

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Estrutura de pastas para Angular: escalável, limpa e fácil

Estruture seu projeto Angular desta forma e deixe eu projeto escalável, limpo e de fácil entendimento para qualquer desenvolvedor



Belmiro · [Follow](#)

4 min read · May 3, 2021



Share

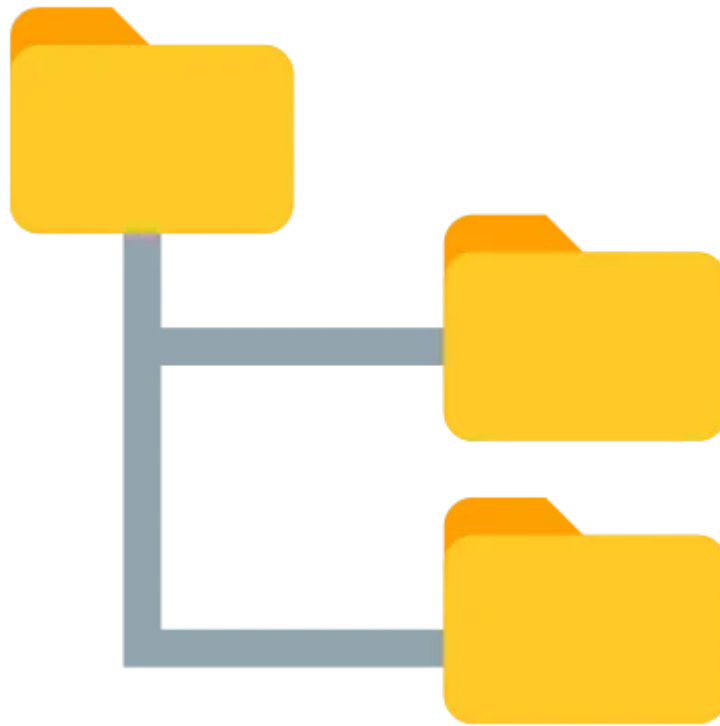


More

Introdução

Conforme o tempo vai passando, vai mudando uma série de preferências, conhecimentos e experiências. Com o tempo fui percebendo que uma forma de organizar me atendia perfeitamente na maioria dos projetos Angular que atuava.

Resolvi compartilha-la aqui. Espero que ajude!



É uma sugestão de estruturação para um projeto em Angular. Adapte-a e use-a da forma que preferir, sintá-se a vontade. É uma ideia!

Open in app ↗



Search



```
|-- app
  |-- core
    |-- [+] components
    |-- [+] guards
    |-- [+] interceptors
    |-- [+] models
    |-- [+] layouts
    |-- [+] config
    |-- [+] translations
    |-- [+] overrides
    |-- variables.scss
    |-- theme.scss
    |-- core.module.ts
  |-- features
    |-- login
      |-- components
        |-- [+] login-form
        |-- login-components.module.ts
      |-- interfaces
        |-- login-data.interface.ts
```

```

|-- login.component.ts
|-- login.component.html
|-- login.component.scss
|-- login.component.spec.ts
|-- login.module.ts
|-- home
|-- [+] components
|-- [+] enums
|-- [+] interfaces
|-- home.component.ts
|-- home.component.html
|-- home.component.scss
|-- home.component.spec.ts
|-- home.module.ts
|-- shared
|-- [+] components
|-- [+] validators
|-- [+] pipes
|-- [+] directives
|-- [+] services
|-- [+] enums
|-- shared.module.ts
|-- app.component.ts
|-- app.component.html
|-- app.component.scss
|-- app.component.spec.ts
|-- app-routing.module.ts
|-- app.module.ts

```

Explicando tintim por tintim

Core

É literalmente sua palavra em inglês: o núcleo (da aplicação). Lá está tudo que é essencial para o funcionamento.

- **Models**, nessa pasta estão os modelos os quais criou para sua aplicação.
- **Components**, aqui estão os componentes universais da aplicação. A *navbar* ou o *footer* que criou são bons exemplos.
- **Guards**, é onde você irá por os guardas de rota que criou, como por exemplo aquele para proteger rotas que o usuário estar autenticado é um pré-requisito.
- **Interceptors**, é a pasta que conterà nossos interceptadores de requisições, um exemplo mais que excelente é um *interceptor* cujo sua responsabilidade é injetar o *token* obtido no *header* da requisição.
- **Layouts**, pasta que armazena os *layouts* pré-definidos criados da aplicação.

