

Criar um repositório com o nome **GalloFlix-Angular**

Clonar repositório localmente

Abrir o repositório no **Visual Studio Code**

No terminal digite o comando abaixo:

```
ng new galloflix
```

Ao surgir a pergunta abaixo digite Y e em seguida [Enter]:

```
? Would you like to add Angular routing? (y/N)
```

Na pergunta seguinte, simplesmente deixe a opção **CSS** marcada e pressione [Enter]:

```
? Which stylesheet format would you like to use?  
> CSS  
  SCSS  
  Sass  
  Less
```

Para executar o projeto no terminal execute os comandos abaixo:

```
cd galloflix
```

```
ng serve -o
```

Para parar a execução pressione no terminal **Ctrl+C**.

Abra o arquivo **src\index.html** e faça as alterações conforme o código abaixo:

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>Galloflix</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSm07SnpJef0486qhLnuZ2cdeRh002iuK6FUUVM" crossorigin="anonymous">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Ubuntu&display=swap"
rel="stylesheet">
</head>
<body>
  <app-root></app-root>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"
integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz"
crossorigin="anonymous"></script>
</body>
</html>
```

Essas alterações incluem o **bootstrap** e a **fonte do google Ubuntu** a nossa aplicação. Essas alterações podem ser feitas manualmente, através do seguinte processo:

1. Acesse o site www.getbootstrap.com clique no link **Docs** e na opção **Introduction** desça até a segunda opção do **Quick start**, e copie as linhas de uso do **bootstrap**.
2. Acesse o site <https://fonts.google.com>, pesquise pela fonte **Ubuntu**, e copie a opção de inclusão da fonte por link.

Abra o arquivo **src\styles.css** e faça as alterações conforme mostra o código abaixo:

```
*{
  margin: 0;
  padding: 0;
}

body{
  font-family: 'Ubuntu', sans-serif;
  background-color: #000000;
  overflow-x: hidden;
}
```

Abra o arquivo `src\app\app-component.html` apague todo o seu conteúdo e inclua a única linha abaixo:

```
<router-outlet></router-outlet>
```

Execute o projeto e verifique que será exibida apenas uma página com fundo preto.

Acesse o site do **bootstrap** e pesquise pelo elemento **navbar**, copiando o código de exemplo e colando no começo do arquivo **src\app\app-component.html**, conforme mostra o código abaixo:

```
<!-- navbar start -->  
<nav class="navbar navbar-expand-lg fixed-top">  
  <div class="container-fluid">  
    <a class="navbar-brand" href="#">GalloFlix</a>  
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-  
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle  
navigation">  
      <span class="navbar-toggler-icon"></span>  
    </button>  
    <div class="collapse navbar-collapse" id="navbarNav">  
      <ul class="navbar-nav ms-auto">  
        <li class="nav-item">  
          <a class="nav-link active" aria-current="page" href="#">Home</a>  
        </li>  
        <li class="nav-item">  
          <a class="nav-link" href="#">Pesquisar</a>  
        </li>  
      </ul>  
    </div>  
  </div>  
</nav>  
<!-- navbar end -->  
  
<p>  
lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem  
ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem  
ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem  
ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem  
ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum  
</p>  
  
<router-outlet></router-outlet>
```

Abra o arquivo `src\app\app-component.css` e inclua o código abaixo:

```
.navbar{
  backdrop-filter: blur(2px);
  background: rgba(0,0,0,0.5);
}

.navbar .navbar-brand{
  color: #E50914;
  font-weight: bold;
}

.navbar-nav .nav-item .nav-link{
  color: #E50914;
}
```

Agora vamos adicionar um evento de “escuta” para quando ocorrer a rolagem da página, a **navbar** mude de fundo transparente para fundo com cor fixa. Abra o arquivo `src\app\app-component.ts` e inclua o código abaixo:

```
import { Component, HostListener } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'galloflix';
  navbg: any;

  @HostListener('document:scroll') scrollover() {
    if (document.body.scrollTop > 0 || document.documentElement.scrollTop > 0) {
      this.navbg = {
        'background-color': '#000000'
      }
    } else {
      this.navbg = {}
    }
  }
}
```

Abra o arquivo `src\app\app-component.html` e apenas altere a primeira linha do código, incluindo na tag `nav` o estilo `navbg`:

```
<!-- navbar start -->
<nav class="navbar navbar-expand-lg fixed-top" [ngStyle]="navbg">
```

Salve tudo, execute o projeto e verifique o resultado até aqui. Se necessário aumente a quantidade de “Lorem Ipsum” para criar uma barra de rolagem na página inicial.

Abra o terminal e vamos criar os componentes que iram compor as demais páginas da aplicação. Execute os comandos abaixo, um por vez:

```
ng g c pages/home
ng g c pages/search
ng g c pages/movieDetails
```

Ng é a interface de linha de comando do angular, a letra “g” significa “generate” (gerar), e a letra “c” component (componente).

Abra o arquivo `src\app\app-routing.module.ts` e altere o código conforme mostrado abaixo, para incluir as rotas das páginas da aplicação:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './pages/home/home.component';
import { SearchComponent } from './pages/search/search.component';
import { MovieDetailsComponent } from './pages/movie-details/movie-details.component';

const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'search', component: SearchComponent},
  {path: 'movie/:id', component: MovieDetailsComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Volte ao arquivo `app.component.html` e altere apenas os links (tags `a`), trocando o atributo `href` por `routerLink`, conforme mostra as linhas abaixo:

```
<a class="navbar-brand" routerLink="">GalloFlix</a>
<a class="nav-link active" aria-current="page" routerLink="">Home</a>
<a class="nav-link" routerLink="search">Pesquisar</a>
```

Para continuar com o projeto é necessário criar uma conta no site <https://www.themoviedb.org/?language=pt>, que possui informações sobre filmes e séries, além de disponibilizar acesso a essas informações por meio de serviço online (API).

Após criar uma conta e fazer a confirmação por email, acesse no menu, clicando na sua foto de perfil, a opção **Configurações**, ou pelo link <https://www.themoviedb.org/settings/account?language=pt>. No menu lateral escolha a opção **API**, e gere uma nova chave, para utilizarmos em nossa aplicação. Será necessário preencher um formulário com mais algumas informações sobre a aplicação.

Com sua chave criada, volte ao Visual Studio Code, e vamos criar um serviço para acessar os dados da API e disponibilizar esses dados ao restante da aplicação.

Abra o terminal e execute o comando abaixo:

```
ng g s services/movieApi
```

Neste caso, temos a letra “s” para informar ao **Angular CLI**, que queremos criar um **serviço (service)**. O comando cria uma pasta com o nome **services** e os dois arquivos do serviço **movieAPI**.

Para fazer requisições a serviços **API**, precisamos de uma biblioteca específica, desta forma, é necessário alterar o arquivo **src\app\app.module.ts**, para adicionar esta biblioteca, além de disponibilizar nosso serviço (**movieAPI**) ao restante da aplicação.

Abra o arquivo **src\app\app.module.ts** e faça as alterações abaixo, para adicionar as linhas de **import**, do **HttpClientModule** e da **MovieApiService**, em seguida adicionar no vetor de **imports**, o módulo **HttpClient** e nos **providers** o **MovieApiService**:

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './pages/home/home.component';
import { SearchComponent } from './pages/search/search.component';
import { MovieDetailsComponent } from './pages/movie-details/movie-details.component';

import { HttpClientModule } from '@angular/common/http';
import { MovieApiService } from './services/movie-api.service';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    SearchComponent,
    MovieDetailsComponent
  ],
```

```

imports: [
  BrowserModule,
  AppRoutingModule,
  HttpClientModule
],
providers: [
  MovieApiService
],
bootstrap: [AppComponent]
})
export class AppModule { }

