SPEC-DRIVEN DEVELOPMENT

Specky convierte la disciplina de especificaciones en un motor de enforcement programable.

Un servidor MCP open source con 53 herramientas validadas y un pipeline de 10 fases que lleva cualquier entrada hasta el código en producción, de forma determinista, en cualquier IDE con IA.

AUTORA
Paula Silva

CARGO
Software Global Black Belt

DURACIÓN
45 a 60 minutos

FECHA
2026-06-01

AGENDA

Cinco partes, un argumento.

PART I	El problema, por qué el vibe coding genera código rápido y equipos lentos
PART II	El motor, qué es Specky y cómo fuerza el determinismo
PART III	El pipeline, diez fases, requisitos EARS y puertas de revisión humana
PART IV	El ecosistema, cualquier entrada, cualquier IDE, sin vendor lock-in
PART V	La decisión, preparación enterprise y por dónde empezar el lunes

PARTE

I

El problema.

La IA escribe código en segundos. Pero rápido no es lo mismo que correcto, y nadie escribió qué significaba correcto.

EL COSTO DE SALTARSE LA ESPECIFICACIÓN

40%

del tiempo de ingeniería se convierte en retrabajo cuando los requisitos nunca se escribieron.

Le piden al asistente un sistema de login. Genera código al instante, se salta los requisitos, adivina la arquitectura y entrega algo que funciona pero no corresponde a la intención de nadie. No hay forma de verificar el código contra una intención que nunca se capturó.

CÓMO LO LLAMAN		QUÉ PRODUCE ESO
vibe coding	→	código sin intención

DOS FORMAS DE CONSTRUIR CON IA

La corrección no es un prompt mejor. Es un pipeline validado entre intención y código.

VIBE CODING

Código por vibra

- El modelo genera código directo de una frase.
- Los requisitos nunca se capturan, los criterios de aceptación nunca se definen.
- No hay forma de verificar el resultado contra la intención original.
- Funciona en la demo, se rompe en la revisión y en producción.

DESARROLLO DETERMINISTA

Código a partir de specs validadas

- Una especificación estructurada describe qué debe hacer el sistema, primero.
- Cada requisito se valida antes de escribir una línea de código.
- Spec, diseño y tareas quedan alineados y trazables.
- La IA es la operadora. Specky es el motor.

PARTE

II

El motor.

Specky reimplementa la metodología Spec-Kit como 53 herramientas con enforcement, con validación programática en lugar de buenas intenciones.

QUÉ ES SPECKY, EN TRES NÚMEROS

Un servidor MCP open source que canaliza la creatividad de la IA por un pipeline validado.

HERRAMIENTAS MCP VALIDADAS

53

En trece categorías: conversión de entrada, núcleo del pipeline, puertas de calidad, diagramas, infraestructura, pruebas y exportación.

FASES DEL PIPELINE

10

Una máquina de estados que bloquea el salto de fases. No saltan de Init a Design sin completar Specify antes.

IDES COMPATIBLES

cualquiera

Funciona en cualquier IDE con IA que hable MCP. Úsalo en VS Code con GitHub Copilot o en la GitHub Copilot CLI. Un comando npx.

LAS 53 HERRAMIENTAS, ORGANIZADAS

Cincuenta y tres herramientas, trece categorías, agrupadas en las siete familias que de verdad usan.

01 · CONVERSIÓN DE ENTRADA

Convierte transcripciones, documentos, diseños y texto crudo en Markdown limpio que el pipeline lee.

02 · NÚCLEO DEL PIPELINE

Las diez herramientas de fase, de sdd_init a sdd_release, que mueven la máquina de estados adelante.

03 · PUERTAS DE CALIDAD

El validador EARS y el análisis cruzado que bloquean una fase hasta que el trabajo es coherente.

04 · DIAGRAMAS

Generadores de diecisiete tipos de diagrama de ingeniería de software, de C4 a flujos de secuencia.

05 · INFRAESTRUCTURA

Enrutamiento a Terraform y Docker para que la spec llegue a entornos reales, no solo a un documento.

06 · PRUEBAS

Herramientas que atan cada requisito a una prueba, para que la verificación pruebe que el código coincide.

