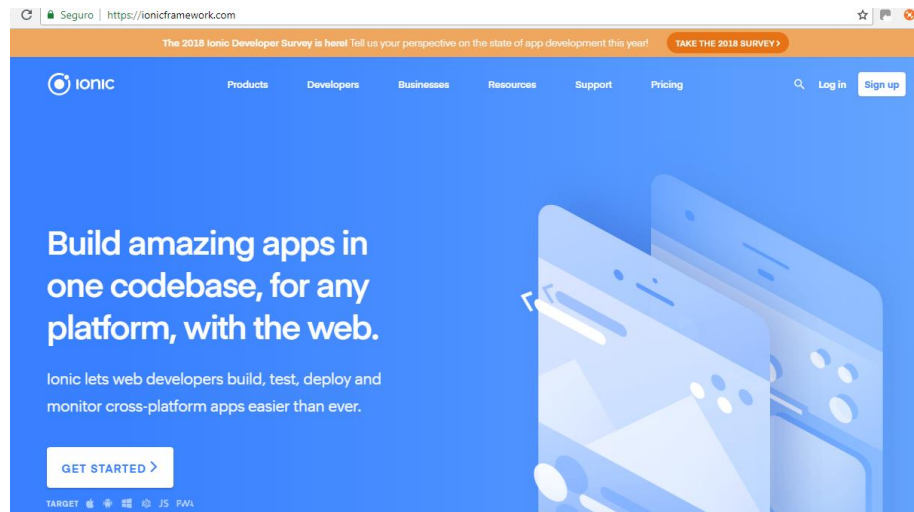


# 1

# IONIC

## MODIFICANDO O PROJETO

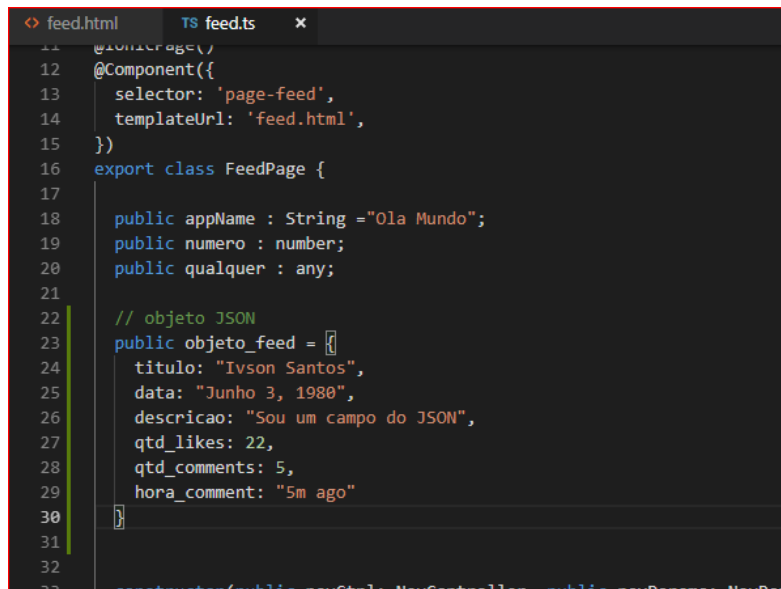


# UTILIZANDO JSON

# CODIFICANDO

## PASSOS

- ▶ Criar o objeto JSON dentro do arquivo .ts, com os campos que são utilizados na tela



```
11 @IonicPage()
12 @Component({
13   selector: 'page-feed',
14   templateUrl: 'feed.html',
15 })
16 export class FeedPage {
17
18   public appName : String ="Ola Mundo";
19   public numero : number;
20   public qualquer : any;
21
22   // objeto JSON
23   public objeto_feed = {
24     titulo: "Ivson Santos",
25     data: "Junho 3, 1980",
26     descricao: "Sou um campo do JSON",
27     qtd_likes: 22,
28     qtd_comments: 5,
29     hora_comment: "5m ago"
30   }
31
32
33   constructor(public navCtrl: NavController, public NavParams: NavParams)
```

