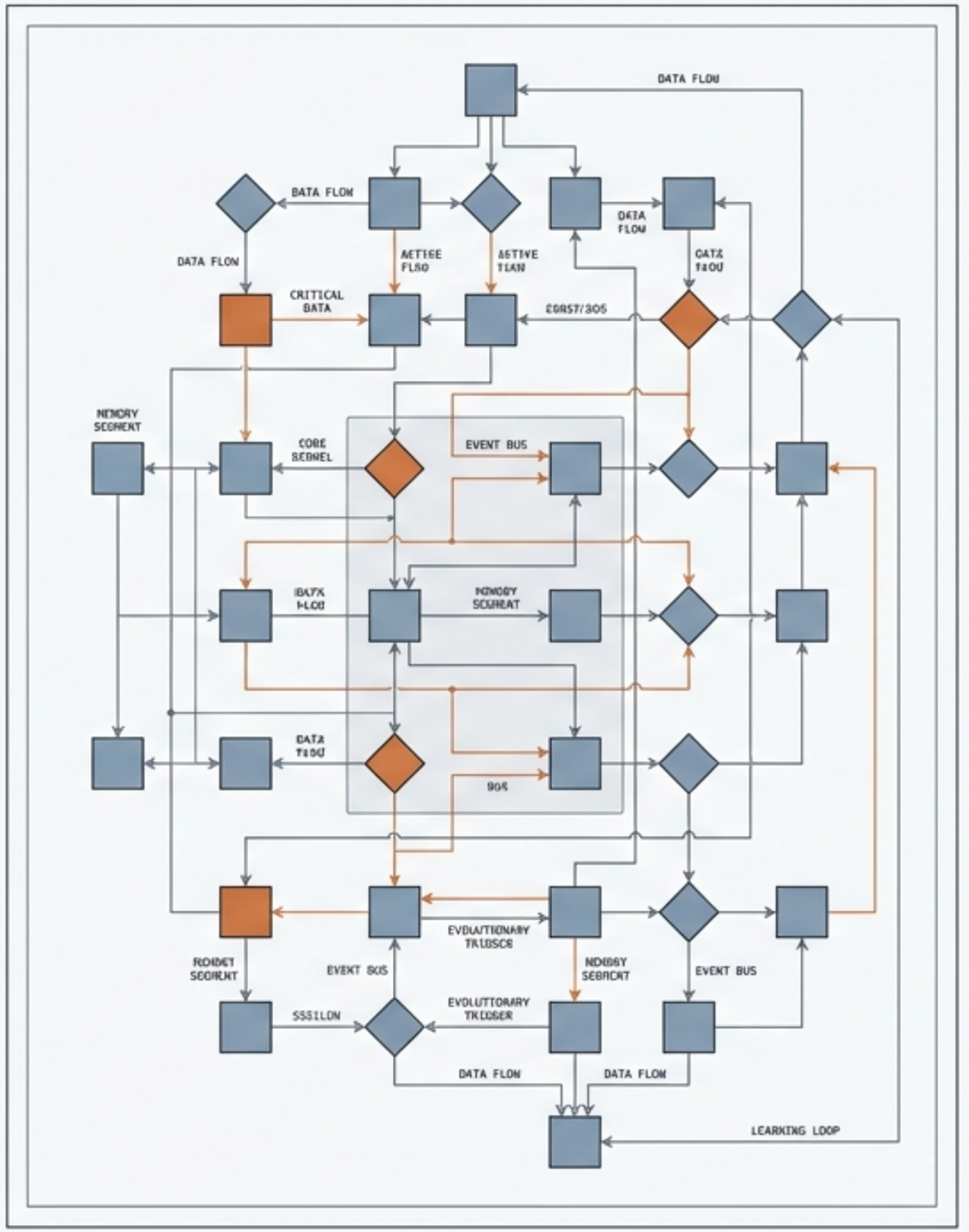
Arquitectura Hermes

Diseño de Agentes Autónomos de Aprendizaje Continuo

[VERSION: v0.7.0] [KERNEL: Multi-Agent Orchestration]

[STORAGE: FTS5 Indexed Memory] [SKILLS: Autonomous Evolution]