07 · EXPORTACIÓN

Enrutamiento a GitHub, Azure DevOps y Jira, para que el plan caiga como issues y pull requests.

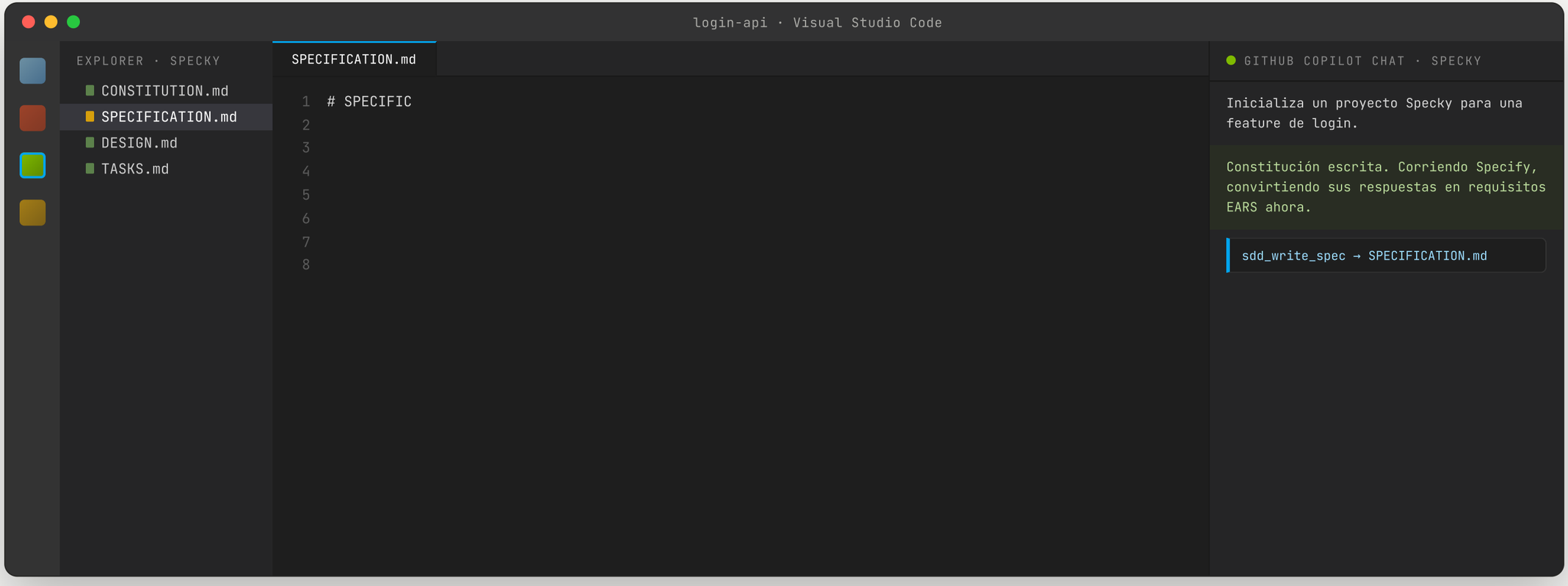
53

herramientas en total, cada una con validación estricta de entrada. Nunca las llaman a mano, el pipeline lo hace.

No memorizan herramientas. Avanzan por fases, y cada fase llama a las herramientas que necesita.

SIMULACIÓN · SPECKY DENTRO DE VS CODE

Vive en el editor que ya usan. Aquí, escribiendo una spec, en vivo.



Reconstrucción ilustrativa. El mismo GitHub Copilot Chat, los mismos archivos, los requisitos se validan mientras se escriben.

METODOLOGÍA Y MOTOR

Spec-Kit define qué hacer. Specky fuerza cómo hacerlo.

SPEC-KIT, LA METODOLOGÍA

Plantillas de prompt y definiciones de agentes

Notación EARS, fases con puertas, el modelo de constitución y plantillas en Markdown. La IA las lee e intenta seguirlas. Ligera, ideal para aprender Spec-Driven Development.