```

Abra o arquivo `src\app\services\movie-api.services.ts` e faça as alterações abaixo:

```

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class MovieApiService {

  constructor(private http: HttpClient) { }

  baseUrl = "https://api.themoviedb.org/3";
  apiKey = "<COLE SUA CHAVE API AQUI>";

  // banner API Data
  bannerApiData(): Observable<any> {
    return
    this.http.get(`${this.baseUrl}/trending/all/week?api_key=${this.apiKey}&language=pt-BR`);
  }
}

```

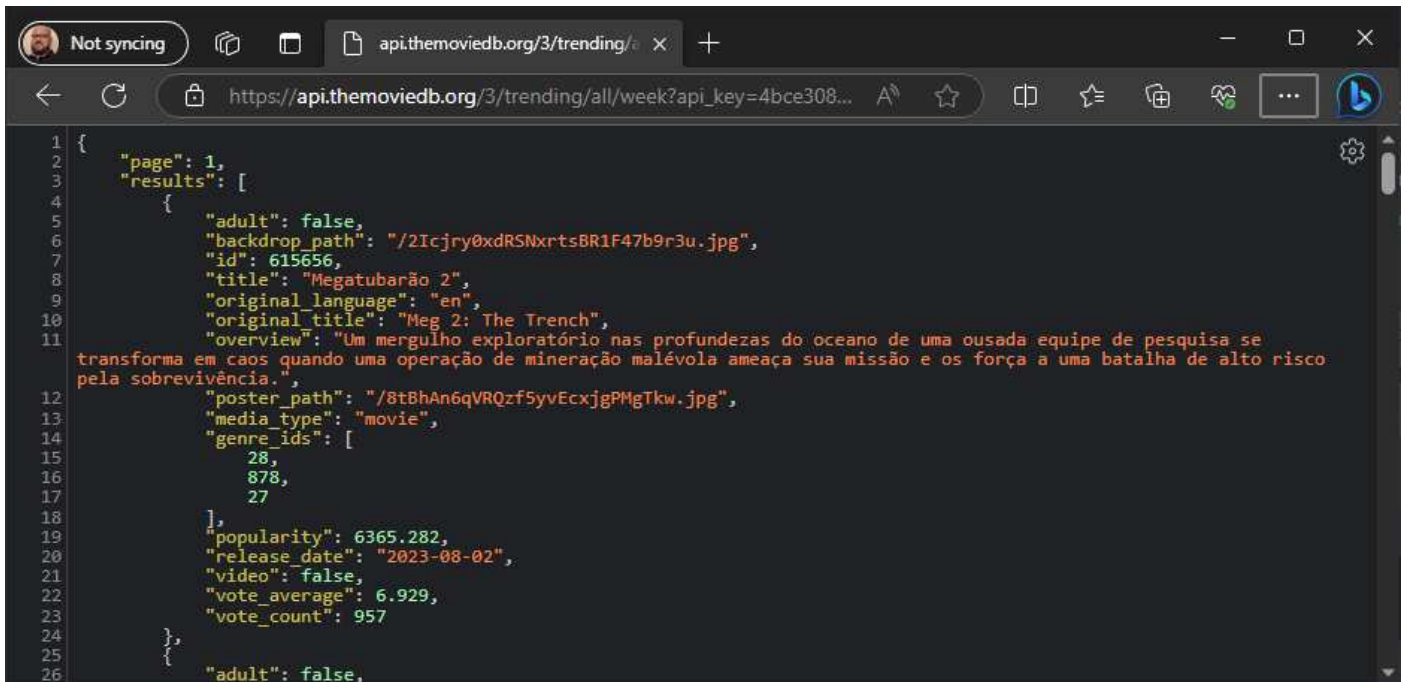
Não esqueça de trocar o texto **COLE SUA CHAVE API AQUI**, pela chave que você criou no site

A função **bannerApiData** usa o serviço injetado no construtor **HttpClient**, para executar a leitura de dados da **API**.

O **'this.http.get'** é um método fornecido pelo módulo **HttpClient** do **Angular** que é usado para fazer requisições **HTTP GET** para um servidor. Essas requisições **GET** são usadas para buscar informações de um servidor, como dados de uma **API**.

Você consegue ver o que será retornado pela função, abrindo um navegador e colando na barra de endereços a rota que usamos na função. Basta copiar e colar o código abaixo no navegador, substituindo o texto em destaque pela sua chave **API**:

https://api.themoviedb.org/3/trending/all/week?api_key=COLE SUA CHAVE API AQUI&language=pt-BR



Abra o arquivo `src\app\app.component.html` e apague totalmente a tag `<p>` com o conteúdo de “Lorem Ipsum”, deixando apenas a tag `nav` (com seu conteúdo) e `router-outlet`.

```
<!-- navbar start -->
<nav class="navbar navbar-expand-lg fixed-top" [ngStyle]="navbg">
  <div class="container-fluid">
    <a class="navbar-brand" routerLink="">GalloFlix</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav ms-auto">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" routerLink="">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" routerLink="search">Pesquisar</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
<!-- navbar end -->

<router-outlet></router-outlet>
```


Abra o arquivo `src\app\pages\home\home.component.ts` e vamos fazer as alterações necessárias para injetar e utilizar o serviço `MovieApi`, buscando os filmes de destaque com o método `bannerApiData()` do serviço:

```
import { Component } from '@angular/core';
import { MovieApiService } from 'src/app/services/movie-api.service';
@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {

  constructor(private service:MovieApiService) { }

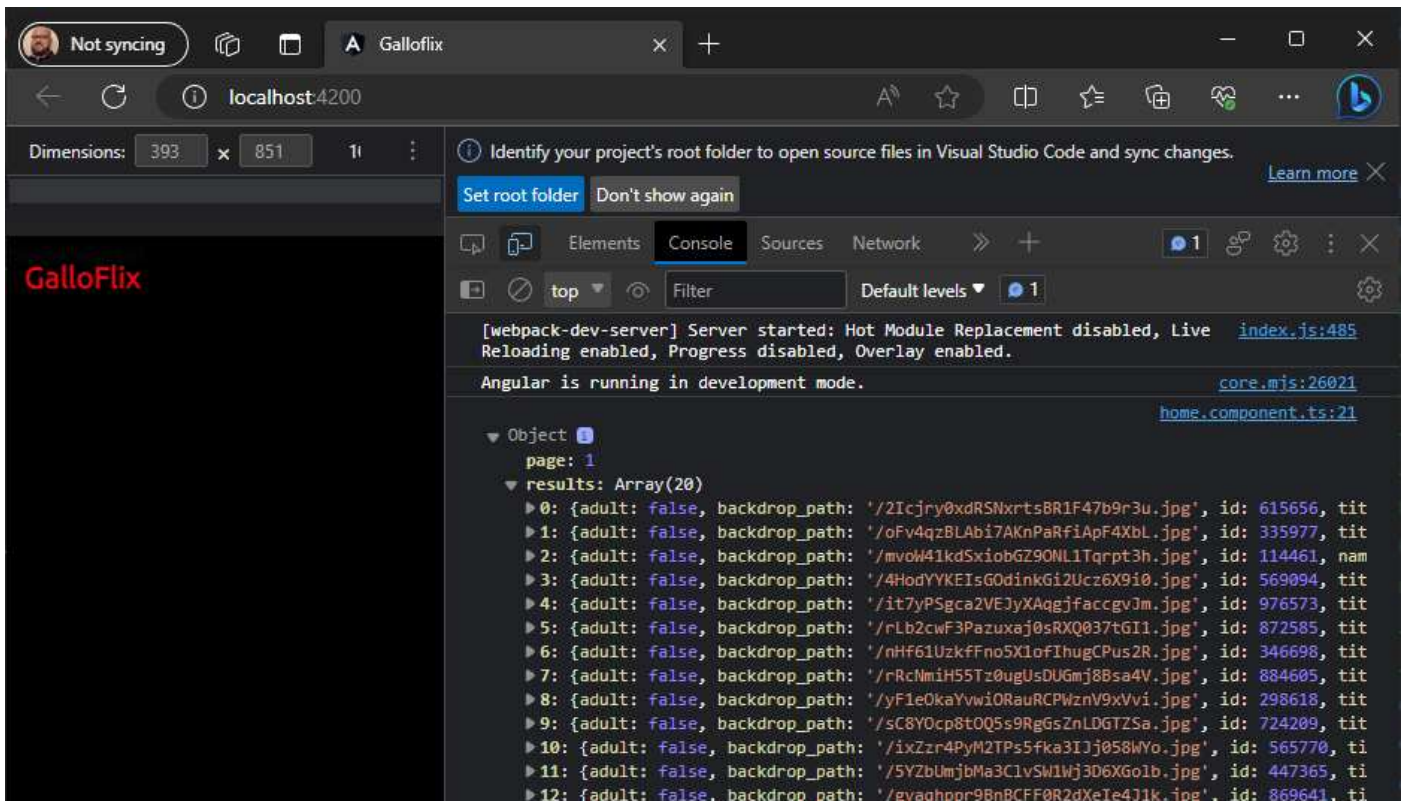
  ngOnInit(): void {
    this.bannerData();
  }

  // Banner Data
  bannerData(){
    this.service.bannerApiData().subscribe((result)=>{
      console.log(result, 'bannerResult#');
    });
  }
}
```

