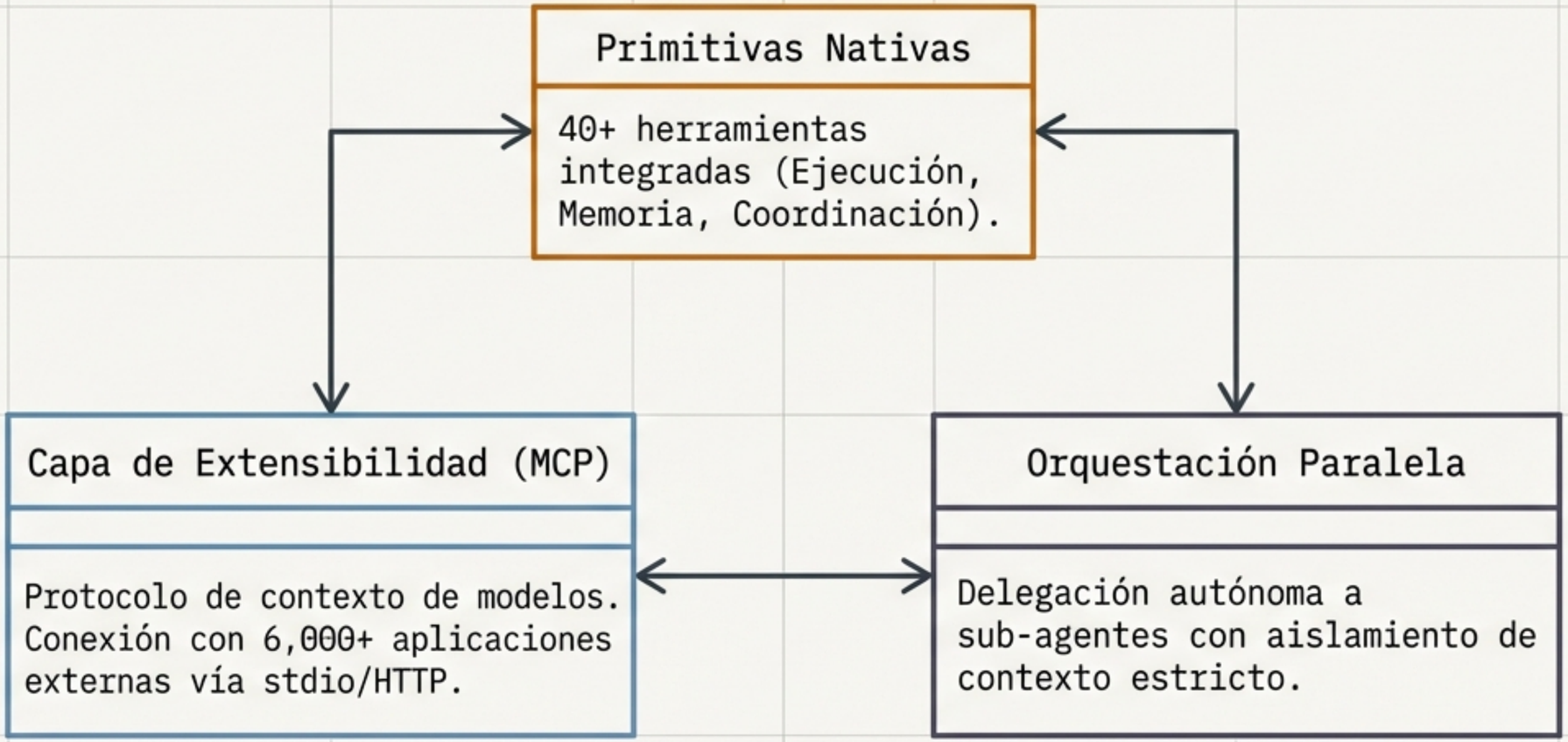
# Arquitectura Hermes: Herramientas, MCP y Delegación Multi-Agente

## Diseño de Sistemas y Orquestación Avanzada

SYSTEM_DOC_REF: CONFIDENCIAL_LATAM // ARCHITECT_BRIEF

# La Tríada de Interactividad



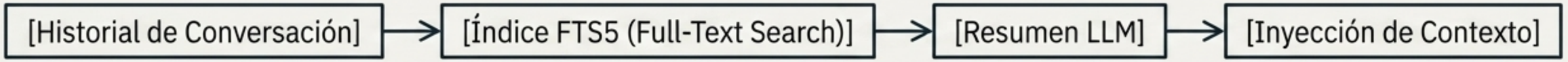

La inteligencia bruta requiere actuadores. Hermes combina herramientas estáticas, APIs dinámicas y delegación paralela para interactuar con entornos reales.