SPECKY, EL MOTOR

Herramientas que validan, fuerzan y generan

Una máquina de estados, un validador de EARS por regex y schemas Zod convierten la metodología en enforcement. Instalen Specky con un comando y toda la metodología Spec-Kit viene integrada.

No hace falta instalar Spec-Kit por separado. La metodología viene dentro del motor.

CUATRO MECANISMOS QUE HACEN REAL EL DETERMINISMO

El determinismo no es una promesa. Lo fuerza código que se niega a dejar el pipeline derivar.

MECANISMO 01

Máquina de estados

Enforcement de fases

- Diez fases obligatorias, sin saltos
- Los archivos obligatorios habilitan el avance

MECANISMO 02

Validador EARS

Calidad del requisito

- Cada requisito verificado contra 6 patrones
- Términos vagos como rápido o bueno se marcan

MECANISMO 03

Análisis cruzado de artefactos

Puntuación de alineación

- Spec, diseño y tareas verificados por consistencia
- Requisitos huérfanos marcados al instante

MECANISMO 04

Enrutamiento MCP-to-MCP

Sin vendor lock-in

- Genera JSON estructurado para cualquier cliente
- Enruta a GitHub, Azure, Jira, Terraform

PARTE

III

El pipeline.

Diez fases. Sin saltos. Revisión humana en cada puerta. Cada fase deja un artefacto trazable atrás.

DE LA INTENCIÓN AL RELEASE, UNA FASE A LA VEZ

La máquina de estados convierte una idea vaga en un recorrido documentado y reproducible.

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

Enmarcar el proyecto · Especificar y diseñar · Planificar y auditar · Construir, verificar, entregar

SIMULACIÓN · LA MÁQUINA DE ESTADOS CORRIENDO

Den iniciar, y las diez fases se ejecutan en orden, cada una dejando un artefacto.

01 ✓ **Init** CONSTITUTION.md	02 ✓ **Discover** 7 preguntas	03 ✓ **Specify** SPECIFICATION.md	04 ✓ **Clarify** árbol de decisión	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 En pausa tras Specify. La máquina no avanza a Design hasta que revisen y respondan LGTM.

Cada fase se pone verde solo cuando su artefacto existe y su puerta pasa. Saltarse es imposible por diseño.

LA SALIDA DE LA FASE 01, DE CERCA

Todo proyecto abre con una constitución. Esto es lo que fija.

La constitución es el reglamento del proyecto. La IA la lee antes de cada fase, así los mismos principios, stack y barras de calidad valen desde el primer requisito hasta el pull request final.

01 Principios que el código debe honrar, como accesibilidad y privacidad por defecto.

02 El stack técnico, para que la IA nunca adivine su lenguaje o framework.

03 Puertas de calidad, la barra a la que se somete cada fase posterior.

CONSTITUTION.MD

```
# Principles
- La accesibilidad no es opcional. WCAG 2.2 AA.
- Ningún dato personal sale del tenant.

# Stack
- TypeScript, React, PostgreSQL.
- Pruebas en Vitest. Lint con ESLint.

# Quality gates
- Cada requisito mapea a una prueba.
- Ninguna fase avanza con una pregunta abierta.
```

Muestra ilustrativa. Escríbanla una vez, y cada fase hereda estas reglas automáticamente.

NOTACIÓN EARS, LOS SEIS PATRONES DE REQUISITO

Cada requisito encaja en una de seis formas. El validador rechaza lo que no encaja.

Ubicuo	El sistema debe...	El sistema debe cifrar todo dato en reposo. Siempre verdadero, sin disparador.
Por evento	Cuando [evento], el sistema debe...	Cuando un usuario envía el login, el sistema debe validar las credenciales.
Por estado	Mientras [estado], el sistema debe...	Mientras está offline, el sistema debe encolar las solicitudes.
Opcional	Donde [condición], el sistema debe...	Donde 2FA está activo, el sistema debe exigir un código de un solo uso.
No deseado	Si [condición], entonces el sistema debe...	Si la sesión expira, entonces el sistema debe redirigir al login.
Complejo	Mientras [estado], cuando [evento]...	Mientras está en mantenimiento, cuando llega una solicitud, el sistema debe encolarla.

