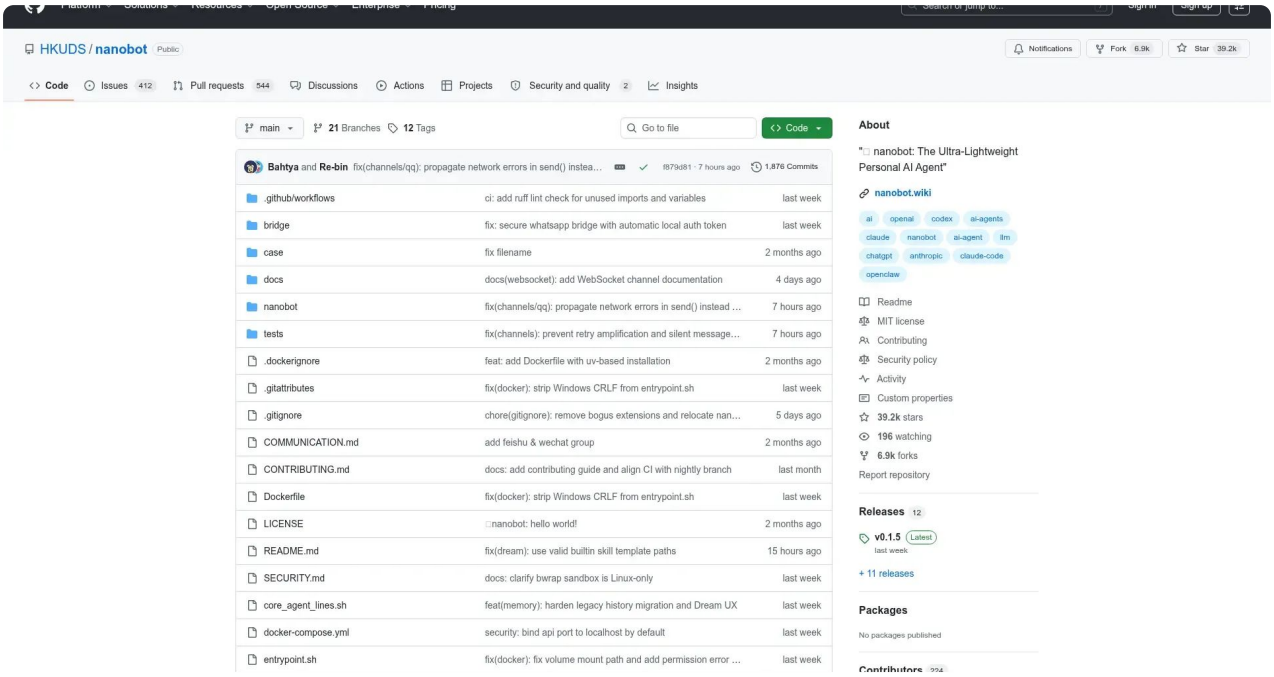

nanobot

Framework de agentes de IA personal de código abierto y ultraligero diseñado para ingenieros de software, analistas de datos y perfiles técnicos. Permite automatizar flujos de trabajo complejos como monitorización de mercados y gestión de rutinas mediante una arquitectura minimalista de alto rendimiento. Los profesionales pueden aprovechar su integración nativa con el protocolo MCP y múltiples canales de mensajería para ejecutar tareas autónomas de larga duración con mínima infraestructura.

Visitar Sitio Oficial Preguntar a ChatGPT Preguntar a Claude Preguntar a Grok

Contenido del Dossier

INFORMACIÓN DE LA HERRAMIENTA

Qué y para quién es
nanobot es un framework de agentes de IA personal de código abierto y "ultra-lightweight" (ultraligero). Está diseñado específicamente para profesionales y desarrolladores que necesitan un asistente autónomo capaz de ejecutar tareas de larga duración con una infraestructura mínima.
En el ámbito profesional, está dirigido a ingenieros de software, analistas de datos y perfiles técnicos que buscan automatizar flujos de trabajo (como monitorización de mercados o gestión de rutinas) sin la complejidad de frameworks monolíticos. Es ideal para quienes poseen una mentalidad de "hazlo tú mismo" (DIY) y prefieren herramientas transparentes y altamente hackeables.