Documentación de Arquitectura de Sistemas



NotebookLM

El Salto Paradigmático: Evolución del Harness Engineering

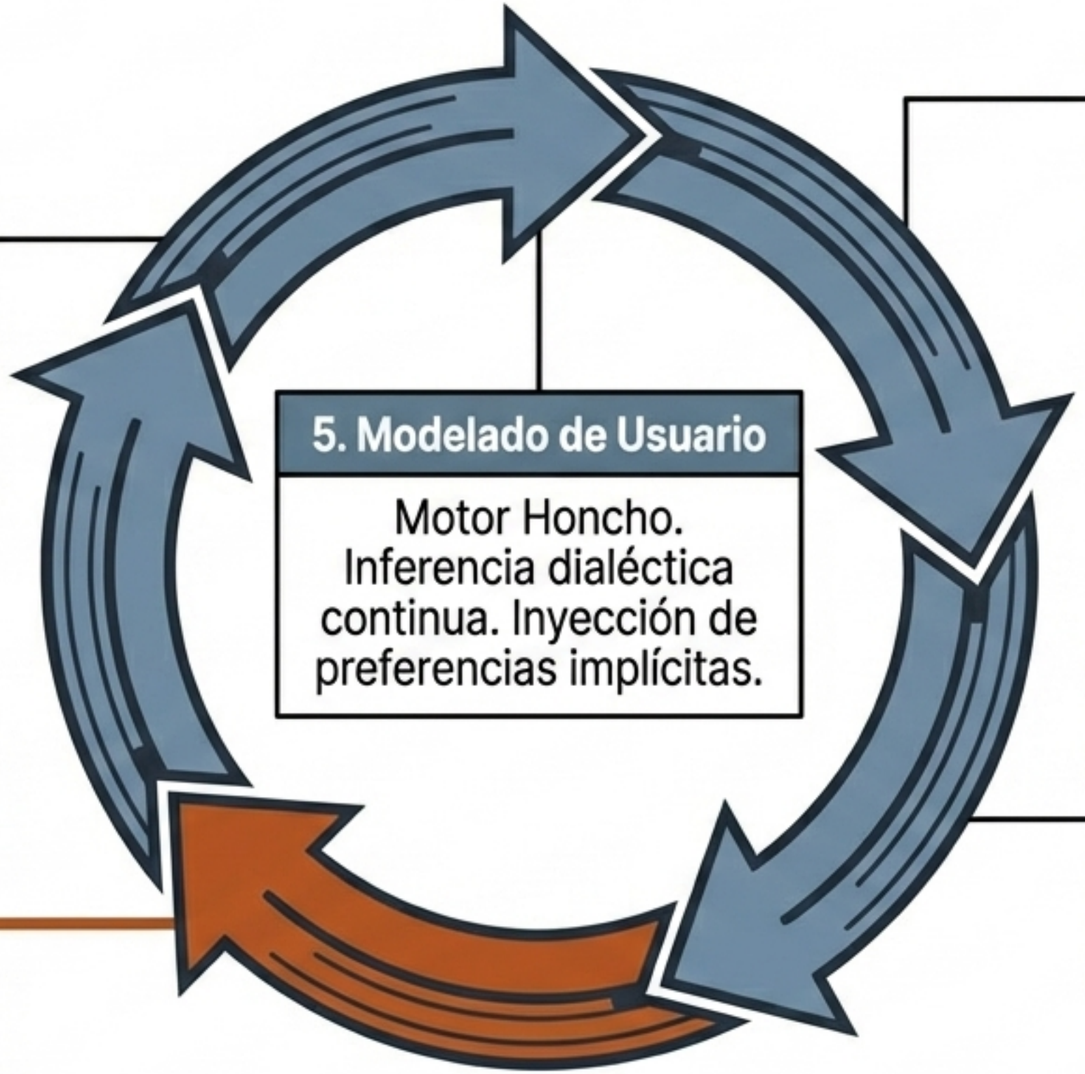
De la configuración humana a la instrumentación auto-sostenible.

Fase Manual (Claude Code)	Fase Configuracional (OpenClaw)	Fase Autónoma (Hermes Agent)
Paradigma Interactividad síncrona. El humano actúa como orquestador.	**Paradigma** Configuración como comportamiento (Configuration-as-behavior).	**Paradigma** Bucle de aprendizaje continuo en segundo plano (24/7 worker).
Gestión de Reglas Archivo `CLAUDE.md` mantenido estáticamente por el desarrollador.	**Gestión de Reglas** Comportamiento definido de forma estricta vía `SOUL.md`. Transparente y predecible.	**Gestión de Reglas** Creación y auto-mejora de archivos `.md` en `~/.hermes/skills/` basados en feedback.
Memoria Dependencia de auto-memoria por proyecto; saturación a corto plazo.	**Memoria** Registros diarios + `MEMORY.md`. Ecosistema masivo (>5,700 skills).	**Memoria** Tres capas estratificadas con indexación local SQLite + FTS5. Evolución sin intervención.

NotebookLM

Arquitectura Core: El Learning Loop

Un motor mecanicista que auto-construye sus propias restricciones.



1. Curación de Memoria

Escritura activa en base de datos. Destilación de Destilaciónes experiencias frente a registro pasivo.

2. Creación Autónoma de Skill

Extracción de soluciones validadas. Generación de archivos `SKILL.md` reutilizables.

3. Auto-mejora (Feedback)

Reescritura del archivo `SKILL.md` basada en métricas de error y correcciones del usuario.

4. Recuperación FTS5

Consulta a `SQLite` previa a tareas. Inyección precisa del contexto histórico bajo demanda.

5. Modelado de Usuario

Motor Honcho. Inferencia dialéctica continua. Inyección de preferencias implícitas.

NotebookLM

Infraestructura de Memoria: El Modelo de 3 Capas

Segmentación de estado para resolver el cuello de botella de la ventana de contexto.

Arquitectura Estratificada

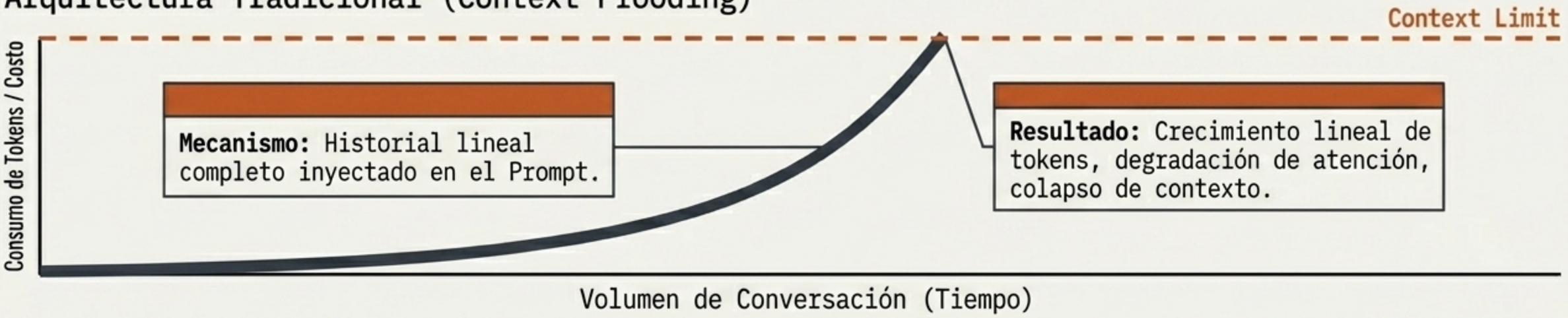
Concepto Cognitivo	Implementación Técnica
Capa Superior: Memoria Episódica (Sesión)	
Cognición: Qué pasó. Historial lineal y resultados de herramientas.	**Implementación:** Base de datos SQLite (**state.db**) + Indexación de búsqueda de texto completo (**FTS5**). Recuperación semántica.
Concepto Cognitivo	**Capa Intermedia: Memoria Semántica (Persistente)**
Cognición: Quién es el usuario. Preferencias operativas y estado duradero.	**Implementación:** Archivos de texto locales (**USER.md / MEMORY.md** en **~/.hermes/memories/**). Persistencia cruzada entre sesiones.
Concepto Cognitivo	**Capa Base: Memoria Procedimental (Skills)**
Cognición: Cómo ejecutar. Metodologías y estándares de operación validados.	**Implementación:** Archivos independientes Markdown gestionados en la ruta **~/.hermes/skills/**.

