

---

Artigo

[Danusa Calixto](#) · Ago. 26, 2022 9min de leitura

## Angular. Dicas gerais

Antes de começarmos com alguns tópicos intermediários e avançados, gostaria de resumir alguns pontos mais gerais. Eles são subjetivos, é claro, então ficarei feliz em discuti-los se você tiver outra opinião ou argumentos melhores para qualquer um deles.

A lista não é abrangente e isso é intencional, pois abordarei alguns tópicos em artigos futuros.

### Dica 1. Siga o guia de estilo oficial

Angular é bastante rigoroso em termos de limitação da arquitetura possível de um aplicativo, mas ainda há muitos lugares nele que permitem que você faça as coisas do seu jeito. A imaginação dos desenvolvedores é ilimitada, mas às vezes dificulta o trabalho de quem trabalha no projeto com você ou depois de você.

A equipe Angular faz um bom trabalho mantendo a própria arquitetura Angular e suas bibliotecas, então eles definitivamente sabem como criar uma base de código estável e suportável.

Eu recomendo seguir seu guia de estilo oficial e se desviar dele apenas se as coisas não funcionarem dessa maneira. Isso tornará as coisas mais fáceis quando você chegar a um novo projeto ou se alguém entrar em seu projeto.

Lembre-se, código e arquitetura de aplicativo com suporte, estável e fácil de entender são mais importantes do que soluções inteligentes, mas enigmáticas, do que ninguém será capaz de acompanhar (possivelmente até você no futuro).

Guia oficial de estilo Angular: <https://angular.io/guide/styleguide>

### Dica 2. Considere comprar o livro Angular Ninja

Você pode pensar que isso é um anúncio, mas é o seguinte: esse é um livro muito bom com todos os principais conceitos de Angular cobertos e o preço fica a seu critério. Você também pode escolher quanto dinheiro irá para escritores e quanto irá para caridade.

Um dos autores deste livro é membro do time Angular, então esta é definitivamente a fonte de informação mais confiável sobre o Angular após a documentação oficial. Você pode ver os capítulos do livro na página do livro e ler capítulos de amostra para decidir se vale a pena seu investimento.

Além disso, o livro é atualizado com o lançamento de uma nova versão do Angular e você obtém todas as atualizações do livro gratuitamente.

O próprio blog do Ninja Squad é uma fonte muito boa de informações sobre Angular, com artigos e notícias sobre novas versões, melhores práticas, recursos experimentais e muito mais.

Livro Angular Ninja: <https://books.ninja-squad.com/angular>

### Dica 3. Leia a documentação oficial

Antes de mergulhar na escrita de algum código do seu aplicativo, é uma boa ideia ler a documentação e os guias oficiais, especialmente se você iniciar seu projeto em uma versão do Angular que não usou antes. As coisas continuam sendo obsoletas o tempo todo, tutoriais e guias na Internet podem estar desatualizados e você pode acabar aumentando sua dívida técnica em vez de usar novas práticas e funcionalidades recomendadas.

Também é um bom hábito verificar para qual versão o guia foi escrito. Se for a versão 2+ atrás da que você está usando, é melhor verificar se as coisas mudaram desde então.

Documentação oficial: <https://angular.io>

## Dica 4. Considere usar o Angular CDK mesmo que você não use o Angular Material

Falarei melhor em artigos futuros, mas sei que muitos desenvolvedores Angular nem conhecem o Angular CDK.

Angular CDK é uma biblioteca de diretivas úteis e classes base que podem ajudá-lo a desenvolver melhores aplicativos. Por exemplo, tem coisas como FocusTrap, Drag & Drop, VirtualScroll e muito mais, que podem ser facilmente adicionados aos seus componentes

