



Como diagnosticar equipes de Tecnologia em 2027?

2027 vai cobrar mais velocidade com o mesmo time. Aqui você aprende a ler uma operação de Produto e Engenharia com sete análises e descobre por onde começar.

NESSE MATERIAL

- Um exemplo real de relatório de diagnóstico.
- O passo a passo e o que você precisa para rodar cada análise.
- 8 skills prontas e 7 tipos de análise.

PARA

Líderes de Tecnologia

LEITURA

15 minutos

Como ler uma operação?

Toda operação de Produto e Engenharia funciona igual por dentro. Pessoas e sistemas sustentam processos. Processos produzem entregas. Entregas movem o resultado de negócio. Simples assim: é nessa cadeia que o diagnóstico caminha.



ONDE A DOR APARECE

"A gente não está atraindo clientes rápido o suficiente."

ONDE A CAUSA ESTÁ

Gente trabalhando em cima de uma arquitetura que não acompanhou.

Como diagnosticar em quatro etapas?

Jogar "me faz um diagnóstico" numa AI devolve um texto bonito e genérico, diferente a cada vez. O que falta é método: uma ordem de leitura e réguas fixas, para você poder comparar o antes com o depois daqui a três meses.

ETAPA 1

Mapear o contexto

Como a operação está hoje?

O QUE SAI

Uma foto crua: dados de repositório, tickets, pipeline e conversas com o time, mais a dor que a liderança sente e as hipóteses que ela carrega.

ETAPA 2

Ler com réguas

Onde ela trava?

O QUE SAI

Achados organizados por camada, cada um amarrado à régua que o produziu.

ETAPA 3

Escolher a restrição

O que atacar primeiro?

O QUE SAI

Uma aposta pequena e concreta, com ponto de partida e prazo claro.

ETAPA 4

Medir e reabrir

Mudou de patamar?

O QUE SAI

O resultado medido na mesma régua do início, e a próxima restrição da fila.

Uma análise para cada camada

Cada análise mede uma camada com uma régua conhecida, o que permite comparar entre times e ao longo do tempo. Nenhuma delas, sozinha, revela a restrição. Ela só aparece quando você cruza os achados.

Resultados

o número de negócio

ENTRA POR DECLARAÇÃO

A dor do negócio, o momento do time e o que a liderança acha que está acontecendo.

Entregas

o que sai

DORA Metrics

O time entrega com que frequência? E com que estabilidade?

Processos

o caminho

Value Stream Mapping

Onde o trabalho fica parado entre a ideia e a produção?

Pessoas

quem faz

DevEx

O que trava o dia a dia de quem coloca a mão no código?

Team Topologies

Quem é dono de quê, e quanta carga carrega?

Sistemas

com o quê

Systems Inventory

O que existe, como se conecta e o que dá para mexer sem dor?

Tech Capabilities

O que sustenta o fluxo hoje, e o que está faltando?

AI Readiness

Corta todas as camadas: a operação tem base para um agente de AI rodar nela?

DORA Metrics

Quatro métricas de entrega vindas do DORA (programa de pesquisa do Google Cloud): frequência de deploy, tempo até produção, taxa de falha e recuperação. É a única régua que permite comparar a sua operação com o resto do mercado.



- 1 Tempo até produção, 23 dias.** Mais de uma semana já é desempenho baixo. O item fica pronto, mas demora para chegar no cliente.
- 2 Taxa de falha, 20%.** Cada deploy carrega 6,5 itens juntos. Com tanta coisa subindo de uma vez, fica difícil saber o que quebrou.
- 3 Tempo de recuperação, 21 horas.** A recuperação é a única métrica boa aqui. O problema real está no tamanho do lote que cada deploy carrega.

REQUISITOS

1. Acesso de leitura ao repositório e à esteira de deploy (CI/CD).
2. Saber exatamente qual evento da esteira marca o deploy em produção.
3. Registro de incidentes. Se não tiver, hotfix e revert funcionam como substituto.

COMO FAZER

1. Defina uma janela de 3 a 6 meses e liste todos os deploys em produção.
2. Frequência: divida o total de deploys pelas semanas da janela.
3. Tempo até produção: meça do primeiro commit ao deploy. Use a mediana, não a média.
4. Taxa de falha: quantos deploys geraram incidente, hotfix ou revert, dividido pelo total.
Recuperação: tempo da falha até o próximo deploy que resolveu.
5. Compare cada número com as faixas do DORA. O pulo do gato: leia velocidade e estabilidade juntas, nunca separadas.

