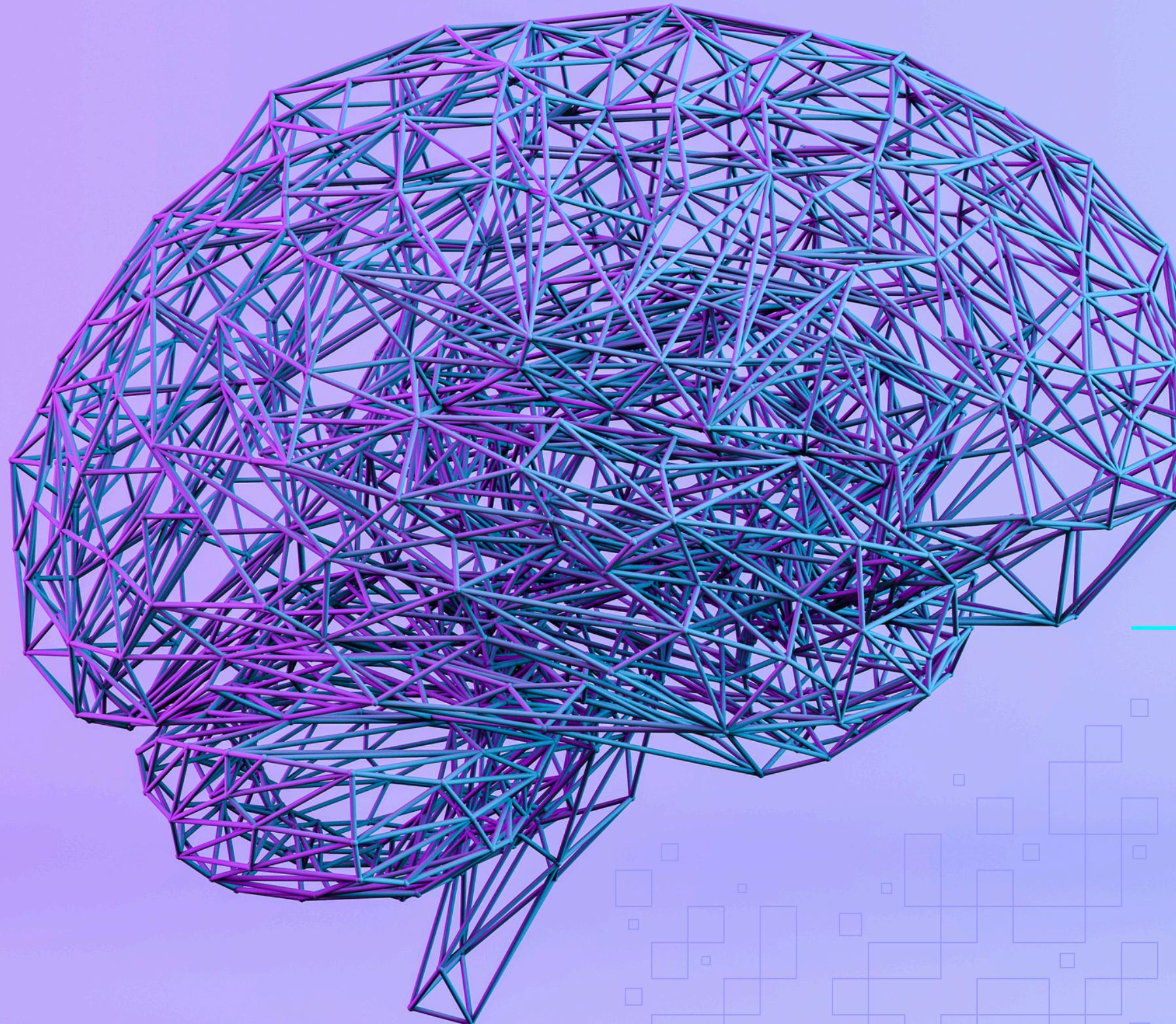




# ASISTENTE INTELIGENTE





# CASO ORGANIZACIONAL

- LA ORGANIZACIÓN CORRESPONDE A UNA UNIVERSIDAD FICTICIA, DE TAMAÑO MEDIO, DEDICADA A LA EDUCACIÓN SUPERIOR. ATIENDE A MÁS DE 10.000 ESTUDIANTES EN DISTINTAS CARRERAS, TANTO DIURNAS COMO VESPERTINAS. SU RUBRO PRINCIPAL ES LA FORMACIÓN ACADÉMICA PROFESIONAL Y TÉCNICA.



# PROBLEMÁTICA

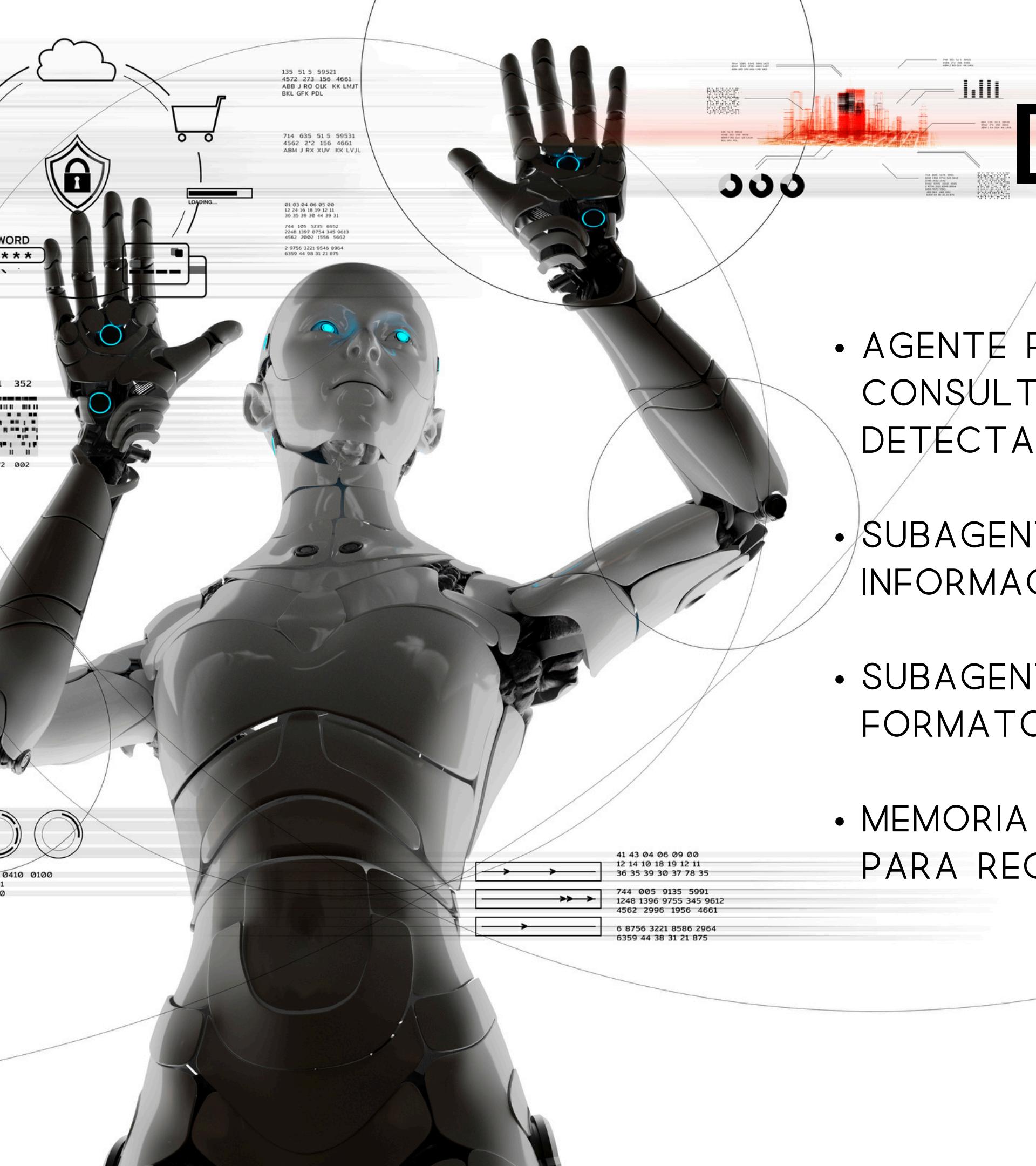
ACTUALMENTE, LOS ESTUDIANTES ENFRENTAN DEMORAS Y DIFICULTADES PARA RESOLVER CONSULTAS FRECUENTES RELACIONADAS CON NOTAS, CALENDARIOS ACADÉMICOS Y BECAS. LA GESTIÓN MANUAL MEDIANTE SECRETARÍAS GENERA SOBRECARGA ADMINISTRATIVA Y BAJA SATISFACCIÓN ESTUDIANTIL. EL DESAFÍO CONSISTE EN IMPLEMENTAR UN ASISTENTE INTELIGENTE CAPAZ DE RESPONDER ESTAS CONSULTAS DE MANERA RÁPIDA, CONFIABLE Y TRANSPARENTE.