CDK angular: <https://material.angular.io/cdk/categories>

## Dica 5. Corrija suas dependências no package.json

Não está particularmente relacionado ao Angular e pode ser importante em qualquer projeto. Quando você faz `npm install --save <something>`, ele será adicionado ao seu `package.json` com `^` ou `~` no início da versão do pacote. Isso significa que seu projeto poderá usar qualquer versão secundária/patch da mesma versão principal da dependência. Isso vai dar errado no futuro com quase 100% de probabilidade. Eu enfrentei esse problema em diferentes projetos tantas vezes. Tempo passando e nova versão menor de alguma dependência saindo e seu aplicativo não será mais compilado. Porque um autor desta dependência atualizou suas dependências em versão menor (ou mesmo patch) e agora elas estão em conflito com sua compilação. Ou pode haver um novo bug apresentado em uma nova versão menor de sua dependência e você não tem problemas em seu código (porque você instalou suas dependências antes da nova versão aparecer), mas qualquer outra pessoa tentando instalar e compilar seu projeto a partir do repositório irá enfrentar o bug que você nem conhece (e não poderá reproduzi-lo em sua máquina).

O `package-lock.json` existe para resolver esse problema e ajudar os desenvolvedores a ter o mesmo conjunto de dependências em todo o projeto. Idealmente, ele deve ser comprometido com o repositório, mas muitos desenvolvedores estão decidindo adicionar esse arquivo ao `.gitignore`. E se acabou, você pode acabar com os problemas acima. Portanto, é melhor não confiar cegamente na resolução de suas dependências e corrigi-la na versão específica, escaneá-la regularmente em busca de vulnerabilidades (com `npm audit`) e depois atualizá-la e testá-la manualmente.

## Dica 6. Tente atualizar seu aplicativo para a nova versão do Angular assim que puder

Angular está evoluindo continuamente e novas versões são lançadas a cada 6 meses ou mais. Manter seu aplicativo atualizado permitirá que você use todos os novos recursos, incluindo compilações mais rápidas e outras otimizações. Além disso, atualizar para a próxima versão principal do Angular não deve ser um grande problema neste caso. Mas se você está 5 versões atrás e seu aplicativo é de tamanho médio ou grande, o processo de atualização para a última versão pode ser difícil, pois o Angular não possui esquemas para atualizar aplicativos pulando versões intermediárias do Angular. Você precisará atualizar todas as versões intermediárias uma a uma, verificando se o aplicativo funciona em cada versão. A atualização direta sem os esquemas da CLI do Angular é possível, mas também pode ser complicada, então sugiro manter seu aplicativo atualizado e atualizado.

## Dica 7. Tente reduzir sua lista de dependências

É fácil cair no hábito de trazer novas dependências em seu aplicativo, pois você precisa de novas funcionalidades. Mas com cada dependência, a complexidade do seu aplicativo está crescendo e o risco de uma avalanche de dívida técnica repentina está aumentando.

Se você decidiu adicionar uma nova dependência em seu aplicativo, pense nisso:

- Quão bem suportada é a dependência?
- Quantas pessoas estão usando?
- Qual o tamanho da equipe de desenvolvimento?
- Com que rapidez eles estão fechando os problemas do GitHub?
- Como a documentação da dependência está escrita?
- Com que rapidez ele é atualizado após o lançamento do novo major de Angular?
- Qual o impacto dessa dependência no desempenho e no tamanho do pacote do seu aplicativo?
- Quão difícil será substituir essa dependência se ela for abandonada ou preterida no futuro?

Se algumas dessas perguntas tiverem resposta de tom negativo, considere se livrar dela ou pelo menos substituí-la por algo mais maduro e bem fundamentado.

## Dica 8. Não use `<any>` como seu tipo “ temporário ”

Novamente, é fácil começar a usar qualquer um enquanto você escreve sua lógica de negócios, porque escrever tipos apropriados pode ser demorado e você precisa terminar sua tarefa neste sprint. Os tipos podem ser adicionados mais tarde, é claro. Mas é escorregadio aumentar a dívida técnica.

Uma arquitetura de seu aplicativo e tipos devem ser definidos antes de escrever qualquer lógica de negócios. Você deve entender claramente quais objetos você terá e onde eles serão usados. Especificação antes do código, certo? (Tom Demarco escreveu um livro sobre isso antes de se tornar popular:

[\[https://www.amazon.com/Deadline-Novel-About-Project-Management/dp/093263...\]](https://www.amazon.com/Deadline-Novel-About-Project-Management/dp/093263...)(  
[https://www.amazon.com/Deadline -Novel-About-Project-Management/dp/0932633390](https://www.amazon.com/Deadline-Novel-About-Project-Management/dp/0932633390))).