# CODIFICANDO

## PASSOS

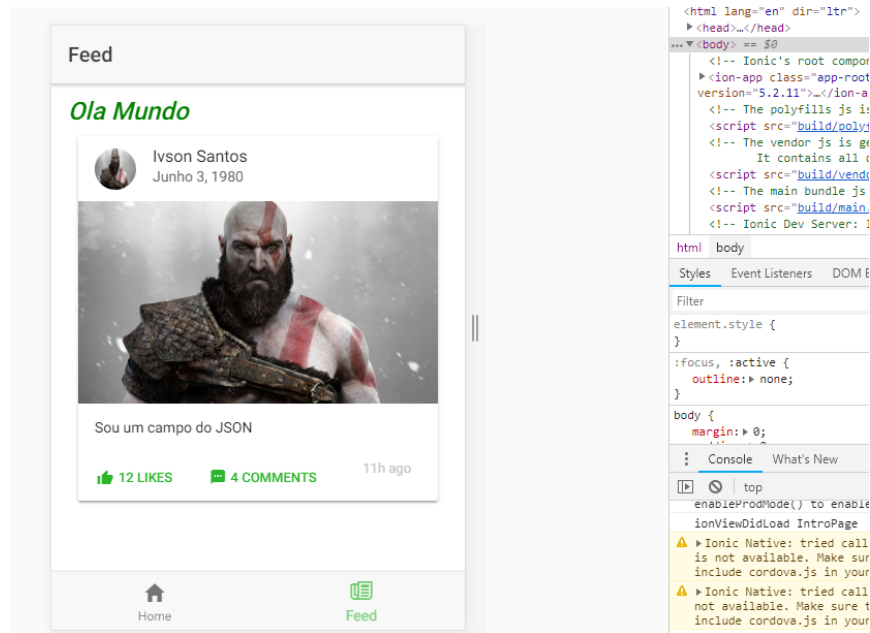
- ▶ Colocar na página HTML os bindings para os campos do objeto JSON

```
14
15
16 <ion-content padding>
17
18   <h2 class="meu_estilo_titulo"> {{appName}} </h2>
19
20   <ion-card>
21
22     <ion-item>
23       <ion-avatar item-start>
24         
25       </ion-avatar>
26       <h2 class="feed_title">{{ objeto_feed.titulo }}</h2>
27       <p>{{ objeto_feed.data }}</p>
28     </ion-item>
29
30     
31
32     <ion-card-content>
33       <p>{{ objeto_feed.descricao }}</p>
34     </ion-card-content>
35
36     <ion-row>
37       <ion-col>
38         <button ion-button icon-start clear small>
39           <ion-icon name="thumbs-up"></ion-icon>
40           <div>{{ qtd_likes.data }} Likes</div>
41         </button>
42       </ion-col>
43       <ion-col>
44         <button ion-button icon-start clear small>
45           <ion-icon name="text"></ion-icon>
46           <div>{{ objeto_feed.qtd_comments }} Comments</div>
47         </button>
48       </ion-col>
49       <ion-col center text-center>
50         <ion-note>
51           {{ objeto_feed.hora_comment }}
52         </ion-note>
53     </ion-row>
```

# CODIFICANDO

## PASSOS

- ▶ Testar a aplicação e verificar os dados vindo do objetos JSON



# CONSUMINDO API

# CODIFICANDO

## PASSOS

► Iremos consultar a API do site <https://www.themoviedb.org>, em “<https://developers.themoviedb.org/3/movies/get-latest-movie>”

The screenshot shows the TMDB API documentation for the `GET /movie/latest` endpoint. On the left is a sidebar with navigation links. The main content area shows the endpoint details, including a description, authentication requirements, query string parameters, and a table of response fields.

**GET /movie/latest**

Get the most newly created movie. This is a live response and will continuously change.

**Authentication**

☒ API Key

**Query String**

Parameter	Type	Default	Required
api_key	string	<<api_key>>	required
language	string	Pass a ISO 639-1 value to display translated data for the fields that support it. minLength: 2 pattern: ^([a-z]{2})-([A-Z]{2})\$ default: en-US	optional

**Responses** application/json

Status	Schema	Example
200	object	
401		
404		

Field	Type	Optional
adult	boolean	optional
backdrop_path	string or null	optional
belongs_to_collection	null	optional
budget	integer	optional
genres	array(object)	optional
id	integer	optional
name	string	optional
homepage	string	optional
id	integer	optional

# CODIFICANDO

## PASSOS

- ▶ Para nosso código, devemos importar a `HttpModule` para nossa aplicação, no arquivo `app.module.ts`, caso não exista

```
TS app.module.ts x
1  import { NgModule, ErrorHandler } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
4  import { MyApp } from './app.component';
5
6  import { AboutPage } from '../pages/about/about';
7  import { ContactPage } from '../pages/contact/contact';
8  import { HomePage } from '../pages/home/home';
9  import { TabsPage } from '../pages/tabs/tabs';
10
11 import { StatusBar } from '@ionic-native/status-bar';
12 import { SplashScreen } from '@ionic-native/splash-screen';
13 import { FeedPageModule } from '../pages/feed/feed.module';
14 import { IntroPageModule } from '../pages/intro/intro.module';
15
16 import { HttpModule } from '@angular/http';
17
18 @NgModule({
19   declarations: [
20     MyApp,
21     AboutPage,
22     ContactPage,
23     HomePage,
24     TabsPage
25   ],
26   imports: [
27     BrowserModule,
28     IonicModule.forRoot(MyApp),
29     FeedPageModule,
30     IntroPageModule,
31
32     HttpModule
33   ],
34   bootstrap: [IonicApp]
```