- **Translations**, armazena as traduções utilizadas na internacionalização.
- **Configs**, pasta onde fica alguns arquivos de configuração.
- **Overrides**, é a pasta que tem os arquivos que utilizamos para sobrescrever uma parte terceira. Um exemplo é um arquivo responsável por sobrescrever uma classe do Bootstrap.
- **Variables.scss**, é um arquivo com algumas variáveis CSS que podem ser necessárias.
- **Theme.scss**, é um arquivo de tema. Muito comum para customizar uma UI que está utilizando como Bootstrap, Angular Material, PrimeNG etc.

É importante que você carregue (importando) o *core.module.ts* no *app.module.ts* da aplicação. Fazemos isso para carregar todo o seu conteúdo na inicialização da aplicação, pois é o conteúdo núcleo dela.

Features

É simplesmente a pasta que contém definitivamente aquilo que dá valor ao sistema, ou seja, as entregas. *Features* são as características do sistema. Imagine um e-commerce, poderíamos listar como *features*:

1. A tela que lista os produtos;
2. A tela que mostra particularmente um produto;
3. A tela de cadastro do usuário;
4. A tela de pagamento.

Todas essas *features* são exemplos de partes funcionais do sistema. Cada item dentro dessa pasta pode ter seus próprios componentes e, até quem sabe, outros *modules*. Tudo depende do seu projeto.

Vamos viajar mais ainda. Imagine que está construindo um pequeno *ERP* para gerenciamento de uma indústria. É claro que de forma bem acadêmica, poderíamos muito bem ter uma pasta *features* com a seguinte organização e composição:

```
|-- features
    |-- clientes
        |-- components
```

```
|-- [+] listagem-clientes
|-- [+] novo-cliente
|-- clientes-components.module.ts
|-- clientes.component.ts
|-- clientes.component.html
|-- clientes.component.scss
|-- clientes.component.spec
|-- clientes.module.ts
|-- produtos
|-- components
|   |-- [+] listagem-produtos
|   |-- [+] novo-produto
|   |-- produtos-components.module.ts
|-- produtos.component.ts
|-- produtos.component.html
|-- produtos.component.scss
|-- produtos.component.spec
|-- produtos.module.ts
|-- vendas
|-- components
|   |-- [+] listagem-vendas
|   |-- [+] nova-venda
|   |-- vendas-components.module.ts
|-- vendas.component.ts
|-- vendas.component.html
|-- vendas.component.scss
|-- vendas.component.spec
|-- vendas.module.ts
|-- dashboard
|-- dashboard.component.ts
|-- dashboard.component.html
|-- dashboard.component.scss
|-- dashboard.component.spec
|-- dashboard.module.ts
```

Vale destacar a importância de cada feature ter seu próprio module, pois isso é fundamental para fazer uma aplicação utilizando o conceito de lazy loading.

Observe que simples é: temos *features* que contemplam e agrupam todo seu sistema por características e entrega de valor. Além disso, caso seja necessário utilizar os componentes de produtos em outras telas basta importar o *module produtos-components.module.ts*.

Shared

Essa é a mais baba. Aqui fica tudo que é compartilhado e reutilizado em toda aplicação, não tem segredo.

As pastas filhas são apenas para agrupar por tipo de conteúdo — *pipes* com *pipes*, *components* com *components* e assim por diante.

É importante que importe a `shared.module.ts` também em seu `app.module.ts` e em todos os outros demais modules criados. Fazemos isso, pois em tese tudo que está na `shared.module.ts` está disponível para ser utilizado.

Aproveite, também, a `shared.module.ts` para importar e exportar outros módulos comumente utilizados em toda aplicação, por exemplo: `HttpClientModule`, `ReactiveFormsModule`, `CommonModule`, módulos do `toolkit` que está utilizando, entre outros módulos que possa fazer sentido.