# Matriz de Capacidades: 40+ Herramientas Integradas

| Categoría | Sub-Sistemas (Core Tools) | Función Arquitectónica |
|---|---|---|
| Ejecución | `terminal, code_execution, file` | Ejecución de comandos, sandboxing de código, I/O de archivos. |
| Información | `web, browser, session_search` | Búsqueda web, automatización de navegador, indexación de historial. |
| Multimedia | `vision, image_gen, tts` | Procesamiento de visión, generación de imágenes, síntesis de voz. |
| Memoria | `memory, skills, todo, cronjob` | Capa de persistencia, gestión de Skills, planificación, cronjobs. |
| Coordinación | `delegation, moa, clarify` | Orquestación de sub-agentes, razonamiento multi-modelo. |

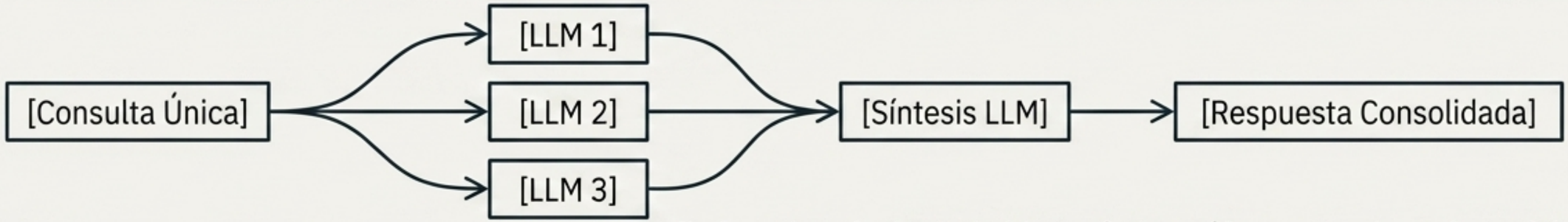
# Análisis Profundo: Mecanismos Internos Avanzados

## session_search



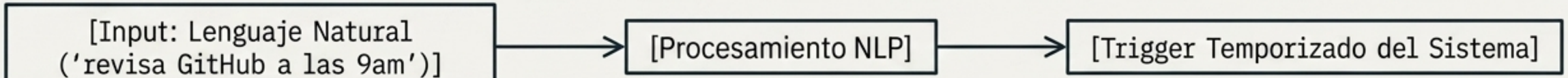

Evita el síndrome de amnesia. Permite al agente recordar "el enfoque de la semana pasada".

## moa (Multi-Model Orchestrated Answering)




Ideal para fact-checking y decisiones técnicas críticas que requieren alta confiabilidad.