# CODIFICANDO

## PASSOS

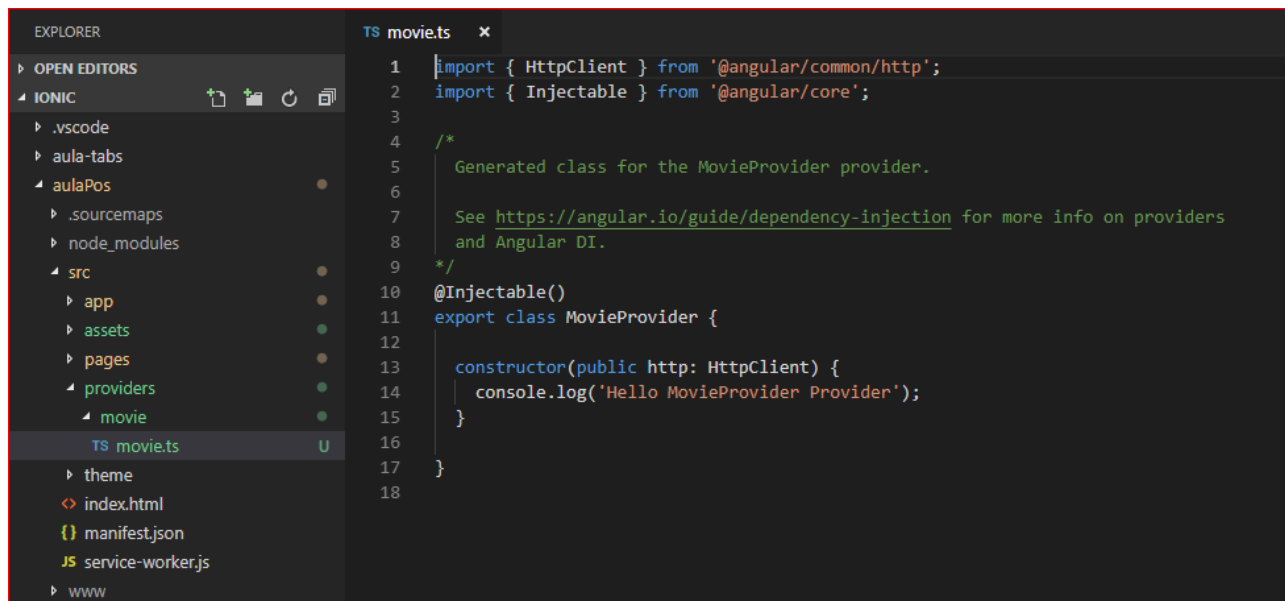
- ▶ Criar um provedor de informação com o comando
- ▶ “ionic generate provider <nome-do-provedor>”

```
[app-scripts] [16:57:42] deeplinks update started ...  
[app-scripts] [16:57:42] deeplinks update finished in 16 ms  
[app-scripts] [16:57:42] transpile update started ...  
[app-scripts] [16:57:42] transpile update finished in 106 ms  
[app-scripts] [16:57:42] webpack update started ...  
[app-scripts] [16:58:01] webpack update finished in 18.93 s  
[app-scripts] [16:58:01] build finished in 19.17 s  
Deseja finalizar o arquivo em lotes (S/N)? S  
PS C:\projetos\ionic\aulaPos> ionic generate provider movie
```

# CODIFICANDO

## PASSOS

- ▶ Será criada uma pasta “providers” com o componente provedor



```
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3
4 /*
5  * Generated class for the MovieProvider provider.
6  *
7  * See https://angular.io/guide/dependency-injection for more info on providers
8  * and Angular DI.
9  */
10 @Injectable()
11 export class MovieProvider {
12
13   constructor(public http: HttpClient) {
14     console.log('Hello MovieProvider Provider');
15   }
16
17 }
18
```

## CODIFICANDO

### **PASSOS**

- ▶ Este componente vai prover informações, quem quiser informação vai chamar este provider e ele se encarrega de dar a informação desejada.

