Tornar-se engenheiro sénior.

Guia prático para programadores full-stack

CONTEÚDO	QTD.
CAPÍTULOS	12
CONCEITOS	134
LISTAS DE VERIFICAÇÃO	8
MODELOS	4
PLANO DE IMPLEMENTAÇÃO	1

feitura.

Tornar-se engenheiro sénior.

Guia prático para programadores full-stack

feitura.com

1.ª edição · 2026

Ficha técnica

Título: *Tornar-se engenheiro sénior — Guia prático para programadores full-stack*
Título original: *Becoming a Senior Engineer — A Practical Handbook for Full-Stack Developers*
Edição: 1.ª edição, 2026 · versão 1.1
Editor: Feitura · feitura.com

Composto em Archivo e IBM Plex Mono.

Índice

00

INTRODUÇÃO

Como usar este guia

O que é este livro

Este livro é um guia autónomo para se tornar um programador full-stack sénior mais sólido. Não é uma coletânea de links. Foi pensado para ser lido sem acesso à internet e para ser útil em qualquer projeto em que esteja a trabalhar.

O conteúdo é uma síntese original, baseada em práticas correntes de engenharia sénior, liderança, arquitetura, testes, operações e gestão de projetos. É deliberadamente prático: cada capítulo deve ajudá-lo a tomar melhores decisões, escrever melhor software, comunicar melhor e criar evidências de que o seu trabalho é de nível sénior.

A ideia central

Um programador sénior não é apenas alguém que sabe mais sintaxe. Um programador sénior melhora os resultados de forma fiável em situações de incerteza.

Por isso, precisa de desenvolver quatro capacidades em conjunto:

1. Profundidade técnica: compreende código, sistemas, dados, testes, desempenho e operações.
2. Discernimento de produto: consegue ligar as escolhas de engenharia aos resultados para o utilizador e para o negócio.
3. Capacidade de entrega: consegue decompor o trabalho, reduzir o risco e concluir incrementos que acrescentam valor.
4. Atitude de liderança: melhora as pessoas e o sistema à sua volta sem esperar que lhe deem autoridade.

Este guia treina estas capacidades em paralelo.

Como estudar

Siga um ciclo semanal:

1. Leia um capítulo ou parte de um capítulo.
2. Identifique três ideias que possa aplicar de imediato.

3. Aplique uma delas no código ou no planeamento do projeto.
4. Escreva uma nota curta a explicar o que mudou na sua forma de pensar.
5. Faça a revisão da semana com o modelo de registo de evidências da Parte V.

Não tente absorver tudo depressa. A senioridade vem de integrar o conhecimento, não de ter contacto com ele. O objetivo não é ler muitas páginas. O objetivo é mudar a forma como concebe, constrói, testa, comunica e decide.

Tempo semanal recomendado

Para 8 a 12 horas por semana:

- 2 horas a ler e a tomar notas.
- 4 horas a construir ou a refatorar alguma coisa.
- 2 horas a testar, a depurar ou a trabalhar no desempenho.
- 1 hora a escrever uma nota de design, um ADR (registo de decisão de arquitetura) ou um resumo de revisão.
- 1 hora a fazer a revisão semanal.

Se tiver menos tempo, mantenha a mesma estrutura, mas reduza o volume. Nunca elimine por completo a escrita e a revisão; é nelas que o discernimento se torna visível.

Aprendizagem baseada em evidências

Todos os meses, deve produzir evidências. As evidências são o que separa uma confiança vaga de uma prova de nível sénior.

Exemplos de evidências:

- Uma funcionalidade entregue com testes.
- Uma migração e um esquema explicados com clareza.
- Um documento de design que expõe os trade-offs.
- Um relatório de bug com a causa raiz e as medidas de prevenção.
- Um refactoring que reduziu a complexidade.

- Uma investigação de desempenho com medições de antes e depois.
- Uma revisão de código que melhorou a facilidade de manutenção.
- Um plano de projeto que identificou cedo os riscos.

Guarde estes artefactos. São eles que vão formar o seu portefólio sénior.

