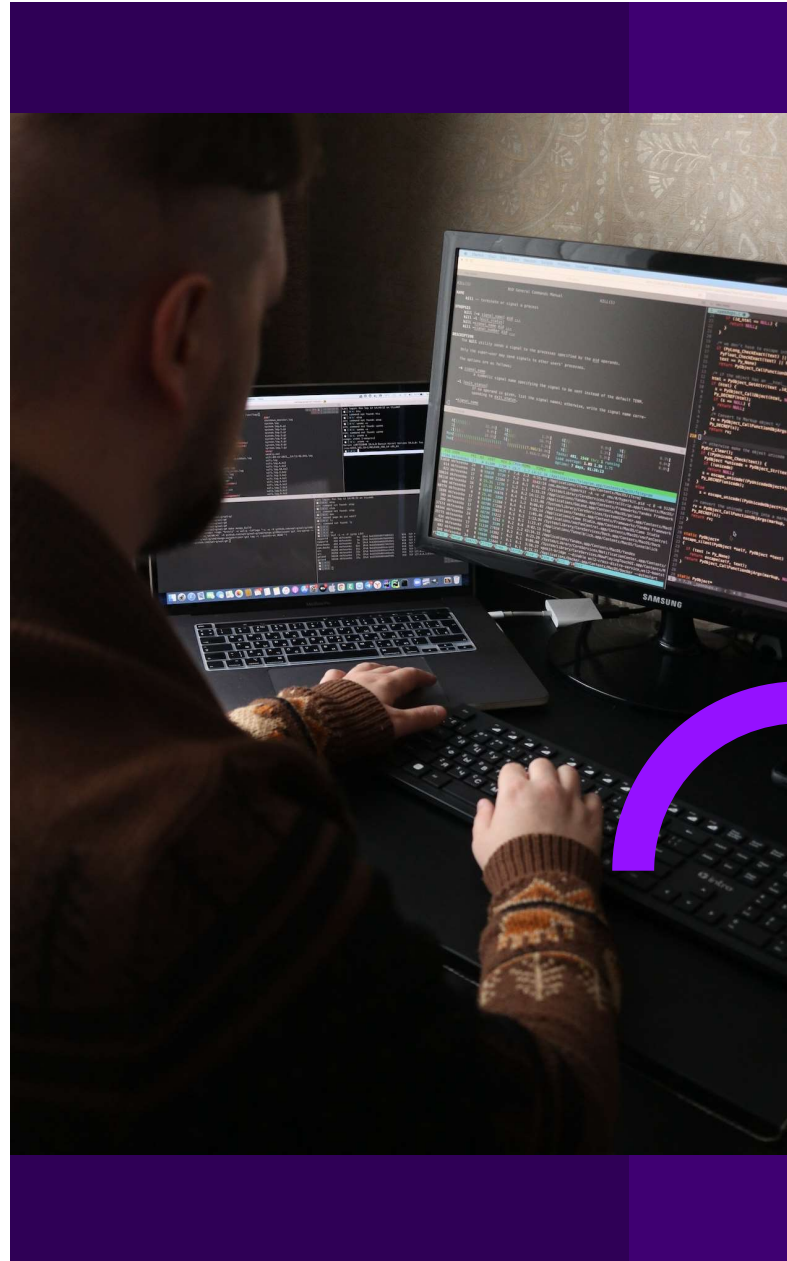
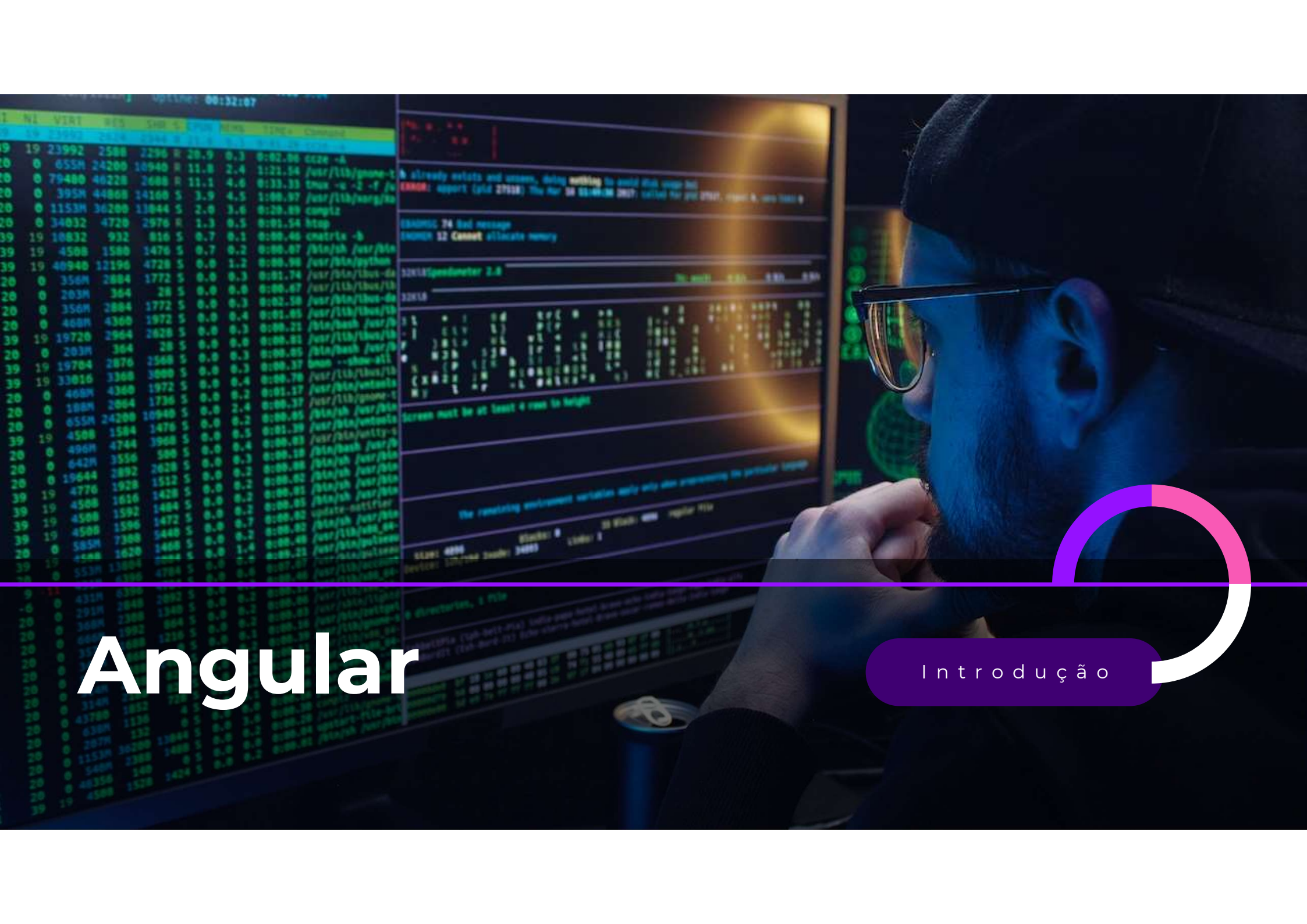