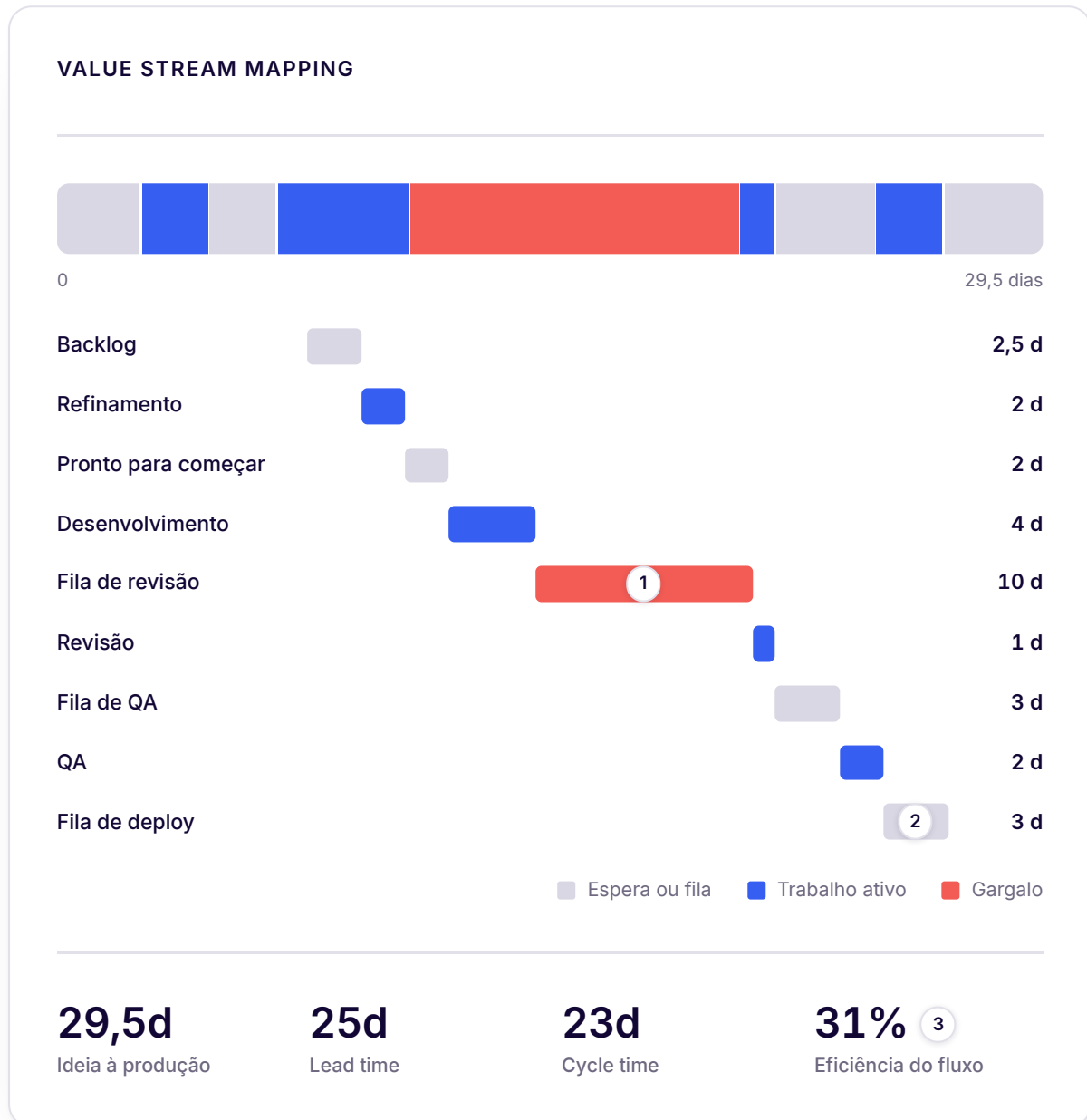
CUIDADOS

1. Só produção conta. Deploy em staging ou teste fica de fora.
2. Poucos deploys fazem o número oscilar muito. Sempre declare o volume da janela.
3. Nunca leia uma métrica isolada. Velocidade boa com falha alta quer dizer que o time entrega mais rápido do que consegue verificar.

↓ Download Skill

Value Stream Mapping

Cada item percorre um caminho no board, da entrada até produção. Essa análise separa quanto tempo é trabalho de verdade e quanto é espera. Vem da Teoria das Restrições e é a que mais encontra gargalo.



- 1 Fila de revisão, 10 dias.** 34% do tempo total. O código fica pronto e ninguém toca nele. É mais do que desenvolvimento, QA e deploy juntos.
- 2 Fila de deploy, 3 dias.** Só 10% do tempo. A ideia de que as releases travam no fim do ciclo não se confirma.
- 3 31% de eficiência do fluxo.** A cada 10 dias, cerca de 7 são fila. O trabalho ativo soma 9 dos 29,5 dias.

REQUISITOS

1. Exportação do board com o histórico de mudança de status e as datas de cada uma.
2. Uma conversa rápida com o time para classificar cada status como espera ou trabalho.
3. Itens que completaram o ciclo inteiro dentro da janela de análise.