Se você estiver escrevendo o código sem tipos predefinidos, poderá acabar com a pior arquitetura de aplicativo e funções que usam objetos muito semelhantes, mas diferentes. Portanto, você precisará criar tipos diferentes para cada função (o que fará as coisas ainda piores) ou gastar seu tempo escrevendo especificações, tipos de refatoração E , o que é uma perda de tempo comparado a se tivesse sido feito antes.

## Dica 9. Gaste seu tempo para entender o processo de construção

Angular faz um ótimo trabalho para facilitar o trabalho do desenvolvedor em relação à construção do projeto. Mas as opções padrão nem sempre são as melhores para todos os casos.

Gaste seu tempo para entender como o processo de compilação funciona no Angular, quais as diferenças entre as compilações de Desenvolvimento e Produção, quais opções o Angular tem para compilações (como otimizações, mapas de origem, agrupamento e muito mais).

## Dica 10. Investigue o que está dentro do seu pacote

Nem todas as bibliotecas fornecem trepidação de árvores e nem sempre importamos da maneira certa, então sempre há uma chance de que algo redundante seja empacotado com nosso aplicativo.

Portanto, é um bom hábito investigar o conteúdo do seu pacote de tempos em tempos.

Existem bons artigos descrevendo o processo usando webpack-bundle-analyzer , então não vou abordar aqui, mas aqui está um link para um deles: <https://www.digitalocean.com/community/tutorials/angular-angular-webpack-bundle-analyzer>

Abordarei este tópico mais detalhadamente mais adiante na série.

## Dica 11. Use a propriedade services providedIn ao invés de importar o serviço no módulo

Muitos exemplos de código Angular na Internet usam a importação dos serviços nos módulos. Mas não é a maneira preferida de declarar serviços desde o Angular 6 ou 7. Devemos usar a propriedade do decorador @Injectable providedIn para habilitar o tree-shake e melhor agrupamento de nossos aplicativos. Angular é inteligente o suficiente para entender em qual pacote o serviço deve ser incluído, quando deve ser inicializado e quantas instâncias do serviço devem ser criadas.

Existem três valores que providedIn aceita. Na maioria das vezes root é suficiente, mas existem mais dois:

- root: o serviço será singleton no escopo de aplicação
- any: uma instância do serviço será criada para todos os módulos carregados ansiosamente, com uma instância diferente criada para cada módulo lento
- plataforma: o serviço será singleton para todas as aplicações rodando na mesma página

## Dica 12. Não esqueça das principais regras de desempenho

- Use trackBy para coleções para reduzir operações de redesenho e execução de JavaScript
- Use a estratégia de detecção de alterações onPush em todos os lugares que puder (abordarei no artigo dedicado)
- Realize cálculos pesados fora do ngZone
- Use padrões de aceleração e debounce com seus eventos para evitar chamadas de servidor desnecessárias e inundação de eventos
- Use a rolagem virtual para exibir conjuntos de big data
- Use tubos puros para transformação de dados em seus modelos
- Use a compilação AoT
- Carregar lentamente as partes do seu aplicativo que não são necessárias para o início do aplicativo
- Evite cálculos e chamadas de função condicionais em seus templates (chamadas de função devem ser usadas apenas para eventos)

Obrigado por ler! Espero que algumas dessas dicas tenham sido úteis para você. Se você tiver quaisquer comentários e observações, por favor, deixe-me saber nos comentários, ficarei feliz em discutir

Até mais!

[#Angular](#) [#Angular2](#) [#Desenvolvimento de IU \(Interface do Usuário\)](#) [#Frontend](#) [#Guia de Codificação](#) [#Outro](#)

---

URL de origem: <https://pt.community.intersystems.com/post/angular-dicas-gerais>