O método `subscribe` é usado para se inscrever em um fluxo de dados assíncrono, como o resultado de uma chamada `HTTP`. Quando você faz uma requisição `HTTP` usando `this.httpClient.get`, a resposta não é retornada imediatamente. Em vez disso, você recebe um objeto observável que representa um fluxo de dados que pode mudar ao longo do tempo. Para acessar efetivamente os dados retornados pela chamada `HTTP`, você precisa se inscrever no observável usando o método `subscribe`.

Em resumo, o fluxo típico ao fazer uma chamada `HTTP` no Angular envolve usar `this.httpClient.get` para iniciar a requisição, e em seguida, usar `subscribe` para se inscrever no observável resultante e receber os dados ou lidar com erros. Isso permite que você trabalhe de maneira assíncrona e reativa com as respostas das chamadas `HTTP` em suas aplicações Angular.

Salve todo o projeto, execute, abra no navegador e observe o resultado apresentado no console das ferramentas do desenvolvedor:



Volte ao arquivo `src\app\pages\home\home.component.ts` e vamos alterar a classe **HomeComponent**, para exibir o resultado da chamada a API na página, vamos adicionar um vetor para receber os resultados:

```
import { Component } from '@angular/core';
import { MovieApiService } from 'src/app/services/movie-api.service';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {

  constructor(private service:MovieApiService) { }

  bannerResult: any = [];

  ngOnInit(): void {
    this.bannerData();
  }

  // Banner Data
  bannerData(){
    this.service.bannerApiData().subscribe((result)=>{
      console.log(result, 'bannerResult#');
      this.bannerResult = result.results;
    });
  }
}
```

Abra o arquivo `src\app\pages\home\home.component.html`, apague o conteúdo existente e faça as alterações abaixo:

```
<!-- banner start -->
<div class="container-fluid">

</div>
<!-- banner end -->
```

Abra o site do bootstrap e pesquise por carousel, ou clique neste link, <https://getbootstrap.com/docs/5.3/components/carousel/>, localize e copie o código de exemplo do **Carousel com Captions**. Cole dentro do container e faça as alterações conforme o código abaixo:

```
<!-- banner start -->
<div class="container-fluid p-0">
  <div id="carouselExampleCaptions" class="carousel slide" data-bs-ride="carousel">
    <div class="carousel-inner">
      <div class="carousel-item active" data-bs-interval="10000" *ngFor="let b of bannerResult">
        
        <div class="carousel-caption d-none d-md-block">
          <h5>First slide label</h5>
          <p>Some representative placeholder content for the first slide.</p>
        </div>
      </div>
    </div>
    <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleCaptions"
data-bs-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Anterior</span>
    </button>
    <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleCaptions"
data-bs-slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Próximo</span>
    </button>
  </div>
</div>
<!-- banner end -->
```

Abra o arquivo `src\app\pages\home\home.component.css`, e adicione o estilo abaixo para limitar o **carousel** ao tamanho máximo da tela.

```
.carousel-inner{
  max-height: 100vh;
}
```

```

.carousel-inner .carousel-item img{
  max-height: 100vh;
  object-fit: cover;
}

.carousel-caption{
  background: rgba(0,0,0,0.5);
  backdrop-filter: blur(5px);
  width: 100%;
  left: 0;
  padding-left: 100px;
  padding-right: 100px;
}

.carousel-caption h2{
  color: #E50914;
  font-weight: bolder;
  font-size: 30px;
}

.carousel-caption p{
  color: #ffffff;
}

```

Salve tudo e execute, você verá uma única imagem e o carousel não funcionará, vamos usar a interpolação (`{{propriedade}}`) para trazer os dados do vetor do banner.

Volte ao arquivo `src\app\pages\home\home.component.html` e altere o código conforme apresentado abaixo:

```

<!-- banner start -->
<div class="container-fluid p-0">
  <div id="carouselExampleCaptions" class="carousel slide" data-bs-ride="carousel">
    <div class="carousel-inner">
      <div class="carousel-item" data-bs-interval="10000" *ngFor="let b of
bannerResult; let i=index"
        [ngClass]="{active:i===0}">
        
        <div class="carousel-caption d-none d-md-block">
          <h2>{{b.title??b.name}}</h2>
          <p>{{b.overview}}</p>
        </div>
      </div>
    </div>
    <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleCaptions"
      data-bs-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>

```

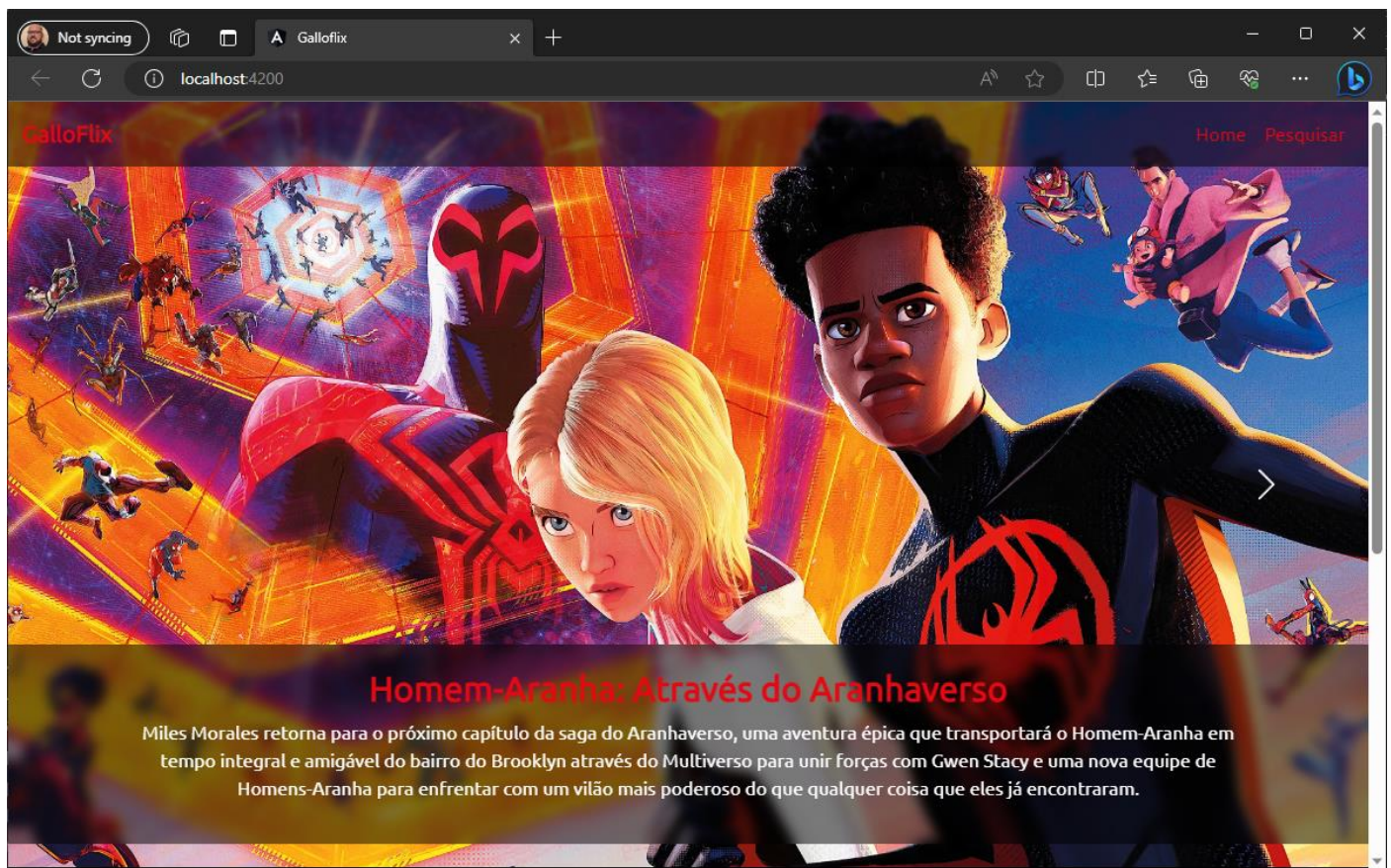


```

        <span class="visually-hidden">Anterior</span>
      </button>
      <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleCaptions"
        data-bs-slide="next">
        <span class="carousel-control-next-icon" aria-hidden="true"></span>
        <span class="visually-hidden">Próximo</span>
      </button>
    </div>
  </div>
<!-- banner end -->