COMO FAZER

1. Liste os status do board na ordem real do trabalho. Marque cada um como espera ou trabalho.
2. Para cada item, calcule quanto tempo ficou em cada status.
3. Resuma cada etapa pela mediana e pelo p85 (o tempo que 85% dos itens não ultrapassam). Fuja da média.
4. Calcule a eficiência do fluxo: tempo de trabalho dividido pelo tempo total.
5. A etapa com mais tempo é o gargalo. Leve o achado para o time como pergunta: por que o item empaca ali?

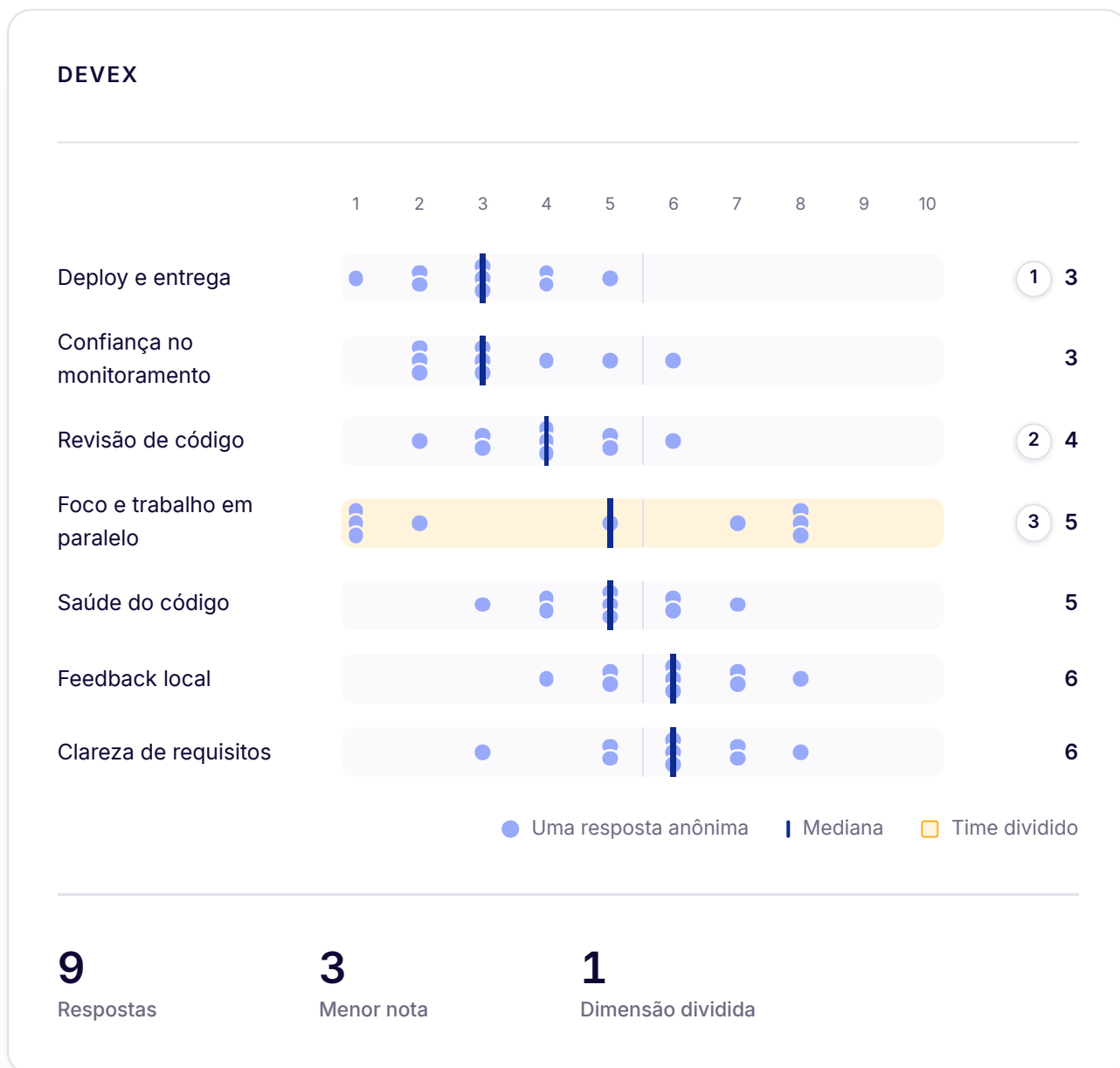
CUIDADOS

1. Board que não reflete o trabalho. Se o time não move o card, o tempo vira ruído. Cheque antes de calcular: isso por si só já é um achado.
2. A média engana. Um item travado 40 dias puxa tudo para cima.
3. Não existe faixa de mercado. Compare a operação com ela mesma ao longo do tempo.

↓ Download Skill

DevEx

Questionário curto e anônimo sobre o que atrapalha o dia a dia de quem desenvolve. Mede aquilo que as ferramentas não capturam, e cada nota é cruzada com um número das outras análises.