Como ler um capítulo

Para cada capítulo, responda:

1. Que problema é que este capítulo me ajuda a resolver?
2. O que já faço bem aqui?
3. O que evito porque me deixa desconfortável?
4. Que comportamento concreto vou praticar esta semana?
5. Que evidências vão provar que o pratiquei?

Associar o guia a um projeto real

O que se lê sem praticar esquece-se. Escolha um projeto em que possa aplicar deliberadamente cada capítulo à medida que avança. Qualquer aplicação não trivial serve: uma ferramenta de gestão de encomendas, um sistema de acompanhamento de projetos, um painel de inventário, um fluxo de faturação, um site de conteúdos, um back-office interno. O projeto final da Parte IV deste livro, um plano de implementação completo, oferece um exemplo pronto a usar que pode construir de ponta a ponta.

Use o projeto para praticar:

- Modelação do domínio com entidades e relações reais.
- Design de API com fronteiras de recursos bem definidas.
- Frontend bem feito, aplicado a fluxos de trabalho reais.
- Estratégia de testes com testes unitários, de funcionalidade e de ponta a ponta.

- Operações com Docker, migrações, logs e documentação de instalação.
- Liderança com registos de decisão e notas de planeamento.

O projeto não precisa de se tornar enorme. Precisa de se tornar um espaço onde pratica hábitos seniores de forma deliberada.

O nível de exigência sénior

Em cada tema, procure atingir este nível de exigência:

- Consigo explicar a ideia de forma simples.
- Consigo aplicá-la no código.
- Consigo identificar os trade-offs.
- Consigo ensiná-la a outra pessoa.
- Consigo produzir evidências de que a usei bem.

Quando conseguir fazer estas cinco coisas de forma consistente em arquitetura, testes, entrega, comunicação e liderança, já não está apenas a acumular conhecimento. Está a construir discernimento sénior.

CONCEITOS DESTA INTRODUÇÃO

PARTE I

Pensar como um sénior

01

CAPÍTULO 01

A mentalidade do engenheiro sénior

Das entregas aos resultados

No início de uma carreira, o sucesso significa muitas vezes concluir corretamente as tarefas atribuídas. O trabalho de nível sénior é diferente. Para um sénior, a pergunta não é apenas «Construí o que me pediram?» É também:

- Isto resolveu o problema certo?
- Reduziu ou aumentou o risco futuro?
- A equipa consegue compreendê-lo e mantê-lo?
- Tornei visíveis os trade-offs?
- Deixei o sistema mais fácil de alterar?

Um programador sénior continua a escrever código. Mas o código faz parte de uma responsabilidade maior: criar resultados fiáveis com discernimento técnico.

A senioridade é discernimento perante restrições

Os projetos reais raramente têm informação perfeita, tempo perfeito ou requisitos perfeitos. A senioridade é a capacidade de tomar boas decisões mesmo assim.

Restrições comuns:

- O prazo é fixo, mas o âmbito não é claro.
- A base de código tem comportamentos legados que ninguém compreende por completo.
- A arquitetura ideal é demasiado cara para a fase atual.
- O negócio quer rapidez, mas o sistema já tem problemas de qualidade.
- A equipa não está de acordo quanto à abordagem.

Um programador sénior não espera que toda a incerteza desapareça. Reduz a incerteza deliberadamente.

Perguntas úteis para um sénior:

- Qual é a coisa mais pequena que podemos construir para aprender?
- Que decisão é reversível?
- Que decisão é cara de reverter?

- O que tem de ser verdade para esta abordagem funcionar?
- O que pode falhar sem darmos por isso?
- O que estamos a tentar otimizar neste momento?

Os três horizontes

Os programadores seniores pensam em três horizontes ao mesmo tempo.

Horizonte 1: hoje

Conseguimos entregar o trabalho atual com segurança? Os testes estão a verde? Há utilizadores bloqueados? Há algum problema em produção? O próximo passo está claro?

Horizonte 2: este mês

Estamos a acumular dívida técnica que nos vai atrasar? Estão a surgir padrões? A equipa está alinhada? Estamos a construir as bases certas para as próximas funcionalidades?