# OBJETIVOS

- Reducir los tiempos de respuesta a consultas estudiantiles.
- Disminuir la carga administrativa de secretarías.
- Mejorar la satisfacción y confianza de los estudiantes.
- Garantizar transparencia en la entrega de información institucional.



# COMPONENTES



- AGENTE PRINCIPAL (CONTROLLER): ESTE ES QUIEN RECIBE LA CONSULTA, ACTIVA LOS SUBAGENTES SEGÚN LA INTENCIÓN DETECTADA (CONSULTA, BÚSQUEDA, REDACCIÓN, VALIDACIÓN).
- SUBAGENTE DE BÚSQUEDA: EMPLEA RAG PARA RECUPERAR INFORMACIÓN EN FUENTES INSTITUCIONALES.
- SUBAGENTE DE REDACCIÓN: ESTRUCTURA LA RESPUESTA EN FORMATO EMPÁTICO Y ACADÉMICO.
- MEMORIA CONTEXTUAL: USA ALMACENAMIENTO TEMPORAL PARA RECORDAR CONSULTAS PREVIAS DENTRO DE LA SESIÓN.

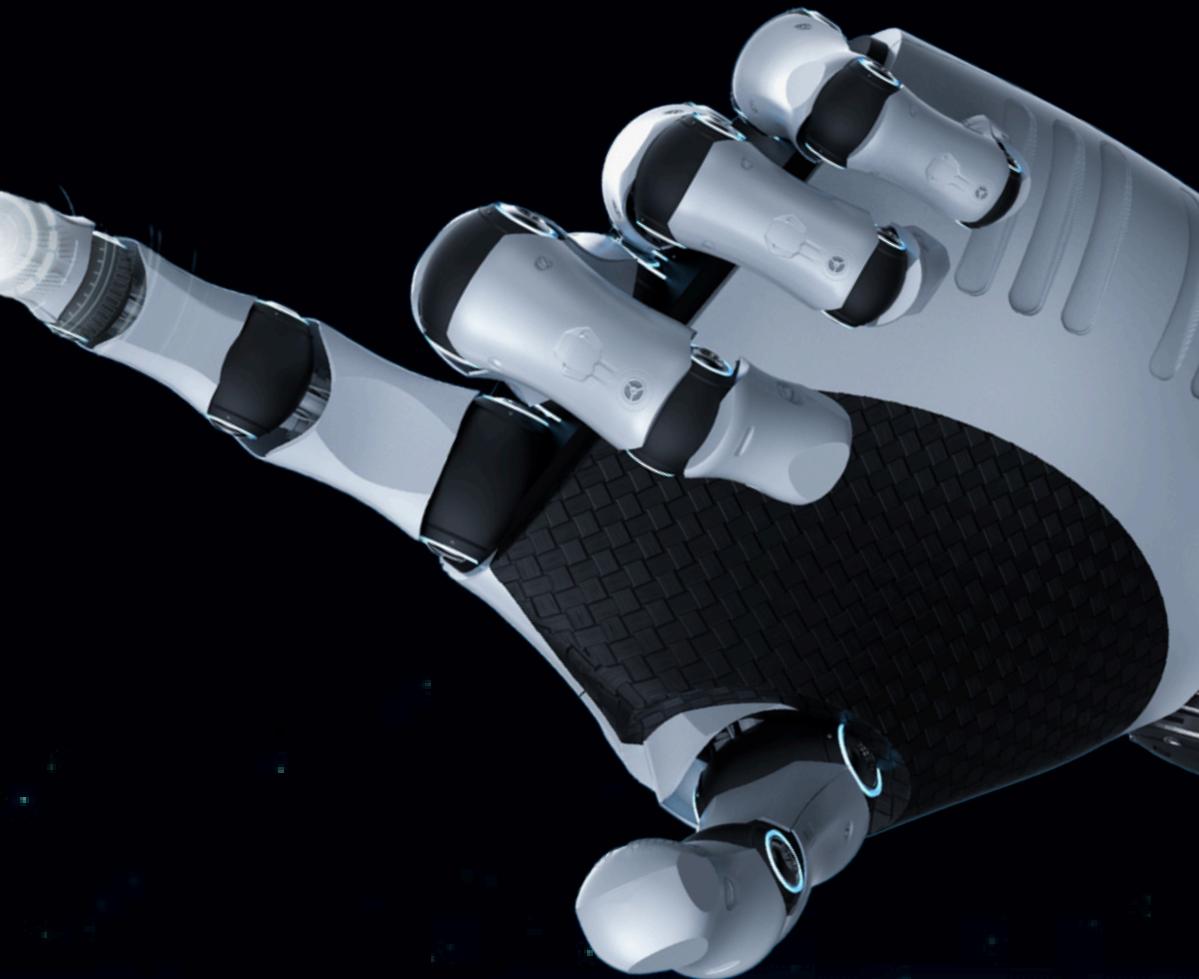
# LOGGING ESTRUCTURADO

*Todas las acciones del agente se registran en logs/agent.log usando formato JSON:*

- *Mensaje generado*
- *Latencia en milisegundos*
- *Tokens usados*
- *Rol (usuario / sistema / asistente)*
- *Identificadores (trace\_id, span\_id)*
- *Herramienta o subagente activado*

*El uso de logging estructurado facilita:*

- *Auditoría de consultas*
- *Diagnóstico de fallos*
- *Medición del costo del modelo (tokens)*
- *Depuración detallada del flujo interno*