- ① **Deploy e monitoramento, nota 3.** As duas menores notas batem com o DORA: muita falha e o cliente descobrindo o problema antes do time.
- ② **Revisão de código, nota 4.** O time sente na pele a fila de 10 dias que o Value Stream Mapping mediu.
- ③ **Foco dividido.** Quatro pessoas dão 1 ou 2, e quatro dão 7 ou 8. São dois modos de trabalhar dentro do mesmo time, com 33 itens abertos para 9 pessoas.

REQUISITOS

1. Um formulário anônimo que leva uns 10 minutos por pessoa.
2. O time precisa saber que o resultado avalia o processo, nunca as pessoas.
3. Os números do DORA e do Value Stream Mapping, para cruzar.

COMO FAZER

1. Escolha de 6 a 8 dimensões: deploy, revisão, monitoramento, foco, saúde do código, feedback local, clareza de requisitos.
2. Faça uma pergunta por dimensão, com nota de 1 a 10. Nota alta sempre significa melhor.
3. Busque no mínimo 80% de resposta.
4. Olhe a mediana e a distância entre a maior e a menor nota. Mais de 5 pontos de diferença indica time dividido (convenção deste guia).
5. Cruze cada dimensão com um número concreto: revisão com a fila de revisão, deploy com o DORA.

CUIDADOS

1. Amostra pequena. Com 9 respostas, declare a amostra e trate a mediana como sinal, não como verdade absoluta.
2. Não avalia pessoas. Se for usado para avaliar alguém, o questionário para de funcionar.
3. Nota e dado podem divergir. Quando divergem, os dois entram na investigação.

 [Download Skill](#)

Team Topologies

Quem muda cada parte do código, e se a fronteira de cada time confere com o que ele de fato mantém. Mostra o dono de cada módulo e onde o trabalho cruza times. Avalia o desenho, nunca as pessoas.

TEAM TOPOLOGIES

MÓDULO	PEDIDOS	CATÁLOGO	PLAT.	DE FORA
checkout	42%	25%	33%	58% ¹
compartilhado	16%	21%	63%	37% ²
integrações	12%	6%	82%	18%
pedidos	84%	7%	9%	16%
loja virtual	89%	11%	.	11%
autenticação	10%	.	90%	10%
catálogo	.	96%	4%	4%
integração com o ERP	.	.	100%	0%
infraestrutura	.	.	100%	0%

■ Fronteira respeitada (até 12%)
■ Porosa (13% a 35%)
■ Desalinhada (acima de 35%)

329

Pull requests

79%

Do time dono

2

Módulos desalinhados

2

Módulos porosos

¹ **Checkout, 58% de fora.** Três times mexem no mesmo módulo. Cada mudança pede revisão de alguém de outro time, e é daí que nasce a fila de revisão.

² **Compartilhado, 37% de fora.** Mesmo padrão, num módulo que todo mundo usa.

REQUISITOS

1. Histórico de pull requests: quem escreveu e quem revisou.
2. A lista de times e quem faz parte de cada um.
3. Os módulos do código: as partes que mudam de forma independente.

COMO FAZER

1. Defina os módulos e ligue cada pessoa ao seu time.
2. Em cada módulo, calcule a parcela de pull requests de cada time.
3. O time com a maior fatia é o dono. O resto é atravessamento.
4. Classifique a fronteira: até 12% de atravessamento, respeitada; de 13% a 35%, porosa; acima de 35%, desalinhada (convenção deste guia).
5. Nos módulos desalinhados, pergunte aos times como trabalham juntos ali: no mesmo código, por contrato, ou ajudando o outro a aprender.

