

Ingeniería de Software Agéntica en VS Code

Arquitecturas, Protocolos y Patrones de Implementación



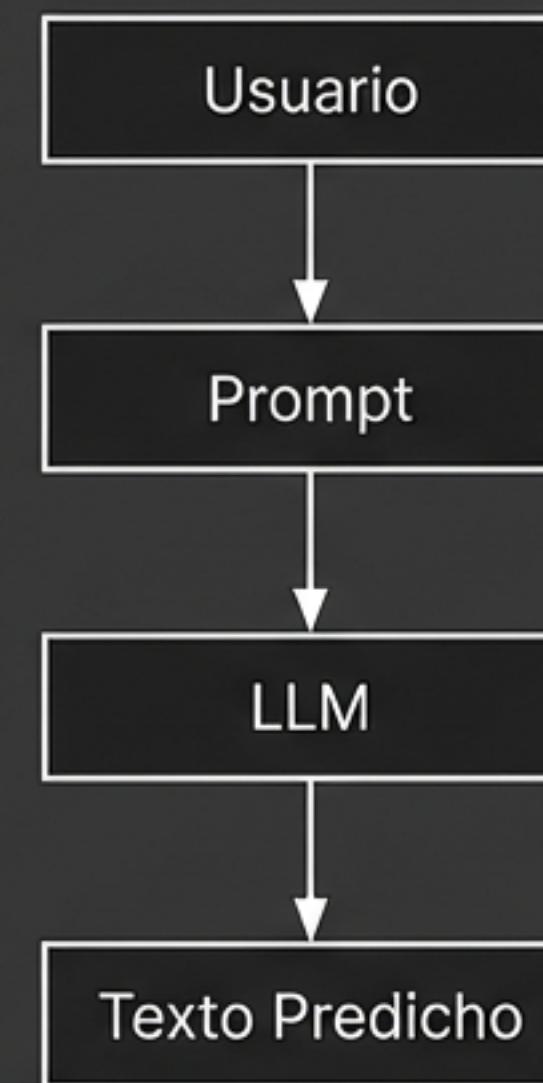
Basado en arquitecturas de Sistemas Multi-Agente (MAS)
y Protocolo de Contexto de Modelo (MCP)

v1.106+

NotebookLM

Del Prompt Lineal al Bucle Recursivo

INTERACCIÓN PASIVA (CHAT)



Predicción de texto estática.

INTERACCIÓN AGÉNTICA (REACT)



Insight Clave: El modelo no responde de inmediato; entra en un ciclo de resolución de problemas.

La Falacia del Agente Monolítico

Un solo agente se ve abrumado por la saturación de contexto y la ambigüedad.
La solución es la descomposición del sistema en roles especializados.

Característica	Agente Monolítico	Sistema Multi-Agente (MAS)
Contexto	✗ Ventana saturada, pérdida de foco	✓ Carga cognitiva distribuida por rol
Fiabilidad	✗ Propenso a alucinaciones	✓ Validación cruzada entre agentes
Ejecución	✗ Secuencial rígida	✓ Escalabilidad modular y paralela

“Dejar de usar la IA como una máquina de escribir inteligente y empezar a diseñarla como un compañero de equipo.”

```
# Agent1.process()  
# Agent2.validate()  
# MAS.orchestrate()
```

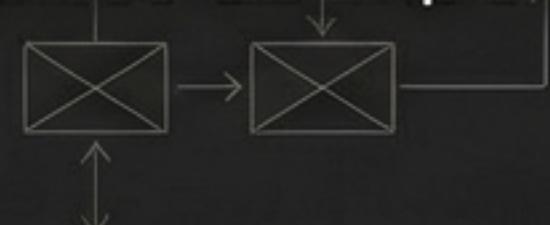
Rol 1: El Cerebro Estratégico (Planner)

Separando el 'Qué' (Arquitectura) del 'Cómo' (Código)

