SPEC-DRIVEN DEVELOPMENT

Specky transforma a disciplina de especificações em um motor de enforcement programável.

Um servidor MCP open-source com 53 ferramentas validadas e um pipeline de 10 fases que leva qualquer entrada até o código em produção, de forma determinística, em qualquer IDE com IA.

AUTORA
Paula Silva

CARGO
Software Global Black Belt

DURAÇÃO
45 a 60 minutos

DATA
2026-06-01

AGENDA

Cinco partes, um argumento.

PART I	O problema, por que vibe coding gera código rápido e times lentos
PART II	O motor, o que é o Specky e como ele força determinismo
PART III	O pipeline, dez fases, requisitos EARS e portões de revisão humana
PART IV	O ecossistema, qualquer entrada, qualquer IDE, sem vendor lock-in
PART V	A decisão, prontidão enterprise e por onde começar na segunda

PARTE

I

O problema.

A IA escreve código em segundos. Mas rápido não é o mesmo que certo, e ninguém escreveu o que certo significava.

O CUSTO DE PULAR A ESPECIFICAÇÃO

40%

do tempo de engenharia vira retrabalho quando os requisitos nunca foram escritos.

Você pede ao assistente um sistema de login. Ele gera código na hora, pula os requisitos, adivinha a arquitetura e entrega algo que roda mas não corresponde à intenção de ninguém. Não há como verificar o código contra uma intenção que nunca foi capturada.

COMO CHAMAM ISSO		O QUE ISSO PRODUZ
vibe coding	→	código sem intenção

DOIS JEITOS DE CONSTRUIR COM IA

A correção não é um prompt melhor. É um pipeline validado entre intenção e código.

VIBE CODING

Código por vibe

- O modelo gera código direto de uma frase.
- Requisitos nunca são capturados, critérios de aceite nunca definidos.
- Não há como verificar o resultado contra a intenção original.
- Funciona na demo, quebra na revisão e em produção.

DESENVOLVIMENTO DETERMINÍSTICO

Código a partir de specs validadas

- Uma especificação estruturada descreve o que o sistema deve fazer, primeiro.
- Cada requisito é validado antes de uma linha de código ser escrita.
- Spec, design e tarefas ficam alinhados e rastreáveis.
- A IA é a operadora. O Specky é o motor.

PARTE

II

O motor.

O Specky reimplementa a metodologia Spec-Kit como 53 ferramentas com enforcement, com validação programática no lugar de boas intenções.

O QUE É O SPECKY, EM TRÊS NÚMEROS

Um servidor MCP open-source que canaliza a criatividade da IA por um pipeline validado.

FERRAMENTAS MCP VALIDADAS

53

Em treze categorias: conversão de entrada, núcleo do pipeline, portões de qualidade, diagramas, infraestrutura, testes e exportação.

FASES DO PIPELINE

10

Uma máquina de estados que bloqueia o pulo de fases. Você não salta de Init para Design sem completar Specify antes.

IDES SUPORTADAS

qualquer

Funciona em qualquer IDE com IA que fale MCP. Use no VS Code com GitHub Copilot ou na GitHub Copilot CLI. Um comando npx.

AS 53 FERRAMENTAS, ORGANIZADAS

Cinquenta e três ferramentas, treze categorias, agrupadas nas sete famílias que você de fato usa.

01 · CONVERSÃO DE ENTRADA

Transforma transcrições, documentos, designs e texto cru em Markdown limpo que o pipeline lê.

02 · NÚCLEO DO PIPELINE

As dez ferramentas de fase, de sdd_init a sdd_release, que movem a máquina de estados adiante.

03 · PORTÕES DE QUALIDADE

O validador EARS e a análise cruzada que travam uma fase até o trabalho ficar coerente.

04 · DIAGRAMAS

Geradores de dezessete tipos de diagrama de engenharia de software, de C4 a fluxos de sequência.

05 · INFRAESTRUTURA

Roteamento para Terraform e Docker para a spec chegar a ambientes reais, não só a um documento.

06 · TESTES

Ferramentas que ligam cada requisito a um teste, para a verificação provar que o código bate com a spec.