# IMPLEMENTACION DE AGENTE

**El agente está diseñado bajo una arquitectura modular**

**Cerebro (Core Engine):** Impulsado por GPT-4o, gestiona el razonamiento y la toma de decisiones.

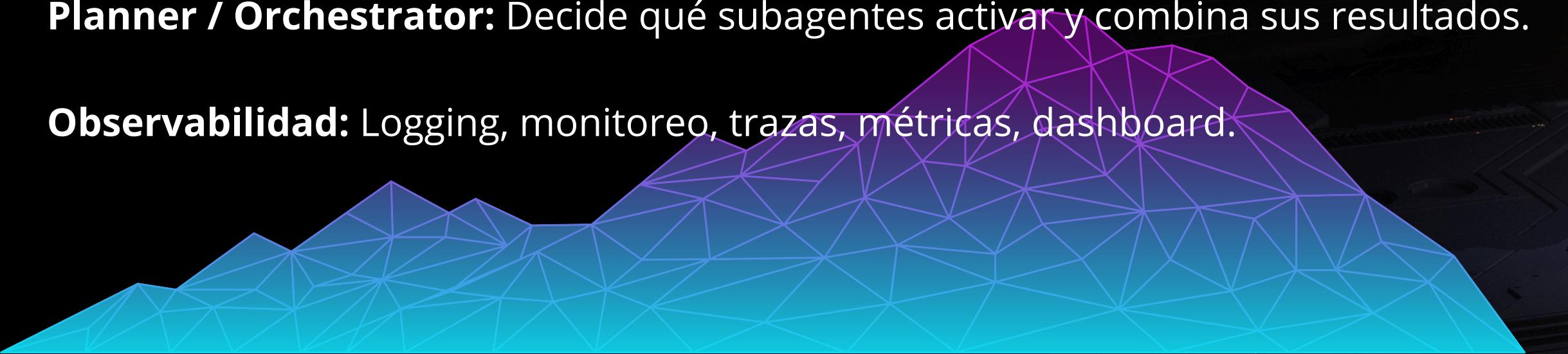
**Memoria (Memory):** Implementada mediante almacenamiento temporal de contexto y registro histórico de conversaciones, garantizando coherencia entre consultas prolongadas.

**Herramientas (Tools):** Acceso a APIs, búsqueda documental y orquestación entre subagentes.