 **Modelo:** Razonamiento alto
(o1, Claude 3.5 Sonnet 'thinking')

 **Output:** JSON o Grafo de Tareas (No código final)

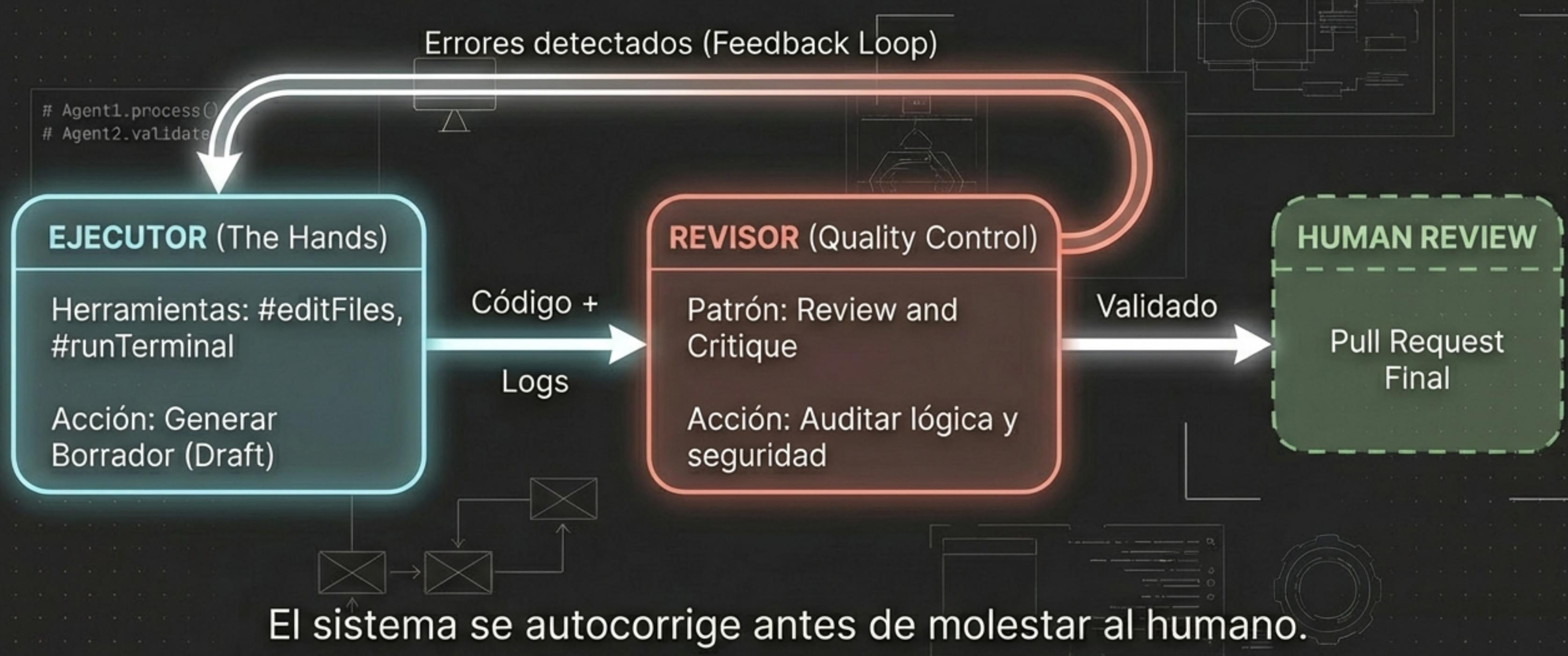
 **Objetivo:** Identificar dependencias y conflictos antes de implementar



```
{ } plan.json ×  
{  
  "goal": "Refactor auth system",  
  "steps": [  
    {  
      "id": 1,  
      "action": "analyze_dependencies",  
      "target": "src/auth/login.ts"  
    },  
    {  
      "id": 2,  
      "action": "create_interface",  
      "status": "pending"  
    }  
  "constraints": ["No breaking changes", "Update tests"]  
}
```