# CODIFICANDO

## PASSOS

- ▶ Para ele buscar o método desejado, devemos criar o mesmo.
- ▶ Iremos utilizar o http para acessar os métodos desejados

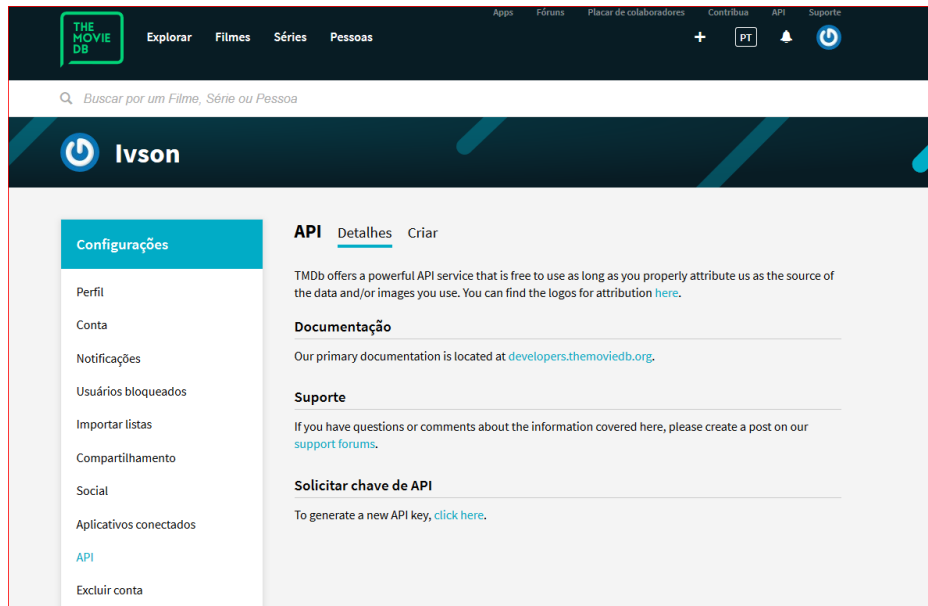
```
1 import { HttpClientModule } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3
4 /**
5  * Generated class for the MovieProvider provider.
6  *
7  * See https://angular.io/guide/dependency-injection for more info on providers
8  * and Angular DI.
9  */
10 @Injectable()
11 export class MovieProvider {
12
13   constructor(public http: HttpClient) {
14     console.log('Hello MovieProvider Provider');
15   }
16
17   getLatestMovies() {
18     this.http
19       .delete
20       .get
21       .head
22       .jsonp
23       .options
24       .patch
25       .post
26       .put
27       .request
```

(method) HttpClient.delete(url: string, options?: { headers?: HttpHeaders | { [header: string]: string | string[]; }; observe?: "body"; params?: HttpParams | { [param: string]: string | string[]; }; reportProgress?: boolean; responseType: "arraybuffer"; withCredentials?: boolean; })

# CODIFICANDO

## PASSOS

- ▶ Para utilizar a API do IMDB deve-se criar uma conta e pegar a ma chave da API gerada em <https://www.themoviedb.org/settings/api>



# CODIFICANDO

## PASSOS

- ▶ Colocar a URL da API no código da app

```
TS movie.ts x TS feed.ts <> feed.html
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { Http } from '@angular/http';
4
5 @Injectable()
6 export class MovieProvider {
7
8     public api_url = "https://api.themoviedb.org/3/";
9
10
11     constructor(public http: Http) {
12         console.log('Hello MovieProvider Provider');
13     }
14
15     getLatestMovies() {
16         return this.http.get(this.api_url + "movie/latest?api_key=" + this.api_key);
17     }
18 }
```

# CODIFICANDO

## PASSOS

- ▶ Na página que for utilizar o provider, incluir ele como um componente

```
TS movie.ts  TS feed.ts  x
1
2 import { Component } from '@angular/core';
3 import { IonicPage, NavController, NavParams } from 'ionic-angular';
4
5 import { MovieProvider } from '../providers/movie/movie';
6
7 @IonicPage()
8 @Component({
9   selector: 'page-feed',
10  templateUrl: 'feed.html',
11
12  providers: [
13    MovieProvider
14  ]
15
16 })
17 export class FeedPage {
18
```

## CODIFICANDO

### PASSOS

- ▶ Criar uma variável para ele no construtor

```
movie.ts  TS feed.ts  x
27         descricao: "Sou um campo do JSON",
28         qtd_likes: 22,
29         qtd_comments: 5,
30         hora_comment: "5m ago"
31     }
32
33
34     constructor(
35         public navCtrl: NavController,
36         public navParams: NavParams,
37         private movieProvider: MovieProvider ) {
38     }
39
```