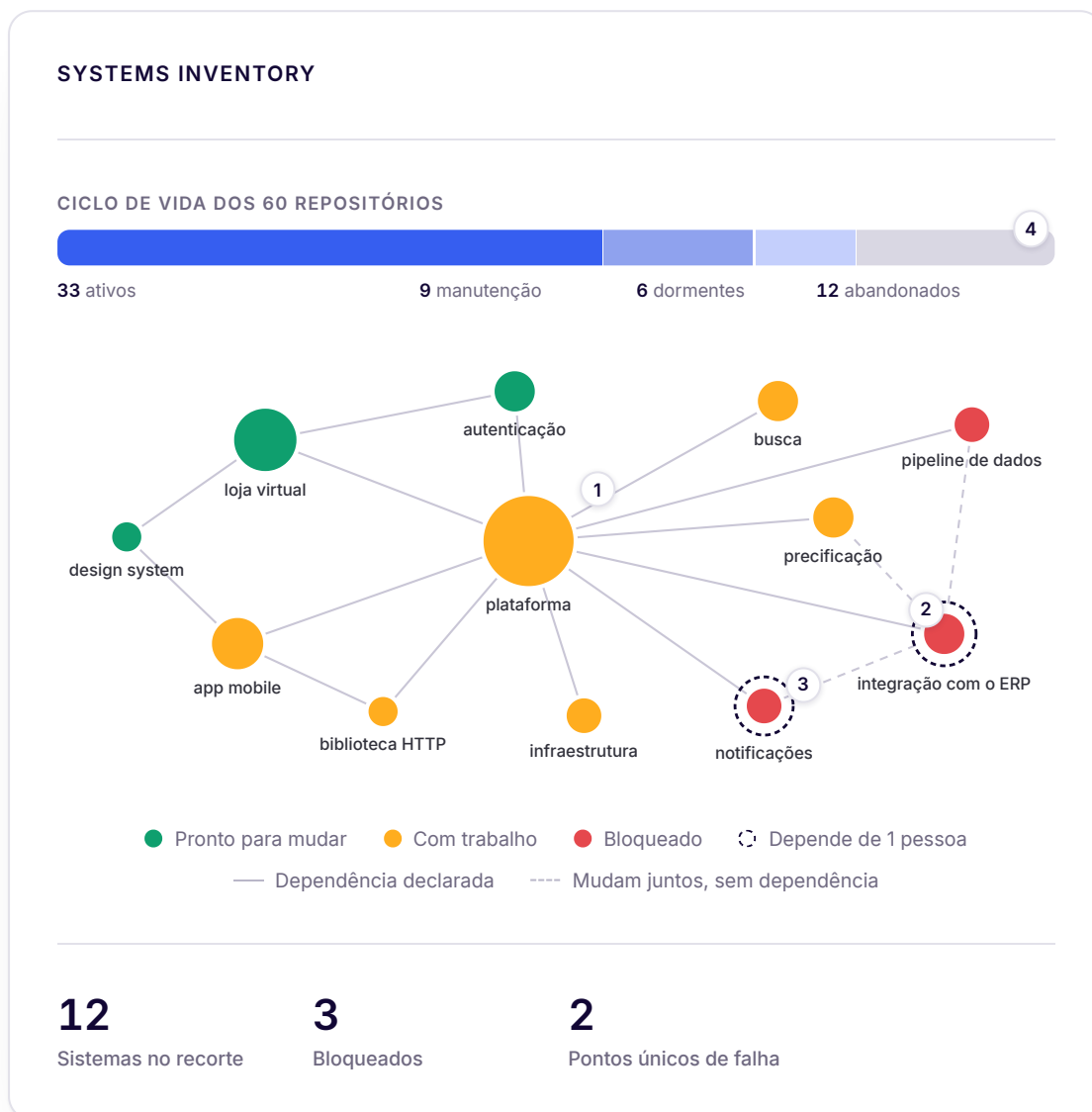
CUIDADOS

1. Nunca vira ranking de pessoas. O time é a unidade e ninguém é nomeado.
2. Colaboração pontual é saudável. O alerta é a colaboração permanente num módulo que deveria ter um dono claro.
3. O organograma pode estar desatualizado. Confirme os times com a liderança antes de rodar.

↓ Download Skill

Systems Inventory

O mapa de tudo que a operação mantém: quais sistemas existem, como se conectam, quem sabe mexer neles e quais estão prontos para receber mudança. Todo plano de transformação passa por aqui.



- 1 **Plataforma.** Muda junto com quase todos os outros sistemas: é o centro de tudo. Checkout e compartilhado, os módulos sem dono do Team Topologies, ficam dentro dela. Está com trabalho porque falta dono claro nesses dois módulos.
- 2 **Integração com o ERP.** Leva os pedidos ao financeiro. Uma pessoa fez 80% das mudanças, e o sistema muda junto com outros 3 sem nenhuma dependência declarada. Está bloqueado para mudar.
- 3 **Notificações.** Mesmo risco: se a pessoa que mexe nelas sair, ninguém mais sabe como mudar.
- 4 **12 repositórios abandonados.** Três deles ainda rodam relatórios e pedidos antigos, e ninguém responde por eles.

REQUISITOS

1. Acesso de leitura a todos os repositórios da organização.
2. A lista dos sistemas que vivem fora de repositório (ERP, ferramentas contratadas) e de quem cuida de cada um.
3. Uma conversa de uns 30 minutos com quem conhece a arquitetura.

COMO FAZER

1. Classifique cada repositório pela última mudança: ativo (até 90 dias), em manutenção (até 180), dormente (até 540) ou abandonado (mais de 540).
2. Mapeie as conexões: dependências declaradas no código e pares que mudam juntos. Acima de 0,5 de co-mudança sem dependência declarada é uma ligação que ninguém desenhou.
3. Para cada sistema, conte quantas pessoas fizeram 80% das mudanças. Se foi uma só, é ponto único de falha.
4. Classifique a prontidão com três perguntas: tem dono claro? Tem teste no caminho crítico? Tem interface documentada? Três sim = pronto. Um não = com trabalho. Dois ou mais não, ou ponto único de falha = bloqueado.
5. Analise a fundo só os sistemas do recorte e valide o mapa com o time.