Roles 2 y 3: Ejecución y Crítica

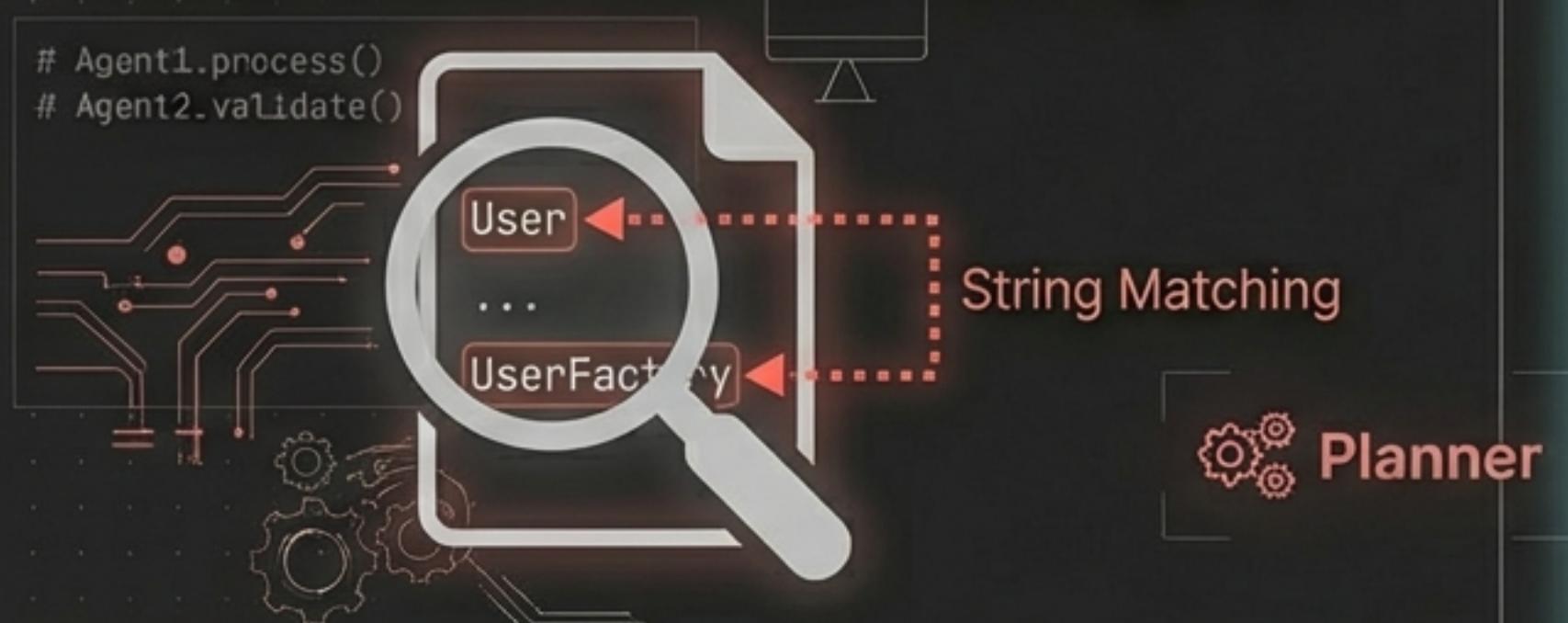
El ciclo de Autocorrección



Infraestructura: El Contexto es Rey (LSP vs. Grep)

Transformando texto plano en símbolos con significado.

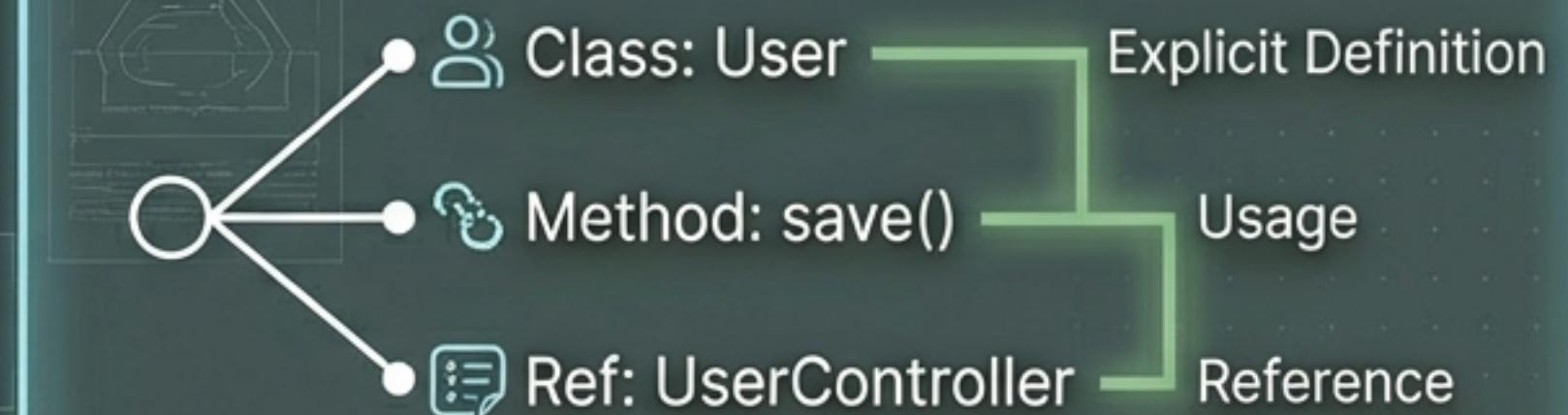
Búsqueda de Texto (Grep/Regex)



- Coincidencias literales
- Adivina referencias
- Alta alucinación en bases de código grandes

Convierte al agente de un autocompletador en un desarrollador junior con un mapa.

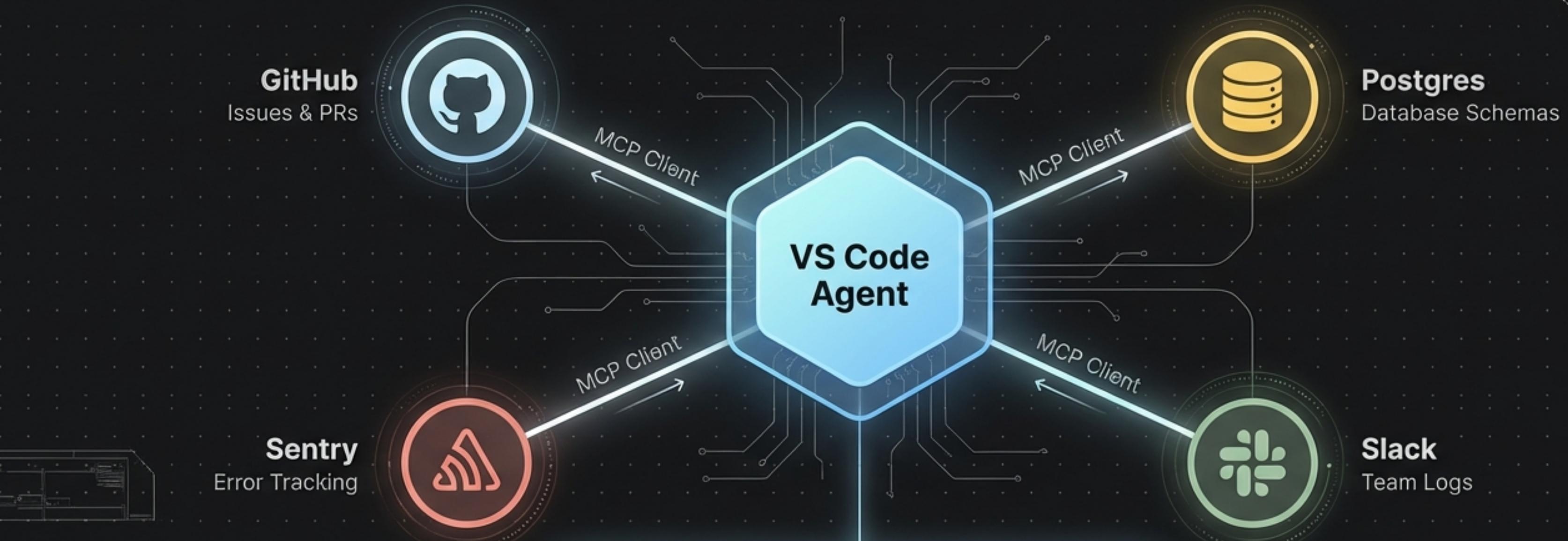
Comprensión Simbólica (LSP)