EARS EN LA PRÁCTICA

El mismo requisito, rechazado y aceptado.

RECHAZADO POR EL VALIDADOR

"El login debe ser rápido y seguro."

- Rápido no nombra número, así ninguna prueba lo puede verificar.
- Seguro no tiene disparador ni condición.
- No encaja en ninguna de las seis formas EARS.

ACEPTADO POR EL VALIDADOR

"Cuando un usuario envía el login, el sistema debe responder en 500 ms. Si las credenciales fallan tres veces, entonces el sistema debe bloquear la cuenta por 15 minutos."

- Forma por evento, con un 500 ms medible.
- Una forma de comportamiento no deseado para el bloqueo.
- Cada cláusula es ahora una prueba que pueden escribir.

El validador no los hace mejores redactores. Se niega a dejar que una frase no testeable se vuelva requisito.

UNA FEATURE, POR EL PIPELINE

Vean una sola feature de login ir de una frase a un pull request.

Discover 7 questions	Siete preguntas revelan lo no dicho: SSO o contraseña, política de bloqueo, duración de sesión, auditoría.
Specify SPECIFICATION.md	Cada respuesta se vuelve un requisito EARS: REQ-001 validar credenciales, REQ-002 bloquear tras tres fallos.
Design DESIGN.md	La arquitectura: un servicio de auth, un rate limiter, un store de sesión, cada uno atado a un requisito.
Tasks TASKS.md	Un plan ordenado y verificable. Cada tarea apunta al requisito que satisface y a la prueba que necesita.
Verify y release PR + work items	La verificación confirma que el código coincide con la spec, y abre un pull request con los work items adjuntos.

Una frase entra, un rastro trazable sale. Cada caja de arriba es un archivo real que un colega leerá el año que viene.

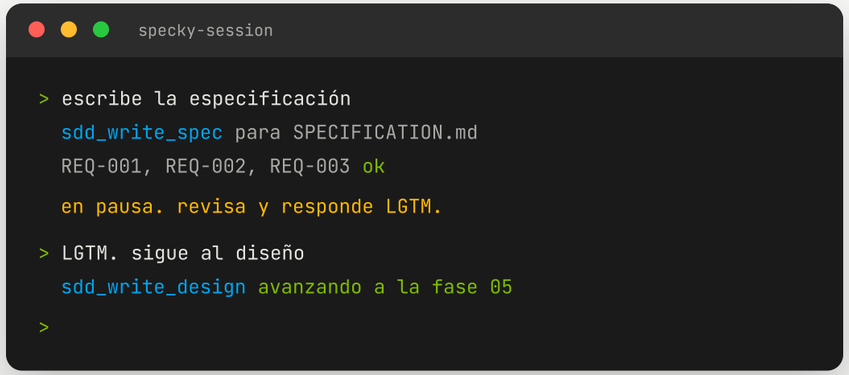
EL HUMANO PERMANECE EN EL CIRCUITO

Después de cada fase principal, el pipeline pausa y espera que ustedes escriban LGTM.

Specify, Design y Tasks terminan cada uno en una puerta de revisión. La IA no avanza sola. Si la verificación detecta después una divergencia entre la spec y el código, Specky los lleva de vuelta a corregir antes de seguir.

3 puertas de revisión

0 fases saltadas



PARTE

IV

El ecosistema.

Cualquier entrada se vuelve spec. Cualquier IDE ejecuta. Specky enruta el trabajo a las herramientas que ya usan, sin lock-in.

SEIS FORMAS DE EMPEZAR

Lo que ya tienen se vuelve el punto de partida de una especificación.

01 · LENGUAJE NATURAL

Un prompt en el chat

Escriban la idea directo. sdd_init y sdd_discover la estructuran. Ideal para greenfield.

02 · TRANSCRIPCIÓN DE REUNIÓN

VTT, SRT, TXT, MD

Extrae decisiones, ítems de acción y requisitos brutos de Teams, Zoom o Meet.

03 · DOCUMENTOS

PDF, DOCX, PPTX

Importa RFPs, documentos de requisito y decks. Los convierte a Markdown y alimenta el pipeline.

