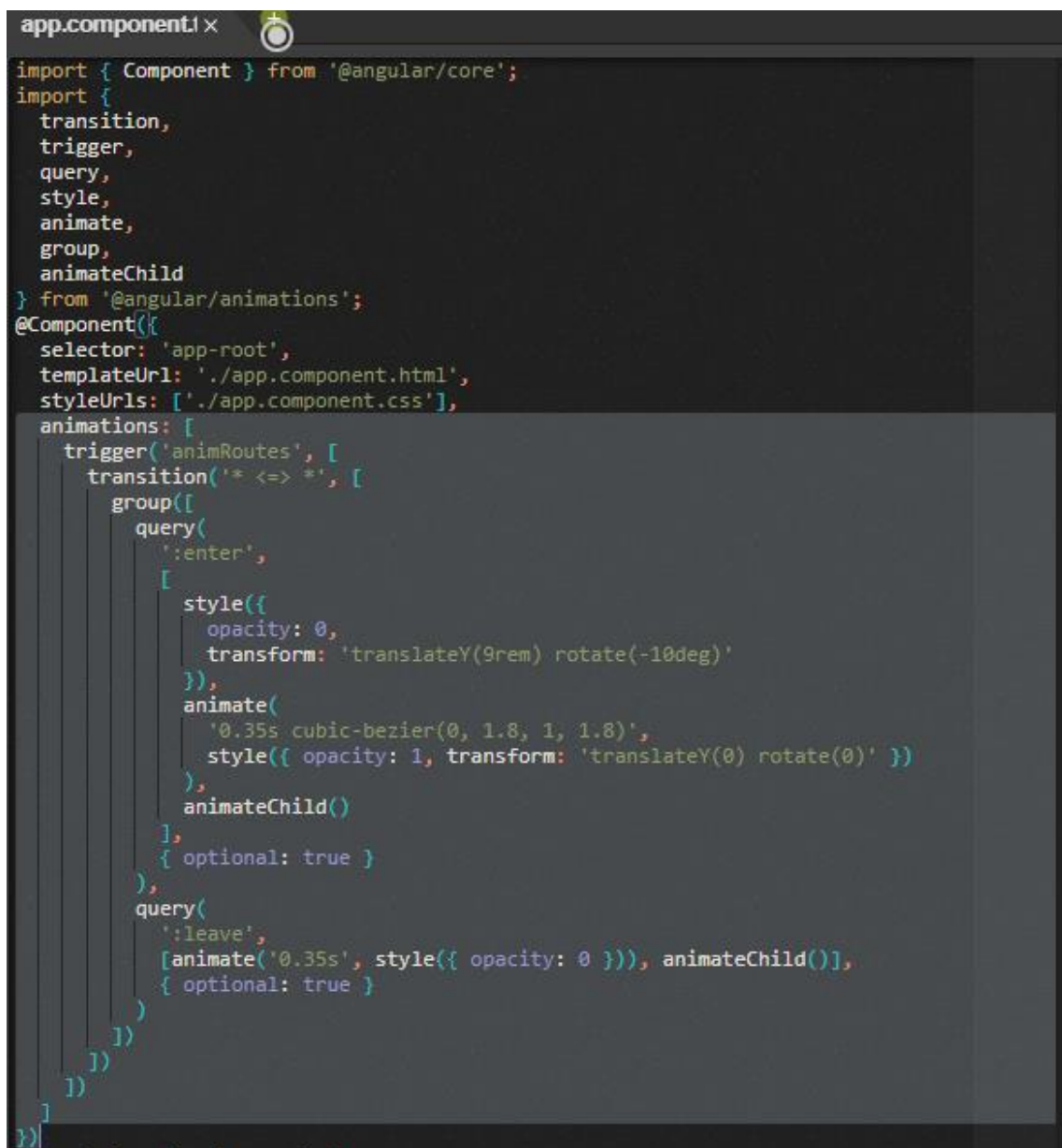
Animação

Além da animação de seleção dos botões, foi adicionada uma animação por roteamento ao trocar de páginas, com um simples atraso no display de cada página.



```
app.module.ts x
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
```

Figura 8: instalando-se modulo para animação. Também é importado pelo @NgModule.