- Mapa determinista del territorio
- Sabe exactamente dónde está la definición
- Cero coste de tokens para navegación precisa

Conectividad Externa: Model Context Protocol (MCP)

Un estándar abierto para conectar asistentes de IA a sistemas donde viven los datos.



Desacopla la lógica del agente de la herramienta.
Escribe el conector una vez, úsalo en cualquier
cliente (Claude, Cursor, VS Code).

El Peligro del 'Over-tooling'

Gestión de Tokens y Contaminación de Contexto

Coste Oculto de Definiciones:

```
# Agent1.process()  
# Agent2.validate()
```

+40k tokens

26k tokens

GitHub Server

Slack + Jira

Resultado: Contaminación del contexto
("Needle in the haystack"). Aumenta
latencia, degrada razonamiento.



Estrategias de Mitigación:

1. Carga Diferida (Tool Search)

El agente busca herramientas bajo demanda. Reducción del 85% de tokens.

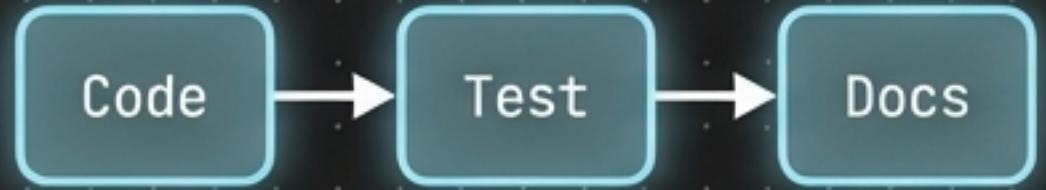


2. Llamadas Programáticas

Ejecutar scripts aislados en lugar de volcar logs crudos al chat.