# CODIFICANDO

## PASSOS

- ▶ Criar uma função para chamar o provider

```
TS movie.ts  TS feed.ts  x
27   descricao: 'Sou um campo do JSON',
28   qtd_likes: 22,
29   qtd_comments: 5,
30   hora_comment: '5m ago'
31 }
32
33
34 constructor(
35   public navCtrl: NavController,
36   public navParams: NavParams,
37   private movieProvider: MovieProvider ) {
38 }
39
40
41 ionViewDidLoad() {
42   this.movieProvider.getLatestMovies().subscribe(
43     data => {
44       console.log(data);
45     },
46     error => {
47       console.log(error);
48     }
49   )
50 }
51
```

## CODIFICANDO

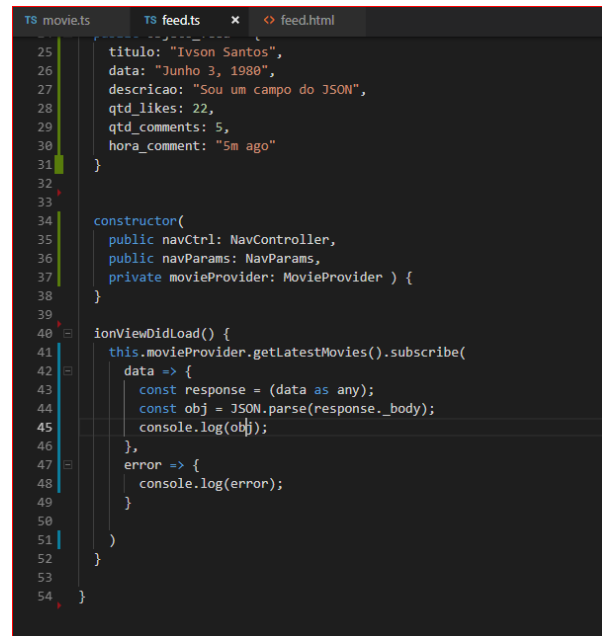
### **PASSOS**

- ▶ Antes de testar, remover os códigos anteriores do HTML
- ▶ Testar

# CODIFICANDO

## PASSOS

- ▶ Converter o objeto de resposta para JSON



```
TS movie.ts    TS feed.ts    x    feed.html
25     titulo: "Ivson Santos",
26     data: "Junho 3, 1988",
27     descricao: "Sou um campo do JSON",
28     qtd_likes: 22,
29     qtd_comments: 5,
30     hora_comment: "5m ago"
31   }
32
33
34   constructor(
35     public navCtrl: NavController,
36     public navParams: NavParams,
37     private movieProvider: MovieProvider ) {
38   }
39
40   ionViewDidLoad() {
41     this.movieProvider.getLatestMovies().subscribe(
42       data => {
43         const response = (data as any);
44         const obj = JSON.parse(response._body);
45         console.log(obj);
46       },
47       error => {
48         console.log(error);
49       }
50     )
51   }
52 }
53
54 }
```


# CODIFICANDO

## PASSOS


- ▶ Converter o objeto de resposta para JSON

Feed


Ola Mundo




Ola Mundo



Ola Mundo

 OLA MUNDO LIKES

 5 COMMENTS

5m ago

```
include cordova.js in your index.html
⚠ Ionic Native: tried calling SplashScreen.hide, but Cord
not available. Make sure to a) run in a real device or s
include cordova.js in your index.html

Hello MovieProvider Provider

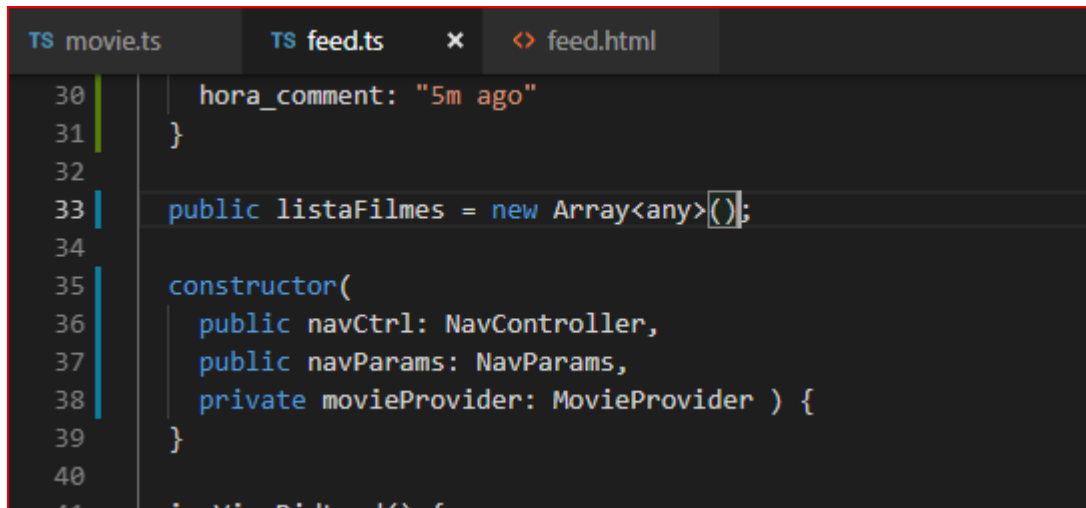
{adult: false, backdrop_path: null, belongs_to_collecti
genres: Array(1), ...}
  adult: false
  backdrop_path: null
  belongs_to_collection: null
  budget: 0
  genres: [{...}]
  homepage: null
  id: 544825
  imdb_id: null
  original_language: "en"
  original_title: "The Woman Who Loves Giraffes"
  overview: "Dr. Anne Innis Dagg re-traces the steps of
  popularity: 0
  poster_path: null
  production_companies: []
  production_countries: []
  release_date: ""
  revenue: 0
  runtime: 81
  spoken_languages: []
  status: "Released"
  tagline: ""
  title: "The Woman Who Loves Giraffes"
  video: false
  vote_average: 0
  vote_count: 0
  __proto__: Object
```