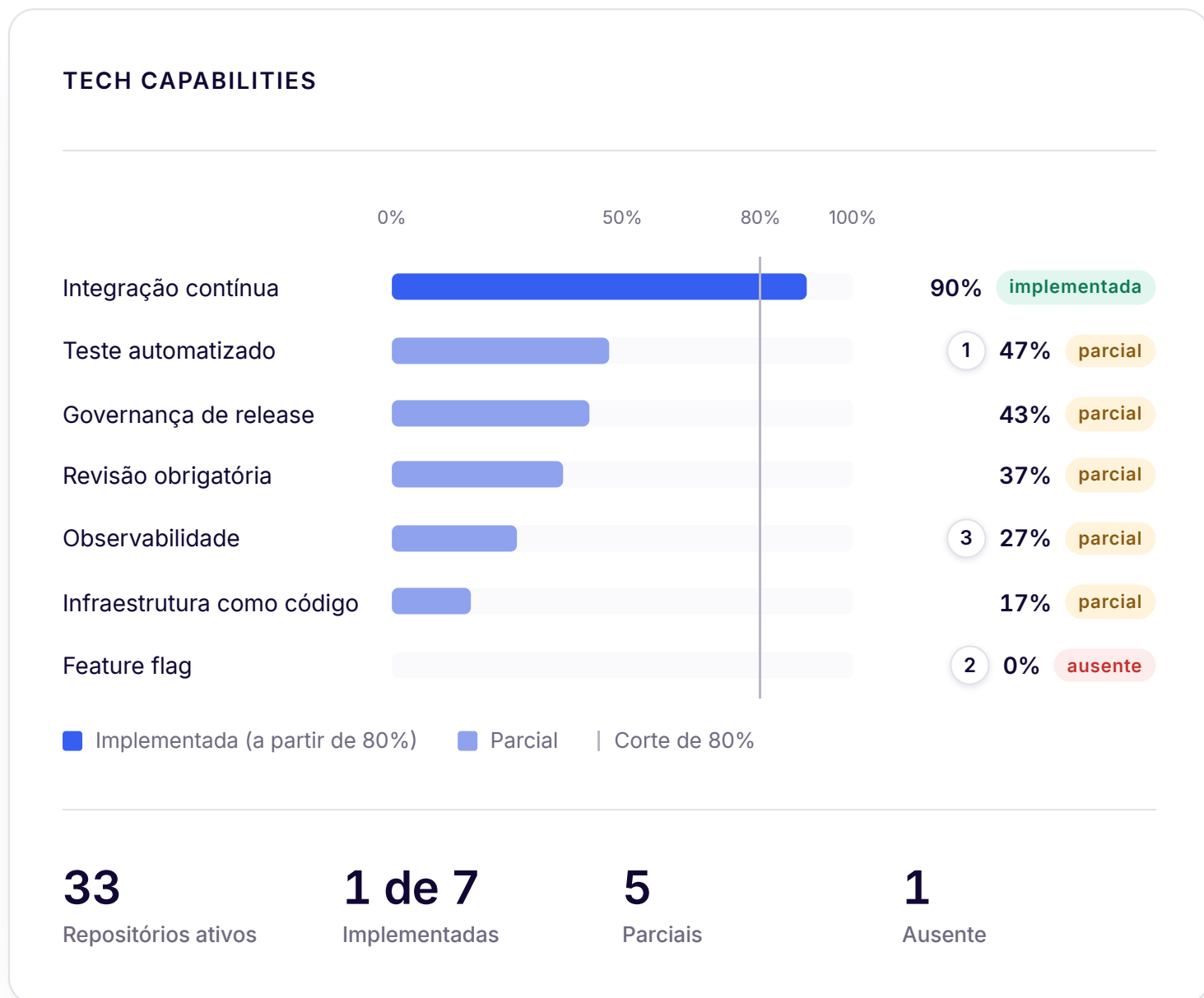
CUIDADOS

1. Abandonado não é desligado. Confirme se ainda roda algo em produção.
2. Escopo largo dilui o sinal. Inventarie tudo, mas vá fundo só no recorte.
3. Os limiares são convenção deste guia. Ajuste ao ritmo da sua operação e deixe claro.

↓ Download Skill

Tech Capabilities

As práticas técnicas que sustentam o fluxo, medidas pelo que de fato aparece no código e na esteira. Para cada prática, a parcela dos repositórios em que ela existe. O resultado é uma lista do que construir.



- 1 Teste automatizado, 47%.** Metade do código sobe sem teste. É daí que vem a falha em 1 de cada 5 deploys.
- 2 Feature flag, 0%.** Sem como subir código desligado, todo deploy libera tudo de uma vez, e o lote cresce.
- 3 Observabilidade, 27%.** O cliente costuma ver a falha antes do time. A recuperação até é rápida, mas depende de alguém avisar.