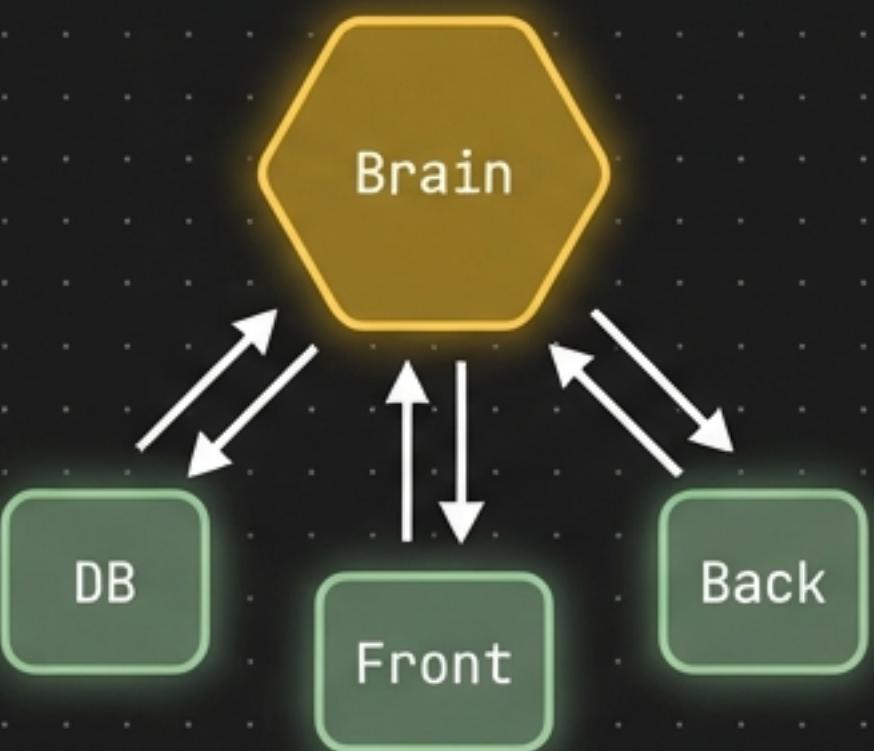
Patrones de Colaboración Multi-Agente

SECUENCIAL



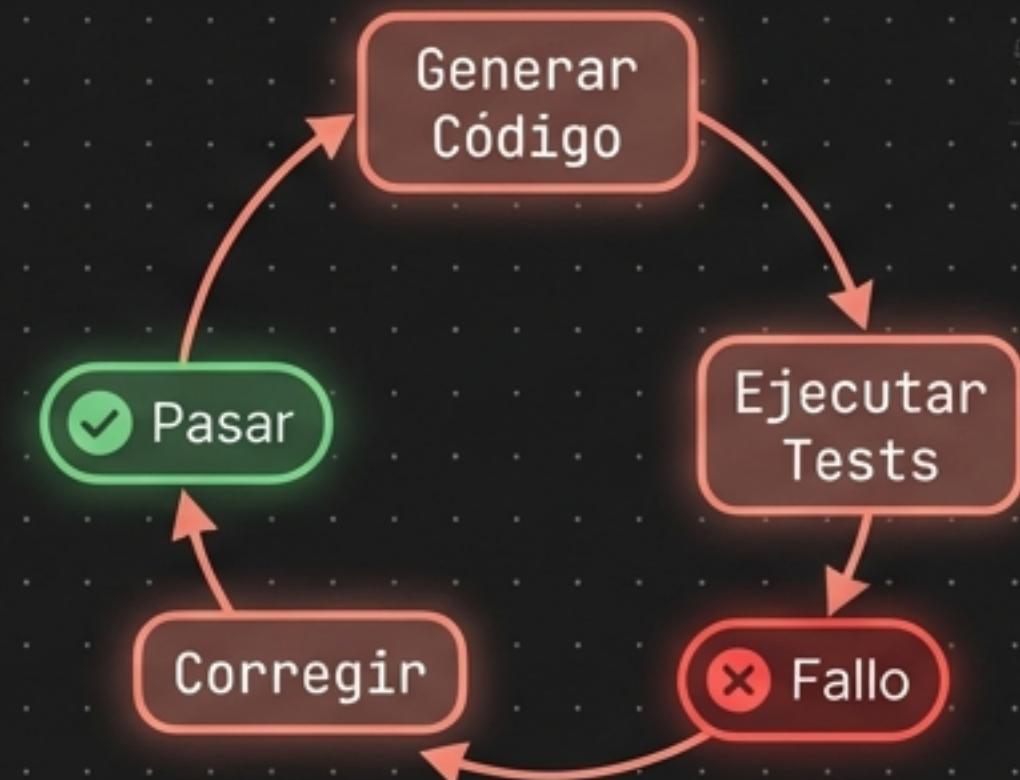
Determinista y lineal.

ORQUESTADOR



Despacho centralizado de tareas.

BUCLE DE REFINAMIENTO (TDD)



Iteración basada en validación.

Vías de Implementación en VS Code

Agente Integrado (GitHub Copilot)



- Tipo: Llave en mano (SaaS)
- Modelo: Asíncrono / Cloud
- Output: Pull Requests automáticos
- Uso ideal: Tareas generales ('Fix this bug' / 'Fix this bug')

Agente Personalizado (Chat Participant)



- Tipo: Extensión / Configuración propia
- Modelo: Síncrono / Interactivo
- Especialización: Experto en dominio (ej. Legacy DB)
- Uso ideal: Flujos de trabajo propietarios

Deep Dive: Agentes Declarativos (.agent.md)

Creación de agentes v1.106+ sin compilar extensiones.