```

Salve tudo e execute novamente.



Vamos agora criar as áreas de filmes em destaque.

Abra o arquivo `src\app\services\movie-api.services.ts` e adicione abaixo do método `bannerApiData()` o código a seguir:

```

// trending Movies API Data
trendingMovieApiData(): Observable<any> {
  return
this.http.get(`${this.baseUrl}/trending/movie/day?api_key=${this.apiKey}&language=pt-BR`);
}

```

Volte ao arquivo `src\app\pages\home\home.component.ts` e vamos alterar a classe `HomeComponent`, para exibir o resultado da chamada a API na página, vamos adicionar um vetor para receber os resultados, faça as alterações em destaque (fundo azul):

```
import { Component } from '@angular/core';
import { MovieApiService } from 'src/app/services/movie-api.service';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {

  constructor(private service: MovieApiService) { }

  bannerResult: any = [];
  trendingMovieResult: any = [];

  ngOnInit(): void {
    this.bannerData();
    this.trendingData();
  }

  // Banner Data
  bannerData() {
    this.service.bannerApiData().subscribe((result) => {
      this.bannerResult = result.results;
    });
  }

  // Trending Data
  trendingData() {
    this.service.trendingMovieApiData().subscribe((result) => {
      this.trendingMovieResult = result.results;
    });
  }
}
```

No arquivo `src\app\pages\home\home.component.html`, adicione ao final do código, a `div` abaixo para exibir o resultado da chamada a API:

```
<!-- banner end -->

<div class="container my-5">
  <!-- trending movies -->
  <div class="row">
    <h5 class="text-white">Filmes em Destaque</h5>
    <div class="rowposter" *ngFor="let t of trendingMovieResult">
```

```

        
    </div>
</div>
</div>

```

Salve tudo e execute, você terá como resultado várias imagens dos filmes de destaque um embaixo do outro, vamos usar CSS e uma alteração no código para melhorar o resultado.

Abra o arquivo `src\app\pages\home\home.component.css`, e adicione o código abaixo ao final do arquivo (**não apague o conteúdo atual**):

```

.rowposter{
  display: flex;
  overflow-y: hidden;
  overflow-x: scroll;
}

.rowimg {
  max-height: 250px;
  object-fit: contain;
  width: 100%;
  box-sizing: content-box;
  transition: transform 450ms;
  margin-right: 7px;
}

.rowimg:hover{
  transform: scale(1.10);
  cursor: pointer;
}

.largeposter{
  max-height: 300px;
}

.rowposter::-webkit-scrollbar{
  display: none;
}

```

Volte ao arquivo `src\app\pages\home\home.component.html` e altere o código da `div` de **Destaque** trocando pelo código abaixo:

```

<!-- trending movies -->
<div class="row">
  <h5 class="text-white">Filmes em Destaque</h5>
  <div class="rowposter mt-3 p-2">
    <ng-container *ngFor="let t of trendingMovieResult">

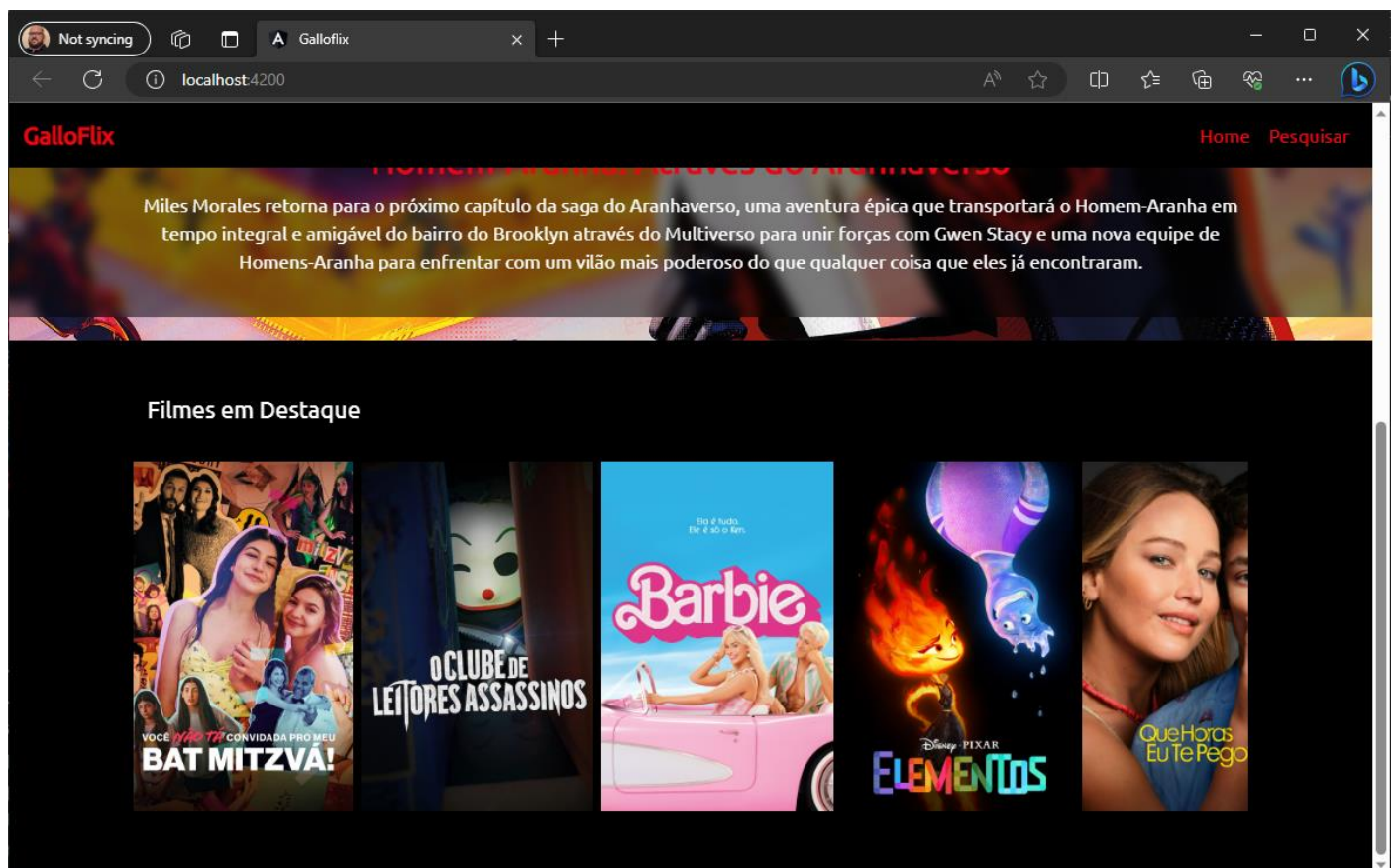
```

```


</ng-container>
</div>
</div>

```

Salve tudo e execute novamente, desça a página principal até o final onde você poderá ver os Filmes em Destaque. Pare o ponteiro do mouse sobre um dos filmes segure a tecla **Shift** e role o **wheel** do mouse (rodinha) para percorrer a lista de filmes.