Apresentação Pessoal

Leone Costa Rocha

1. **Idade:** 35 anos
2. **Experiência:** 10 anos na área de desenvolvimento de sistemas
3. **Certificações:**
 - AZ-900: Fundamentos do Microsoft Azure
 - 70-515: Web Applications Development with Microsoft .NET Framework .
 - 498-361: Software Development Fundamentals
4. **Basic technical skills :** Back-End (C#, ASP.NET), Front-End (JavaScript, CSS, Type Script)
5. **GitHub:** <https://github.com/LeoneRocha>
6. **LinkedIn:** <https://www.linkedin.com/in/leone-costa-rocha-14049722/>
7. **E-mails :**
 - leone.rocha@cognizant.com
 - leocr_lem@yahoo.com.br





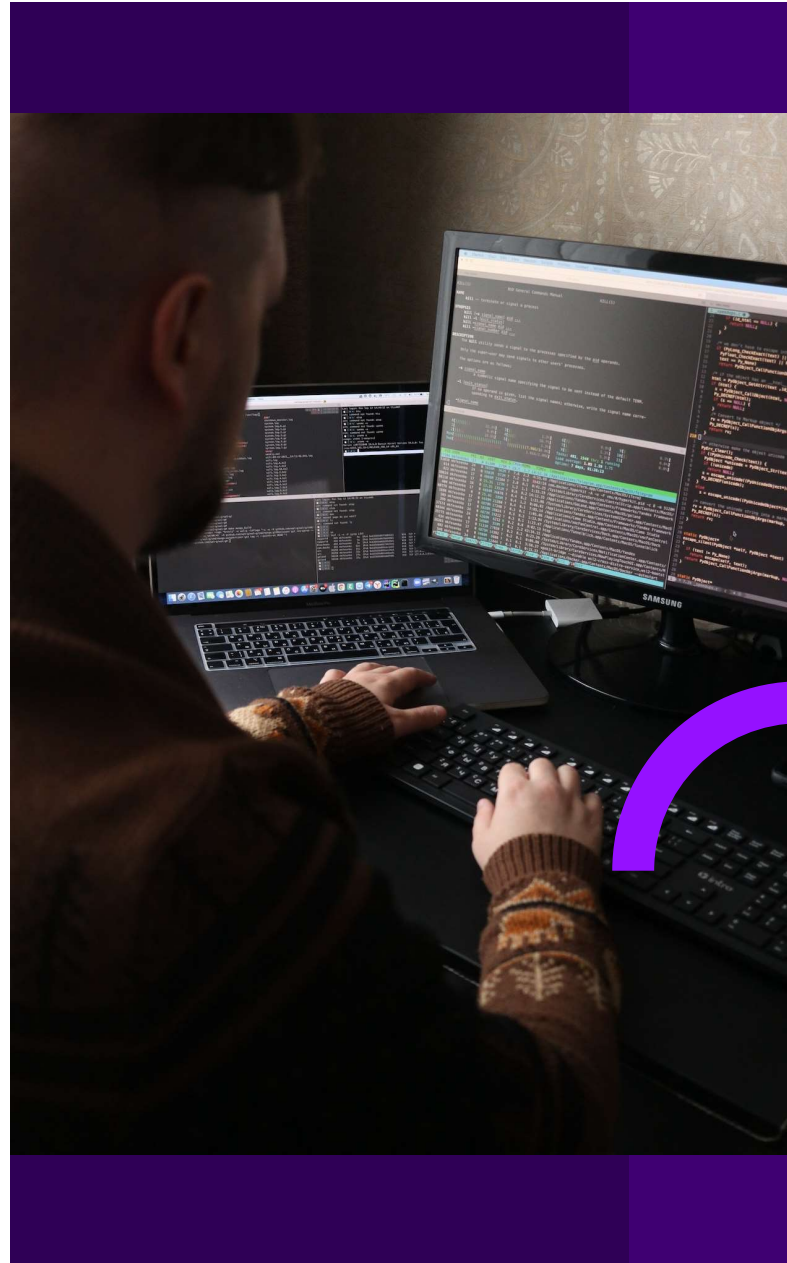
Angular

Introdução

Angular

Conteúdo

1. Introdução
2. Componentes
3. Templates
4. Tipos de bind
5. Diretivas
6. Serviços
7. Módulos



Introdução



Angular é um framework de desenvolvimento web em JavaScript criado pela Google.

TypeScript é tipo uma versão melhorada do JavaScript, com superpoderes de detecção de erros e organização. É tipo um JavaScript turbinado!. Com recursos de **tipagem** estática e a **orientação a objetos**.

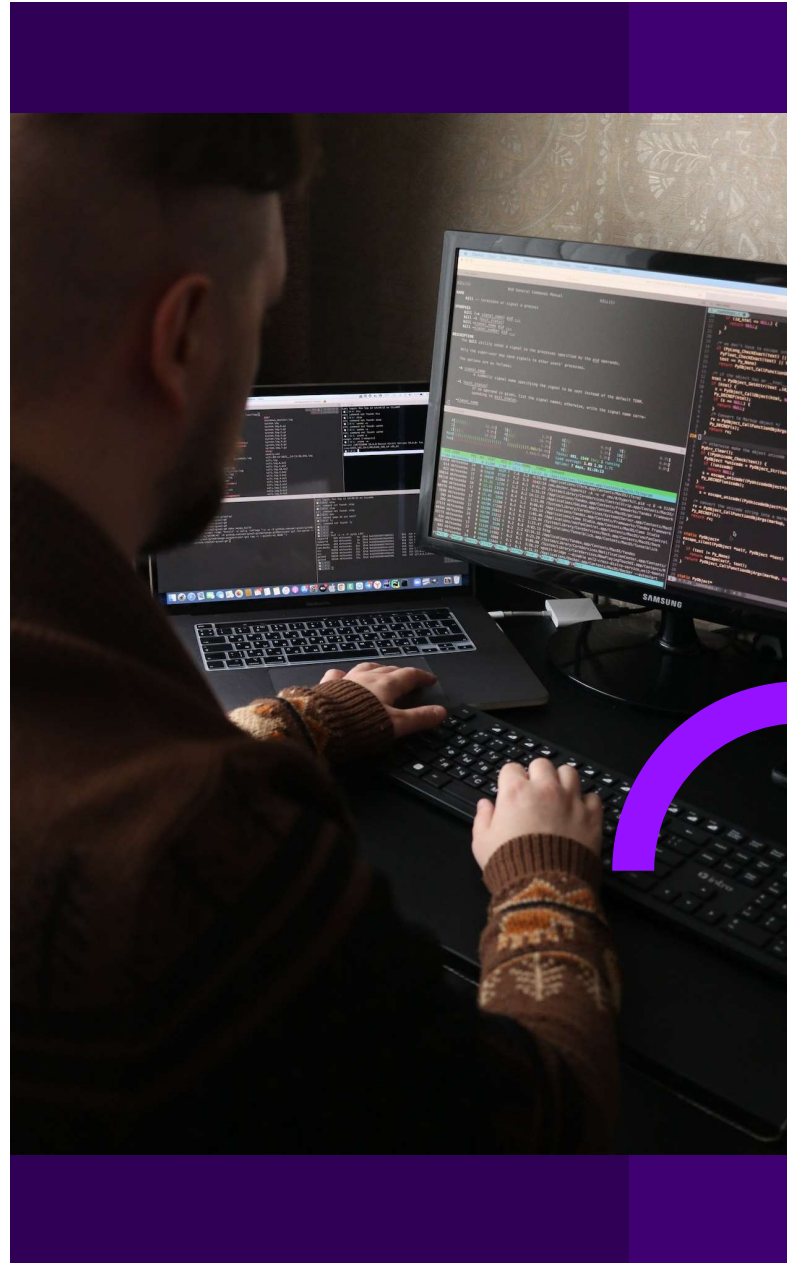
Diferenças

Angular JS

1. Surgiu para ser simples
2. Performace ruim
3. Api Cresceu incosistentemente
4. Conceitos confuses e repetidos
5. ES5

Angular (2+)