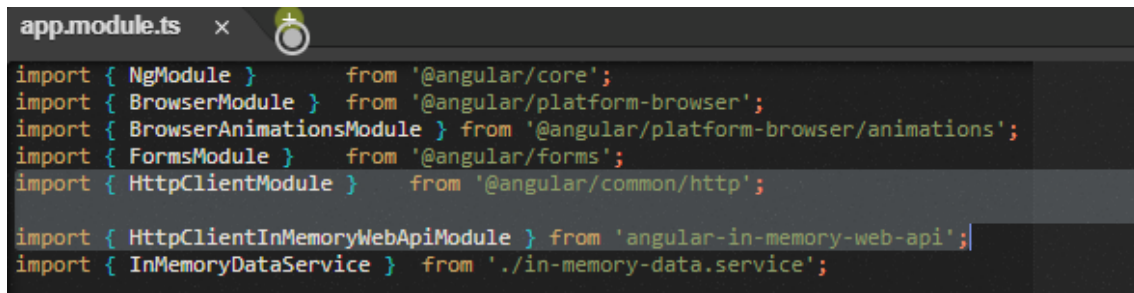
```
app.component.ts x
import { Component } from '@angular/core';
import {
  transition,
  trigger,
  query,
  style,
  animate,
  group,
  animateChild
} from '@angular/animations';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  animations: [
    trigger('animRoutes', [
      transition('* <=> *', [
        group([
          query(
            ':enter',
            [
              style({
                opacity: 0,
                transform: 'translateY(9rem) rotate(-10deg)'
              }),
              animate(
                '0.35s cubic-bezier(0, 1.8, 1, 1.8)',
                style({ opacity: 1, transform: 'translateY(0) rotate(0)' })
              ),
              animateChild()
            ],
            { optional: true }
          ),
          query(
            ':leave',
            [animate('0.35s', style({ opacity: 0 })), animateChild()],
            { optional: true }
          )
        ])
      ])
    ])
  ]
})
```

Figura 9: adição das propriedades de animação ao decorador @Component().

Serviço HTTP

Foi utilizado o `HttpClient` para buscar informações dos filmes com requisições HTTP. Dessa forma os usuários podem adicionar, editar e deletar os filmes, além de pesquisar os filmes por nome.

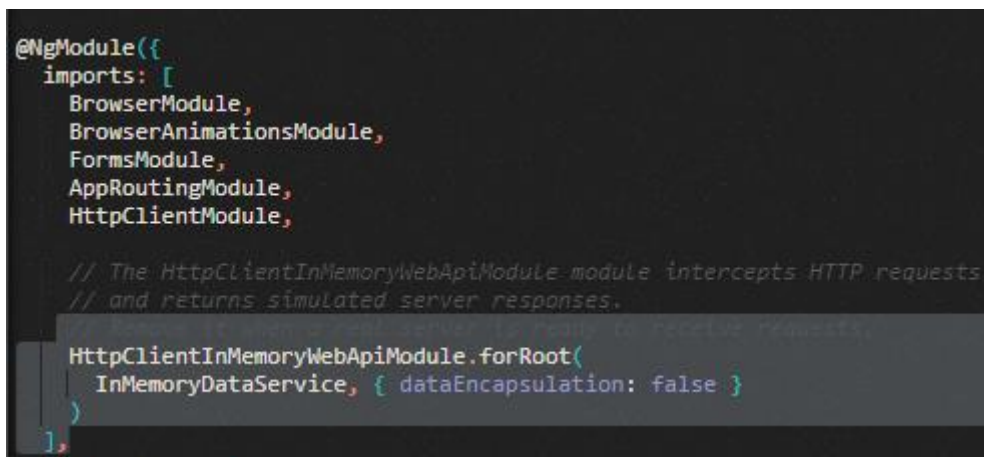
A seguir é demonstrado como foram importados os módulos de web API e injetado o `HttpClient` ao construtor utilizando-se a propriedade `http`.



```
app.module.ts x
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

import { HttpClientInMemoryWebApiModule } from 'angular-in-memory-web-api';
import { InMemoryDataService } from './in-memory-data.service';
```