## cronjob



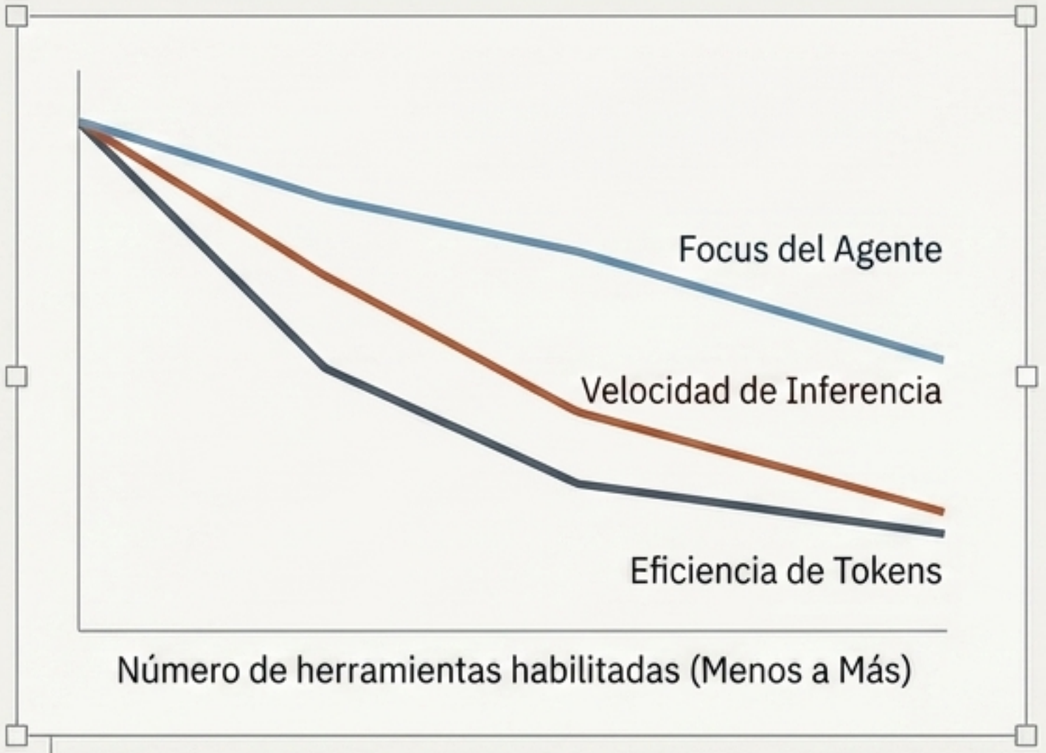

Sin expresiones cron, sin configurar schedulers externos.

# Optimización de Recursos vía Toolsets

Code Configuration

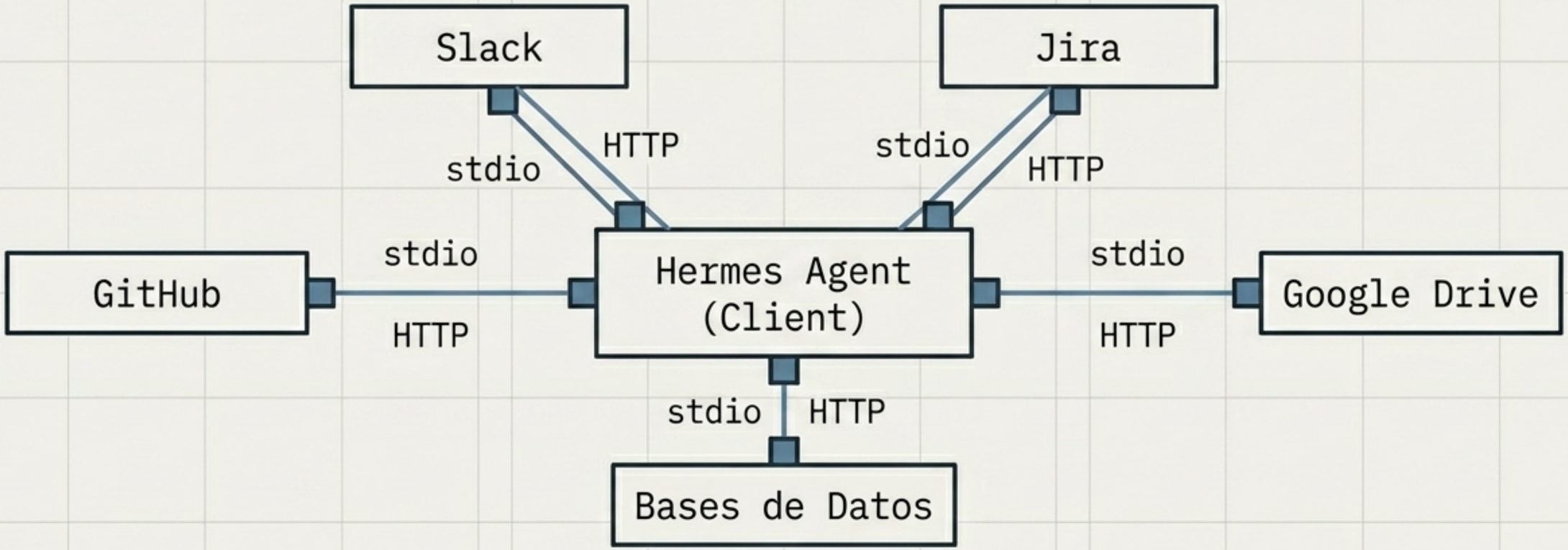
```
toolsets:
  - web
  - terminal
  - file
  # - homeassistant
```