Fim

Sem muita ladainha, espero ter te ajudado com algum conhecimento. Fico feliz caso essa sugestão tenha te ajudado ou então dado base para você mudar, adaptar ou até mesmo criar a sua própria.

Se você curtiu, considere comprar um café como um sinal de agradecimento.

Obrigado pela leitura e até mais!

[Angular](#)[Desenvolvimento Web](#)[Programação](#)[Frontend](#)[JavaScript](#)[Follow](#)

Written by Belmiro

88 Followers

Software Engineer. LinkedIn: <https://www.linkedin.com/in/belmiroflavio/>

More from Belmiro



Belmiro in React Brasil

10 APIs grátis e legais para você consumir

Fala pra mim, tá de bobeira aí?

2 min read · Mar 7, 2021



143



5



Belmiro in React Brasil

Minha primeira experiência com o Svelte (SvelteKit)

Tirei umas horinhas para dar uma olhada nesse tal Svelte

10 min read · Feb 28, 2022

 23  1



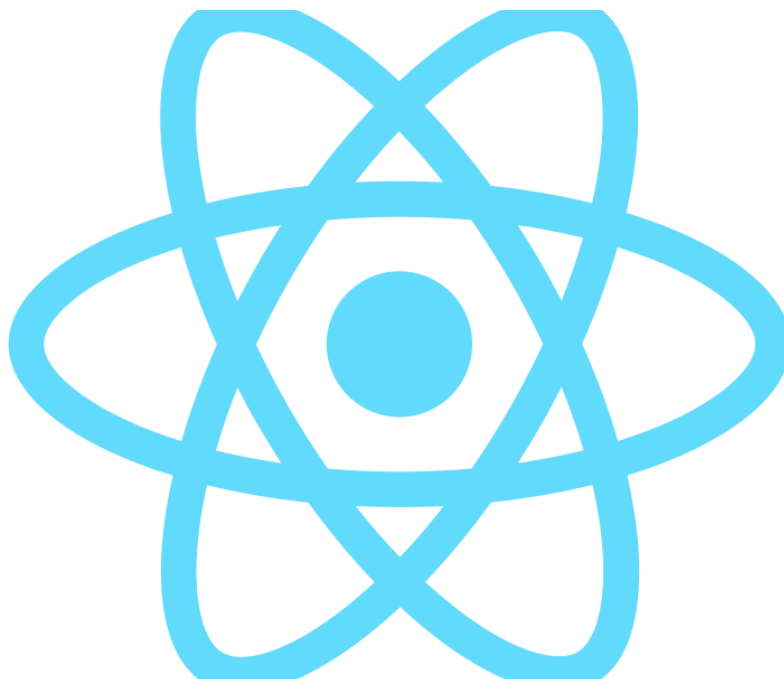
 Belmiro in React Brasil

Frontenders, bora falar de acessibilidade?

8 min read · Jan 5, 2021

 78 



Belmiro in React Brasil

Minha primeira experiência com o React (Native)

Me perdoa Angular?!

7 min read · Jan 15, 2021



91



See all from Belmiro

Recommended from Medium



Minko Gechev in Angular Blog

Introducing Angular v17

Last month marked the 13th anniversary of Angular's red shield. AngularJS was the starting point for a new wave of JavaScript frameworks...

17 min read · Nov 8



3.9K



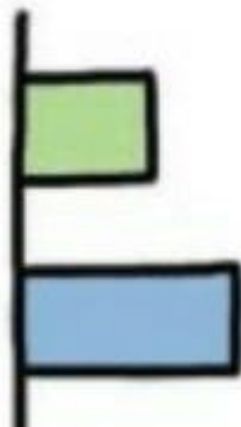
40



FEELINGS OF POWER

MONEY

STATUS



Fabio Zuin

Angular 17: Inject or constructor? Signals or Observables? ChangeDetection?

Angular evolves, and so do the approaches to creating components. Join me on a journey through a guide to component development

4 min read · Dec 12



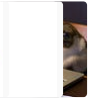
154



2



Lists



Stories to Help You Grow as a Software Developer

19 stories · 654 saves



General Coding Knowledge

20 stories · 718 saves



Apple's Vision Pro

7 stories · 38 saves



Modern Marketing

49 stories · 319 saves

{ **JSON** } is slow?

```
{
  "name": "JSON is slow!",
  "blog": true,
  "writtenAt": 1695884403,
  "topics": ["JSON", "Javascript"]
}
```



Alternatives?