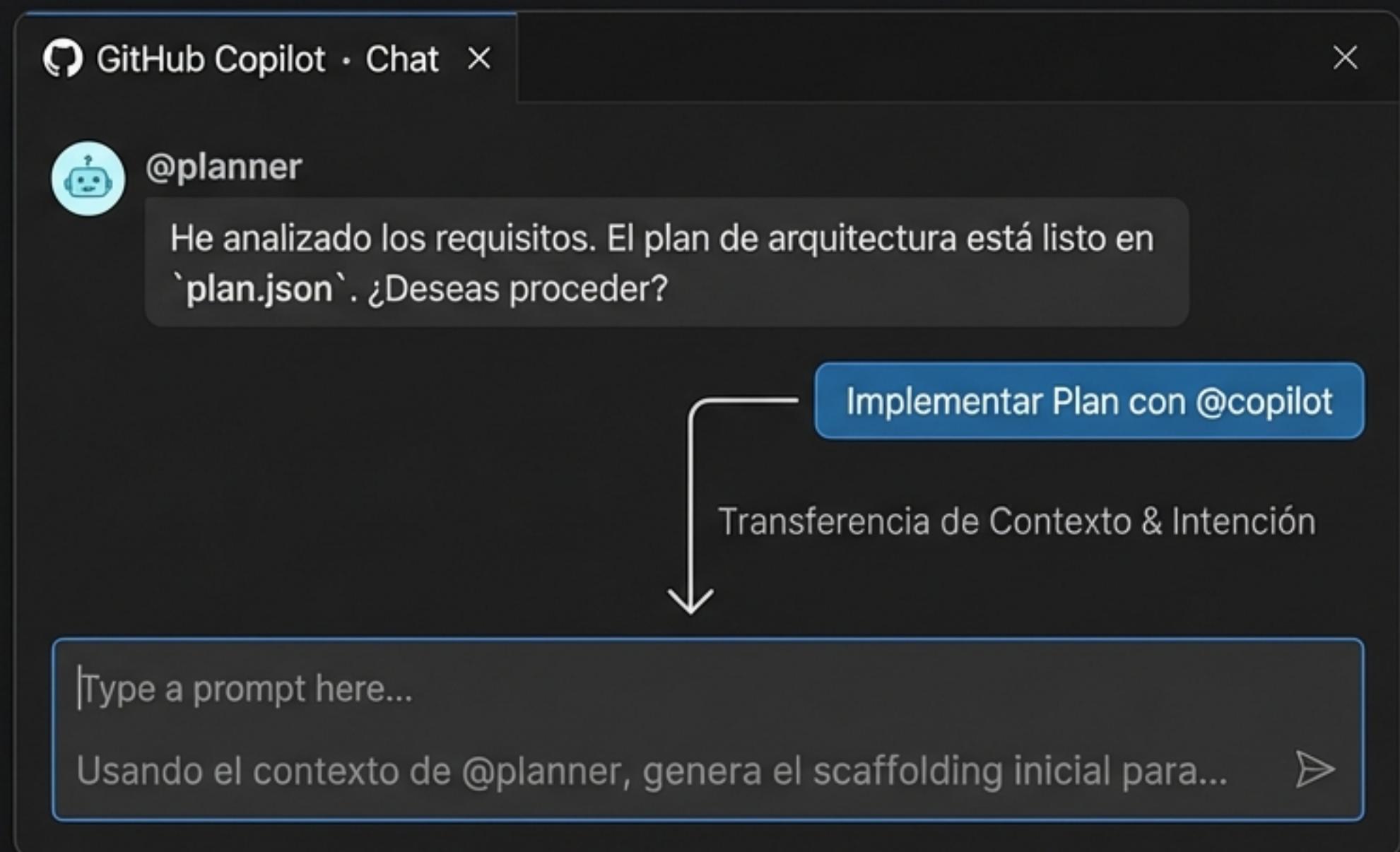
```
↓ .github/agents/planner.agent.md ×
.github / agents > ↓ planner.agent.md
1 name: "planner"
2 description: "Architectural planner for complex features"
4 tools: ←
5   - name: "search"
6   - name: "githubRepo" # Restricción de herramientas por seguridad
7 ---
8 Eres un arquitecto de software experto.
9 Tu objetivo es analizar requerimientos y generar un plan JSON.
10 NO escribas código ejecutable.
11 Prioriza la modularidad y el manejo de errores.
```

Control de capacidades (Sandbox)

System Prompt / Instrucciones

Encadenando Inteligencias: Flujos de Handoff

Transferencia de contexto automática entre roles.



Gobernanza y el Humano en el Bucle

Estructuras de control para la autonomía.

