

Ingeniería de Bucles y el Coste Real de la Autonomía

Diseñando sistemas de agentes desde el script de Bash hasta la integración nativa.

Basado en el análisis de Cristian Palau (Nov 2026)

Gemini Notebook

El misterio de los 505.730 tokens adicionales

Un solo prompt

82.695 tokens



7.1x Sobrecoste por el mismo resultado exacto

Mismo trabajo en bucle

588.425 tokens

Tres tareas pequeñas en un repositorio de Python. Ambos métodos pasaron los tests y resolvieron la trampa del redondeo. ¿Por qué el sistema autónomo quemó siete veces más recursos?

Gemini Notebook

El becario disciplinado pero sin memoria

1. Agente (El Becario)

Claude Code / Codex. Trabaja en una sola tarea por vuelta.



2. Contexto Limpio (La Amnesia)

El entorno se resetea. No recuerda absolutamente nada de la sesión anterior.



3. Memoria en Disco (La Libreta)

Única fuente de verdad superviviente. Archivos prd.json (plan) y progress.txt (registro).



4. Verificador (La Revisión)

Pruebas automatizadas que dictan si la tarea está completada o requiere otra vuelta.



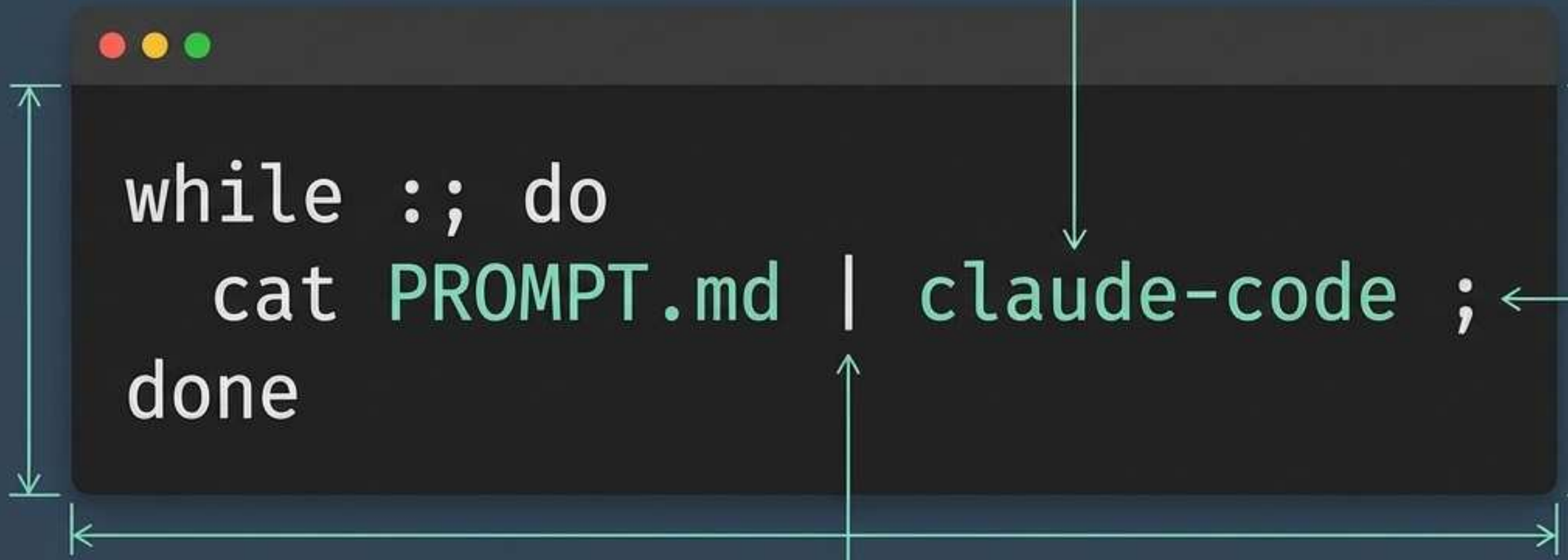
Nota: El Loop Engineering consiste en diseñar este sistema cerrado en lugar de escribir prompts individuales.

Gemini Notebook

Ralph Wiggum y la fuerza bruta del while

1. Una tarea por vuelta

El prompt fuerza al modelo a elegir solo la historia de mayor prioridad en el JSON.



2. Aislamiento

La tubería levanta una sesión totalmente nueva por iteración. Contexto inmaculado.

3. Persistencia

Cada éxito se graba en el historial de Git antes de la siguiente vuelta.