LISTANDO

# CODIFICANDO

## PASSOS

- ▶ Primeiro criem uma variável do tipo lista



```
TS movie.ts    TS feed.ts    x    <> feed.html
30 |             hora_comment: "5m ago"
31 |         }
32 |
33 |         public listaFilmes = new Array<any>();
34 |
35 |         constructor(
36 |             public navCtrl: NavController,
37 |             public navParams: NavParams,
38 |             private movieProvider: MovieProvider ) {
39 |         }
40 |
41 |     }
```

# CODIFICANDO

## PASSOS

- ▶ Passar a lista de retorno para a lista criada

```
TS movie.ts  TS feed.ts  x  feed.html
38 |         private movieProvider: MovieProvider ) {
39 |     }
40 |
41 |     ionViewDidLoad() {
42 |         this.movieProvider.getPopular().subscribe(
43 |             data => {
44 |                 const response = (data as any);
45 |                 const obj = JSON.parse(response._body);
46 |
47 |                 // o objeto retorna a lista de filmes (results)
48 |                 this.listaFilmes = obj.results;
49 |                 console.log(obj);
50 |             },
51 |             error => {
52 |                 console.log(error);
53 |             }
54 |         )
55 |     }
56 | }
57 |
58 | }
```

# CODIFICANDO

## PASSOS

- ▶ Criar mais um método para pegar uma lista de objetos

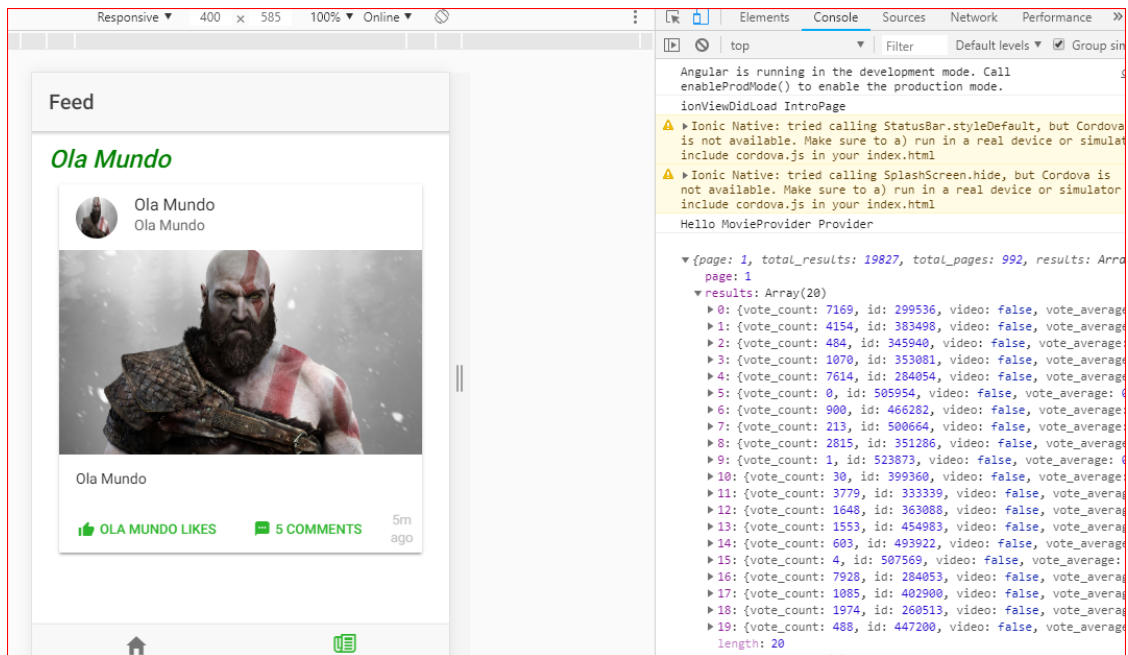
```
TS movie.ts x TS feed.ts <> feed.html
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { Http } from '@angular/http';
4
5 @Injectable()
6 export class MovieProvider {
7
8   public api_url = "https://api.themoviedb.org/3/";
9
10
11   constructor(public http: Http) {
12     console.log('Hello MovieProvider Provider');
13   }
14
15   getLatestMovies() {
16     return this.http.get(this.api_url + "movie/latest?api_key=" + this.api_key);
17   }
18
19   getPopular() {
20     return this.http.get(this.api_url + "movie/popular?api_key=" + this.api_key);
21   }
22 }
```