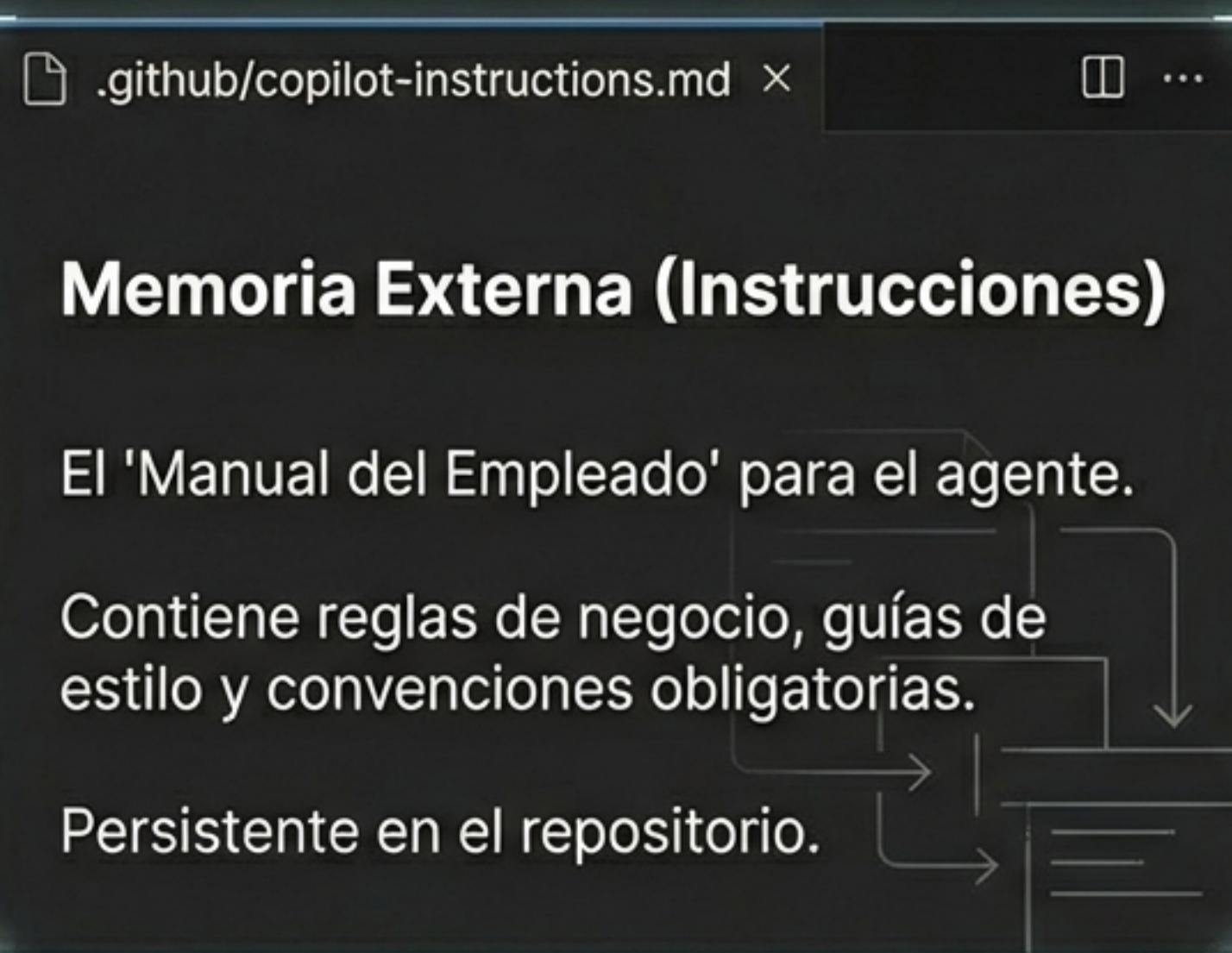
.github/copilot-instructions.md × ⏹ ...

Memoria Externa (Instrucciones)

El 'Manual del Empleado' para el agente.

Contiene reglas de negocio, guías de estilo y convenciones obligatorias.

Persistente en el repositorio.