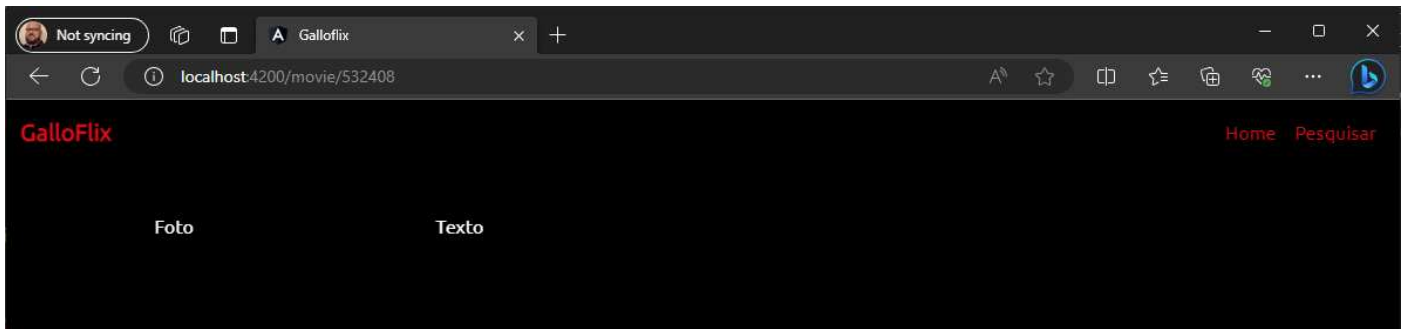
Vamos agora criar a página de detalhes do filme, para isso abra o arquivo `app\src\pages\movie-details\movie-details.component.html`, apague o conteúdo atual do arquivo e insira o código abaixo:

```

<div class="container mt-5 pt-5">
  <div class="row text-white">
    <div class="col-md-3">
      Foto
    </div>
    <div class="col-md-9">
      Texto
    </div>
  </div>
</div>

```


Se você salvar tudo e executar, ao clicar em um dos filmes em destaque será redirecionado ao componente.



Abra o arquivo `src\app\services\movie-api.services.ts` e adicione abaixo do método `trendingMovieApiData()` o código a seguir, que será responsável por trazer os dados do filme clicado pelo usuário:

```
// Movie Details API Data
movieDetails(data: any): Observable<any> {
  return
this.http.get(`${this.baseUrl}/movie/${data}?api_key=${this.apiKey}&language=pt-BR`);
}
```

Abra o arquivo `src\app\pages\movie-details\movie-details.component.ts` e faça as alterações abaixo para adicionar o serviço `MovieApi`, e receber o resultado da requisição `HTTP` para exibição dos detalhes do filme:

```
import { Component } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { MovieApiService } from 'src/app/services/movie-api.service';

@Component({
  selector: 'app-movie-details',
  templateUrl: './movie-details.component.html',
  styleUrls: ['./movie-details.component.css']
})
export class MovieDetailsComponent {

  constructor(private service:MovieApiService, private router:ActivatedRoute) { }

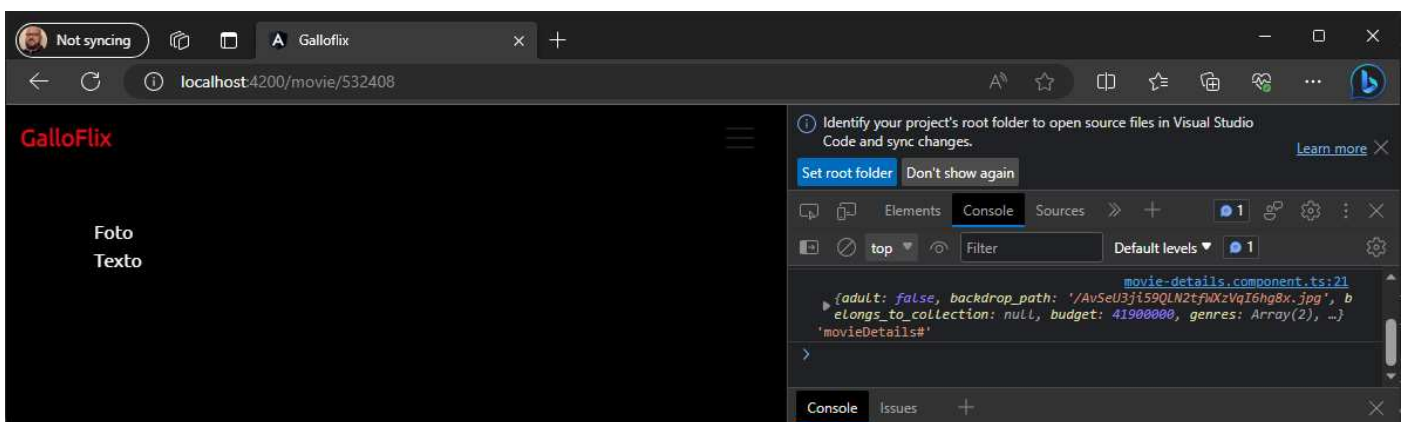
  ngOnInit(): void{
    let id = this.router.snapshot.paramMap.get('id');
    this.getMovie(id);
  }

  getMovie(id:any){
    this.service.movieDetails(id).subscribe((result)=>{
      console.log(result, 'movieDetails#');
    });
  }
}
```

Aqui, evento de inicialização do componente (**ngOnInit**), utilizamos o objeto **privado router**, que é do tipo **ActivatedRoute**, para extrair da rota de navegação o id do filme clicado pelo usuário.

O **ActivatedRoute** é uma classe fornecida pelo **Angular** que representa o estado atual da rota ativada. Ela contém informações sobre os parâmetros, consultas e dados associados a uma rota específica. Em outras palavras, o **ActivatedRoute** é usado para acessar os detalhes da rota que está sendo acessada no momento, permitindo que você obtenha informações como os parâmetros da **URL** e dados associados a essa rota específica. Isso é útil para construir lógica dinâmica e personalizada com base na **URL** atual.

Salvando tudo e executando temos como resultado no **console.log**, os dados do filme clicado pelo usuário, conforme visto na imagem abaixo.



Voltando ao arquivo **src\app\pages\movie-details\movie-details.component.ts**, vamos criar um objeto e popular este objeto com os dados recebidos pela chamada da **API**.

```
import { Component } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { MovieApiService } from 'src/app/services/movie-api.service';

@Component({
  selector: 'app-movie-details',
  templateUrl: './movie-details.component.html',
  styleUrls: ['./movie-details.component.css']
})
export class MovieDetailsComponent {

  constructor(private service:MovieApiService, private router:ActivatedRoute) { }

  movieResult: any;

  ngOnInit(): void{
    let id = this.router.snapshot.paramMap.get('id');
    this.getMovie(id);
  }
}
```

```

getMovie(id:any){
  this.service.movieDetails(id).subscribe((result)=>{
    //console.log(result, 'movieDetails#');
    this.movieResult = result;
  });
}
}

```

Agora basta alterar a página de detalhes do filme, alterando o arquivo `app\src\pages\movie-details\movie-details.component.html`, conforme o código abaixo:

```

<div class="container mt-5 pt-5">
  <div class="row" *ngIf="movieResult">
    <div class="col-md-4">
      
    </div>
    <div class="col-md-8 text-white">
      <h1>{{movieResult.title}}</h1>
      <h5 class="mt-4">Título Original: {{movieResult.original_title}}</h5>
      <p>Data de Estreia: {{movieResult.release_date | date:'dd/MM/yyyy'}}</p>
      <p>{{movieResult.overview}}</p>
    </div>
  </div>
</div>

```

Salve execute e clique em um dos filmes em destaque



Agora vamos adicionar novos métodos ao **movieApi**, para recuperar do filme, o trailer e os atores e atrizes que participaram do filme selecionado pelo usuário.