# CODIFICANDO

## PASSOS

- ▶ Testar o retorno da lista



# CODIFICANDO

## PASSOS

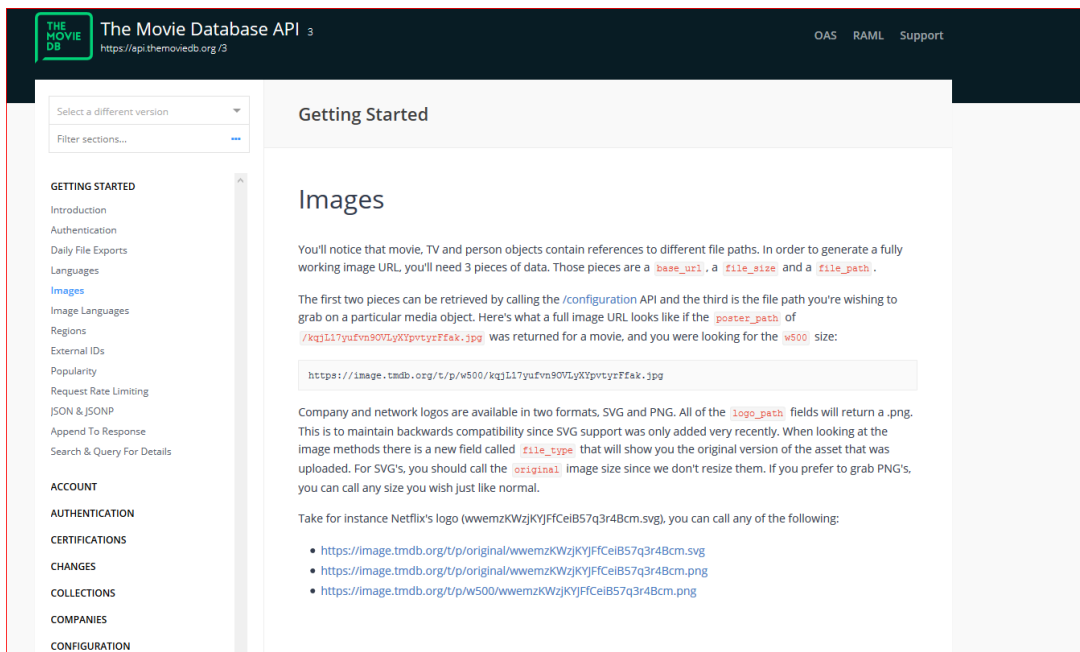
- ▶ Colocar a diretriva de lista para a página HTML

```
TS moviets  TS feedts  feed.html x
14  <!-- diretriva estrutural, para alterar a estrutura dos itens da pagina
15  |     repete o elemento
16  |     pegar o nome da variavel em TS
17  -->
18  <ion-card *ngFor="let movie of listaFilmes">
19
20      <ion-item>
21          <ion-avatar item-start>
22              
23          </ion-avatar>
24          <h2 class="feed_title">{{ movie.original_title }}</h2>
25          <p>{{ movie.release_date }}</p>
26      </ion-item>
27
28      
29
30      <ion-card-content>
31          <p>{{ movie.overview }}</p>
32      </ion-card-content>
33
34      <ion-row>
35          <ion-col>
36              <button ion-button icon-start clear small>
37                  <ion-icon name="thumbs-up"></ion-icon>
38                  <div>{{ appName }} Likes</div>
39              </button>
40          </ion-col>
41          <ion-col>
42              <button ion-button icon-start clear small>
43                  <ion-icon name="text"></ion-icon>
44                  <div>{{ objeto_feed.qtd_comments }} Comments</div>
45              </button>
46          </ion-col>
47          <ion-col center text-center>
48              <ion-note>
49                  {{ objeto_feed.hora_comment }}
50              </ion-note>
51          </ion-col>
```