1. Mais aderente a padrões
2. Padrão para criar qualquer coisa (pipes, componentes, services
3. Olhando a era de componentes
4. ES6/ES2015
5. Em typescript o codigo é transpilado para JavaScript



Como identificar AngularJS (versão 1.x) ou Angular (versão a partir do 2+)

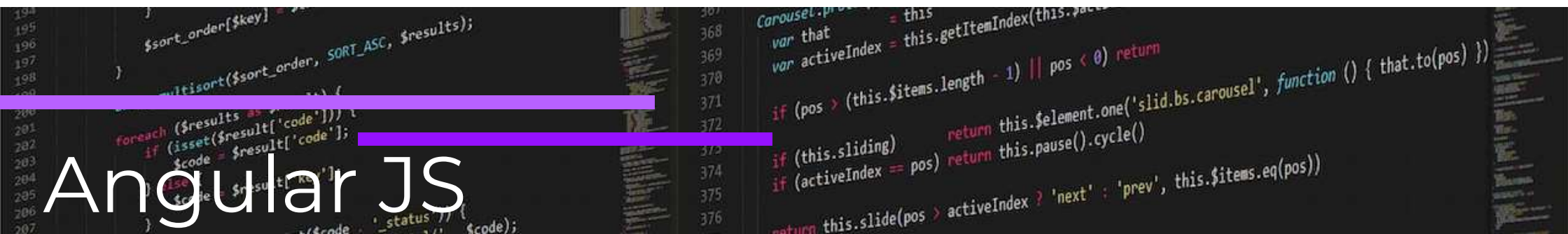
- 1. Estrutura de diretórios:** O AngularJS geralmente tem uma estrutura de diretórios diferente do Angular2+. No AngularJS, os arquivos JavaScript e HTML são agrupados em uma estrutura baseada em funcionalidade, enquanto no Angular, os arquivos TypeScript e HTML são organizados em componentes e módulos.
- 2. Sintaxe de código:** O AngularJS usa uma sintaxe diferente em comparação com o Angular. O AngularJS usa diretivas com prefixo "ng-" (por exemplo, ng-controller, ng-model), enquanto o Angular usa uma abordagem baseada em componentes com diretivas personalizadas (por exemplo, *ngIf, [ngModel]).
- 3. Módulos:** No Angular, o conceito de módulos é fundamental. Se você encontrar uma definição de módulo no código, é provável que esteja trabalhando com Angular. No AngularJS, não existe um conceito de módulos tão distintos.



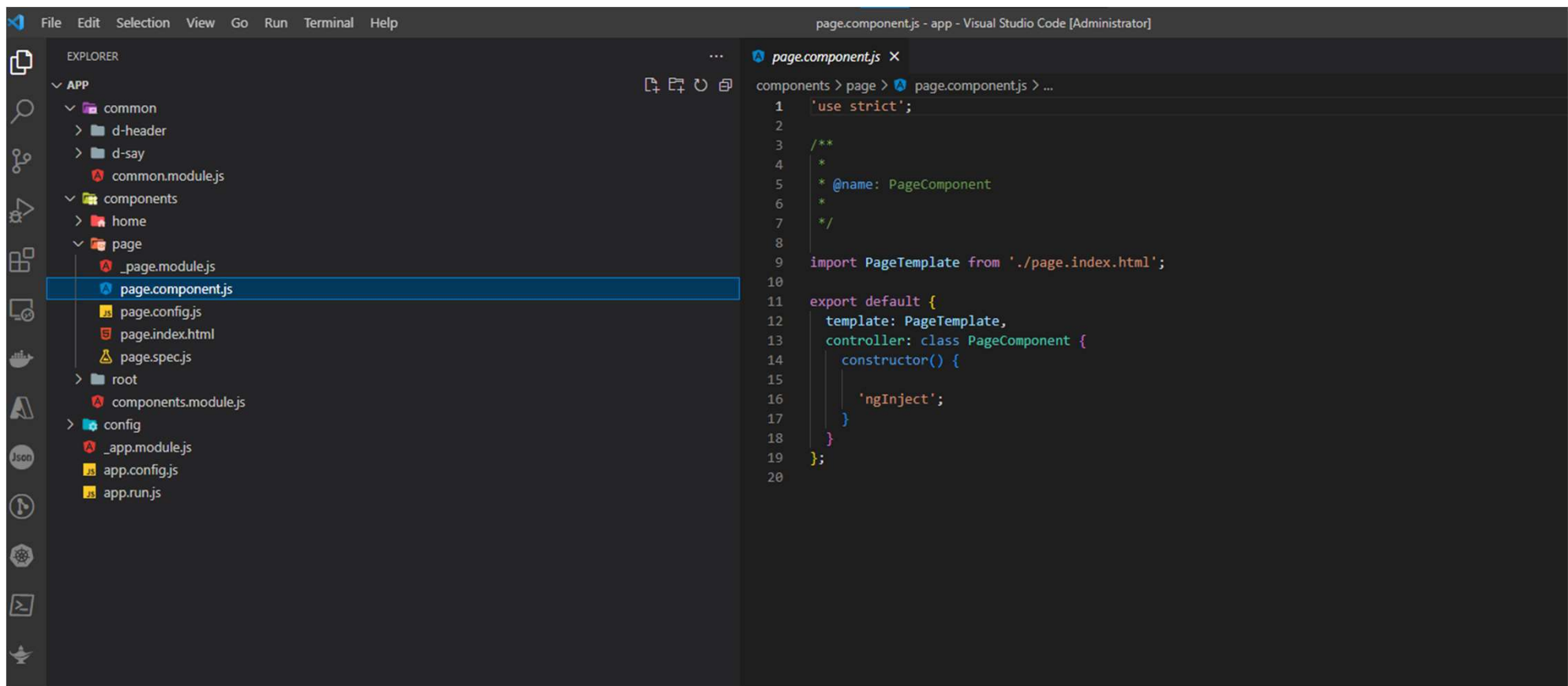
Como identificar AngularJS (versão 1.x) ou Angular (versão a partir do 2+)

- 4. Angular CLI:** O Angular possui uma ferramenta de linha de comando chamada Angular CLI, que facilita a criação e o gerenciamento de projetos Angular. Se você estiver usando o Angular CLI para criar ou executar seu projeto, é provável que esteja trabalhando com o Angular.
- 5. Versões do pacote:** Verifique as versões dos pacotes instalados em seu projeto. O AngularJS tem versões como 1.x (por exemplo, 1.7.9), enquanto o Angular possui versões a partir do 2.x (por exemplo, 12.2.5).
6. Portanto, verifique a estrutura do projeto, a sintaxe de código, a presença de módulos, o uso do Angular CLI e as versões dos pacotes para determinar se você está trabalhando com AngularJS ou Angular TypeScript.
7. Quando você realiza o build de um projeto Angular em TypeScript, o Angular utiliza o Angular CLI para realizar uma série de etapas para transformar seu código TypeScript em um aplicativo Angular pronto para ser executado em um navegador.