04 · DISEÑO EN FIGMA

Diseño a spec

Lee componentes e interacciones vía Figma MCP. Ideal para trabajo design-first.

05 · SCAN DE CODEBASE

Contexto brownfield

Detecta lenguaje, framework y estructura antes de especificar una feature nueva.

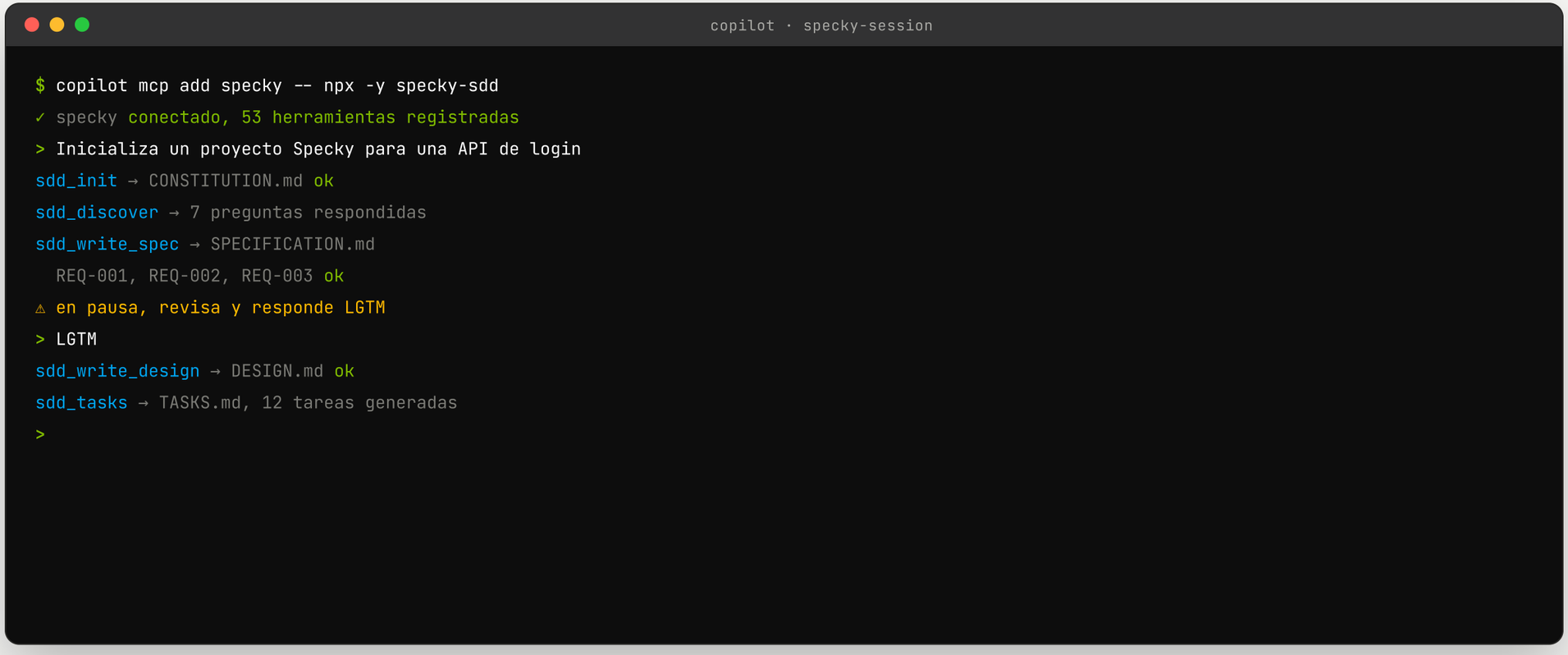
06 · TEXTO CRUDO

Peguen cualquier cosa

Sin archivo. Peguen un correo del cliente o notas y toda herramienta de import lo acepta directo.