07 · EXPORTAÇÃO

Roteamento para GitHub, Azure DevOps e Jira, para o plano virar issues e pull requests.

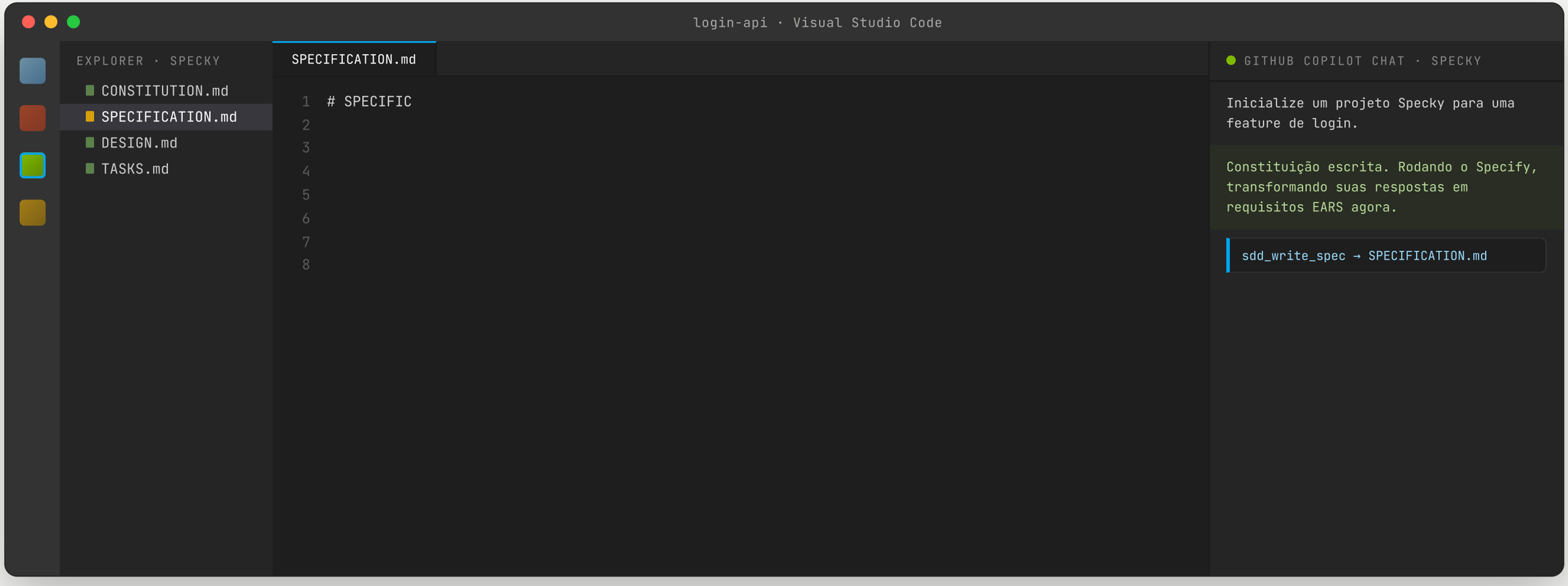
53

ferramentas no total, cada uma com validação estrita de entrada. Você nunca as chama na mão, o pipeline chama.

Você não decora ferramentas. Você passa pelas fases, e cada fase chama as ferramentas que precisa.

SIMULAÇÃO · SPECKY DENTRO DO VS CODE

Ele vive no editor que você já usa. Aqui, escrevendo uma spec, ao vivo.



Reconstrução ilustrativa. Mesmo GitHub Copilot Chat, mesmos arquivos, os requisitos são validados conforme são escritos.

METODOLOGIA E MOTOR

O Spec-Kit define o que fazer.
O Specky força como fazer.

SPEC-KIT, A METODOLOGIA

Templates de prompt e definições de agentes

Notação EARS, fases com portões, o modelo de constituição e templates em Markdown. A IA os lê e tenta seguir. Leve, ideal para aprender Spec-Driven Development.

SPECKY, O MOTOR

Ferramentas que validam, forçam e geram

Uma máquina de estados, um validador de EARS por regex e schemas Zod transformam a metodologia em enforcement. Instale o Specky com um comando e toda a metodologia Spec-Kit vem embutida.