NotebookLM

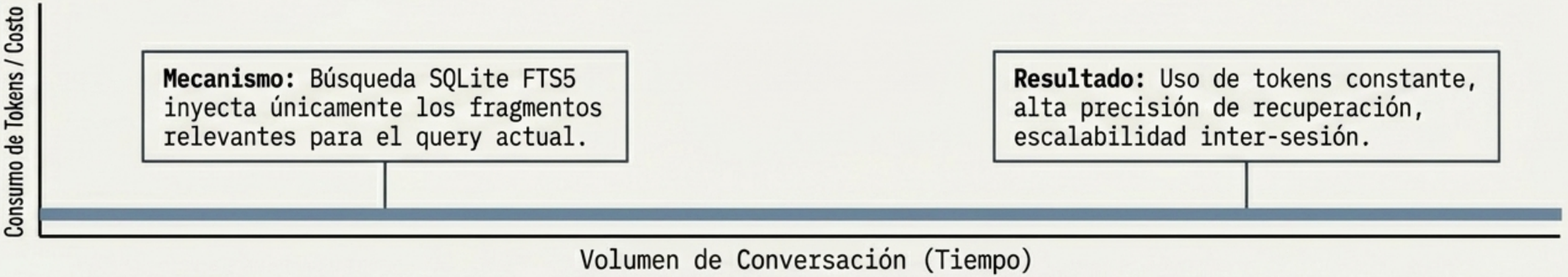
Recuperación Bajo Demanda vs. Inundación de Contexto

La eficiencia asintótica de la indexación sobre la fuerza bruta.

Arquitectura Tradicional (Context Flooding)



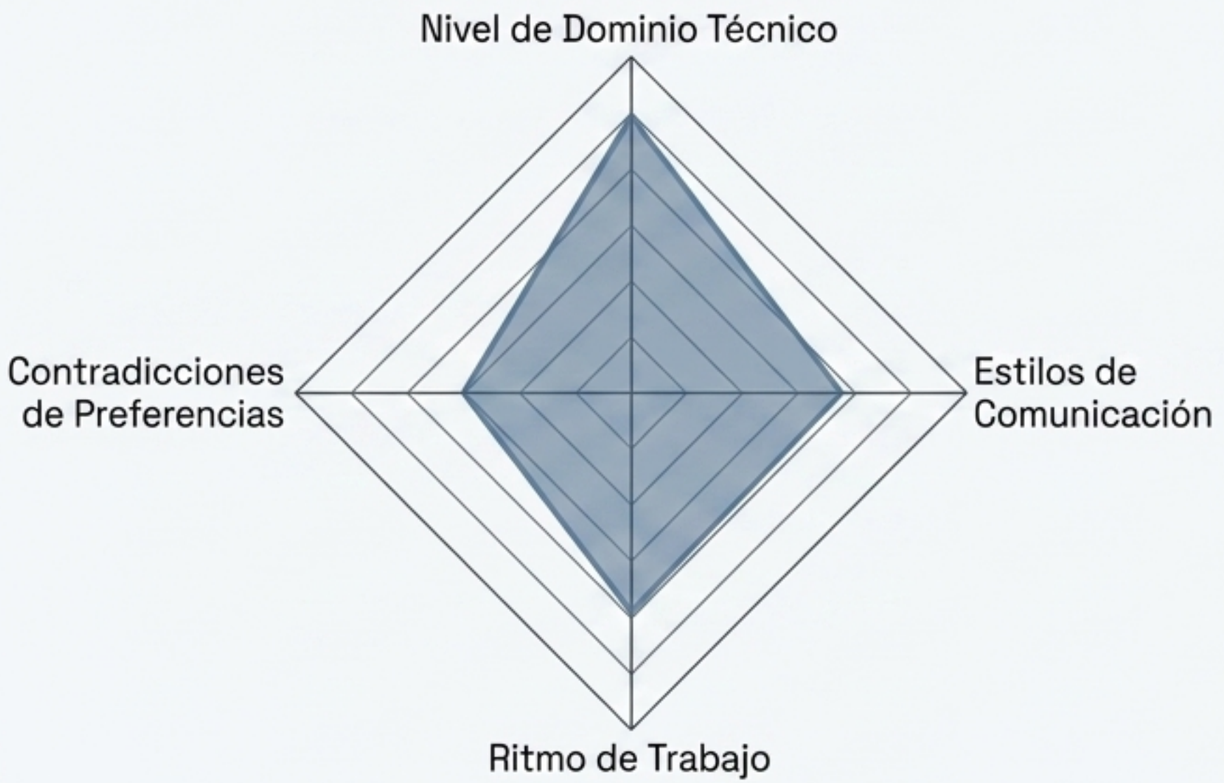
Arquitectura Hermes (FTS5 On-Demand)



NotebookLM

Modelado Dialéctico de Usuario (Motor Honcho)

Inferencia de identidad de 12 capas sin configuración explícita.