Gemini Notebook

Prueba de estrés: La calculadora de facturas

Repositorio Base

- Target: factura.py (Vacío al inicio)
- Plan: prd.json (3 Historias de Usuario: IVA, Descuentos, Total)
- Verificador: test_factura.py (7 tests que fallan inicialmente)

La Trampa Técnica

El error de coma flotante en Python. round(2.675, 2) devuelve 2.67, no 2.68.

Requiere usar el módulo decimal.

>_ Resultados del experimento:



El agente resolvió la trampa y los 7 tests usando las 3 metodologías:

- Prompt único
- Bucle Ralph
- Comando /goal

Gemini Notebook

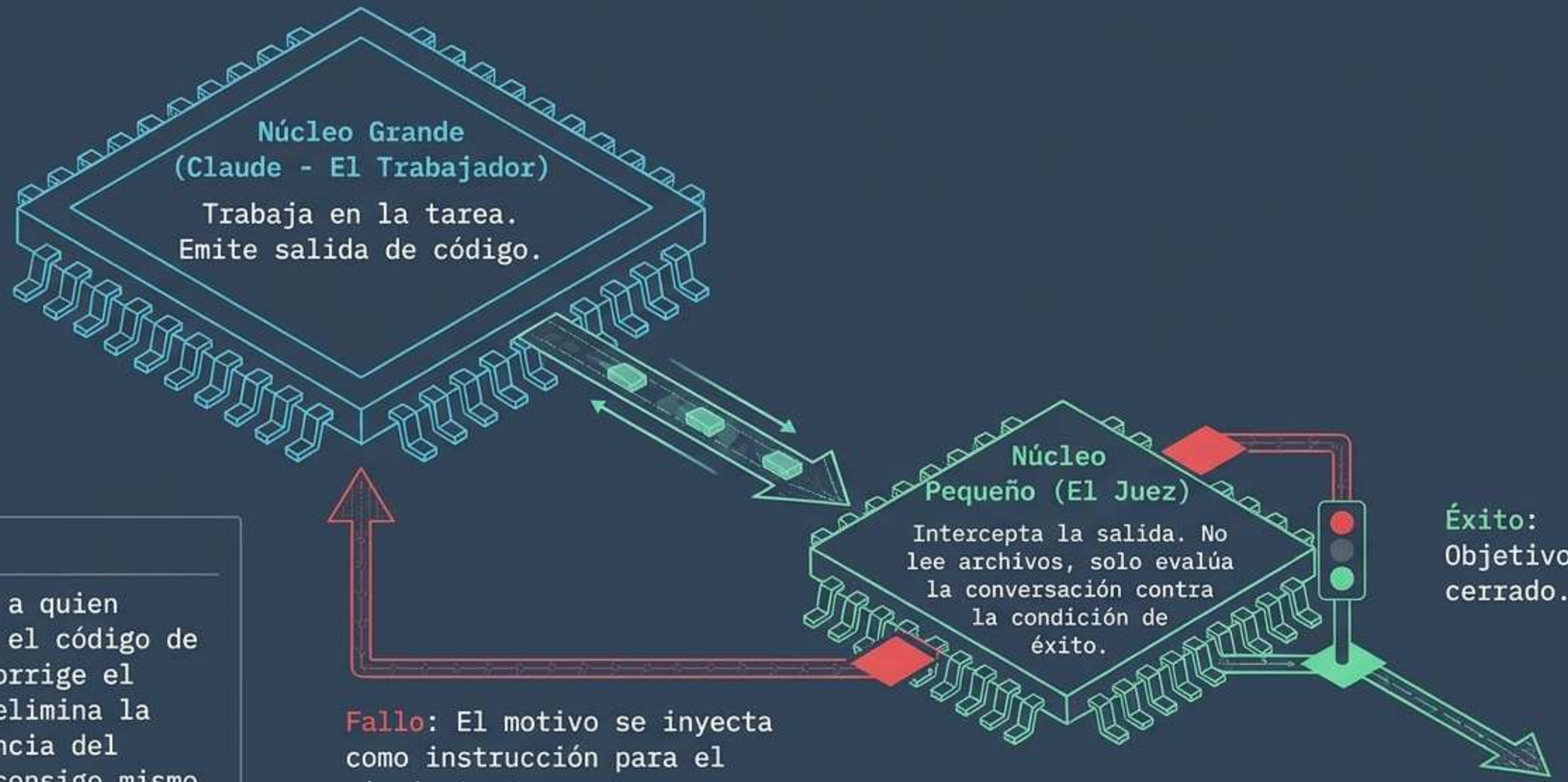
El peaje oculto del contexto inmaculado



Ralph previene la degradación cognitiva (Context Rot), pero el precio de levantar una sesión limpia en cada vuelta es releer las instrucciones del sistema, las herramientas y el proyecto una y otra vez.