SIMULACIÓN · LA MISMA CORRIDA EN LA GITHUB COPILOT CLI

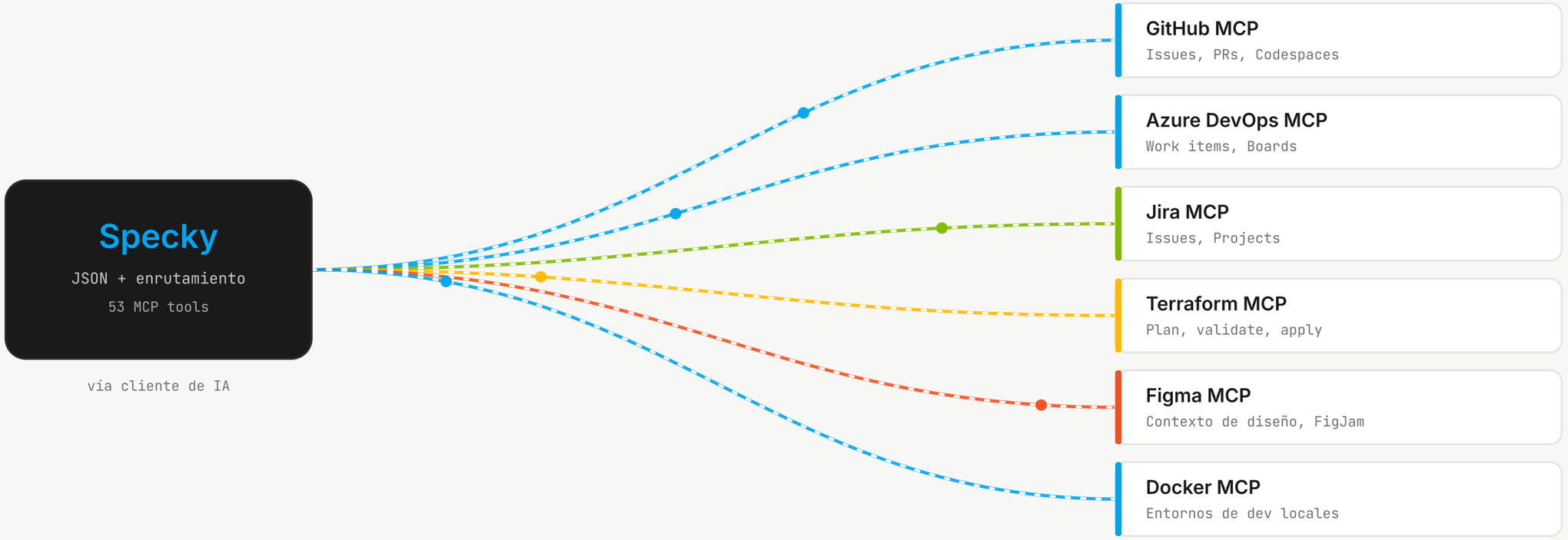
Sin IDE. La terminal corre el pipeline idéntico, línea por línea.



Reconstrucción ilustrativa. Las mismas herramientas, los mismos artefactos, la misma puerta LGTM. Solo cambió la superficie.

ARQUITECTURA MCP-TO-MCP

Specky emite JSON estructurado. Su cliente enruta a las herramientas que ya pagan.



Dos dependencias de runtime. Cero solicitudes de red salientes del propio Specky. Sus datos quedan locales.

UN PIPELINE, TRES PUNTOS DE PARTIDA

El pipeline es el mismo. Lo que cambia es dónde empiezan.

GREENFIELD

Empezar de cero

Una aplicación nueva, sin código existente. Init, discover con siete preguntas, specify, design y entrega.

BROWNFIELD

Agregar a código existente

Scan de la codebase primero. Las preguntas de discovery entonces cargan su stack real como contexto, y el chequeo de drift mantiene código y spec alineados.

MODERNIZACIÓN

Evaluar y actualizar el legado

Scan, import en lote de docs y transcripciones antiguas, chequeo de compliance, y generación de artefactos de migración con runbooks de rollback.

PARTE

V

La decisión.

Por qué una capa de SDD determinista importa para una empresa, y por dónde empezar el lunes.

HECHO PARA ADOPCIÓN ENTERPRISE

Gobernanza y una superficie de ataque mínima, validadas por pruebas, no por marketing.