Não é preciso instalar o Spec-Kit à parte. A metodologia vem dentro do motor.

QUATRO MECANISMOS QUE TORNAM O DETERMINISMO REAL

Determinismo não é uma promessa. É forçado por código que se recusa a deixar o pipeline derivar.

MECANISMO 01

Máquina de estados

Enforcement de fases

- Dez fases obrigatórias, sem pular
- Arquivos obrigatórios liberam o avanço

MECANISMO 02

Validador EARS

Qualidade do requisito

- Cada requisito conferido contra 6 padrões
- Termos vagos como rápido ou bom são sinalizados

MECANISMO 03

Análise cruzada de artefatos

Pontuação de alinhamento

- Spec, design e tarefas conferidos por consistência
- Requisitos órfãos sinalizados na hora

MECANISMO 04

Roteamento MCP-to-MCP

Sem vendor lock-in

- Gera JSON estruturado para qualquer cliente
- Roteia para GitHub, Azure, Jira, Terraform

PARTE

III

O pipeline.

Dez fases. Sem pular. Revisão humana em cada portão. Cada fase deixa um artefato rastreável para trás.

DA INTENÇÃO AO RELEASE, UMA FASE DE CADA VEZ

A máquina de estados transforma uma ideia vaga em uma jornada documentada e reproduzível.

01 **Init** CONSTITUTION.md	02 **Discover** 7 perguntas	03 **Specify** SPECIFICATION.md	04 **Clarify** árvore de decisão	05 **Design** DESIGN.md
06 **Tasks** TASKS.md	07 **Analyze** ANALYSIS.md	08 **Implement** checkpoints	09 **Verify** VERIFICATION.md	10 **Release** PR + work items

Enquadrar o projeto · Especificar e desenhar · Planejar e auditar · Construir, verificar, entregar

SIMULAÇÃO · A MÁQUINA DE ESTADOS RODANDO

Aperte iniciar, e as dez fases executam em ordem, cada uma deixando um artefato.

01 **Init** CONSTITUTION.md	02 **Discover** 7 perguntas	03 **Specify** SPECIFICATION.md	04 **Clarify** árvore de decisão	05 **Design** DESIGN.md
06 **Tasks** TASKS.md	07 **Analyze** ANALYSIS.md	08 **Implement** checkpoints	09 **Verify** VERIFICATION.md	10 **Release** PR + work items

LGTM Pausado após o Specify. A máquina não avança para o Design até você revisar e responder LGTM.

Cada fase fica verde só quando seu artefato existe e seu portão passa. Pular é impossível por design.

A SAÍDA DA FASE 01, DE PERTO

Todo projeto abre com uma constituição. É isso que ela fixa.

A constituição é o livro de regras do projeto. A IA a lê antes de cada fase, então os mesmos princípios, stack e padrões de qualidade valem do primeiro requisito ao pull request final.

01 Princípios que o código deve honrar, como acessibilidade e privacidade por padrão.

02 O stack técnico, para a IA nunca adivinhar sua linguagem ou framework.

03 Portões de qualidade, a barra à qual toda fase seguinte é submetida.

CONSTITUTION.MD

```
# Principles
- Acessibilidade não é opcional. WCAG 2.2 AA.
- Nenhum dado pessoal sai do tenant.

# Stack
- TypeScript, React, PostgreSQL.
- Testes em Vitest. Lint com ESLint.

# Quality gates
- Cada requisito mapeia para um teste.
- Nenhuma fase avança com uma questão em aberto.
```

Amostra ilustrativa. Escreva uma vez, e toda fase herda essas regras automaticamente.

NOTAÇÃO EARS, OS SEIS PADRÕES DE REQUISITO

Cada requisito encaixa em uma de seis formas. O validador rejeita o que não encaixa.

Ubíquo	O sistema deve...	O sistema deve criptografar todo dado em repouso. Sempre verdadeiro, sem gatilho.
Por evento	Quando [evento], o sistema deve...	Quando um usuário envia o login, o sistema deve validar as credenciais.
Por estado	Enquanto [estado], o sistema deve...	Enquanto offline, o sistema deve enfileirar as requisições.
Opcional	Onde [condição], o sistema deve...	Onde 2FA está ativo, o sistema deve exigir um código de uso único.
Indesejado	Se [condição], então o sistema deve...	Se a sessão expirar, então o sistema deve redirecionar para o login.
Complexo	Enquanto [estado], quando [evento]...	Enquanto em manutenção, quando uma requisição chega, o sistema deve enfileirá-la.