Límite de Seguridad Establecido




**Habilitar todo de golpe es ineficiente.**
Agrupar por Toolsets garantiza un agente más rápido, enfocado y económico.

# La Capa de Extensibilidad: Model Context Protocol (MCP)

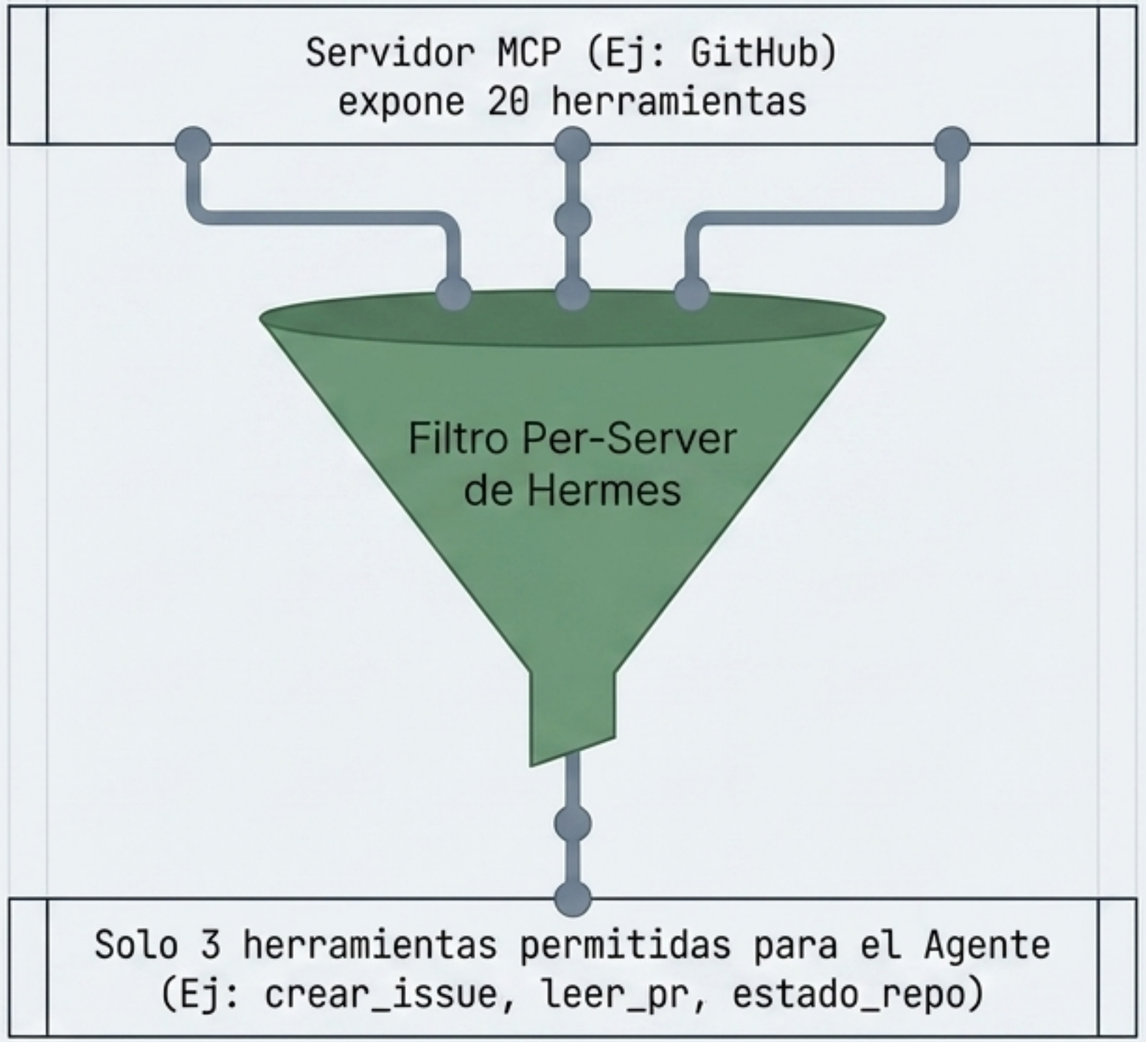



**MCP** es el protocolo abierto estándar de comunicación entre agentes de IA y herramientas externas. Otorga acceso plug-and-play a un ecosistema de **más de 6,000 aplicaciones** sin necesidad de escribir integraciones personalizadas.

