DESARROLLO AGÉNTICO Y SPEC PROGRAMMING

Glosario

El vocabulario de Spec-Driven Development que usaremos hoy. Tenlo a mano durante los ejercicios.

A · CONCEPTO

< Spec-Driven Development (SDD)
Metodología donde la especificación es la fuente de verdad y el código, un artefacto generado y verificable a partir de ella. "La spec es el contrato".

< Harness
El andamiaje (CLI, comandos, estructura de carpetas) que envuelve la disciplina de SDD. spec-kit y cc-sdd son harnesses; la disciplina existe sin ellos.

< Vibe coding
Guiar al agente con prompts sueltos, sin spec. Legítimo y ágil para explorar o prototipar; se queda corto cuando la intención debe preservarse y el proyecto va a iterar.

< spec-anchored
Nivel de rigor recomendado: la spec guía, pero el código sigue siendo la fuente de verdad y las pruebas hacen cumplir la spec.

B · EL FLUJO Y SUS ARTEFACTOS

< Constitución
Principios no negociables del proyecto (calidad, testing, arquitectura, UX) que restringen todas las fases. En Claude Code suele vivir en `CLAUDE.md`.

< Spec (especificación)
El qué y el porqué de un feature: problema, usuarios, comportamiento y criterios de aceptación — deliberadamente sin hablar de stack.

< Criterios de aceptación
Condiciones concretas y verificables que definen "terminado". Si no se pueden comprobar, no sirven.

< Clarificación
La fase que casi todos saltan: el agente lista ambigüedades, huecos y supuestos, y tú los resuelves antes de diseñar. Donde más retrabajo se evita.

< Plan
El cómo: stack, arquitectura, modelo de datos y contratos. Conecta los principios con las decisiones técnicas.

< Tareas
El plan descompuesto en una lista ordenada por dependencias, marcando lo que corre en paralelo.

< Implementación
Ejecutar tarea por tarea con spec, plan y principios en contexto, idealmente con pruebas primero (TDD).

< Compuerta (gate)
Revisión humana al cierre de cada fase: apruebas el artefacto antes de alimentar el siguiente. La disciplina central de SDD.

C · NOTACIONES DE REQUISITOS

< EARS
Easy Approach to Requirements Syntax. Requisitos sin ambigüedad en cinco patrones ("El sistema deberá...", "Cuando... deberá...", "Si... entonces deberá...", "Mientras...", "Donde..."). La usa cc-sdd.

< Given / When / Then
Criterios de aceptación estilo BDD: dado un contexto, cuando ocurre algo, entonces el resultado esperado. Los usa spec-kit con user stories.

D · CONCEPTOS DE LAS HERRAMIENTAS

< Steering
En cc-sdd, el conocimiento persistente del proyecto (producto, tecnología, estructura) que el agente consulta sin que se lo repitas.

< Boundary-first
Diseño (cc-sdd) que hace explícitas las fronteras del código: qué toca cada feature de lo ya construido.

< Revisor separado
Un agente distinto al que implementa verifica el trabajo, para que nadie "califique su propia tarea". La mejor práctica más desaprovechada de SDD.

< Greenfield / brownfield
Proyecto nuevo desde cero vs. agregar sobre un sistema existente. La disciplina de iteración cambia entre uno y otro.

< spec-kit & cc-sdd
Las dos herramientas del taller. spec-kit (GitHub): user stories + Given/When/Then, artefactos modulares. cc-sdd (gotalab, inspirado en Kiro): EARS, boundary-first, ciclo autónomo con revisor.