REQUISITOS

1. Acesso de leitura aos repositórios ativos.
2. A configuração da esteira (CI/CD) e as regras de proteção de branch.
3. A lista de bibliotecas e ferramentas de monitoramento que estão em uso.

COMO FAZER

1. Escolha de 6 a 8 práticas: integração contínua, teste automatizado, revisão obrigatória, governança de release, observabilidade, infraestrutura como código, feature flag.
2. Em cada repositório ativo, procure o sinal de cada prática.
3. Registre de onde veio o sinal: comportamento observado, configuração ou sinal indireto.
4. Calcule a parcela de repositórios com sinal positivo.
5. Classifique: implementada a partir de 80%, parcial abaixo disso, ausente quando nenhum repo tem, não observada quando faltou acesso (convenção deste guia).

CUIDADOS

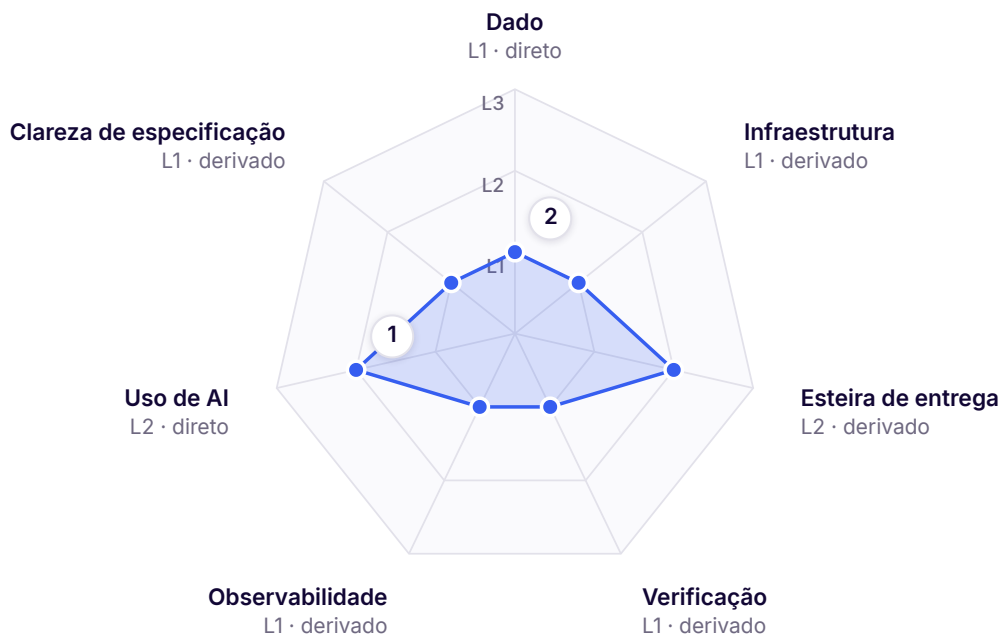
1. Configurado não é usado. O que aparece no comportamento vale mais do que o que está declarado.
2. Não observada é diferente de ausente. Se você não teve acesso ao sinal, diga isso.
3. É uma lista do que construir. Não vira nota nem ranking de time.

↓ Download Skill

AI Readiness

Sete eixos que mostram se a operação tem base para um agente de AI rodar nela. Cinco medem engenharia, dois medem o uso de AI em si. Colocar agente em cima de operação frágil só amplifica a fragilidade.

AI READINESS



5

Eixos em L1

2

Eixos em L2

0

Eixos em L3

2

Com evidência direta

1 **Uso de AI em L2, verificação em L1.** O time já usa AI no dia a dia, mas sem política escrita e com teste em só 47% dos repos. A AI está acelerando o que já é frágil.

2 **Dado em L1.** Sem catálogo e sem dono por domínio, o agente trabalha em cima de dado que ninguém garante.

REQUISITOS

1. Os resultados das outras seis análises. Cinco dos sete eixos saem de lá.
2. Um questionário curto sobre como o time usa AI e se existe alguma política.
3. Uma conversa sobre governança de dados com quem cuida dessa parte.

COMO FAZER

1. Avalie cada eixo em três níveis. L1: não existe ou cobre até um terço da operação. L2: existe, mas não é padrão. L3: é o jeito padrão de fazer as coisas.
2. Use evidência direta quando tiver. Nos outros eixos, derive das análises anteriores, mas deixe claro de onde veio.
3. Ordene os eixos pelo que trava os demais. A sequência natural: dado e infraestrutura primeiro, depois verificação e observabilidade, por último uso de AI.
4. Não calcule média nem nota única. O que importa são as combinações entre os eixos.
5. Escreva a política de AI do time, se ainda não existir. O pior caso é "sai usando".