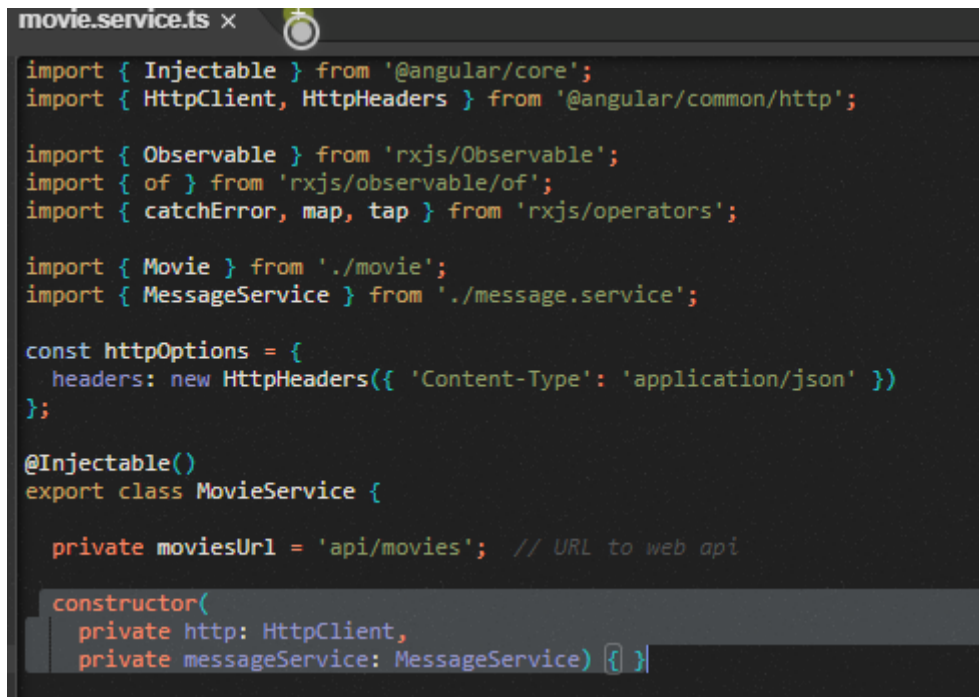
Figura 10: importados as classes `InMemoryWebApiModule` e `InMemory`



```
@NgModule({
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    FormsModule,
    AppRoutingModule,
    HttpClientModule,

    // The HttpClientInMemoryWebApiModule module intercepts HTTP requests
    // and returns simulated server responses.
    // Keep this module if you need to run the application in development.
    HttpClientInMemoryWebApiModule.forRoot(
      InMemoryDataService, { dataEncapsulation: false }
    )
  ],
  providers: []
})
```

Figura 11: adição dos imports ao `@NgModule`.



```

movie.service.ts x
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable } from 'rxjs/Observable';
import { of } from 'rxjs/observable/of';
import { catchError, map, tap } from 'rxjs/operators';

import { Movie } from './movie';
import { MessageService } from './message.service';

const httpOptions = {
  headers: new HttpHeaders({ 'Content-Type': 'application/json' })
};

@Injectable()
export class MovieService {

  private moviesUrl = 'api/movies'; // URL to web api

  constructor(
    private http: HttpClient,
    private messageService: MessageService) {}

```

Figura 12: injeção do HttpClient ao construtor.

Conclusão

Foi observado uma alta praticidade no uso das ferramentas do Angular4, incluindo com o apoio dos passos do tutorial Tour Of Heroes. Foram realizados testes comparativos de linguagem com outras versões, como AngularJS e Angular 2, porém não foi possível chegar ao nível de complexidade de aplicação como utilizando-se o Angular4.