Interacción Bruta

Interacción Bruta: la modelea en ... user modelo.

Análisis de Patrones

Análisis de Patrones y compe... análisis.

Deducción de Identidad

Identifad MS capa: Deducción de Identidad es una contracta: y ... ad de Identidad.

Inyección como Contexto Invisible

Inyección como Contexto Invisible en ... comportamiento ... usuario.

Nota Arquitectónica: El modelo ajusta implícitamente el comportamiento futuro sin depender de instrucciones manuales del usuario.

NotebookLM

Sistema de Skills Evolutivas: Anatomía y Mutación

De la programación imperativa a las reglas auto-modificables.

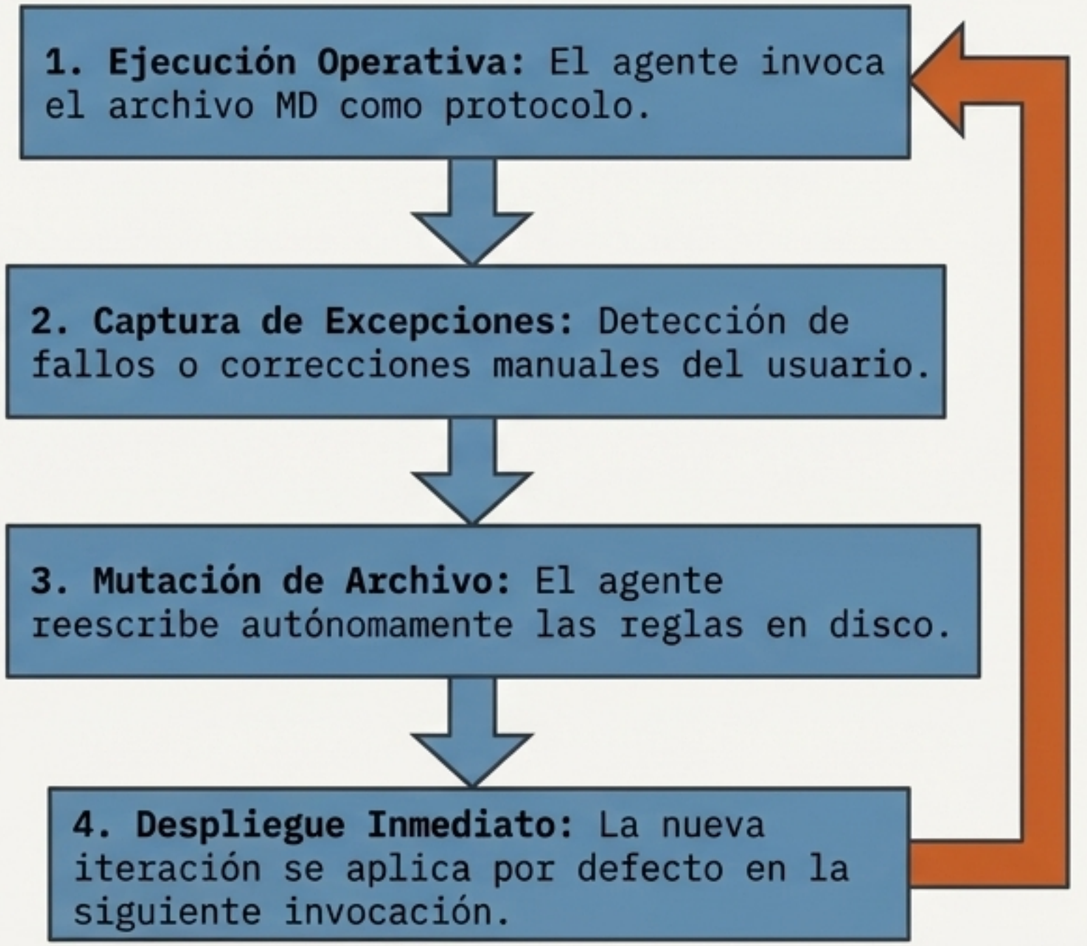
Estructura del Archivo SKILL.md

Trigger
Condición semántica de activación (Ej. Cuando el usuario pide hacer commit).

Rules
Pasos de ejecución estandarizados (Ej. Convención Angular).

Constraints
Límites estrictos (Sandboxing lógico).

Bucle de Ciclo de Vida (Skill Mutability)