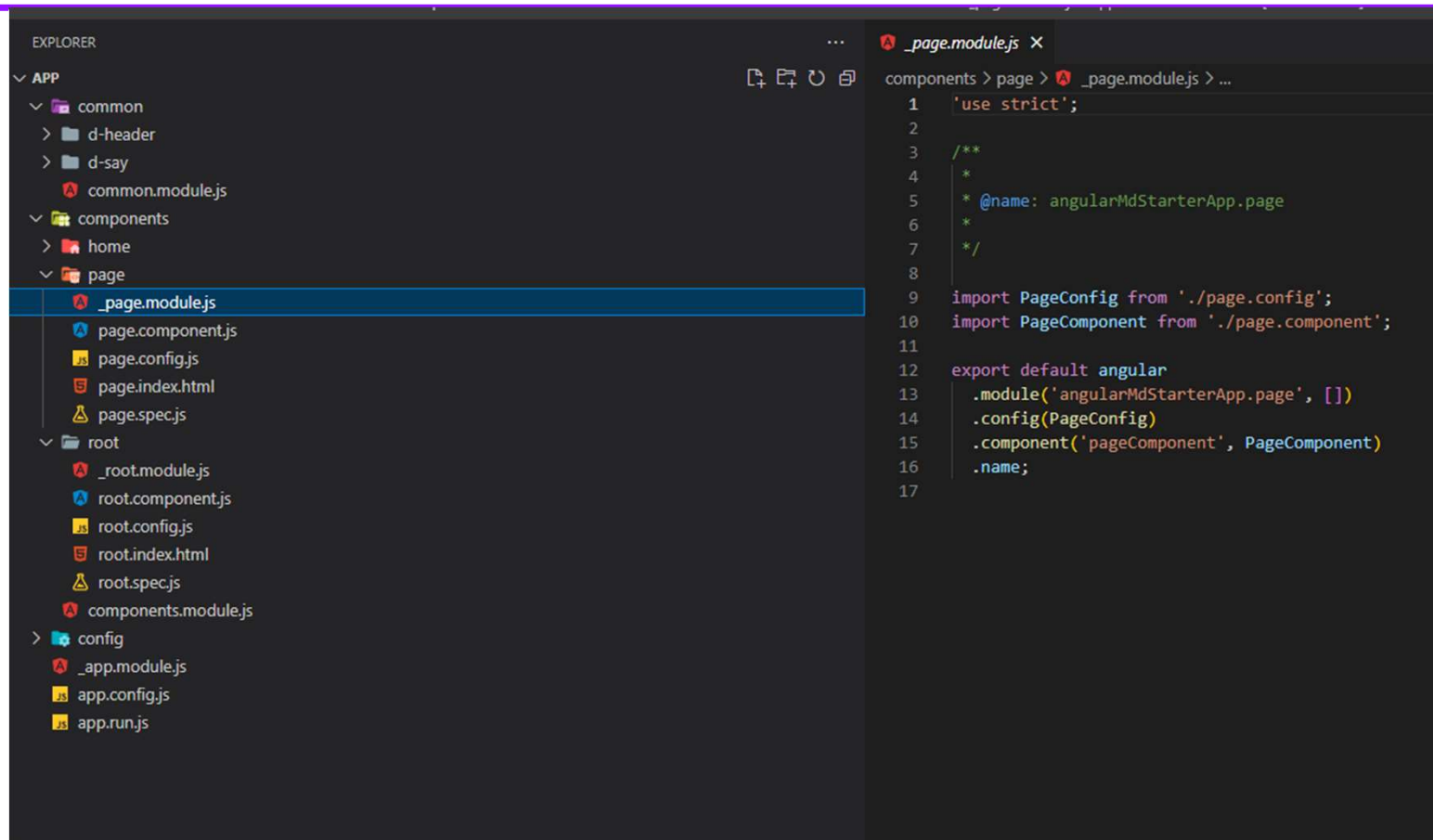


Angular JS





Angular JS





Angular JS

```
8
9 function RootConfig($stateProvider, $urlRouterProvider) {
10
11     'ngInject';
12
13     $stateProvider
14     .state('root', {
15         abstract: true,
16         url: '/',
17         template: '<root-component></root-component>',
18     });
19
20     $urlRouterProvider.otherwise('/');
21 }
22
23 export default RootConfig;
```

```
components.module.js X
components > components.module.js > ...
1 'use strict';
2
3 /**
4  *
5  * @name: angularMdStarterApp.components
6  *
7  */
8
9 import RootModule from './root/_root.module';
10 import HomeModule from './home/_home.module';
11 import PageModule from './page/_page.module';
12
13 export default angular.module('angularMdStarterApp.components', [
14     RootModule,
15     HomeModule,
16     PageModule,
17 ])
18 .name;
```

Tipos de dados

- Variáveis com tipagem de dados
- Erros de compilação
- Dados dinâmicos

```
const message: string = "Criando nossa primeira string tipada"  
console.log(message)
```

```
let idade: number = 4  
// idade = '1' // gera erro ao compilar
```

```
console.log("idade " + idade)  
idade = idade + 1  
console.log("idade " + idade)
```

```
// Não é obrigatório tipar um dado  
// Funcionando como JavaScript puro  
let dadoDinamico  
dadoDinamico = "meu nome"  
dadoDinamico = 1
```

Entendendo a estrutura do app

O projeto gerado segue exatamente uma estrutura de aplicação NodeJS. Existe um **package.json** que contém todas as dependências que ele instalou.

Vamos analisar as pastas e principais arquivos que foram criadas com o comando que executamos

Entendendo a estrutura do app

- **.angular-cli.json:** arquivo que define como o Angular executará com todas as suas configurações
- **webpack:** module bundler para compilar todos os assets da nossa aplicação. Como você pode ter visto ao executar o comando ``ng serve`` foi exibido todos os bundles que ele gerou para rodar a aplicação
- **styles.css:** CSS globais da aplicação
- **assets:** não sofrerá nenhum impacto durante a compilação

Entendendo a estrutura do app

- **polyfills.ts:** para habilitar as features para browser mais antigos que precise dar suporte
- **main.ts:** módulo de definição principal
- **index.html:** html que será renderizado



Instalação e configuração do ambiente de desenvolvimento

Para começar a desenvolver com Angular e TypeScript, é necessário instalar o Node.js e o Angular CLI. Você pode fazer isso digitando

```
"npm install -g @angular/cli"
```

Na linha de comando. Em seguida, siga as instruções da documentação oficial do Angular para instalar e configurar o ambiente de desenvolvimento em seu sistema operacional.



Criação de um projeto em Angular com TypeScript

Um projeto em Angular possui uma estrutura específica, com arquivos e pastas como "src" (código fonte), "node_modules" (dependências) e "angular.json" (configurações). Para criar um novo projeto em Angular com TypeScript, basta digitar na linha de comando.

ng new meu-projeto

npm run build

npm start

Componentes



Componentes são elementos essenciais do Angular usados para criar **partes** da **interface** do usuário, como **botões**, **caixas** de texto ou **tabelas** de dados. Essas estruturas são blocos reutilizáveis que combinam código, estilo e template para criar partes da interface do usuário de forma modular e organizada. Encapsulam funcionalidades específicas e podem ser combinados para construir aplicações complexas.

O componente

- **html:** template HTML que será renderizado
- **css:** estilos CSS para este componente
- **ts:** componente em si que foi criado

Exemplo de um componente

```
import { Component } from "@angular/core"

@Component ({ // -> decorator
  selector: 'app-first',
  templateUrl: './my-first-component.html'
  // ou
  // template: '<h1> Test </h1>'
})

export class MyFirstComponent {
  constructor ( Image
}
```

Utilizando um componente

Quando o componente estiver pronto é necessário informar ao módulo da sua aplicação que esse componente existe.