EARS NA PRÁTICA

O mesmo requisito, rejeitado e aceito.

REJEITADO PELO VALIDADOR

"O login deve ser rápido e seguro."

- Rápido não nomeia número, então nenhum teste consegue checar.
- Seguro não tem gatilho e não tem condição.
- Não encaixa em nenhuma das seis formas EARS.

ACEITO PELO VALIDADOR

"Quando um usuário envia o login, o sistema deve responder em até 500 ms. Se as credenciais falharem três vezes, então o sistema deve bloquear a conta por 15 minutos."

- Forma por evento, com um 500 ms mensurável.
- Uma forma de comportamento indesejado para o bloqueio.
- Cada cláusula agora é um teste que você consegue escrever.

O validador não te faz escrever melhor. Ele se recusa a deixar uma frase não testável virar requisito.

UMA FEATURE, PELO PIPELINE

Veja uma única feature de login ir de uma frase a um pull request.

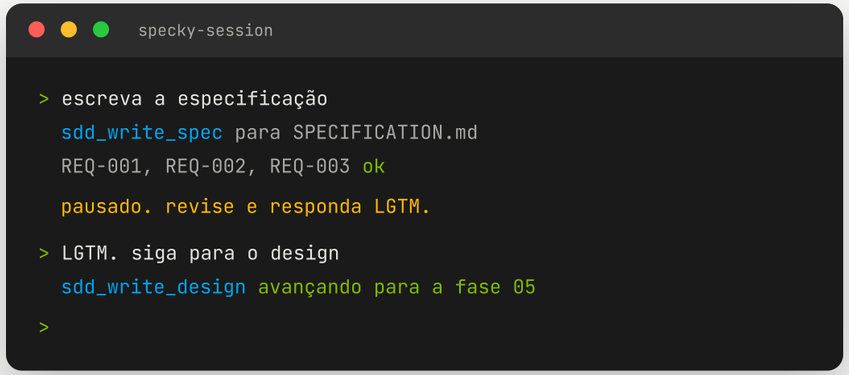
Discover 7 questions	Sete perguntas revelam o não dito: SSO ou senha, política de bloqueio, duração de sessão, auditoria.
Specify SPECIFICATION.md	Cada resposta vira um requisito EARS: REQ-001 validar credenciais, REQ-002 bloquear após três falhas.
Design DESIGN.md	A arquitetura: um serviço de auth, um rate limiter, um store de sessão, cada um ligado a um requisito.
Tasks TASKS.md	Um plano ordenado e conferível. Cada tarefa aponta o requisito que satisfaz e o teste que precisa.
Verify e release PR + work items	A verificação confirma que o código bate com a spec, e abre um pull request com os work items anexados.

Uma frase entra, uma trilha rastreável sai. Cada caixa acima é um arquivo real que um colega lê no ano que vem.

O HUMANO FICA NO CIRCUITO

Depois de cada fase principal, o pipeline pausa e espera você digitar LGTM.

Specify, Design e Tasks terminam cada um num portão de revisão. A IA não avança sozinha. Se a verificação detectar depois uma divergência entre a spec e o código, o Specky te leva de volta para corrigir antes de prosseguir.



3 portões de revisão

0 fases puladas

PARTE

IV

O ecossistema.

Qualquer entrada vira spec. Qualquer IDE executa. O Specky roteia o trabalho para as ferramentas que você já usa, sem lock-in.

SEIS FORMAS DE COMEÇAR

O que você já tem vira o ponto de partida de uma especificação.

01 · LINGUAGEM NATURAL

Um prompt no chat

Digite a ideia direto. sdd_init e sdd_discover a estruturam. Ideal para greenfield.

02 · TRANSCRIÇÃO DE REUNIÃO

VTT, SRT, TXT, MD

Extrai decisões, itens de ação e requisitos brutos do Teams, Zoom ou Meet.