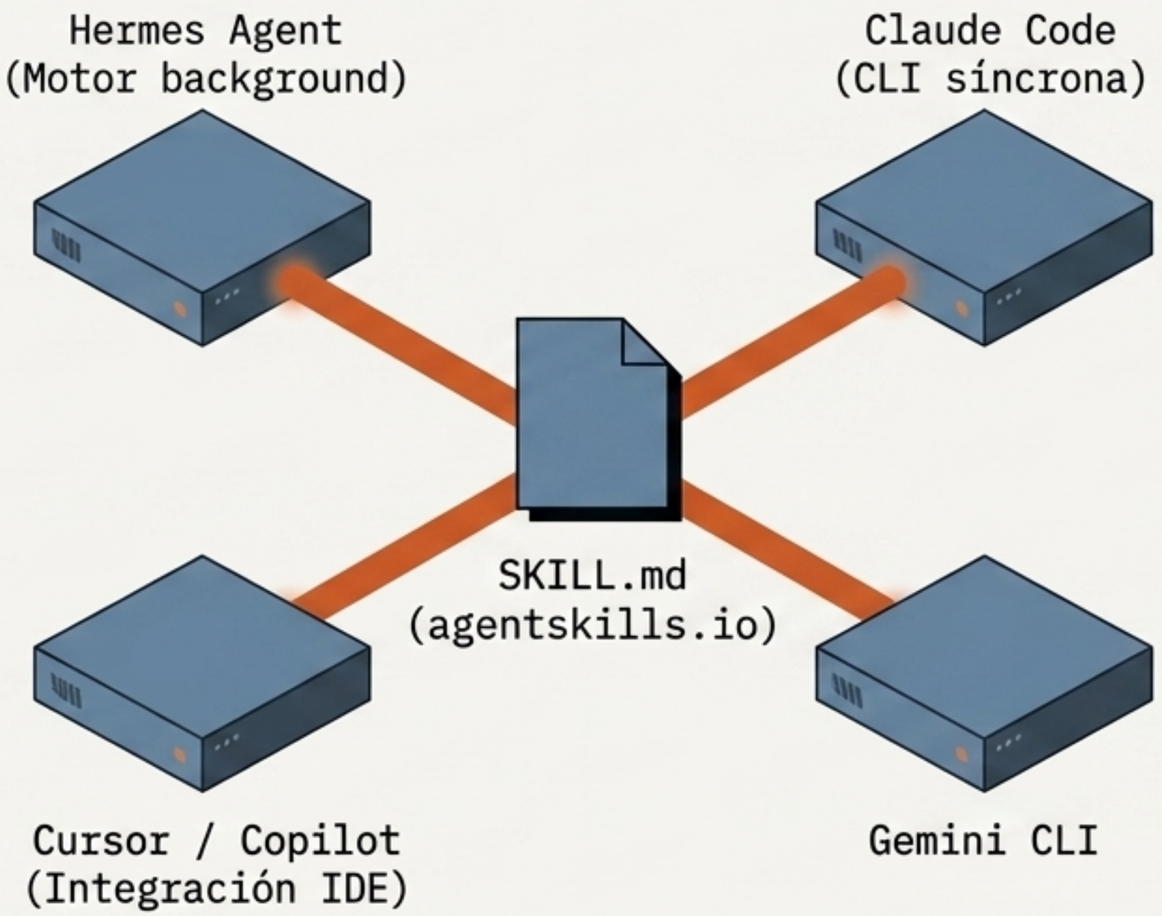
NotebookLM

Interoperabilidad Universal: El Estándar agentskills.io

Eliminación del Vendor Lock-in mediante portabilidad de memoria procedimental.

Mecanismo Core:

- Actúa como un puerto universal para unidades de capacidad procedimental.
- Emplea una semántica estricta basada puramente en Markdown. Sin frameworks abstractos.



Conclusión Arquitectónica:

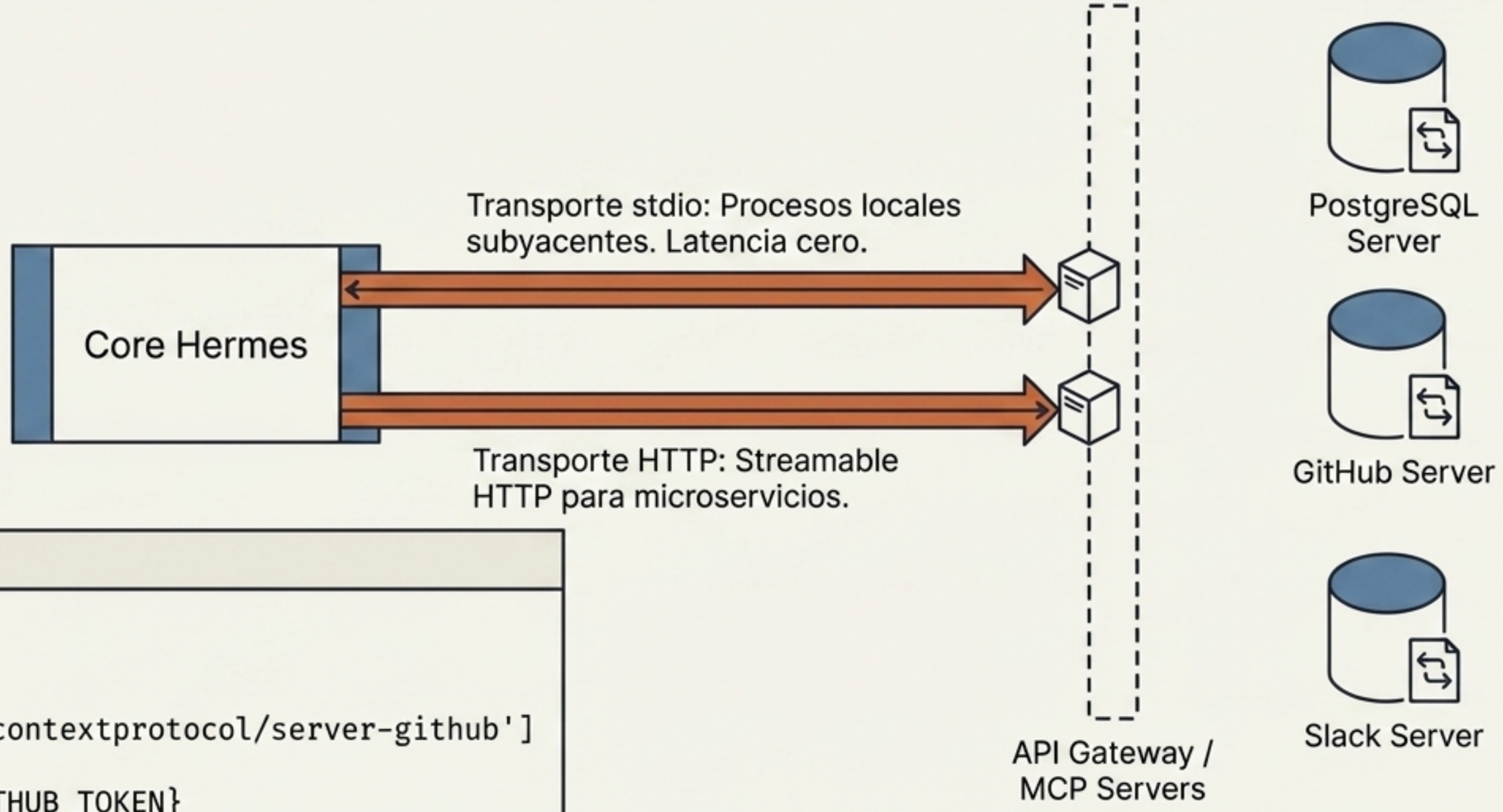
El usuario retiene la propiedad total del asset cognitivo.