Assim você informa ao Angular a qual módulo um determinado componente pertence.

```
// app.module.ts
```

```
@NgModule({  
  declarations: [MyFirstComponent]  
})
```

```
export class AppModule {}
```

Image

Templates



O template no Angular é onde você cria a aparência visual de um componente, usando uma mistura de **HTML**. Ele pode ser definido diretamente no arquivo do componente ou em um arquivo HTML externo. É a parte de interface visual de componentes.

HTML do componente

```
<!-- header.component.html -->  
<header>  
  <h1>Primeiro App Angular</h1>  
</header>
```

Image

```
194 }
195 $sort_order[$key] = $results[$key];
196 }
197 multisort($sort_order, SORT_ASC, $results);
198 }
199 }
200 }
201 foreach ($results as $result) {
202     if (isset($result['code'])) {
203         $code = $result['code'];
204     } else {
205         $code = $result['key'];
206     }
207     $code = ($code == '_status') ? $code : ($code . ' ' . $code);
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
```

Criando um componente

ng generate component post-list

```
import { Component, OnInit } from '@angular/core';
import { PostService } from '../post.service';

@Component({
  selector: 'app-post-list',
  templateUrl: './post-list.component.html',
  styleUrls: ['./post-list.component.css']
})
export class PostListComponent implements OnInit {
  posts: any[] = [];

  constructor(private postService: PostService) { }

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

  getPosts(): void {
    this.postService.getPosts()
      .subscribe(posts => {
        this.posts = posts;
      });
  }

  deletePost(id: number): void {
    this.postService.deletePost(id)
      .subscribe(() => {
        this.posts = this.posts.filter(post => post.id !== id);
      });
  }
}
```


Tipos de BIND



No Angular, existem quatro tipos principais de bindings (ligações) que permitem a comunicação e atualização de dados entre o componente e o template:


Property Binding

É quando você deseja conectar um valor de uma propriedade de um elemento a uma expressão Angular, realizando assim essa associação.

A marcação no HTML que determina que uma propriedade estará conectada ao Angular estará usando a sintaxe `[]` e pode ser aplicada a qualquer propriedade de um elemento HTML.

Uma alteração que é feita no componente e será renderizado pelo template. **PORÉM** apenas nesse sentido

```
// no componente  
user = { name: 'Thiago Dorneles' }  
  
// no template html  
<input type="text" [value]="user.name" />
```



Retângulo

Exemplos Property Binding

```
<!-- escondendo o botao caso usuario não esteja logado -->  
<button [hidden]="!user.isLogged" value="Logout" />  
  
<!-- adiciona classe disabled caso usuario esteja desabilitado -->  
<input type="text" [class.disabled]="user.disabled" [value]="user.name" />  
  
↓  
  
<!-- conteudo renderizado pelo codigo acima -->  
<input type="text" class="disabled" value="Thiago Dorneles" />
```


One-Way Binding

Uma alteração que é feita no componente e será renderizado pelo template.

PORÉM apenas nesse sentido. Neste exemplo o valor no componente não será atualizado no componente caso o valor seja alterado dentro do HTML

```
// no componente
user = { name: 'Thiago Dorneles' }

// no template html
<input type="text" [value]="user.name" />
```



Two-Way Binding

Uma alteração que é feita no componente e será renderizado pelo template.

PORÉM apenas nesse sentido. Neste exemplo o valor no componente não será atualizado no componente caso o valor seja alterado dentro do HTML

Usando `[(ngModel)]`, você pode criar uma ligação bidirecional entre uma propriedade do componente e um elemento HTML. Por exemplo, `[(ngModel)]="nome"` mantém o valor da propriedade "nome" do componente e atualiza automaticamente o campo de entrada do elemento HTML.

```
<!-- habilitando o two-way binding -->  
<input  
  type="text"  
  [(value)]="user.name" />
```


Decorator Component

- Dentro do próprio decorator

Component definimos uma propriedade chamada **inputs** com a relação de itens que serão recebidos

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

@Component({
  selector: 'ttt-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css'],
  inputs: [ 'title' ]
})
export class HeaderComponent {
  title: string

  constructor() { }
}
```

```
<!-- header.component.html -->
<header>
  <h1>{{title}}</h1>
</header>

<!-- app.component.html -->
<ttt-header title="Primeiro App"></ttt-header>
```

Decorator Input

- Precisamos importar o decorator **Input**
- Criamos um atributo dentro do componente e o decoramos com o **@Input**
- Por padrão, o decorator Input coloca o mesmo nome do atributo

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

@Component({
  selector: 'ttt-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent {
  @Input() title: string

  constructor() { }
}

export class HeaderComponent {
  @Input('value') title: string

  constructor() { }
}
```

Diretivas



As diretivas no Angular são recursos que permitem estender o HTML para adicionar comportamentos personalizados. Elas **podem alterar a aparência, comportamento ou estrutura** dos elementos HTML. Existem dois tipos: **diretivas de atributo** (alteram elementos existentes) e **diretivas estruturais** (alteram a estrutura do DOM). As diretivas são fundamentais para criar aplicativos dinâmicos e reutilizáveis.

Directives

- **ngIf:** exibir um conteúdo apenas quando necessário
- **ngFor:** utilizada para repetição de uma lista/array de informações
- **ngSwitch:** mesmo comportamento de um switch/case de todas linguagens conhecidas
- **ngClass:** adiciona uma classe de estilo no elemento

Exemplo de um ngIf

```
<div *ngIf="exibirConteudo">  
  conteudo a ser exibido  
</div>
```


Exemplo de um ngFor

```
<ul>  
  <li *ngFor="let menu of menus;">{{menu}}</li>  
</ul>
```

Serviços



Services no Angular são classes que fornecem **funcionalidades compartilhadas** e **lógica de negócio reutilizável**. Eles ajudam a manter o código **organizado**, facilitam a **reutilização** e promovem a separação de preocupações. Os Services são injetados nos componentes e podem ser usados para realizar tarefas como chamadas a **APIs**, **manipulação de dados** e **gerenciamento de estado**.

Exemplo de um serviço

```
import { Injectable } from '@angular/core'
import { Http } from '@angular/http'

@Injectable()
export class MyHttpService {
  constructor (private http: Http) {}

  get () {
    return this.http.get('/')
  }
}
```

ng generate service post

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

@Injectable({
  providedIn: 'root'
})
export class PostService {
  private apiUrl = 'https://jsonplaceholder.typicode.com/posts';