Abra o arquivo **src\app\services\movie-api.services.ts** e adicione abaixo do método **movieDetails()** as duas funções abaixo:

```
// Movie Trailer API Data
movieVideo(data: any): Observable<any> {
  return
this.http.get(`${this.baseUrl}/movie/${data}/videos?api_key=${this.apiKey}&language=pt-BR`);
}

// Movie Cast API Data
movieCast(data: any): Observable<any> {
  return
this.http.get(`${this.baseUrl}/movie/${data}/credits?api_key=${this.apiKey}&language=pt-BR`);
}
```

Voltando ao arquivo **src\app\pages\movie-details\movie-details.component.ts**, vamos criar dois objetos e inserir os dados recebidos pelas chamadas da API.

```
import { Component } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { MovieApiService } from 'src/app/services/movie-api.service';

@Component({
  selector: 'app-movie-details',
  templateUrl: './movie-details.component.html',
  styleUrls: ['./movie-details.component.css']
})
export class MovieDetailsComponent {

  constructor(private service:MovieApiService, private router:ActivatedRoute) { }

  movieResult: any;
  movieVideoResult: any;
  movieCastResult: any;

  ngOnInit(): void{
    let id = this.router.snapshot.paramMap.get('id');
    this.getMovie(id);
    this.getVideo(id);
    this.getCast(id);
  }

  getMovie(id:any){
    this.service.movieDetails(id).subscribe((result)=>{
      //console.log(result, 'movieDetails#');
      this.movieResult = result;
    });
  }
}
```


Para conseguirmos exibir o trailer do YouTube, vamos precisar fazer uma configuração que permita a execução segura, desta forma, abra o arquivo `src\app\app.module.ts`, e faça as alterações em destaque:

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './pages/home/home.component';
import { SearchComponent } from './pages/search/search.component';
import { MovieDetailsComponent } from './pages/movie-details/movie-details.component';

import { HttpClientModule } from '@angular/common/http';
import { MovieApiService } from './services/movie-api.service';

import { DomSanitizer } from "@angular/platform-browser";
import { Pipe, PipeTransform } from "@angular/core";

@Pipe({ name: 'safe' })
export class SafePipe implements PipeTransform {
  constructor(private sanitizer: DomSanitizer) { }
  transform(url: string) {
    return this.sanitizer.bypassSecurityTrustResourceUrl(url);
  }
}

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    SearchComponent,
    MovieDetailsComponent,
    SafePipe
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [
    MovieApiService
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Agora volte ao arquivo `src\app\pages\movie-details\movie-details.component.ts`, para alterar as funções criadas anteriormente, para enviar os resultados recebidos pela API, aos objetos serão exibidos na página.

```

getVideo(id:any){
  this.service.movieVideo(id).subscribe((result)=>{
    result.results.forEach((elem:any) => {
      if (elem.type == "Trailer")
      {
        this.movieVideoResult = "https://www.youtube.com/embed/" + elem.key;
      }
    });
    console.log(this.movieVideoResult, 'movieVideo#');
  });
}

getCast(id:any){
  this.service.movieCast(id).subscribe((result)=>{
    this.movieCastResult = result.cast;
    console.log(this.movieCastResult, 'movieVideo#');
  });
}

```

E por fim, podemos adicionar ao final do arquivo `app\src\pages\movie-details\movie-details.component.html`, as `divs` abaixo para exibir o trailer e o elenco do filme.

```

<div class="container my-5 text-white" *ngIf="movieVideoResult">
  <h3 class="my-3">Trailer Oficial</h3>
  <iframe width="100%" height="500" [src]="movieVideoResult | safe"></iframe>
</div>

<div class="container my-5 text-white" *ngIf="movieCastResult">
  <h3 class="my-3">Elenco</h3>
  <div class="row">
    <div class="col-lg-4 col-sm-6" *ngFor="let c of movieCastResult">
      <div class="row">
        <div class="col-lg-3 mb-3" *ngIf="c.profile_path">
          
        </div>
        <div class="col-lg-9 mb-3">
          <h3>{{c.original_name}}</h3>
          <p>{{c.character}}</p>
        </div>
      </div>
    </div>
  </div>
</div>

```


Salve tudo e execute novamente, desça a página para ver as novas informações disponíveis.

Not syncing

Gallofix

localhost:4200/movie/615656

HomePesquisar




MEGATUBARÃO 2

Trailer Oficial


MEGATUBARÃO 2 - Trailer Oficial

Assistir m...Compartilh...




Assistir no YouTube


Elenco



Jason Statham
Jonas Taylor



Wu Jing
Jiuming



Shuya Sophia Cai
Meiyang

Vamos agora montar a página de pesquisa.

Abra o arquivo `src\app\pages\search\search.component.html`, apague seu conteúdo e insira o código abaixo:

```
<div class="container mt-5 pt-5">
  <form>
    <div class="input-group mb-3">
      <input type="text" class="form-control" placeholder="Pesquisar Filme...">
      <button class="btn btn-danger" type="button">Pesquisar</button>
    </div>
  </form>
</div>
```

Abra o arquivo `src\app\pages\search\search.component.css`, e insira o código de estilização abaixo:

```
button {
  background-color: #E50914;
  color: #ffffff;
}

#movie{
  cursor: pointer;
}

img{
  width: 100%;
}

#details{
  color: #ffffff;
}
```

Salve tudo e veja uma prévia da página de pesquisa.

Para realizar as pesquisas, vamos usar o **ReactiveFormsModule** do Angular que são uma parte da biblioteca de formulários reativos do Angular. Eles fornecem uma maneira mais poderosa e flexível de gerenciar e validar formulários em comparação com os formulários **template-driven** (baseados em modelos).

Para isso abra o arquivo `src\app\app.module.ts`, e adicione apenas as linhas em destaque abaixo:

```
import { ReactiveFormsModule } from '@angular/forms';

@NgModule({
  imports: [
    ReactiveFormsModule
  ],
  providers: [
```

Agora abra o arquivo `src\app\pages\search\search.component.ts`, faça as alterações abaixo, para criar o formulário reativo e receber as informações enviadas pelo formulário no evento `submitForm()`:

```
import { Component } from '@angular/core';
import { FormControl, FormGroup } from '@angular/forms';

@Component({
  selector: 'app-search',
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.css']
})
export class SearchComponent {

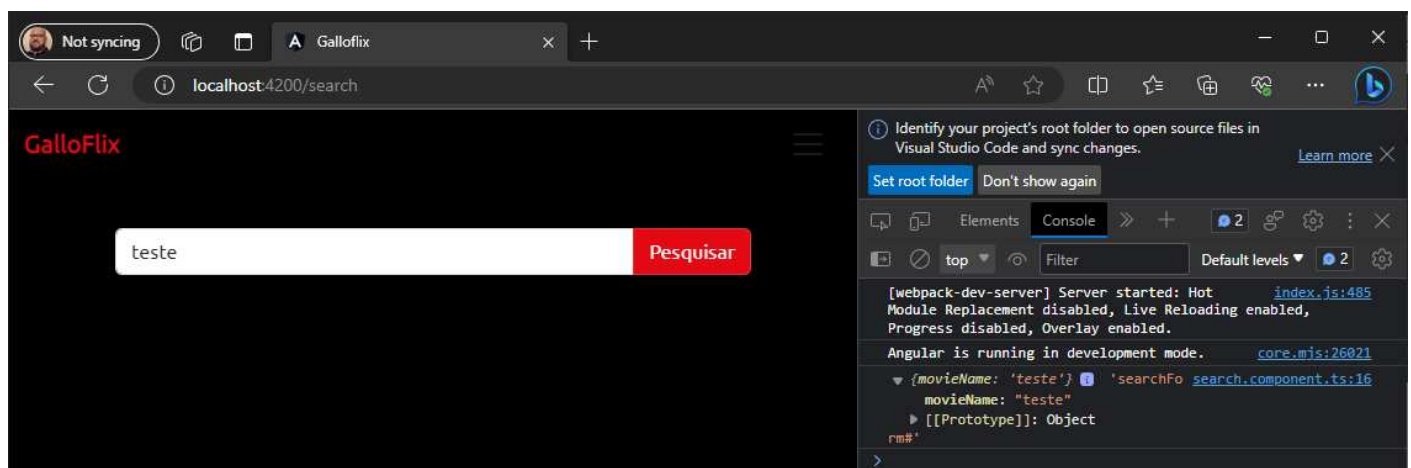
  searchForm = new FormGroup({
    'movieName': new FormControl(null)
  });

  submitForm(){
    console.log(this.searchForm.value, 'searchForm#');
  }
}
```