El conocimiento procedimental se desarrolla una vez y se invoca desde cualquier nodo del ecosistema.

NotebookLM

Conectividad y Extensibilidad: Model Context Protocol (MCP)

Interfaz unificada hacia un ecosistema de más de 6,000 aplicaciones.



~/.hermes/config.yaml

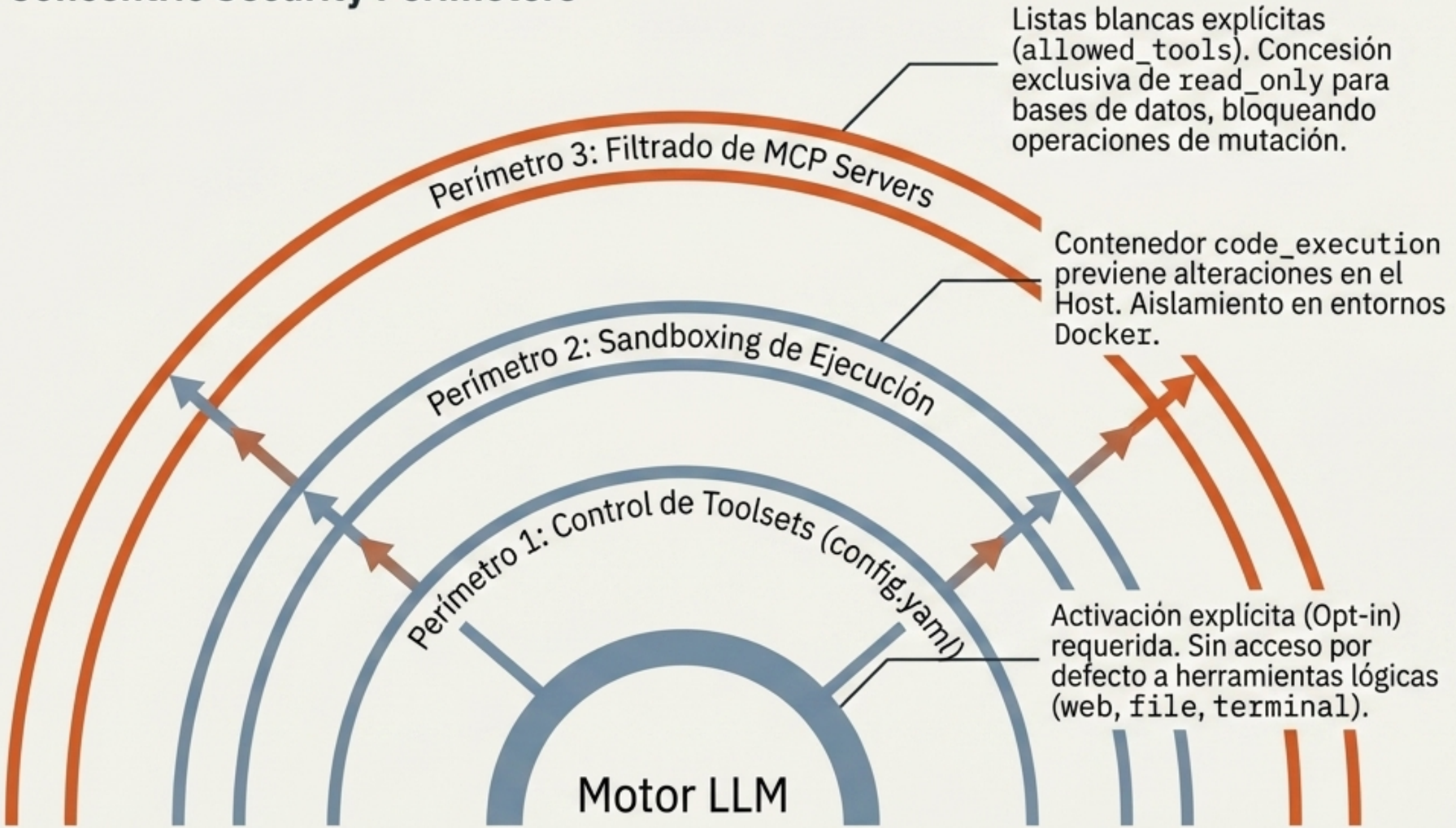
```
mcp_servers:
  github:
    command: npx
    args: ['-y', '@modelcontextprotocol/server-github']
    env:
      GITHUB_TOKEN: ${GITHUB_TOKEN}
```

NotebookLM

Restricciones y Seguridad: El Constraint Layer

Implementación del **Principio de Privilegios Mínimos** en sistemas autónomos.