  constructor(private http: HttpClient) { }

  getPosts() {
    return this.http.get<any[]>(this.apiUrl);
  }

  getPost(id: number) {
    return this.http.get<any>(`${this.apiUrl}/${id}`);
  }

  addPost(post: any) {
    return this.http.post<any>(this.apiUrl, post);
  }

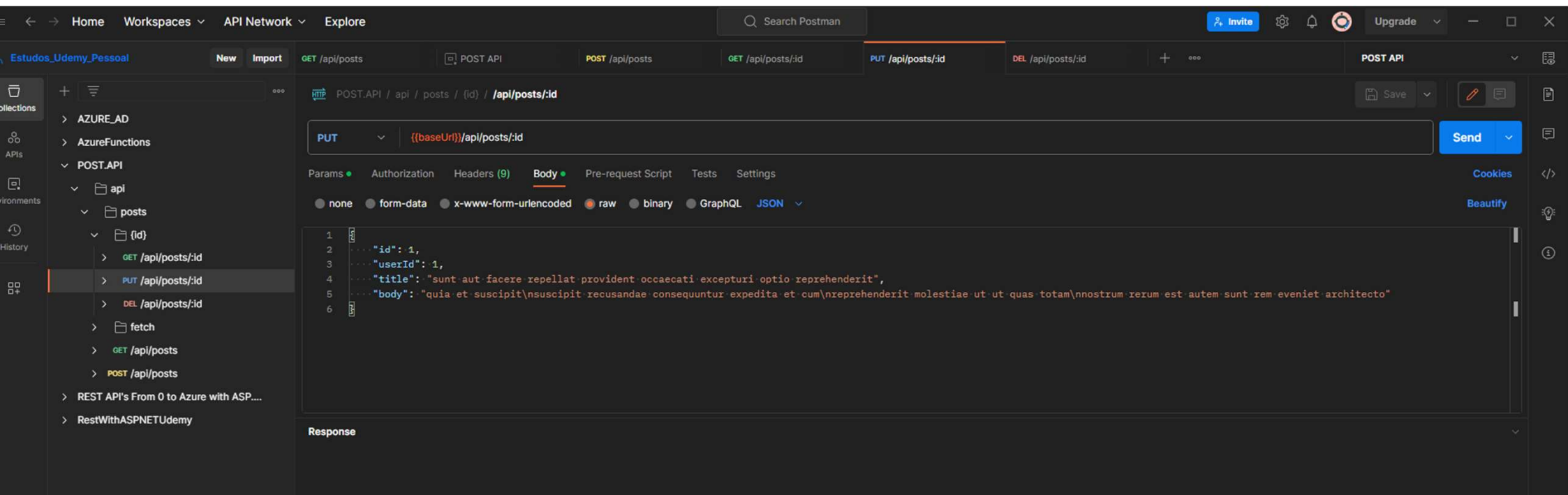
  updatePost(post: any) {
    return this.http.put<any>(`${this.apiUrl}/${post.id}`, post);
  }

  deletePost(id: number) {
    return this.http.delete<any>(`${this.apiUrl}/${id}`);
  }
}
```