Gemini Notebook

El juez asimétrico: La arquitectura de /goal



Separar a quien escribe el código de quien corrige el examen elimina la indulgencia del modelo consigo mismo.

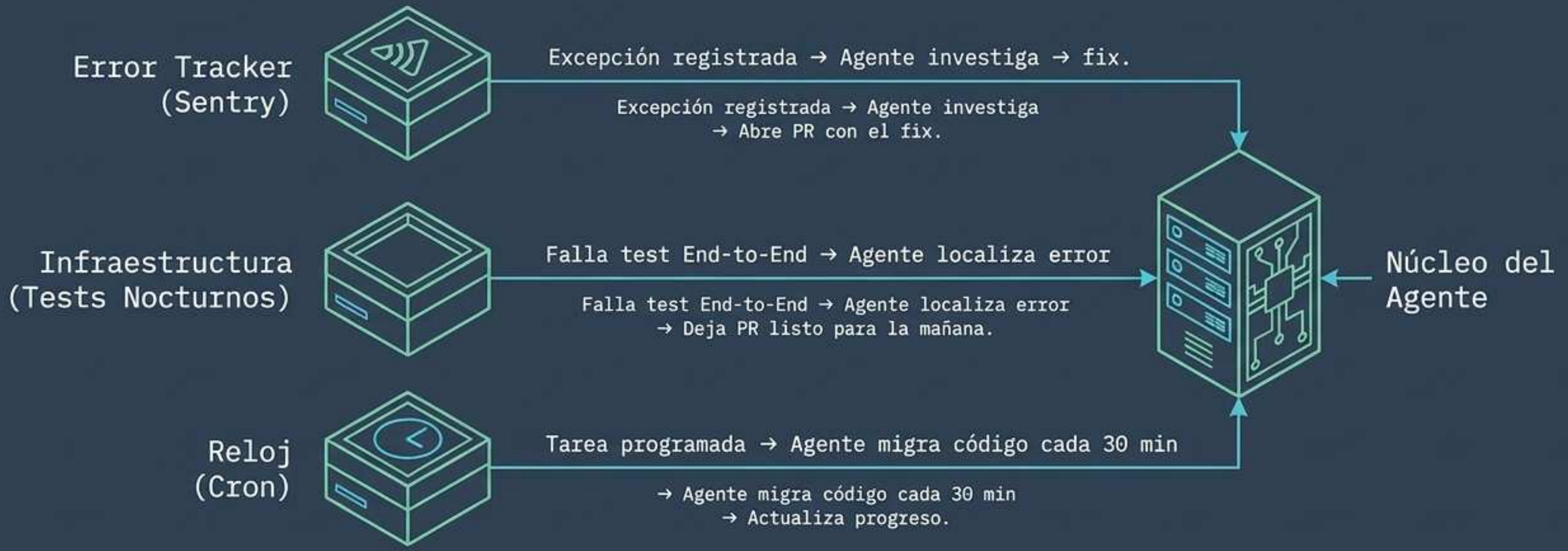
Gemini Notebook

Anatomía comparada de la autonomía

	Bucle Ralph	/goal	/loop
Disparador	Script manual	Fin del turno anterior	Temporizador (Cron)
Condición de Parada	Escribir <promise>COMPLETE</promise>	Evaluador aprueba la condición	Intervención manual
Estado del Contexto	Limpio en cada iteración	Sesión acumulativa	Sesión acumulativa
Caso de Uso Óptimo	Trabajo largo multidimensional	Tarea con final comprobable empíricamente	Vigilar sistemas en constante cambio (CI/CD)