Concentric Security Perimeters



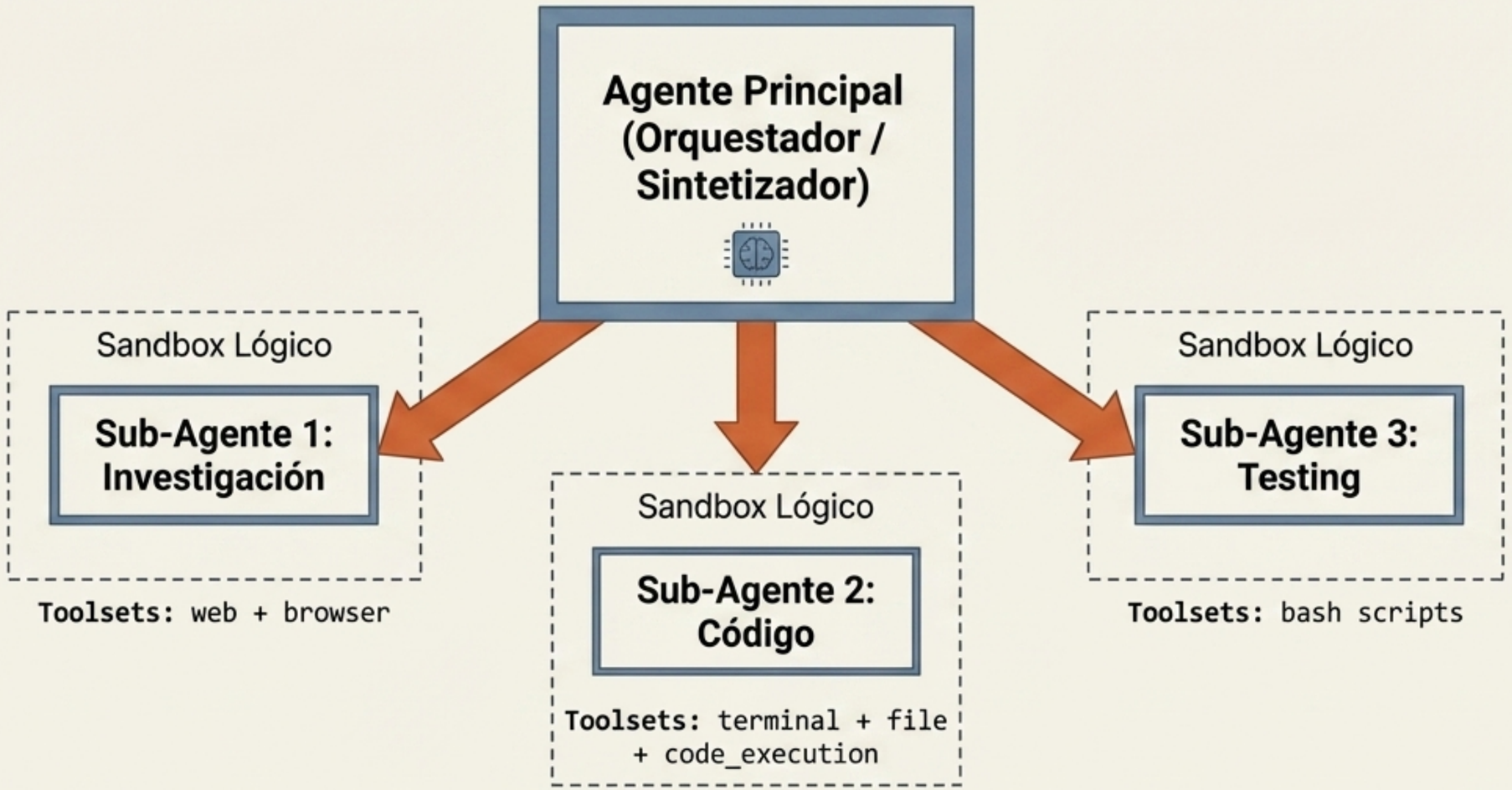
Diseño de Mínimos Privilegios:

A medida que la intención viaja desde el LLM hacia el exterior, debe atravesar filtros restrictivos explícitos, garantizando que la autonomía no comprometa el entorno local.

NotebookLM

Orquestación Multi-Agente: Herramienta delegate_task

Paralelismo de hilos de ejecución con contextos aislados (Límite: 3 Nodos).



Resolución de Cuellos de Botella:

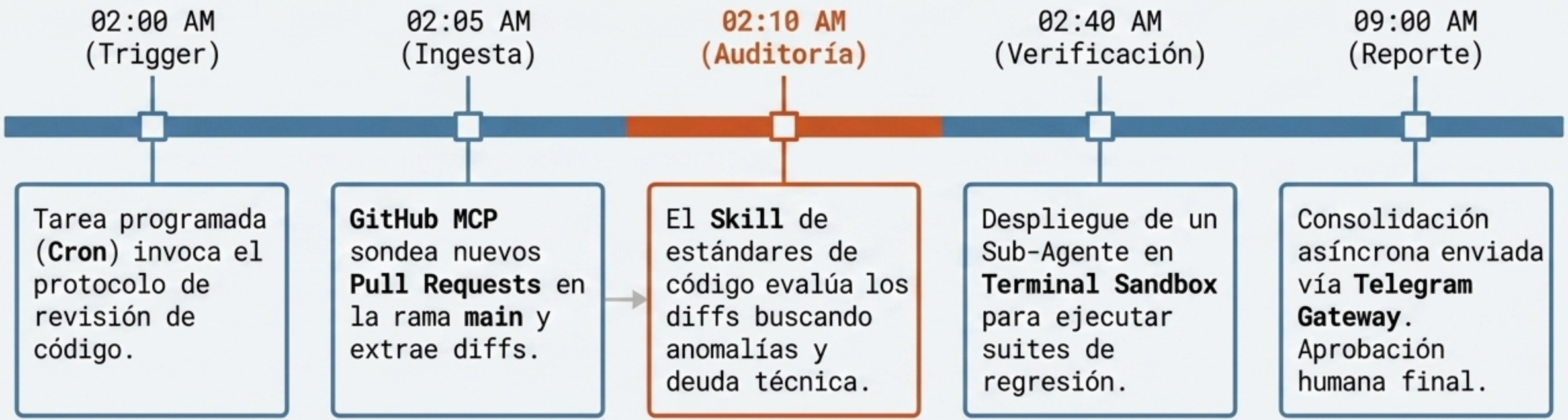
1. Context Explosion: Previene la contaminación semántica aislando los historiales. El código de producción no se mezcla con logs de investigación.