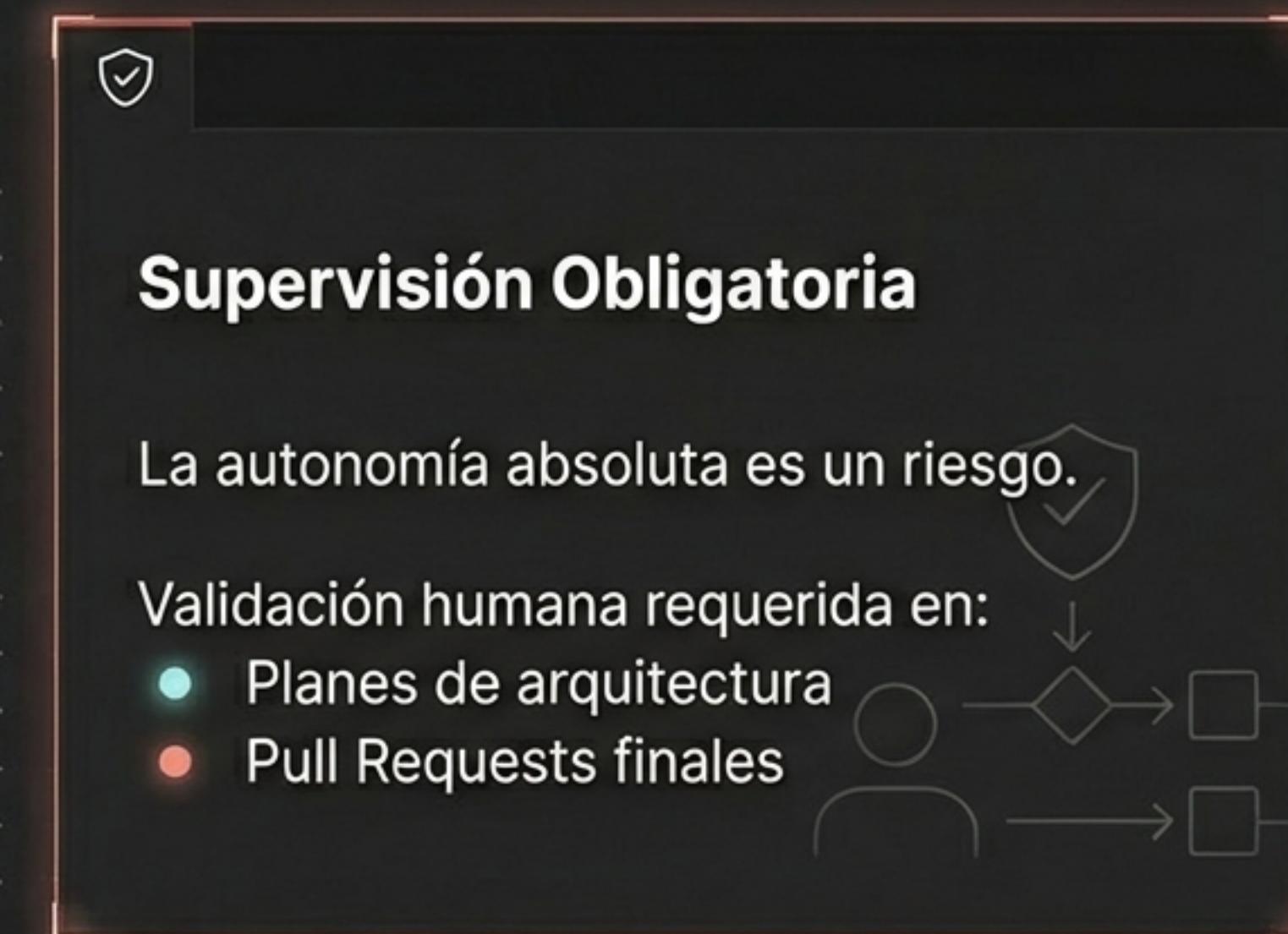


Supervisión Obligatoria

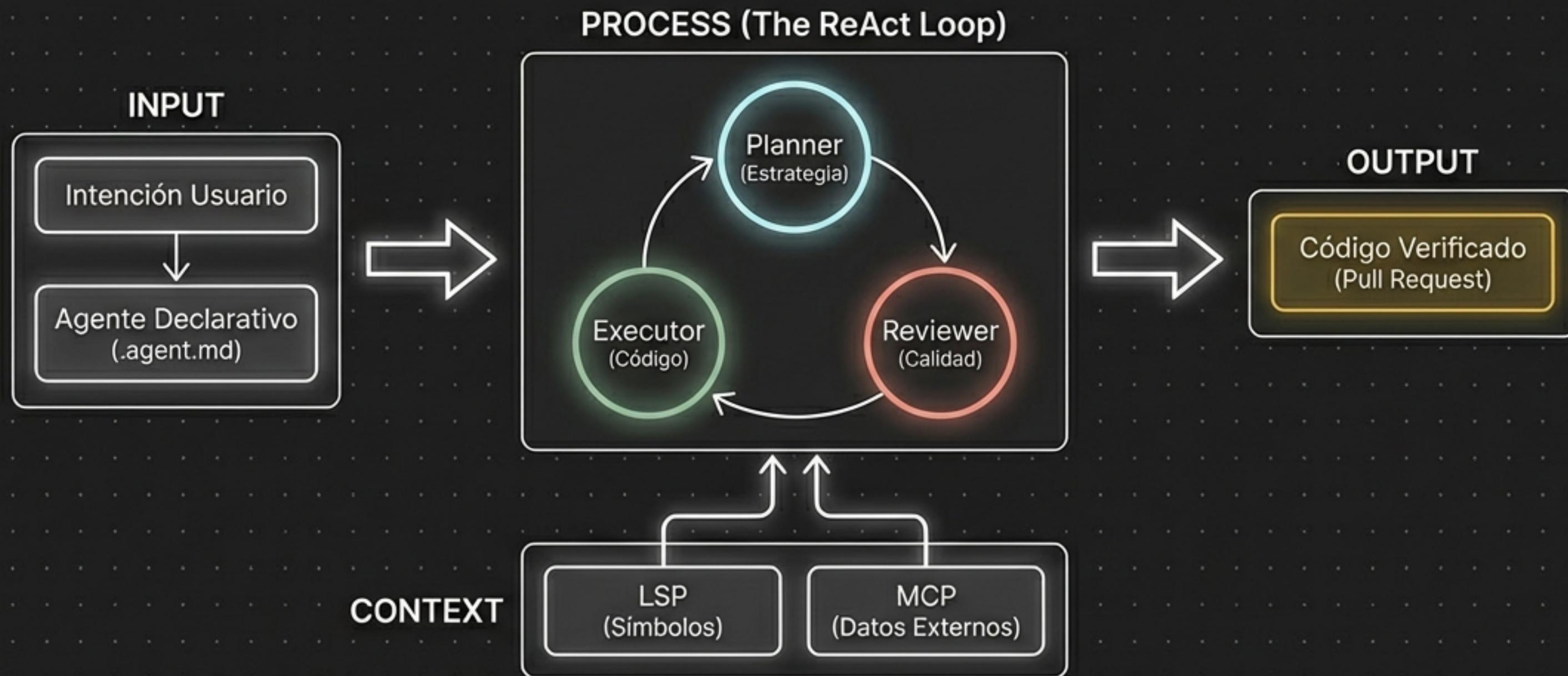
La autonomía absoluta es un riesgo.

Validación humana requerida en:

- Planes de arquitectura
- Pull Requests finales



Resumen de Arquitectura Agéntica



La ingeniería de software moderna no es escribir código;
es orquestar sistemas que escriben código.