# CODIFICANDO

## PASSOS

- ▶ Colocar imagens para o sistema



The screenshot shows the 'The Movie Database API' documentation page. The header includes the logo, the title 'The Movie Database API', the URL 'https://api.themoviedb.org/3', and links for 'OAS', 'RAML', and 'Support'. A left sidebar contains a table of contents with categories like 'GETTING STARTED', 'ACCOUNT', 'AUTHENTICATION', 'CERTIFICATIONS', 'CHANGES', 'COLLECTIONS', 'COMPANIES', and 'CONFIGURATION'. The 'GETTING STARTED' section is expanded, showing 'Introduction', 'Authentication', 'Daily File Exports', 'Languages', 'Images' (highlighted), 'Image Languages', 'Regions', 'External IDs', 'Popularity', 'Request Rate Limiting', 'JSON & JSONP', 'Append To Response', and 'Search & Query For Details'. The main content area is titled 'Getting Started' and 'Images'. It explains that movie, TV, and person objects contain references to different file paths and lists the three pieces of data needed to generate a fully working image URL: `base_url`, `file_size`, and `file_path`. It provides an example of a `poster_path` and shows the resulting URL: `https://image.tmdb.org/t/p/w500/kqjLl7yufvn90VlyK1pvtzrFfak.jpg`. It also mentions that company and network logos are available in SVG and PNG formats and lists example URLs for Netflix's logo.

The Movie Database API <sup>3</sup>  
https://api.themoviedb.org/3

OAS RAML Support

Select a different version  
Filter sections...

GETTING STARTED

- Introduction
- Authentication
- Daily File Exports
- Languages
- Images**
- Image Languages
- Regions
- External IDs
- Popularity
- Request Rate Limiting
- JSON & JSONP
- Append To Response
- Search & Query For Details

ACCOUNT

AUTHENTICATION

CERTIFICATIONS

CHANGES

COLLECTIONS

COMPANIES

CONFIGURATION

### Getting Started

## Images

You'll notice that movie, TV and person objects contain references to different file paths. In order to generate a fully working image URL, you'll need 3 pieces of data. Those pieces are a `base_url`, a `file_size` and a `file_path`.

The first two pieces can be retrieved by calling the `/configuration` API and the third is the file path you're wishing to grab on a particular media object. Here's what a full image URL looks like if the `poster_path` of `/kqjLl7yufvn90VlyK1pvtzrFfak.jpg` was returned for a movie, and you were looking for the `w500` size:

```
https://image.tmdb.org/t/p/w500/kqjLl7yufvn90VlyK1pvtzrFfak.jpg
```

Company and network logos are available in two formats, SVG and PNG. All of the `logo_path` fields will return a .png. This is to maintain backwards compatibility since SVG support was only added very recently. When looking at the image methods there is a new field called `file_type` that will show you the original version of the asset that was uploaded. For SVG's, you should call the `original` image size since we don't resize them. If you prefer to grab PNG's, you can call any size you wish just like normal.

Take for instance Netflix's logo (wwemzKWzjKjYfCeIB57q3r4Bcm.svg), you can call any of the following:

- <https://image.tmdb.org/t/p/original/wwemzKWzjKjYfCeIB57q3r4Bcm.svg>
- <https://image.tmdb.org/t/p/original/wwemzKWzjKjYfCeIB57q3r4Bcm.png>
- <https://image.tmdb.org/t/p/w500/wwemzKWzjKjYfCeIB57q3r4Bcm.png>

# CODIFICANDO

## PASSOS

- ▶ Colocar imagens para o sistema

```
TS movie.ts    TS feed.ts    <> feed.html x
11
12 <h2 class="meu_estilo_title"> {{appName}} </h2>
13
14 <!-- diretriva estrutural, para alterar a estrutura dos itens da pagina
15      repete o elemento
16      pegar o nome da variavel em TS
17 -->
18 <ion-card *ngFor="let movie of listaFilmes">
19
20   <ion-item>
21     <ion-avatar item-start>
22       
23     </ion-avatar>
24     <h2 class="feed_title">{{ movie.original_title }}</h2>
25     <p>{{ movie.release_date }}</p>
26   </ion-item>
27
28   <img [src]="`https://image.tmdb.org/t/p/w300/` + movie.backdrop_path">
29
```

# CODIFICANDO

## PASSOS

- ▶ Colocar imagens para o sistema