Horizonte 3: este trimestre

A arquitetura vai continuar a suportar o crescimento esperado? Estamos a criar silos de conhecimento? Estamos a investir em ferramentas, documentação e fiabilidade antes que a dor se transforme em crise?

Um erro comum é viver apenas no Horizonte 1. Isso dá rapidez a curto prazo e lentidão a longo prazo. Outro erro é viver apenas no Horizonte 3. Isso produz diagramas impressionantes e pouca coisa entregue. O discernimento sénior equilibra os três.

Responsabilidade sem controlo

Muitas vezes vai ser responsável por resultados sem ter controlo total sobre todas as variáveis. É normal.

A responsabilidade (ownership) significa:

- Detetar os problemas cedo.
- Comunicar os riscos com clareza.

- Propor opções em vez de se limitar a relatar obstáculos.
- Levar as coisas até ao fim.
- Tornar visível o trabalho que não se vê.

A responsabilidade não significa fazer tudo sozinho. Aliás, fazer tudo sozinho pode ser um modo de falha. Um programador sénior melhora a organização do trabalho para que os outros possam contribuir.

Bom gosto em engenharia

Os bons engenheiros seniores desenvolvem gosto técnico. O gosto não é uma preferência pessoal disfarçada de autoridade. É reconhecimento de padrões adquirido com a experiência.

Exemplos de bom gosto:

- Escolher tecnologia aborrecida — comprovada, sem surpresas — quando a fiabilidade importa mais do que a novidade.
- Evitar abstrações até a duplicação provar que são necessárias.
- Dar nomes às coisas com base no significado do domínio e não em detalhes de implementação.
- Escrever testes centrados no comportamento e não no funcionamento interno.
- Desenhar API difíceis de usar de forma errada.
- Pensar no deploy e no rollback antes de ir para produção.

O gosto apura-se ao rever as próprias decisões. Pergunte-se: o que é que esta escolha facilitou, o que é que dificultou e o que me surpreendeu mais tarde?

Os modos de falha do sénior

Cuidado com estas armadilhas:

A armadilha do herói

Resolve tudo sozinho, depressa e em silêncio. Parece produtivo, mas a equipa fica dependente de si. O trabalho de nível sénior deve aumentar a capacidade da equipa e não apenas a produtividade individual.

A armadilha do astronauta da arquitetura

Cria complexidade para possibilidades futuras que podem nunca se concretizar. A arquitetura de nível sénior não é a abstração máxima. É a estrutura adequada às necessidades atuais e às necessidades futuras plausíveis.

A armadilha do cínico

Já viu muitos fracassos e, por isso, passa a desvalorizar tudo. A experiência deve criar discernimento, não desprezo. Os melhores engenheiros seniores conseguem ser céticos e construtivos ao mesmo tempo.

A armadilha da otimização local

Otimiza o seu ticket, mas prejudica o sistema como um todo. Os programadores seniores preocupam-se com o fluxo de valor em toda a equipa e em todo o produto.

Um método prático para decidir como um sénior

Perante uma decisão, escreva respostas curtas a estas perguntas:

1. Problema: que problema estamos a resolver?
2. Contexto: que restrições importam?
3. Opções: que duas ou três abordagens viáveis existem?
4. Trade-offs: qual é o custo de cada opção?
5. Risco: o que pode correr mal?
6. Reversibilidade: podemos desfazer isto mais tarde?

7. Decisão: o que escolhemos agora?
8. Revisão: quando vamos voltar a analisá-la?

Isto pode tornar-se um pequeno ADR (registo de decisão de arquitetura). Não precisa de ser sofisticado. O valor está em explicitar o raciocínio.

EXERCÍCIO PRÁTICO

Escolha uma decisão técnica do projeto atual. Escreva uma nota de uma página com:

- A decisão.
- Porque foi necessária.
- As opções consideradas.
- Porque escolheu a opção atual.
- O que teria de acontecer para mudar de ideias.

Este exercício treina o músculo central de um sénior: o discernimento explícito.

CONCEITOS DESTE CAPÍTULO