Spec-Driven Development

Dirigir agentes de IA con método, no a puro pálpito.

DANIEL PÉREZ

EL PLAN

Lo que vamos a hacer hoy

1. Qué es SDD
2. Por qué y cuándo usarlo
3. Glosario
4. El flujo, en crudo
5. Las dos herramientas
6. Práctica con el café
7. Cierre

HACKLAB ORIENTE

Qué es Spec-Driven Development

La idea, antes que cualquier herramienta.

01 · LA IDEA

La spec es el contrato

Antes le escribías el código a la máquina. Hoy se lo pides a un agente y él lo escribe. El problema es que quedas respondiendo por código que no alcanzas a leer completo.

SDD le da la vuelta. Lo importante pasa a ser la spec, el documento donde dices qué quieres. El código sale de ahí, y como tienes la spec, lo puedes revisar contra ella.

01 · OJO CON ESTO

Las herramientas son andamiaje

spec-kit y cc-sdd solo automatizan un flujo que ya existe sin ellas. Uno podría hacer SDD con puros archivos de texto.

Por eso hoy primero entendemos el flujo. Después vienen las herramientas.

HACKLAB ORIENTE

Por qué y cuándo usarlo

Y cuándo mejor no.

02 · LO QUE GANAS

Qué te da SDD

Intención escrita	No se pierde entre un prompt y otro.
Sabes cuándo quedó	Con criterios claros, no con el “creo que ya”.
Contexto guardado	Vive en archivos, no en la conversación.
Iteración ordenada	Un feature a la vez.
Artefactos que muestras	Sirve si trabajas con clientes.

02 · PERO…

No es para todo

Si es un arreglo chiquito o algo desechable, montar toda la spec cuesta más de lo que ayuda.

Regla fácil: si te molestaría que el agente lo entienda distinto a lo que querías, escribe la spec. Si lo arreglas con un prompt más, go vibe.

El patrón sano: **explorar suelto, y cuando va en serio, pasar a spec.**

HACKLAB ORIENTE

Glosario

La terminología necesaria.

HACKLAB ORIENTE

El flujo, en crudo

Cómo hacerlo sin herramientas

04 · EL FLUJO

Seis pasos, puro texto

1 Principios
Las reglas del proyecto. Van en el CLAUDE.md.

2 Spec
El qué y el porqué. Sin hablar de tecnología.

3 Clarificación
El agente pregunta lo ambiguo. Tú resuelves.

4 Plan
Ahora sí: stack, arquitectura, datos.

5 Tareas
El plan partido en pasos.

6 Implementación
Paso a paso, con la prueba primero.

Cada comando de las herramientas hace uno de estos pasos.

HACKLAB ORIENTE

spec-kit y cc-sdd

Qué automatiza cada una.

05 · LAS DOS

De dónde vienen

spec-kit

De GitHub. Historias de usuario con Given/When/Then. Los artefactos van en archivos separados. Necesita uv y Python.

cc-sdd

De gotalab, inspirado en Kiro. Notación EARS. Diseño por fronteras. Solo necesita Node.

05 · LADO A LADO

Las diferencias que importan

	spec-kit	cc-sdd
Principios	Constitución, paso aparte	Steering, persistente
Requisitos	Given / When / Then	EARS
Clarificación	Fase con comando	En las compuertas
Diseño	En varios archivos	Concentrado, por fronteras
Implementación	El plan de una	Tarea por tarea, con revisor
Iteración	Le recuerdas el contexto	El steering lo recuerda

HACKLAB ORIENTE

Construyamos algo

El mismo proyecto en las dos herramientas.

06 · SETUP

Instalación de cc-sdd y spec-kit

cafe ~ setup

```
$ mkdir cafe-ccsdd
$ cd cafe-ccsdd
$ git init
$ npx cc-sdd@latest --claude-skills --lang es
$ claude
```

cafe ~ setup

```
$ mkdir cafe-speckit
$ cd cafe-speckit
$ git init
$ uv tool install specify-cli —from
git+https://github.com/github/spec-
kit.git@v0.12.4
$ specify self check
$ specify init . --integration claude
$ claude
```

06 · LOS COMANDOS

El flujo, comando por comando

cc-sdd

```
/kiro-discovery
/kiro-spec-init
/kiro-spec-requirements
(clarificas en la compuerta)
/kiro-spec-design
/kiro-spec-tasks
/kiro-impl
```

spec-kit

```
/speckit-constitution
/speckit-specify
/speckit-clarify
/speckit-plan
/speckit-analyze
/speckit-tasks
/speckit-implement
```

HACKLAB ORIENTE

Preguntas?

Gracias