```

194 }
195 $sort_order[$key] = $results[$key];
196 }
197 multisort($sort_order, SORT_ASC, $results);
198 }
199
200 foreach ($results as $result) {
201     if (isset($result['code'])) {
202         $code = $result['code'];
203     } else {
204         $code = $result['key'];
205     }
206     $status = $result['_status'];
207     $code = $code . $status;
208 }
209
210 $results = array();
211 foreach ($results as $result) {
212     if (isset($result['code'])) {
213         $code = $result['code'];
214     } else {
215         $code = $result['key'];
216     }
217     $status = $result['_status'];
218     $code = $code . $status;
219 }
220
221 $results = array();
222 foreach ($results as $result) {
223     if (isset($result['code'])) {
224         $code = $result['code'];
225     } else {
226         $code = $result['key'];
227     }
228     $status = $result['_status'];
229     $code = $code . $status;
230 }
231
232 $results = array();
233 foreach ($results as $result) {
234     if (isset($result['code'])) {
235         $code = $result['code'];
236     } else {
237         $code = $result['key'];
238     }
239     $status = $result['_status'];
240     $code = $code . $status;
241 }
242
243 $results = array();
244 foreach ($results as $result) {
245     if (isset($result['code'])) {
246         $code = $result['code'];
247     } else {
248         $code = $result['key'];
249     }
250     $status = $result['_status'];
251     $code = $code . $status;
252 }
253
254 $results = array();
255 foreach ($results as $result) {
256     if (isset($result['code'])) {
257         $code = $result['code'];
258     } else {
259         $code = $result['key'];
260     }
261     $status = $result['_status'];
262     $code = $code . $status;
263 }
264
265 $results = array();
266 foreach ($results as $result) {
267     if (isset($result['code'])) {
268         $code = $result['code'];
269     } else {
270         $code = $result['key'];
271     }
272     $status = $result['_status'];
273     $code = $code . $status;
274 }
275
276 $results = array();
277 foreach ($results as $result) {
278     if (isset($result['code'])) {
279         $code = $result['code'];
280     } else {
281         $code = $result['key'];
282     }
283     $status = $result['_status'];
284     $code = $code . $status;
285 }
286
287 $results = array();
288 foreach ($results as $result) {
289     if (isset($result['code'])) {
290         $code = $result['code'];
291     } else {
292         $code = $result['key'];
293     }
294     $status = $result['_status'];
295     $code = $code . $status;
296 }
297
298 $results = array();
299 foreach ($results as $result) {
300     if (isset($result['code'])) {
301         $code = $result['code'];
302     } else {
303         $code = $result['key'];
304     }
305     $status = $result['_status'];
306     $code = $code . $status;
307 }
308
309 $results = array();
310 foreach ($results as $result) {
311     if (isset($result['code'])) {
312         $code = $result['code'];
313     } else {
314         $code = $result['key'];
315     }
316     $status = $result['_status'];
317     $code = $code . $status;
318 }
319
320 $results = array();
321 foreach ($results as $result) {
322     if (isset($result['code'])) {
323         $code = $result['code'];
324     } else {
325         $code = $result['key'];
326     }
327     $status = $result['_status'];
328     $code = $code . $status;
329 }
330
331 $results = array();
332 foreach ($results as $result) {
333     if (isset($result['code'])) {
334         $code = $result['code'];
335     } else {
336         $code = $result['key'];
337     }
338     $status = $result['_status'];
339     $code = $code . $status;
340 }
341
342 $results = array();
343 foreach ($results as $result) {
344     if (isset($result['code'])) {
345         $code = $result['code'];
346     } else {
347         $code = $result['key'];
348     }
349     $status = $result['_status'];
350     $code = $code . $status;
351 }
352
353 $results = array();
354 foreach ($results as $result) {
355     if (isset($result['code'])) {
356         $code = $result['code'];
357     } else {
358         $code = $result['key'];
359     }
360     $status = $result['_status'];
361     $code = $code . $status;
362 }
363
364 $results = array();
365 foreach ($results as $result) {
366     if (isset($result['code'])) {
367         $code = $result['code'];
368     } else {
369         $code = $result['key'];
370     }
371     $status = $result['_status'];
372     $code = $code . $status;
373 }
374
375 $results = array();
376 foreach ($results as $result) {
377     if (isset($result['code'])) {
378         $code = $result['code'];
379     } else {
380         $code = $result['key'];
381     }
382     $status = $result['_status'];
383     $code = $code . $status;
384 }
385
386 $results = array();
387 foreach ($results as $result) {
388     if (isset($result['code'])) {
389         $code = $result['code'];
390     } else {
391         $code = $result['key'];
392     }
393     $status = $result['_status'];
394     $code = $code . $status;
395 }
396
397 $results = array();
398 foreach ($results as $result) {
399     if (isset($result['code'])) {
400         $code = $result['code'];
401     } else {
402         $code = $result['key'];
403     }
404     $status = $result['_status'];
405     $code = $code . $status;
406 }
407
408 $results = array();
409 foreach ($results as $result) {
410     if (isset($result['code'])) {
411         $code = $result['code'];
412     } else {
413         $code = $result['key'];
414     }
415     $status = $result['_status'];
416     $code = $code . $status;
417 }
418
419 $results = array();
420 foreach ($results as $result) {
421     if (isset($result['code'])) {
422         $code = $result['code'];
423     } else {
424         $code = $result['key'];
425     }
426     $status = $result['_status'];
427     $code = $code . $status;
428 }
429
430 $results = array();
431 foreach ($results as $result) {
432     if (isset($result['code'])) {
433         $code = $result['code'];
434     } else {
435         $code = $result['key'];
436     }
437     $status = $result['_status'];
438     $code = $code . $status;
439 }
440
441 $results = array();
442 foreach ($results as $result) {
443     if (isset($result['code'])) {
444         $code = $result['code'];
445     } else {
446         $code = $result['key'];
447     }
448     $status = $result['_status'];
449     $code = $code . $status;
450 }
451
452 $results = array();
453 foreach ($results as $result) {
454     if (isset($result['code'])) {
455         $code = $result['code'];
456     } else {
457         $code = $result['key'];
458     }
459     $status = $result['_status'];
460     $code = $code . $status;
461 }
462
463 $results = array();
464 foreach ($results as $result) {
465     if (isset($result['code'])) {
466         $code = $result['code'];
467     } else {
468         $code = $result['key'];
469     }
470     $status = $result['_status'];
471     $code = $code . $status;
472 }
473
474 $results = array();
475 foreach ($results as $result) {
476     if (isset($result['code'])) {
477         $code = $result['code'];
478     } else {
479         $code = $result['key'];
480     }
481     $status = $result['_status'];
482     $code = $code . $status;
483 }
484
485 $results = array();
486 foreach ($results as $result) {
487     if (isset($result['code'])) {
488         $code = $result['code'];
489     } else {
490         $code = $result['key'];
491     }
492     $status = $result['_status'];
493     $code = $code . $status;
494 }
495
496 $results = array();
497 foreach ($results as $result) {
498     if (isset($result['code'])) {
499         $code = $result['code'];
500     } else {
501         $code = $result['key'];
502     }
503     $status = $result['_status'];
504     $code = $code . $status;
505 }
506
507 $results = array();
508 foreach ($results as $result) {
509     if (isset($result['code'])) {
510         $code = $result['code'];
511     } else {
512         $code = $result['key'];
513     }
514     $status = $result['_status'];
515     $code = $code . $status;
516 }
517
518 $results = array();
519 foreach ($results as $result) {
520     if (isset($result['code'])) {
521         $code = $result['code'];
522     } else {
523         $code = $result['key'];
524     }
525     $status = $result['_status'];
526     $code = $code . $status;
527 }
528
529 $results = array();
530 foreach ($results as $result) {
531     if (isset($result['code'])) {
532         $code = $result['code'];
533     } else {
534         $code = $result['key'];
535     }
536     $status = $result['_status'];
537     $code = $code . $status;
538 }
539
540 $results = array();
541 foreach ($results as $result) {
542     if (isset($result['code'])) {
543         $code = $result['code'];
544     } else {
545         $code = $result['key'];
546     }
547     $status = $result['_status'];
548     $code = $code . $status;
549 }
550
551 $results = array();
552 foreach ($results as $result) {
553     if (isset($result['code'])) {
554         $code = $result['code'];
555     } else {
556         $code = $result['key'];
557     }
558     $status = $result['_status'];
559     $code = $code . $status;
560 }
561
562 $results = array();
563 foreach ($results as $result) {
564     if (isset($result['code'])) {
565         $code = $result['code'];
566     } else {
567         $code = $result['key'];
568     }
569     $status = $result['_status'];
570     $code = $code . $status;
571 }
572
573 $results = array();
574 foreach ($results as $result) {
575     if (isset($result['code'])) {
576         $code = $result['code'];
577     } else {
578         $code = $result['key'];
579     }
580     $status = $result['_status'];
581     $code = $code . $status;
582 }
583
584 $results = array();
585 foreach ($results as $result) {
586     if (isset($result['code'])) {
587         $code = $result['code'];
588     } else {
589         $code = $result['key'];
590     }
591     $status = $result['_status'];
592     $code = $code . $status;
593 }
594
595 $results = array();
596 foreach ($results as $result) {
597     if (isset($result['code'])) {
598         $code = $result['code'];
599     } else {
```



Módulos



Um módulo no Angular é uma estrutura que **agrupa** componentes, serviços e outros recursos relacionados, fornecendo um contexto para o desenvolvimento de uma funcionalidade específica em uma aplicação.

Interface Gráfica

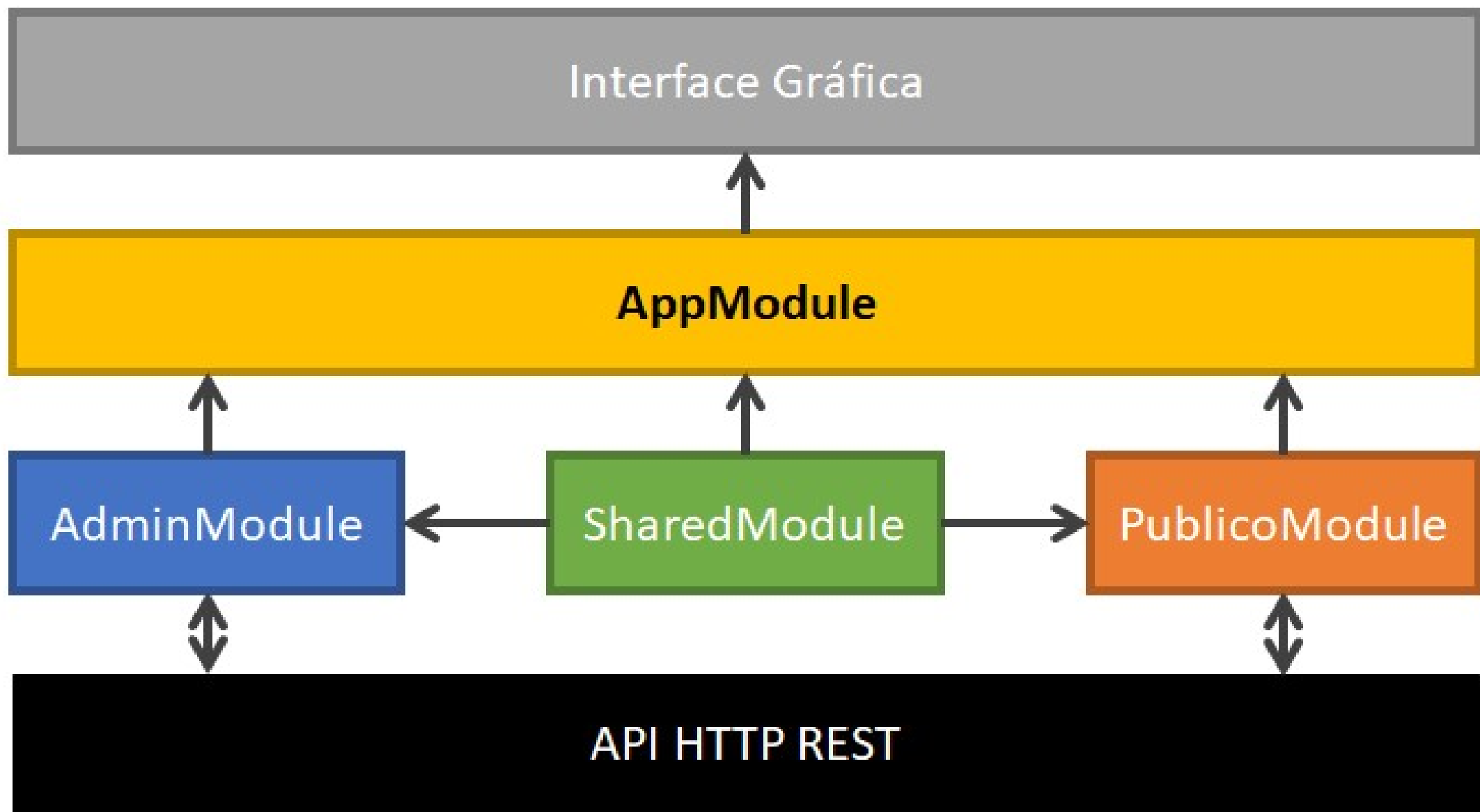
AppModule

AdminModule

SharedModule

PublicoModule

API HTTP REST