Volte ao arquivo `src\app\pages\search\search.component.html`, e faça as alterações abaixo, para ligar o formulário reativo do código ao `html` da página:

```
<div class="container mt-5 pt-5">
  <form [formGroup]="searchForm" (ngSubmit)="submitForm()">
    <div class="input-group mb-3">
      <input type="text" class="form-control"
        formControlName="movieName"
        placeholder="Pesquisar Filme...">
      <button class="btn btn-danger" type="submit">Pesquisar</button>
    </div>
  </form>
</div>
```

Salve execute e veja o resultado de realizar uma pesquisa no `console.log`:



Agora vamos precisamos adicionar um método ao **movieApi**, para realizar a pesquisa do texto digitado pelo usuário.

Abra o arquivo **src\app\services\movie-api.services.ts** e adicione abaixo do método **movieCast()** a função abaixo:

```
// search Movie API Data
searchMovie(data: any): Observable<any> {
  return
this.http.get(`${this.baseUrl}/search/movie?api_key=${this.apiKey}&query=${data.movieName}&language=pt-BR`);
}
```

Agora volte ao arquivo **src\app\pages\search\search.component.ts**, faça as alterações abaixo, para adicionar o serviço **MovieApi** a classe e executar a função de pesquisa, trazendo os resultados da pesquisa para o **console** da página:

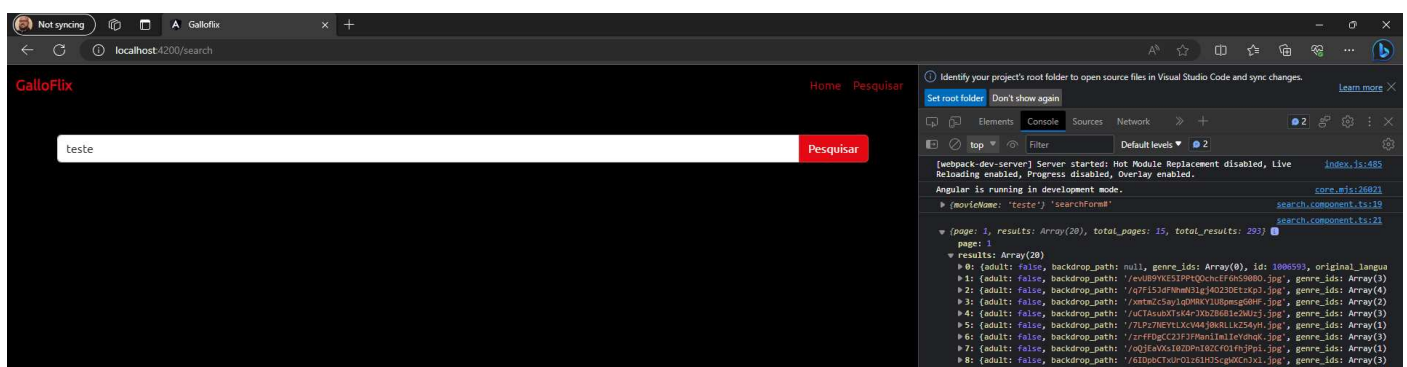
```
import { Component } from '@angular/core';
import { FormControl, FormGroup } from '@angular/forms';
import { MovieApiService } from 'src/app/services/movie-api.service';

@Component({
  selector: 'app-search',
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.css']
})
export class SearchComponent {

  constructor(private service:MovieApiService) { }

  searchForm = new FormGroup({
    'movieName': new FormControl(null)
  });

  submitForm(){
    console.log(this.searchForm.value, 'searchForm#');
    this.service.searchMovie(this.searchForm.value).subscribe((result)=>{
      console.log(result, 'searchMovieSubmit#');
    });
  }
}
```



Agora volte ao arquivo `src\app\pages\search\search.component.ts`, e adicione o objeto `searchResult` e altere a função `submitForm()`, conforme os código abaixo:

```
searchResult: any;

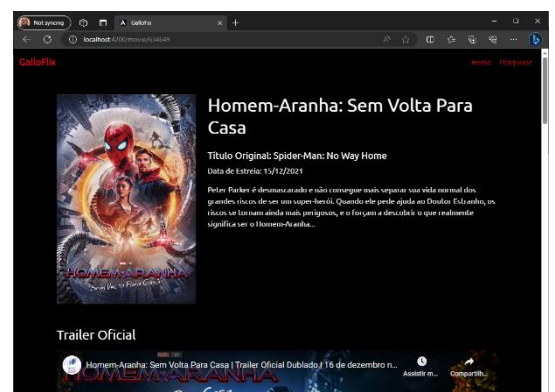
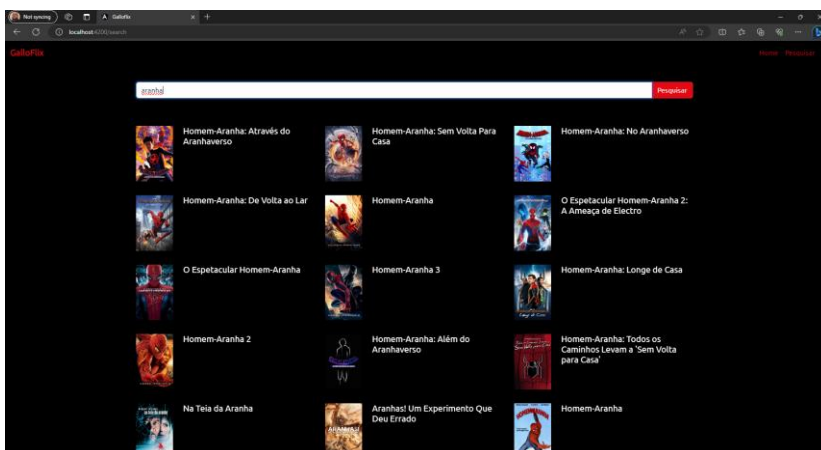
submitForm(){
  console.log(this.searchForm.value, 'searchForm#');
  this.service.searchMovie(this.searchForm.value).subscribe((result)=>{
    console.log(result, 'searchMovieSubmit#');
    this.searchResult = result.results;
  });
}
```

E para exibir os resultados da pesquisa, abra o arquivo `src\app\pages\search\search.component.html`, e faça as alterações abaixo:

```
<div class="container mt-5 pt-5">
  <form [formGroup]="searchForm" (ngSubmit)="submitForm()">
    <div class="input-group mb-3">
      <input type="text" class="form-control"
        FormControlName="movieName"
        placeholder="Pesquisar Filme...">
      <button class="btn btn-danger" type="submit">Pesquisar</button>
    </div>
  </form>

  <div class="row mt-5">
    <div class="col-lg-4" *ngFor="let s of searchResult">
      <div id="movie" class="row" [routerLink]="['/movie',s.id]">
        <div class="col-lg-3 my-3" *ngIf="s.poster_path">
          
        </div>
        <div id="details" class="col-lg-9 my-3">
          <h5>{{s.title}}</h5>
        </div>
      </div>
    </div>
  </div>
</div>
```