# Integración MCP y Filtrado Granular

Code Configuration

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




Precisión granular. Integración inmediata, pero control absoluto sobre la superficie de ataque expuesta al modelo.

# Orquestación Multi-Agente: El Paradigma de los Tres Caballos

## Aislamiento de Contexto

Cada sub-agente posee su propio espacio de memoria y contexto de conversación independiente. El agente principal solo inyecta el background necesario.

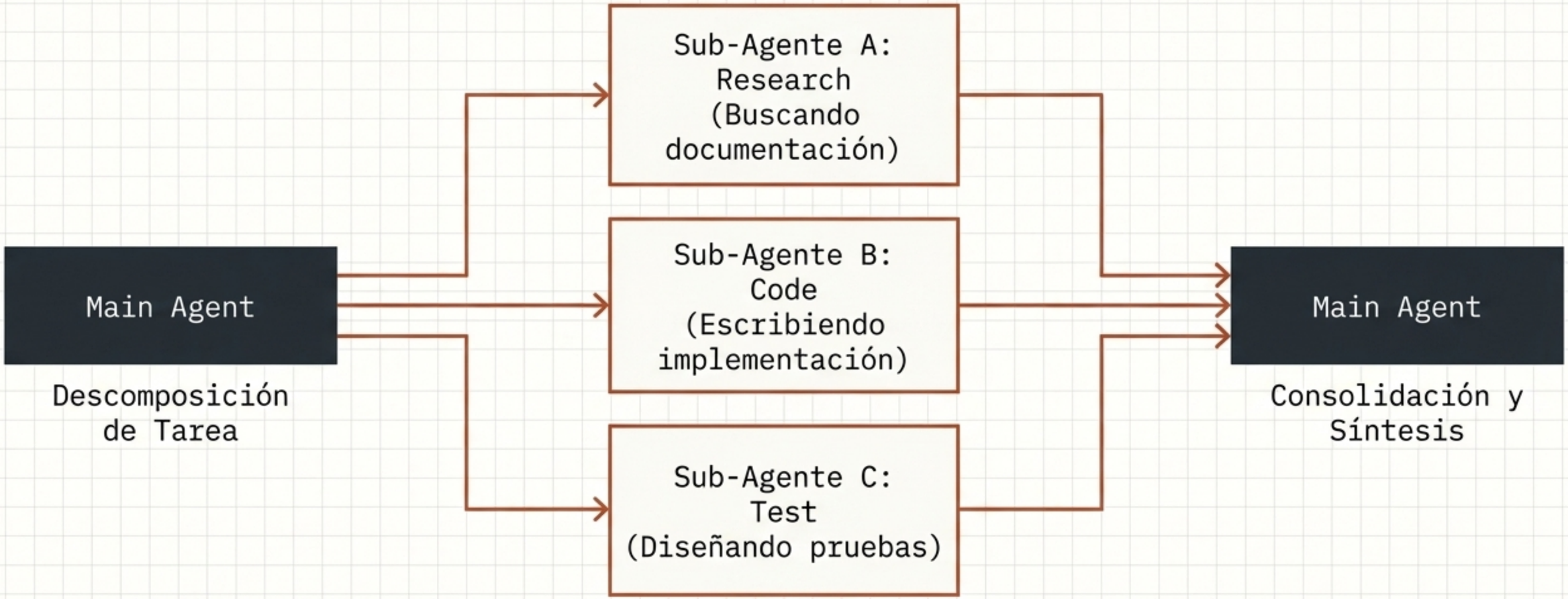
## Toolsets Restringidos

Asignación de permisos por tarea (Ej: El sub-agente de investigación solo recibe acceso a web y browser). Maximiza eficiencia y seguridad.