Gemini Notebook

Disparadores: Autonomía fuera del portátil



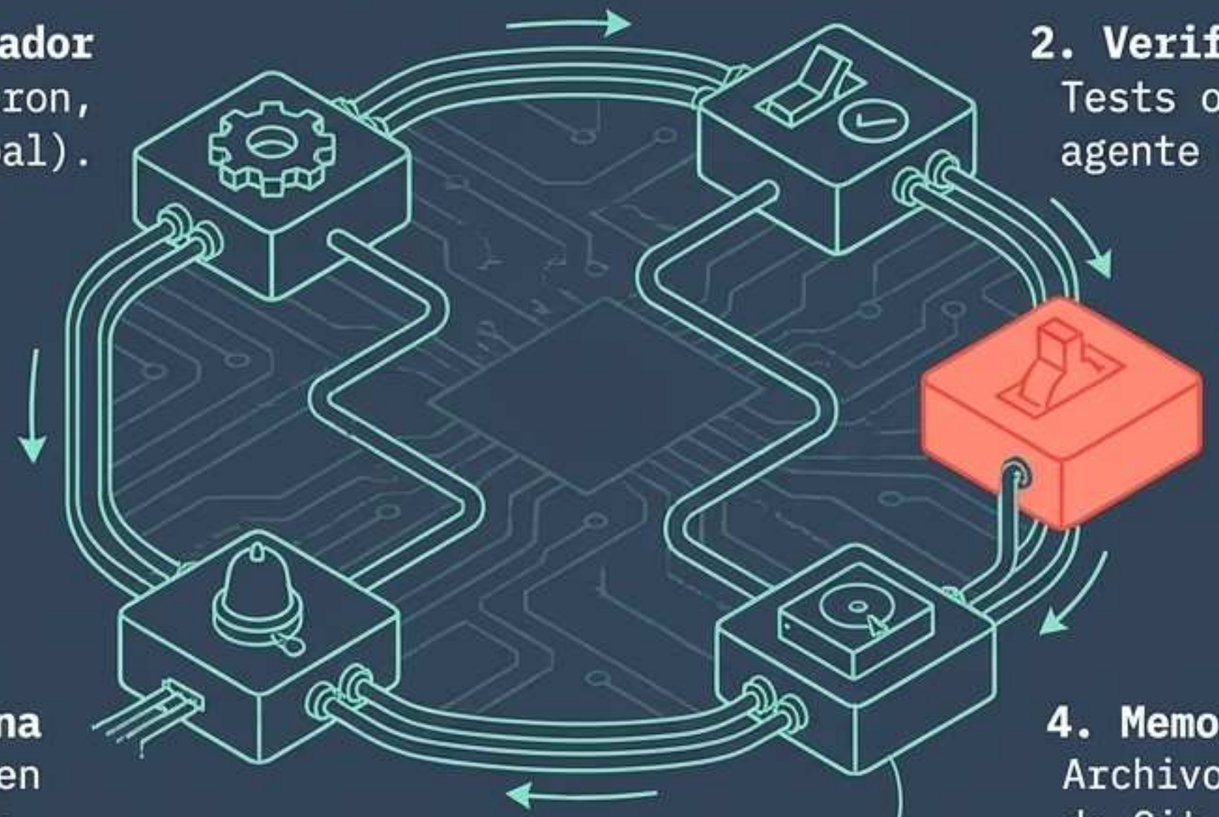
Nota: El cron existe desde los 70s. La revolución es que el paso intermedio ya no es un script fijo, sino un agente que deduce y adapta.

Gemini Notebook

Las cinco piezas que evitan el desbocamiento

1. El Disparador
Inicia el ciclo (cron, evento o /goal).

2. Verificador Independiente
Tests o linter. Nunca el mismo agente que programó.



3. El Freno / Presupuesto
Tope duro de vueltas, tokens (ej. 20 turnos) o tiempo.

5. Notificación Humana
Alerta al operador en caso de éxito o bloqueo.

4. Memoria en Disco
Archivo de progreso y commits de Git (no en memoria volátil).

Sandbox Extra: Git Worktree para evitar colisiones de agentes.

Gemini Notebook

La deriva, la deuda y el Tokenmaxxing



Permisos Excesivos

Una prueba sin permisos automáticos chocó con bash, forzando al modelo a buscar atajos creativos. El gasto saltó de 158k a 411.271 tokens por la fricción.



La Deriva (Drift)

En ejecuciones largas sin guías estrictas, el agente optimiza código periférico que 'parece correcto' pero no avanza el objetivo principal.



Deuda de Comprensión

A medida que el bucle escupe código no supervisado a gran velocidad, la brecha entre el repositorio y el modelo mental humano se vuelve inmanejable.

Gemini Notebook

No son bucles, es Ingeniería de Contexto

Chat Tradicional / Sesión Única

Metodología Loop



El Loop Engineering es simplemente Ingeniería de Contexto con otro nombre. Tanto Ralph como /goal son válvulas diseñadas con un solo propósito:elegir el conjunto más pequeño de información que dirija al modelo hacia la solución, minimizando la degradación cognitiva.

Gemini Notebook

El Blueprint de Decisión: Cuándo delegar el control

Situación	Solución
Cabe en una sesión y tiene tests listos.	Un buen prompt. (Más rápido, inmensamente más barato).
Tiene un final comprobable, pero el número de intentos es impredecible.	Comando /goal (Con un tope estricto de turnos).
Tarea masiva, requiere muchas piezas pequeñas y no cabe en una sola sesión.	Bucle Ralph (Con PRD en JSON, progreso en disco y Git).
Hay que vigilar un sistema en constante mutación (Colas, Despliegues, CI).	Comando /loop o Cron con Agente.

Mejorar el código sin una métrica de verificación no requiere un bucle, requiere definir primero la condición.

Gemini Notebook