CUIDADOS

1. Adoção não é prontidão. Todo mundo usando Copilot não diz nada sobre se a base está pronta.
2. Eixo derivado herda a precisão da análise que alimentou ele. Nada de casas decimais.
3. O nível serve para localizar, não para dar nota. A conversa que importa é sobre o que trava.

O QUE AMPLIFICA A AI

Em 2025, o DORA publicou sete capacidades que amplificam o efeito da AI. Nenhuma delas se compra pronta.

1. Posição clara sobre AI, comunicada para o time inteiro.
2. Ecossistema de dados organizado e saudável.
3. Dados internos acessíveis para a AI consumir.
4. Controle de versão sólido.
5. Trabalho em lotes pequenos.
6. Foco no usuário.
7. Plataforma interna de qualidade.

↓ Download Skill

O que aparece quando você cruza as análises?

Sozinha, cada análise mostra um sintoma. O diagnóstico de verdade aparece quando os achados se conectam numa cadeia, da dor que a liderança sente até a causa lá embaixo.

UM NÚMERO POR ANÁLISE

● Crítico ● Pede atenção

● DORA 20% Dos deploys falham	● VSM 10 dias Na fila de revisão	● DEVEX Score: 4 Para a revisão de código	● TEAM TOP. 58% Das mudanças no checkout vêm de fora
● INVENTORY 3 de 12 Sistemas bloqueados para mudar	● TECH CAP. 47% Dos repositórios com teste	● AI READINESS L1 Em verificação e dado	

DA DOR À CAUSA

ENTREGAS	23 dias até produção, e 1 em cada 5 deploys com falha.
PROCESSOS	10 dias na fila de revisão: 34% do tempo do item.
PESSOAS	O time dá nota 4 para revisão, e no checkout 58% das mudanças vêm de fora do time dono.
SISTEMAS	Os dois módulos sem dono ficam dentro da plataforma, por onde tudo passa.



A causa: módulos compartilhados sem dono claro. Qualquer mudança neles precisa de revisão de outro time, e essa revisão espera.

O QUE A LIDERANÇA ACHAVA VS. O QUE O DADO MOSTROU

HIPÓTESE DA LIDERANÇA	VEREDITO	O DADO
A revisão de código é o gargalo.	Confirmada	10 dias esperando revisão, 34% do tempo.
As releases travam no fim do ciclo.	Em parte	A fila de deploy é só 10% do tempo, mas 1 em cada 5 deploys falha. O fim do ciclo é instável, não lento.
A plataforma compartilhada trava o squad.	Em parte	No geral, a fronteira funciona (79% das mudanças são do time dono). O problema está em dois módulos específicos.
A instabilidade está custando venda.	Sem dado	A falha está medida. Conectar falha a venda exige dado de negócio que não estava nas ferramentas.

SEIS FRENTES EM 13 SEMANAS



No máximo três frentes ao mesmo tempo. As metas saem da imersão com o time.

25 → 18 dias

A métrica que acompanha tudo: o lead time do compromisso, do momento em que o time se compromete com o item até produção. O objetivo: tirar 7 dos 10 dias de espera por revisão.

COMO FAZER A LEITURA GLOBAL

1. Coloque um número de cada análise lado a lado.
2. Monte a cadeia de cima para baixo, até chegar em pessoas ou sistemas.
3. Teste cada hipótese da liderança contra o dado: confirmada, em parte, refutada ou sem dado.
4. Escolha uma restrição e aplique os cinco passos.
5. Monte o plano em frentes, com camada, ponto de partida, meta, prazo e a data da próxima leitura.

Download Skill

O diagnóstico de hoje é uma foto.
O objetivo é transformar em filme.

A maioria das leituras de operação termina num relatório grande que mostra como as coisas estavam naquele dia. Só que a operação muda toda semana, e a foto envelhece. O filme é a mesma leitura, repetida a cada ciclo, com a mesma régua.

	DIAGNÓSTICO PONTUAL	DIAGNÓSTICO CONTÍNUO
QUANDO	Uma vez, num projeto com data para acabar	A cada ciclo, com a próxima leitura já na agenda
O QUE MOSTRA	Como a operação estava no dia	Como a operação está agora
RÉGUA	Muda a cada leitura, e o antes não se compara com o depois	A mesma do ponto de partida, do primeiro ao último ciclo
RESTRIÇÃO	Fica presa na primeira que apareceu	Muda de lugar, e a leitura vai atrás
RESPONDE	O que está errado	Se a aposta de fato mudou a operação de patamar

Quer rodar essa leitura na sua operação?

A BossaBox faz o diagnóstico junto com o seu time: dados das ferramentas que vocês já usam, conversa com quem faz acontecer e as hipóteses da liderança testadas na prática. Você sai com a restrição, o primeiro movimento e a régua para medir se ele funcionou.

 [Agendar com Especialista](#)