```
@NgModule ({
  declarations: [
    FooterModule,
    HeaderModule,
    ListModule,
  ],
  imports: [
    CommonModule,
    RouterModule,
    FormsModule,
    ReactiveFormsModule,
    NgbModule,
  ],
  exports: [
    FooterModule,
    HeaderModule,
    ListModule,
  ],
})
export class SharedModule {}
```

```

367     Carousel.prototype = this
368     var that = this
369     var activeIndex = this.getItemIndex(this.$active)
370     if (pos > (this.$items.length - 1) || pos < 0) return
371     if (this.sliding) return this.$element.one('slid.bs.carousel', function () { that.to(pos) })
372     if (activeIndex == pos) return this.pause().cycle()
373     return this.slide(pos > activeIndex ? 'next' : 'prev', this.$items.eq(pos))

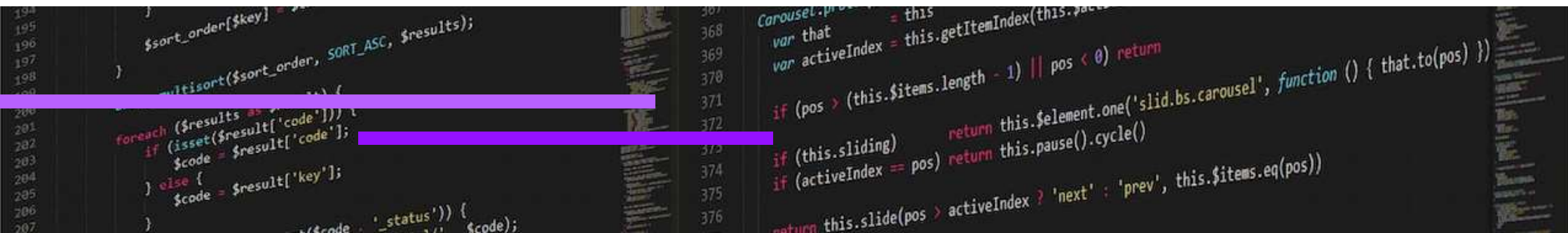
```

ng generate module nome-do-modulo

 Copy code


```
import { HttpClientModule } from '@angular/common/http';
```

```
@NgModule({  
  declarations: [  
    // Componentes  
  ],  
  imports: [  
    // Outros módulos  
    HttpClientModule  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```



Passo a passo

Ter os seguintes programas

1. VISUAL STUDIO CODE
2. Baixar o NODE JS mais recente
3. Ter os pluguins do visual studio code (OPCIONAL)
 - Angular Language Service
 - JavaScript and TypeScript Nightly
 - ESLint

Código Fonte

1. Baixar do GIT o código
2. Baixar o NODE JS
3. Rodar o comando no terminal do visual code
npm install para instalar as dependencias do projetos
os pacote
4. Para executar o projeto rodar o comando **ng serve**
geralmente abre na porta padrão
http://localhost:4200

Links Uteis

Material

1. GitHub:

<https://github.com/LeoneRocha/WORKSHOPCOGNIZANTANGULAR>

2. Documentação Angular: <https://angular.io/>

<https://angular.io/docs>

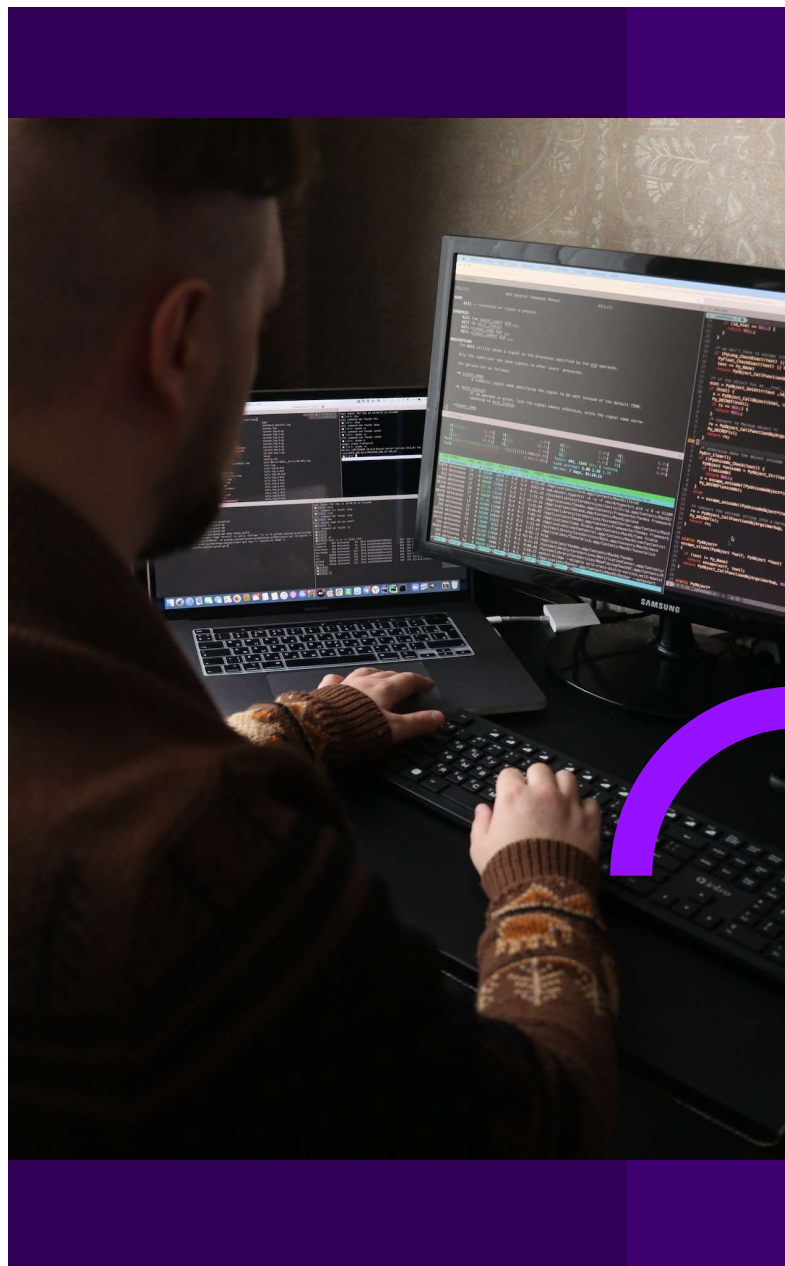
3. NODE JS: <https://nodejs.org/en/download>

4. Visual Studio Code: <https://code.visualstudio.com/>

5. API MOCK: <https://jsonplaceholder.typicode.com/posts>

6. Curso de Angular Udemey Cognizant:

<https://cognizant.udemy.com/course/the-complete-angular-master-class/>





Developer

Thank You