03 · DOCUMENTOS

PDF, DOCX, PPTX

Importa RFPs, documentos de requisito e decks. Converte para Markdown e alimenta o pipeline.

04 · DESIGN NO FIGMA

Design para spec

Lê componentes e interações via Figma MCP. Ideal para trabalho design-first.

05 · SCAN DE CODEBASE

Contexto brownfield

Detecta linguagem, framework e estrutura antes de especificar uma feature nova.

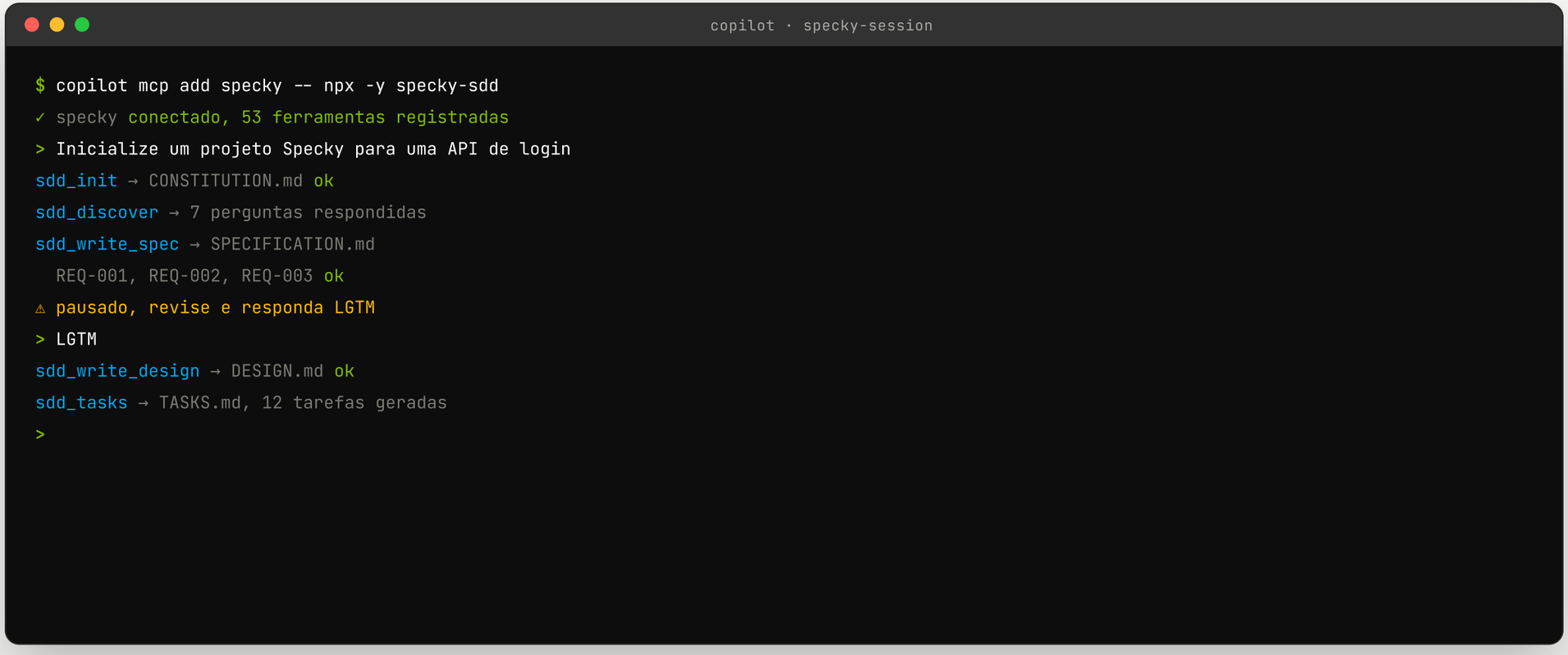
06 · TEXTO CRU

Cole qualquer coisa

Sem arquivo. Cole um e-mail do cliente ou notas e toda ferramenta de import aceita direto.