## Límite de Concurrencia (Max 3)

Límite intencional por diseño. Tres sub-agentes paralelos cubren el 90% de los escenarios de ingeniería sin colapsar la coordinación.

# Flujo de Ejecución Paralela

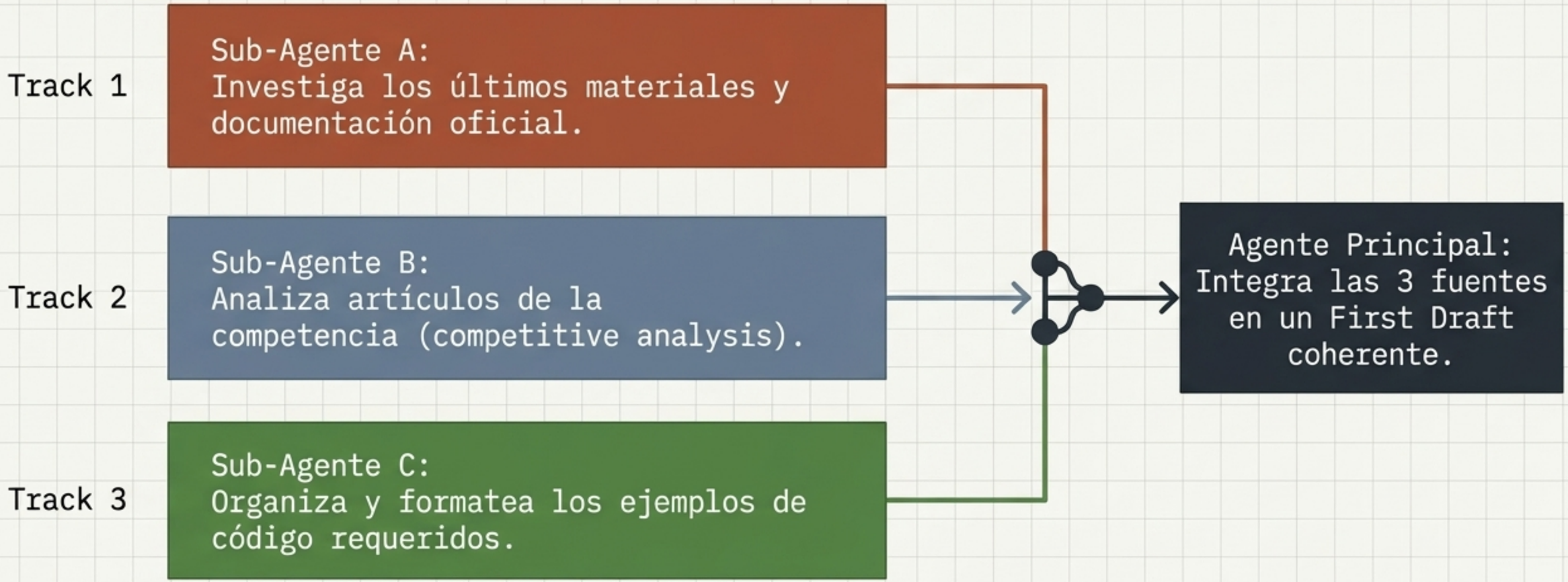


# Delegación Tradicional vs. Autonomía Hermes

| | Tradicional (Múltiples Instancias) | Arquitectura Hermes |
|---|---|---|
| **Lanzamiento de Instancias** | ❌ Manual (el usuario abre múltiples terminales y gestiona el flujo de trabajo). | ✅ Autónomo (el agente decide dinámicamente cuándo y cuántos sub-agentes despachar). |
| **Contexto y Coordinación** | ❌ Nula. Las instancias no se comunican entre sí ni comparten estado. | ✅ Total. El agente principal actúa como orquestador, compartiendo solo el background específico necesario. |
| **Síntesis de Resultados** | ❌ Consolidación manual de código y texto por parte del desarrollador. | ✅ El agente principal compila automáticamente el trabajo de todos los sub-agentes en un resultado unificado. |

# Aplicación en el Mundo Real: Pipeline Paralelo

```
> Prompt: 'Escribe un artículo técnico para el blog sobre la última actualización de React.'
```