**RAG Local (Document Retriever):** Divide documentos en fragmentos, los puntúa por similitud y devuelve contexto relevante al modelo.

**Planner / Orchestrator:** Decide qué subagentes activar y combina sus resultados.

**Observabilidad:** Logging, monitoreo, trazas, métricas, dashboard.



# OBSERVABILIDAD Y MONITOREO

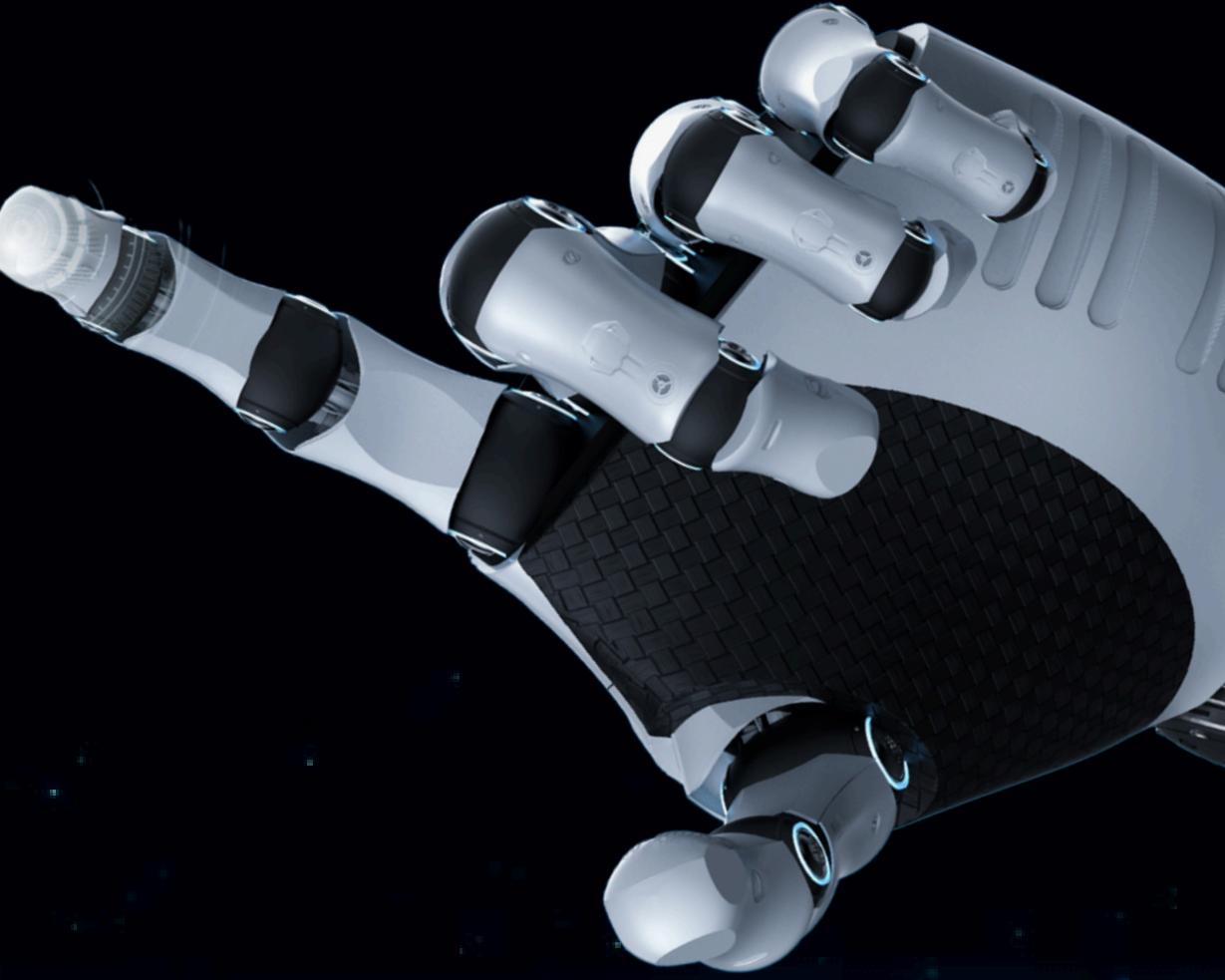
*El agente registra cada interacción usando trazas (traces) que permiten inspeccionar:*