SIMULAÇÃO · A MESMA EXECUÇÃO NA GITHUB COPILOT CLI

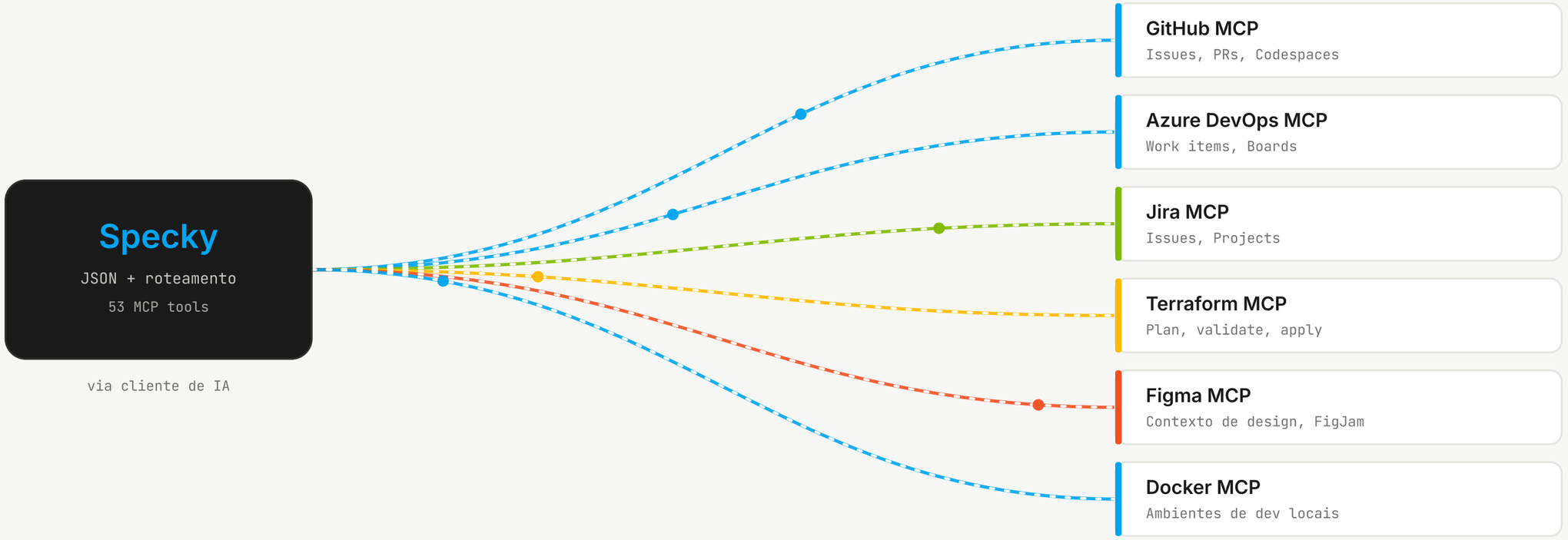
Sem IDE. O terminal roda o pipeline idêntico, linha por linha.



Reconstrução ilustrativa. Mesmas ferramentas, mesmos artefatos, mesmo portão LGTM. Só a superfície mudou.

ARQUITETURA MCP-TO-MCP

O Specky emite JSON estruturado. Seu cliente roteia para as ferramentas que você já paga.



Duas dependências de runtime. Zero requisições de rede de saída do próprio Specky. Seus dados ficam locais.

UM PIPELINE, TRÊS PONTOS DE PARTIDA

O pipeline é o mesmo. O que muda é onde você começa.

GREENFIELD

Começar do zero

Uma aplicação nova, sem código existente. Init, discover com sete perguntas, specify, design e entrega.

BROWNFIELD

Adicionar a código existente

Scan da codebase primeiro. As perguntas de discovery então carregam seu stack real como contexto, e a checagem de drift mantém código e spec alinhados.

MODERNIZAÇÃO

Avaliar e atualizar o legado

Scan, import em lote de docs e transcrições antigas, checagem de compliance, e geração de artefatos de migração com runbooks de rollback.

PARTE

V

A decisão.

Por que uma camada de SDD determinística importa para uma empresa, e por onde começar na segunda.

FEITO PARA ADOÇÃO ENTERPRISE

Governança e uma superfície de ataque mínima, validadas por testes, não por marketing.