# La Capa de Restricción: Gobernanza de Sistemas Autónomos

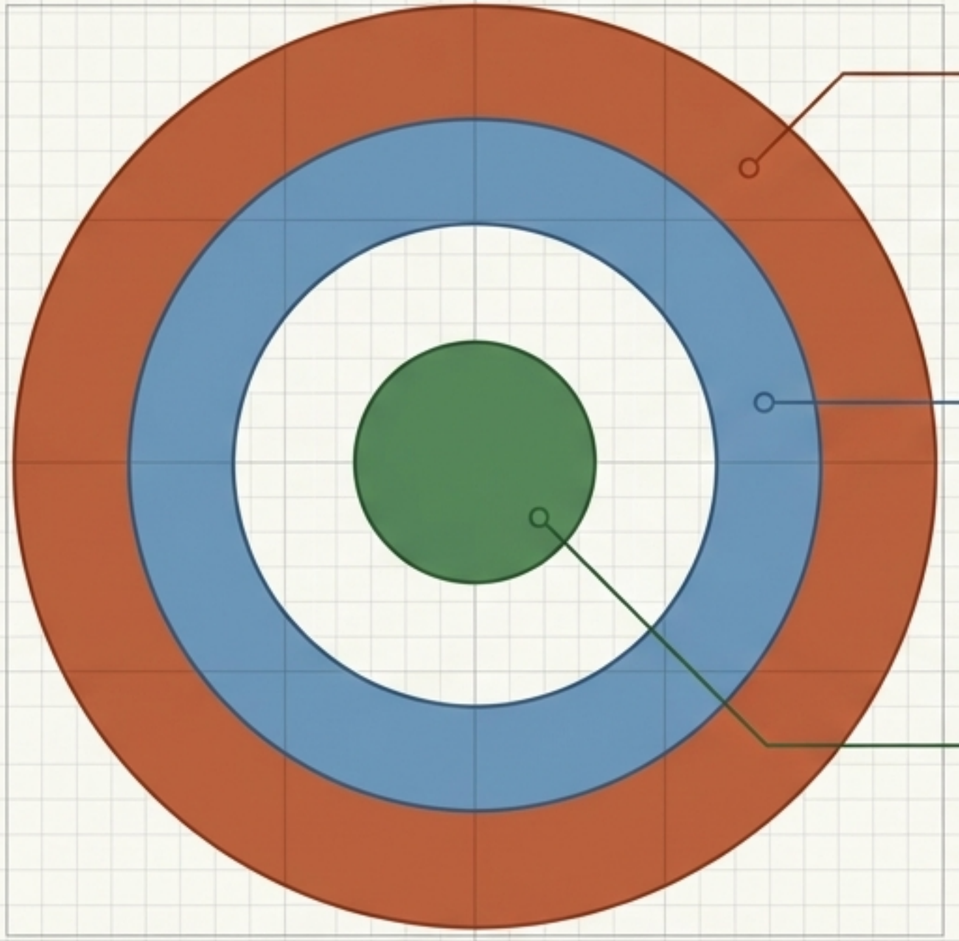

### La Filosofía (The Orange Book)

Inspirado en los principios del Harness Engineering Orange Book. Esta capa de seguridad funciona con la misma rigurosidad que los hooks y linters en pipelines de integración continua.

### Regla de Diseño Arquitectónico

Otorgar al agente la capacidad *exacta* para completar su tarea, ni un permiso más. No se busca un aislamiento teórico perfecto, sino un modelo pragmático entre usabilidad y seguridad de infraestructura.

# Defensa en Profundidad: 3 Capas de Restricción



### Control de Toolsets (Macro)

El interuptor principal. Solo las herramientas explícitamente habilitadas en config.yaml son expuestas al agente global.

### Restricción de Sub-Agentes (Micro)

Permisos granulares. Al delegar, los sub-agentes reciben un subconjunto hiper-limitado del Toolset. Un agente de búsqueda jamás toca la terminal.

### Sandbox de code_execution (Aislamiento)

Barrera dura. El código se ejecuta en un entorno efímero y totalmente aislado. Las operaciones destructivas no pueden afectar el sistema de archivos principal del host.

# Arquitectura de Seguridad para Producción

## Rule 1 : Default to Closed

Para entornos sensibles (servidores de producción), desactiva todos los Toolsets por defecto. Habilita única y estrictamente los vectores necesarios.