2. Time Bottleneck: Ejecución asíncrona concurrente. Retorno final consolidado al Agente Principal para síntesis.

NotebookLM

Caso Práctico 1: Pipeline Autónomo de Dev Automation

Orquestación cruzada entre Cron, MCP y validación algorítmica.

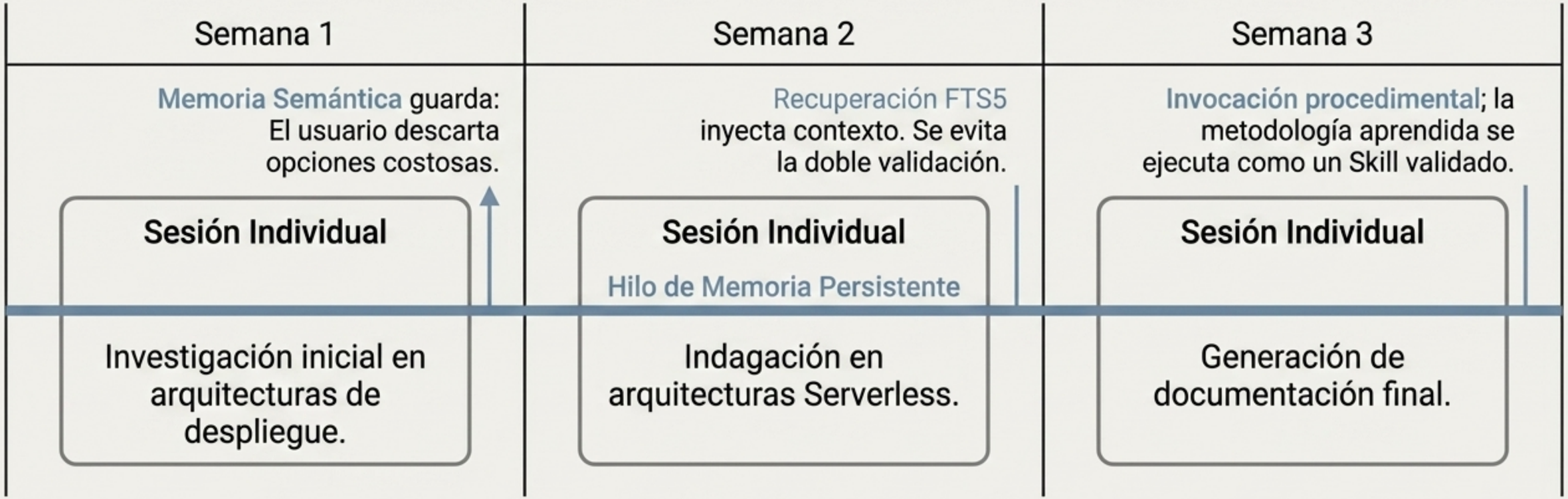


El sistema opera sin supervisión activa, integrando capacidades procedimentales con acceso a herramientas externas.

NotebookLM

Caso Práctico 2: Asistente de Conocimiento a Largo Plazo

Continuidad de contexto mediante la persistencia entre sesiones (Cross-session Memory).



Eficiencia Computacional:
Mantenimiento de la huella de tokens mediante resúmenes de estado semántico (`USER.md`) en lugar de carga histórica secuencial lineal.

NotebookLM

Matriz Arquitectónica del Ecosistema de Agentes

No es una elección binaria, sino una estrategia de combinación de topologías.

	Claude Code (El Artesano)	OpenClaw (La Consola)	Hermes Agent (El Mayordomo)
Modo Operativo	Síncrono / Front-end (CLI)	Síncrono / Bajo Demanda	Asíncrono / Background Loop (24/7)
Gestión de Reglas	Manual (CLAUDE.md)	Configuracional estricta (SOUL.md)	Autodescubrimiento y evolución (Skills)
Memoria	Auto-memoria por proyecto	Registros diarios + Transparencia	3 capas estratificadas + FTS5
Despliegue	Local (Suscripción ligada al usuario)	Local / Hospedado	Arquitectura VPS / Serverless / Local
Caso de Uso Ideal	Pair-programming interactivo, refactorización	Auditoría rígida, agentes de equipo	Monitorización ininterrumpida, automatización de fondo

NotebookLM

Fronteras de Autonomía y Especificaciones de Despliegue

La transición del micro-management a la gobernanza algorítmica (On the Loop).

Espectro de Intervención Humana

Punto de control crítico: Auditoría asíncrona de mutaciones en directorios ~/.hermes/skills/. El usuario evalúa direcciones lógicas en lugar de sintaxis.

Deployment Specs Recomendados

INFRAESTRUCTURA	Servidor VPS estándar ($5/mes, e.g., Ubuntu 22.04 LTS).
RUNTIME	Despliegue en contenedor (Docker Volume Mounting: -v ~/.hermes:/opt/data para persistencia absoluta).
MULTIPLEXING	Proceso único enrutando hacia interfaces móviles (Telegram/Discord Bots) y CLI local.
LOCAL FALLBACK	Soporte para Ollama (Llama 3 / Hermes 3 8B) en nodos con GPU dedicada.

NotebookLM