FRAMEWORKS DE COMPLIANCE

HIPAA	Saúde, 6 controles
SOC 2	SaaS e cloud, 6 controles
GDPR	Dados na UE, 5 controles
PCI-DSS	Cartão, 6 controles
ISO 27001	Gestão de segurança, 6 controles

POSTURA DE SEGURANÇA

- Só duas dependências de runtime. Superfície de ataque mínima.
- Zero requisições de rede de saída. Todo dado fica local.
- Sem eval ou execução dinâmica de código. Path traversal bloqueado.
- Validação estrita de schema em toda entrada de ferramenta.
- **292 testes unitários, 89 por cento de cobertura, exigidos em todo push.**

A SDD MARKET ANALYSIS 2026

Os fluxos de SDD estão empatados. A plataforma enterprise em volta da camada é o diferencial.

GITHUB E MICROSOFT

4.25

Nota de prontidão enterprise, de 5,0

PARIDADE DE FLUXO

4 : 4

A PLATAFORMA DECIDE

KIRO, AWS

2.85

Nota de prontidão enterprise, de 5,0

Segurança, governança, liberdade multi-modelo e compliance são onde a distância se abre. O Specky é o motor aberto que torna essa camada real.

PARA ONDE O SPECKY VAI

Entregando hoje, planejado a seguir, e no horizonte.

V3.0, ATUAL	Respostas interativas enriquecidas, 17 tipos de diagrama de engenharia de software, um template de design C4 de 12 seções e enforcement reforçado em cada transição de fase.	entregando
V2.4, PLANEJADO	Autenticação HTTP por token, observabilidade com OpenTelemetry, templates de spec internacionalizados e feedback de shrinking por IA na refinação da spec.	a seguir
V3.1 EM DIANTE	Controle de acesso por papel, log de auditoria persistente, rate limiting, SSO e SAML, workspaces multi-tenant e um dashboard de qualidade de especificação.	horizonte

O VOCABULÁRIO

Seis termos que destravam todo o resto desta talk.

SDD	Spec-Driven Development	Escreva e valide a especificação primeiro, depois deixe a IA construir contra ela.
EARS	Easy Approach to Requirements Syntax	Seis formas de frase que tornam cada requisito testável, ou rejeitado.
MCP	Model Context Protocol	O padrão aberto que deixa o Specky rodar em qualquer IDE com IA e rotear para outras ferramentas.
Constituição	O livro de regras do projeto	Princípios, stack e portões de qualidade que a IA honra em cada fase.
Drift	Spec e código divergem	Quando o código deixa de bater com a spec. A verificação pega isso e te leva de volta.
Portão LGTM	O checkpoint humano	Depois de uma fase principal o pipeline pausa até alguém digitar LGTM para prosseguir.

MÃO NA MASSA, HOJE

Cinco passos da instalação à sua primeira spec validada.

STEP 01	Adicione o Specky à sua IDE com IA com um comando npx. Sem editor novo, sem migração.
STEP 02	Peça à IA para inicializar um projeto. Ela escreve a constituição com você.
STEP 03	Responda as sete perguntas de discovery. Sua ideia de feature vira intenção estruturada.
STEP 04	Deixe ela escrever a especificação, então leia e responda LGTM no portão de revisão.
STEP 05	Siga por design e tarefas. Você tem agora uma spec rastreável, antes de qualquer código.

Dez minutos para uma especificação validada. A primeira vez é a mais lenta, e ainda é mais rápida que o retrabalho que você evita.

ENCERRAMENTO

Especificações recompensam disciplina de engenharia.

O nome divertido, o motor sério. A IA é a operadora. O Specky é o motor.

CONTATO

Paula Silva
Software Global Black Belt

paulasilva@microsoft.com

COMECE AGORA

Open source, MIT
Specky · open source

`npm: specky-sdd`

INSTALE O SPECKY, COMECE NA SEGUNDA

```
# GitHub Copilot CLI, inside the repo
copilot mcp add specky -- npx -y specky-sdd

# VS Code with GitHub Copilot
# adicione specky ao .vscode/mcp.json

# then, in the AI chat
> Inicialize um projeto Specky
> e me ajude a definir o escopo
```