- *Llamadas al LLM (ChatOpenAI → GPT-4o)*
- *Flujo completo del agente (Waterfall)*
- *Latencia por sub-agente*
- *Tokens usados*
- *Errores y excepciones*

*Se utiliza la plataforma LangSmith para:*

- *Depurar el comportamiento del agente*
- *Visualizar los pasos del planner y subagentes*
- *Medir rendimiento y detectar cuellos de botella*
- *Validar respuestas e iteraciones*

*Cada ejecución del agente genera una traza con trace\_id, span\_id y metadatos, permitiendo reconstruir el proceso de decisión.*

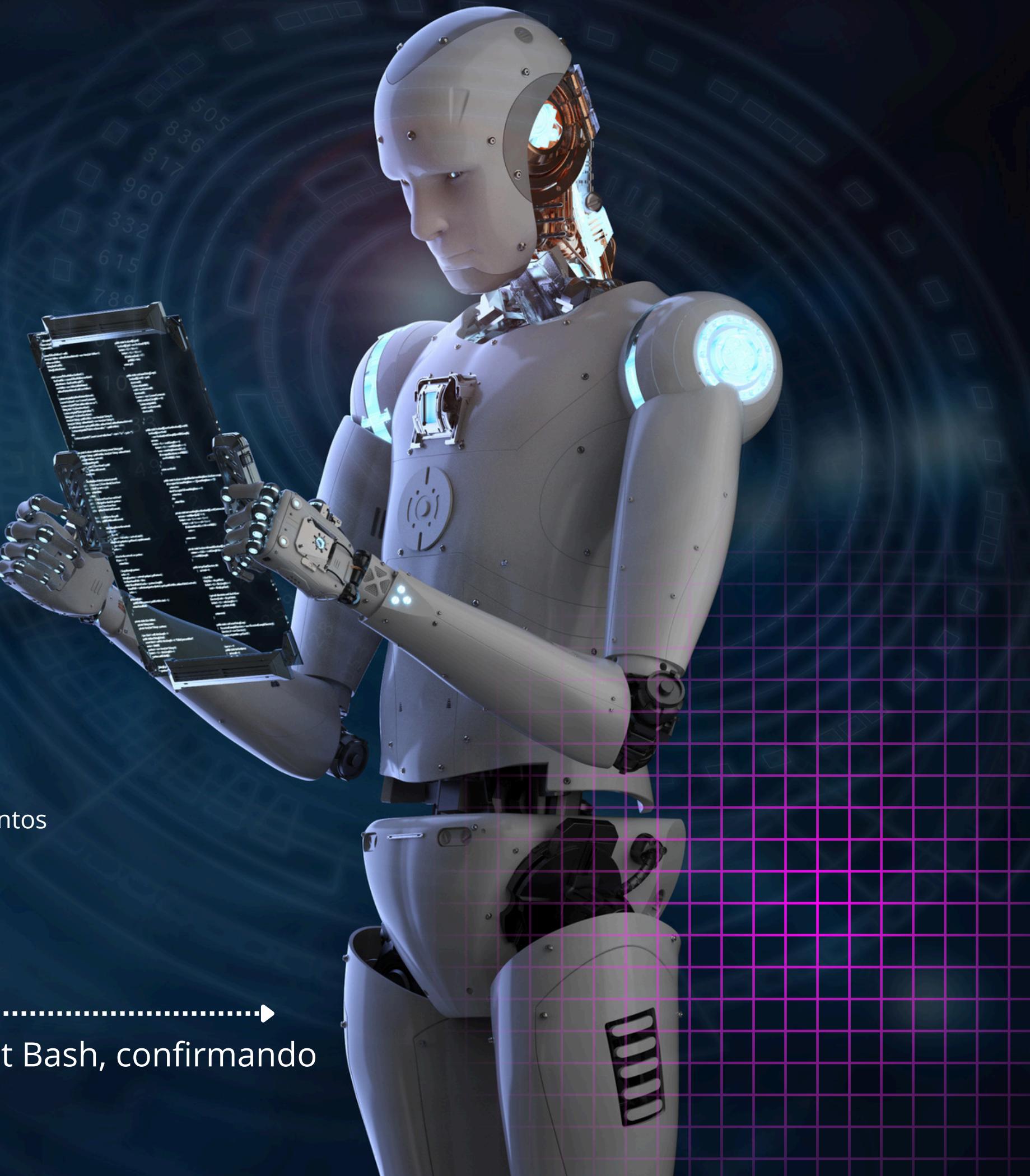


# IMPLEMENTACIÓN

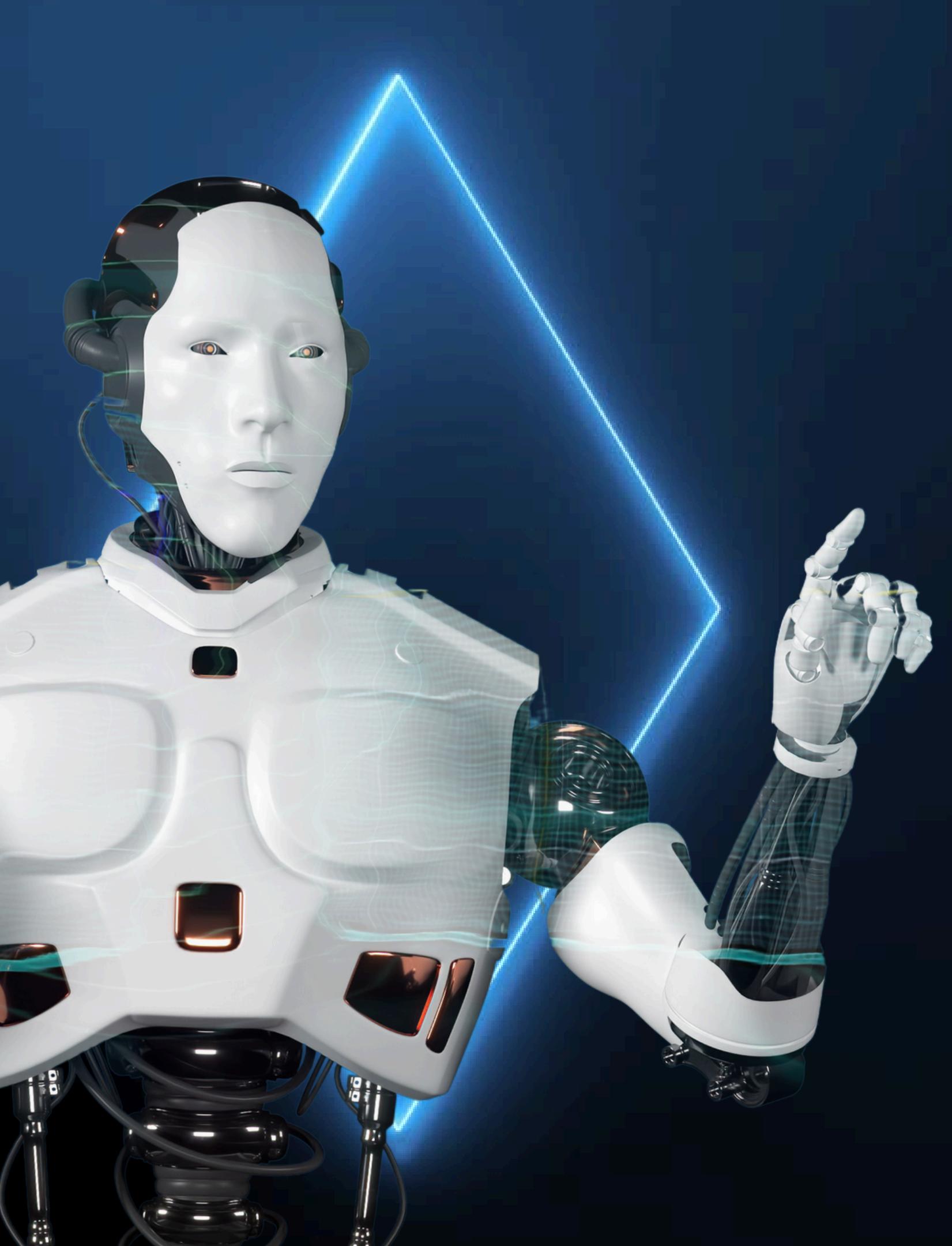
## Estructura:

- `assistant_uni.py` — Controlador principal del agente y punto de entrada del programa.