Salvando e executando:



Para finalizar estes projetos, vamos adicionar novas áreas a página inicial, seguindo a mesma lógica da área de detalhes, para exibir filmes por gênero. Desta forma, vamos adicionando ao arquivo `src\app\services\movie-api.services.ts`, abaixo do método `searchMovie()` as funções abaixo:

```
// Action Movies
fetchActionMovies(): Observable<any> {
  return
this.http.get(`${this.baseUrl}/discover/movie?api_key=${this.apiKey}&with_genres=28&language=pt-BR`);
}

// Adventure Movies
fetchAdventureMovies(): Observable<any> {
  return
this.http.get(`${this.baseUrl}/discover/movie?api_key=${this.apiKey}&with_genres=12&language=pt-BR`);
}

// Animation Movies
fetchAnimationMovies(): Observable<any> {
  return
this.http.get(`${this.baseUrl}/discover/movie?api_key=${this.apiKey}&with_genres=16&language=pt-BR`);
}

// Comedy Movies
fetchComedyMovies(): Observable<any> {
  return
this.http.get(`${this.baseUrl}/discover/movie?api_key=${this.apiKey}&with_genres=35&language=pt-BR`);
}

// Documentary Movies
fetchDocumentaryMovies(): Observable<any> {
  return
this.http.get(`${this.baseUrl}/discover/movie?api_key=${this.apiKey}&with_genres=99&language=pt-BR`);
}

// Science-Fiction Movies
fetchScienceFictionMovies(): Observable<any> {
  return
this.http.get(`${this.baseUrl}/discover/movie?api_key=${this.apiKey}&with_genres=878&language=pt-BR`);
}

// Thriller Movies
fetchThrillerMovies(): Observable<any> {
  return
this.http.get(`${this.baseUrl}/discover/movie?api_key=${this.apiKey}&with_genres=53&language=pt-BR`);
}
```

No arquivo `src\app\pages\home\home.component.ts`, iremos adicionar novos vetores de objetos e as funções para fazer as chamadas à API.

```
import { Component } from '@angular/core';
import { MovieApiService } from 'src/app/services/movie-api.service';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {

  constructor(private service: MovieApiService) { }

  bannerResult: any = [];
  trendingMovieResult: any = [];
  actionMovieResult: any=[];
  adventureMovieResult: any=[];
  animationMovieResult: any=[];
  comedyMovieResult: any=[];
  documentaryMovieResult: any=[];
  scienceFictionMovieResult: any=[];
  thrillerMovieResult: any=[];

  ngOnInit(): void {
    this.bannerData();
    this.trendingData();
    this.actionMovieData();
    this.adventureMovieData();
    this.animationMovieData();
    this.comedyMovieData();
    this.documentaryMovieData();
    this.scienceFictionMovieData();
    this.thrillerMovieData();
  }

  // Banner Data
  bannerData() {
    this.service.bannerApiData().subscribe((result) => {
      console.log(result, 'bannerResult#');
      this.bannerResult = result.results;
    });
  }

  // Trending Data
  trendingData() {
    this.service.trendingMovieApiData().subscribe((result) => {
      console.log(result, 'trendingResult#');
      this.trendingMovieResult = result.results;
    });
  }
}
```

```
// Action Movies
actionMovieData(){
  this.service.fetchActionMovies().subscribe((result)=>{
    this.actionMovieResult = result.results;
  });
}

// Adventure Movies
adventureMovieData(){
  this.service.fetchAdventureMovies().subscribe((result)=>{
    this.adventureMovieResult = result.results;
  });
}

// Animation Movies
animationMovieData(){
  this.service.fetchAnimationMovies().subscribe((result)=>{
    this.animationMovieResult = result.results;
  });
}

// Comedy Movies
comedyMovieData(){
  this.service.fetchComedyMovies().subscribe((result)=>{
    this.comedyMovieResult = result.results;
  });
}

// Documentary Movies
documentaryMovieData(){
  this.service.fetchDocumentaryMovies().subscribe((result)=>{
    this.documentaryMovieResult = result.results;
  });
}

// Science-Fiction Movies
scienceFictionMovieData(){
  this.service.fetchScienceFictionMovies().subscribe((result)=>{
    this.scienceFictionMovieResult = result.results;
  });
}

// Thriller Movies
thrillerMovieData(){
  this.service.fetchThrillerMovies().subscribe((result)=>{
    this.thrillerMovieResult = result.results;
  });
}
}
```

Por fim, vamos alterar o arquivo `src\app\pages\home\home.component.html`, para incluir as novas informações coletadas. Adicione ao seu código as linhas em destaque.

```
<!-- banner start -->
<div class="container-fluid p-0">
  <div id="carouselExampleCaptions" class="carousel slide" data-bs-ride="carousel">
    <div class="carousel-inner">
      <div class="carousel-item" data-bs-interval="10000" *ngFor="let b of
bannerResult; let i=index"
        [ngClass]="{active:i===0}">
        
        <div class="carousel-caption d-none d-md-block">
          <h2>{{b.title??b.name}}</h2>
          <p>{{b.overview}}</p>
        </div>
      </div>
    </div>
    <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleCaptions"
      data-bs-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Anterior</span>
    </button>
    <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleCaptions"
      data-bs-slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Próximo</span>
    </button>
  </div>
</div>
<!-- banner end -->

<div class="container my-5">

  <!-- trending movies -->
  <div class="row">
    <h5 class="text-white">Filmes em Destaque</h5>
    <div class="rowposter mt-3 p-2">
      <ng-container *ngFor="let t of trendingMovieResult">
        
      </ng-container>
    </div>
  </div>

  <!-- action movies -->
  <div class="row mt-5">
    <h5 class="text-white">Filmes de Ação</h5>
```



```

        <div class="rowposter mt-3 p-2">
            <ng-container *ngFor="let a of actionMovieResult">
                
            </ng-container>
        </div>
    </div>

    <!-- adventure movies -->
    <div class="row mt-5">
        <h5 class="text-white">Filmes de Aventura </h5>
        <div class="rowposter mt-3 p-2">
            <ng-container *ngFor="let a of adventureMovieResult">
                
            </ng-container>
        </div>
    </div>

    <!-- animation movies -->
    <div class="row mt-5">
        <h5 class="text-white">Filmes de Animação</h5>
        <div class="rowposter mt-3 p-2">
            <ng-container *ngFor="let a of animationMovieResult">
                
            </ng-container>
        </div>
    </div>

    <!-- comedy movies -->
    <div class="row mt-5">
        <h5 class="text-white">Filmes de Comédia</h5>
        <div class="rowposter mt-3 p-2">
            <ng-container *ngFor="let c of comedyMovieResult">
                
            </ng-container>
        </div>
    </div>

    <!-- documentary movies -->
    <div class="row mt-5">
        <h5 class="text-white">Filmes de Documentário</h5>
        <div class="rowposter mt-3 p-2">
            <ng-container *ngFor="let d of documentaryMovieResult">
                
            </ng-container>
        </div>
    </div>

```

```

    </div>
  </div>

  <!-- science-fiction movies -->
  <div class="row mt-5">
    <h5 class="text-white">Filmes de Ficção-Científica</h5>
    <div class="rowposter mt-3 p-2">
      <ng-container *ngFor="let f of scienceFictionMovieResult">
        
      </ng-container>
    </div>
  </div>

  <!-- thriller movies -->
  <div class="row mt-5">
    <h5 class="text-white">Filmes de Suspense</h5>
    <div class="rowposter mt-3 p-2">
      <ng-container *ngFor="let t of thrillerMovieResult">
        
      </ng-container>
    </div>
  </div>

</div>

```

Salve tudo e execute seu projeto.