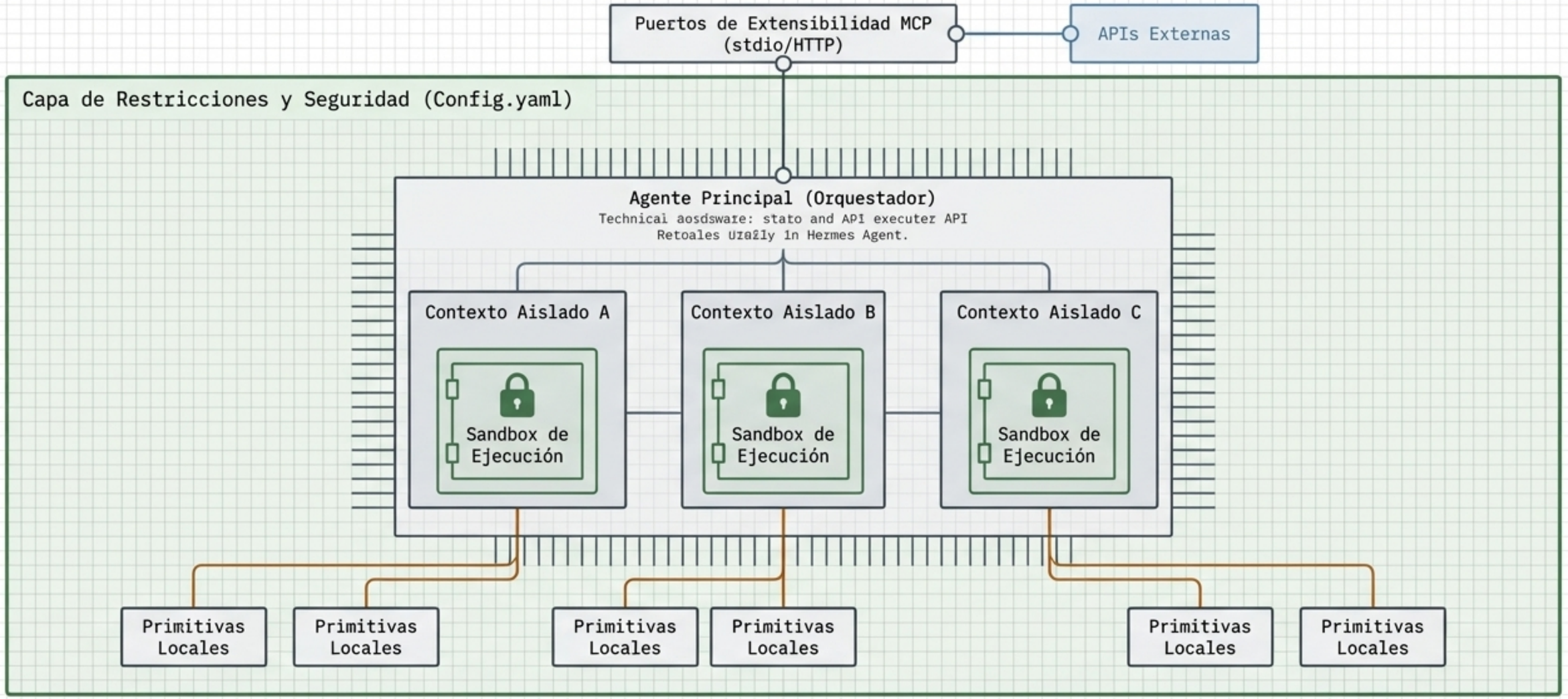
## Rule 2: Filtrado Estricto MCP

Usa obligatoriamente el filtrado per-server de la capa MCP. Restringe meticulosamente la exposición de APIs y endpoints externos hacia el modelo.

## Rule 3: Autorización Explícita

Implementa una arquitectura basada en autorización. Es preferible que el agente detenga la ejecución y solicite: "Necesito permisos XX para continuar", en lugar de operar de manera global con acceso sin restricciones.

# Blueprint de Arquitectura: Hermes Agent




Una arquitectura diseñada no solo para ejecutar comandos, sino para orquestar flujos de trabajo autónomos, seguros y altamente paralelizables en entornos de producción críticos.