FRAMEWORKS DE COMPLIANCE

HIPAA	Salud, 6 controles
SOC 2	SaaS y cloud, 6 controles
GDPR	Datos en la UE, 5 controles
PCI-DSS	Tarjeta, 6 controles
ISO 27001	Gestión de seguridad, 6 controles

POSTURA DE SEGURIDAD

- Solo dos dependencias de runtime. Superficie de ataque mínima.
- Cero solicitudes de red salientes. Todo dato queda local.
- Sin eval ni ejecución dinámica de código. Path traversal bloqueado.
- Validación estricta de schema en toda entrada de herramienta.
- **292 pruebas unitarias, 89 por ciento de cobertura, exigidas en todo push.**

EL SDD MARKET ANALYSIS 2026

Los flujos de SDD están empatados. La plataforma enterprise alrededor de la capa es el diferencial.

GITHUB Y MICROSOFT

4.25

Nota de preparación enterprise, de 5,0

PARIDAD DE FLUJO

4 : 4

LA PLATAFORMA DECIDE

KIRO, AWS

2.85

Nota de preparación enterprise, de 5,0

Seguridad, gobernanza, libertad multi-modelo y compliance son donde se abre la distancia. Specky es el motor abierto que hace real esa capa.

HACIA DÓNDE VA SPECKY

Entregando hoy, planificado a continuación, y en el horizonte.

V3.0, ACTUAL	Respuestas interactivas enriquecidas, 17 tipos de diagrama de ingeniería de software, una plantilla de diseño C4 de 12 secciones y enforcement reforzado en cada transición de fase.	entregando
V2.4, PLANIFICADO	Autenticación HTTP por token, observabilidad con OpenTelemetry, plantillas de spec internacionalizadas y feedback de shrinking por IA en la refinación de la spec.	a continuación
V3.1 EN ADELANTE	Control de acceso por rol, log de auditoría persistente, rate limiting, SSO y SAML, workspaces multi-tenant y un dashboard de calidad de especificación.	horizonte

EL VOCABULARIO

Seis términos que destraban todo lo demás de esta charla.

SDD	Spec-Driven Development	Escriban y validen la especificación primero, luego dejen que la IA construya contra ella.
EARS	Easy Approach to Requirements Syntax	Seis formas de frase que hacen cada requisito testeable, o rechazado.
MCP	Model Context Protocol	El estándar abierto que deja a Specky correr en cualquier IDE con IA y enrutar a otras herramientas.
Constitución	El reglamento del proyecto	Principios, stack y puertas de calidad que la IA honra en cada fase.
Drift	Spec y código divergen	Cuando el código deja de coincidir con la spec. La verificación lo detecta y los lleva de vuelta.
Puerta LGTM	El checkpoint humano	Tras una fase principal el pipeline pausa hasta que una persona escribe LGTM para seguir.

MANOS A LA OBRA, HOY

Cinco pasos de la instalación a su primera spec validada.

STEP 01	Agreguen Specky a su IDE con IA con un comando npx. Sin editor nuevo, sin migración.
STEP 02	Pidan a la IA que inicialice un proyecto. Escribe la constitución con ustedes.
STEP 03	Respondan las siete preguntas de discovery. Su idea de feature se vuelve intención estructurada.
STEP 04	Dejen que escriba la especificación, luego léanla y respondan LGTM en la puerta de revisión.
STEP 05	Sigan por diseño y tareas. Ahora tienen una spec trazable, antes de cualquier código.

Diez minutos a una especificación validada. La primera vez es la más lenta, y aún es más rápida que el retrabajo que evitan.

CIERRE

Las especificaciones recompensan la disciplina de ingeniería.

El nombre divertido, el motor serio. La IA es la operadora. Specky es el motor.

CONTACTO

Paula Silva
Software Global Black Belt

paulasilva@microsoft.com

EMPIECEN AHORA

Open source, MIT
Specky · open source

`npm: specky-sdd`

INSTALEN SPECKY, EMPIECEN EL LUNES

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

# VS Code with GitHub Copilot
# agreguen specky al .vscode/mcp.json

# then, in the AI chat
> Inicializa un proyecto Specky
> y ayúdame a definir el alcance
```