- `.env` — Variables de entorno, API keys, base URL y configuración del modelo.
- `memory.py` — Módulo encargado de la memoria conversacional.
- `planner_agent.py` — Responsable de la planificación de tareas del agente.
- `observability.py` — Funciones de logging, métricas y trazabilidad (si usas archivo separado).
- `dashboard.py` — Dashboard de observabilidad implementado en Streamlit.
- `data/` — Conjunto de archivos para el sistema RAG (ej.: `becas_2026.txt`, `reglamento_academico.txt`).
- `logs/agent.log` — Archivo de registro JSON estructurado donde se almacenan todos los eventos del sistema.
- `requirements.txt` - Dependencias del proyecto.

El sistema fue probado en entorno virtual (`.venv`) y ejecutado desde Git Bash, confirmando respuestas adaptativas en tiempo real.



# CONCLUSIÓN

A white humanoid robot with glowing blue lines and a glowing blue pyramid in the background.

*El Asistente Universitario Inteligente evolucionó de un prototipo basado en prompting a un agente funcional con orquestación, memoria y planificación.*

*El sistema demuestra autonomía parcial en la toma de decisiones y una estructura modular escalable para entornos educativos. Se proyecta integrar una base de datos persistente y conexión a APIs institucionales para ampliar su capacidad de respuesta.*