Vaishnav Manoj in DataX Journal

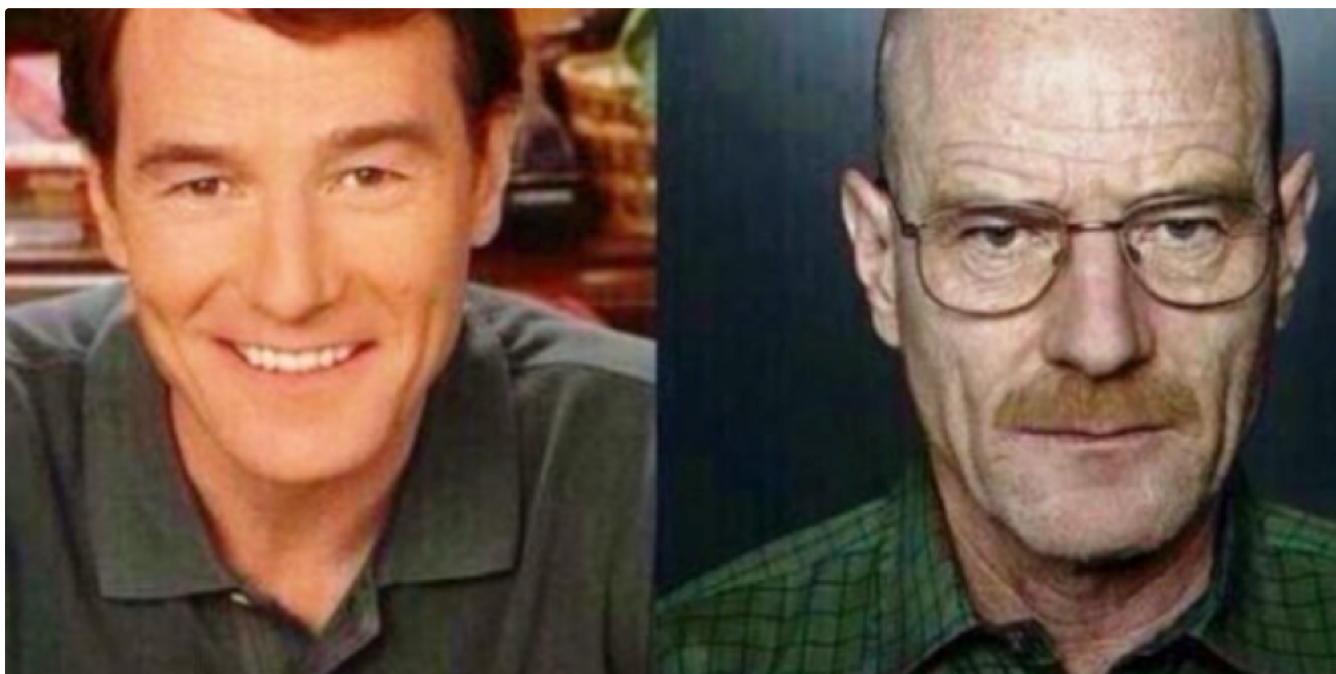
JSON is incredibly slow: Here's What's Faster!

Unlocking the Need for Speed: Optimizing JSON Performance for Lightning-Fast Apps and Finding Alternatives to it!

16 min read · Sep 28

 11.2K  128




 David Goudet

This is Why I Didn't Accept You as a Senior Software Engineer


An Alarming Trend in The Software Industry

🌟 · 5 min read · Jul 25

 7K  72



 Ilir Beqiri in ITNEXT

OnPush and Signals: Local Change Detection in Angular 17

Getting to know about new change detection mode in Angular

9 min read · 3 days ago




169



Directives



 Yuvaraj S

Angular Standalone components 02: Standalone Directives

In my previous blog we discussed about how to create a standalone component.

2 min read · Aug 21



See more recommendations