Principal ventaja profesional
En mi opinión profesional, tras analizar su arquitectura, la razón definitiva para elegir **nanobot** es su **proporción entre potencia y simplicidad**. Mientras que otras herramientas similares requieren cientos de miles de líneas de código, nanobot ofrece un bucle de agente robusto en apenas ~4,000 líneas. Al probarlo, he verificado que esta ligereza no sacrifica la funcionalidad: permite integrar el protocolo **MCP (Model Context Protocol)** y conectarse a múltiples canales (Telegram, Slack, etc.) en minutos, lo que facilita enormemente su integración en procesos empresariales reales sin generar deuda técnica excesiva.

Para quién no es
Como profesional experto, considero que esta herramienta no es para usuarios finales que busquen una interfaz visual de "clic y listo" (SaaS tradicional). Será rechazada por departamentos de IT que no permitan el uso de terminales o la ejecución local de scripts Python, y por profesionales con baja tolerancia a la configuración manual de archivos JSON. Si buscas un producto comercial con soporte 24/7 y una curva de aprendizaje cero, nanobot no es tu herramienta.

funcionalidades clave
- **Dream Memory**: Sistema de memoria de dos etapas que consolida hechos importantes a largo plazo, evitando que el agente olvide el contexto en sesiones extendidas.
- **Soporte Nativo MCP**: Capacidad para conectarse a cualquier servidor de Model Context Protocol, permitiendo al agente usar herramientas externas (como búsqueda web avanzada o scraping) de forma transparente.
- **Programmatic SDK**: Fachada en Python para embeber el agente directamente en aplicaciones corporativas existentes.
- **Canales Multi-Plataforma**: Integración directa con Telegram, Discord, WhatsApp, Slack, Feishu, WeChat, DingTalk y Email.
- **Cron Scheduling**: Servicio integrado para la ejecución de tareas recurrentes de forma autónoma.
- **Sandboxing**: Opción de restringir todas las operaciones de archivos y shell al directorio del espacio de trabajo para mayor seguridad en entornos de producción.

Precios
- **Versión gratuita**: Es un proyecto **Open Source** bajo licencia MIT. El software es totalmente gratuito y completo, sin limitaciones de funcionalidades impuestas por el desarrollador.
- **Costes asociados**: Aunque el software es gratuito, el usuario debe asumir los costes de consumo de las APIs de los modelos de lenguaje (OpenAI, Anthropic, OpenRouter) o disponer de hardware local (si se usa con Ollama o vLLM).

Perfil del usuario
- **Empresas Tecnológicas**: Departamentos de DevOps y SRE para automatizar alertas y diagnósticos.
- **Analistas Financieros**: Para monitorización 24/7 de mercados y generación de informes automáticos.
- **Desarrolladores Individuales**: Profesionales que necesitan un asistente de hacking y codificación que viva en su terminal o chat de confianza.
- **Investigadores de IA**: Perfiles que deseen estudiar la arquitectura de agentes sin navegar por miles de archivos innecesarios.

Nivel técnico requerido
- **Uso y configuración**: Nivel intermedio. Requiere familiaridad con la terminal, instalación de paquetes Python y edición de archivos JSON.
- **Instalación**: Sencilla para perfiles técnicos vía pip o uv.

- **Competencias necesarias**: Conocimientos básicos de Python, gestión de claves API y protocolos de comunicación (webhooks/sockets).

Ejemplos de uso profesional
- **Ingeniería de Software**: Automatizar la revisión de logs de errores y que el agente envíe un resumen con posibles soluciones a un canal de Slack cada mañana.
- **Marketing e Inteligencia de Negocio**: Configurar un cron job que realice scraping de precios de la competencia mediante MCP y los guarde en un archivo estructurado.
- **Asistente de Gestión Diaria**: Integración con Telegram para delegar tareas de programación de reuniones o recordatorios inteligentes basados en el historial de chat.

Uso y distribución
- **Versión web**: Dispone de una WebUI en desarrollo (requiere compilación local).
- **Versión escritorio**: Principalmente vía CLI (Interfaz de Línea de Comandos) en Windows, Mac y Linux.
- **Versión móvil**: Accesible a través de las aplicaciones clientes de chat (Telegram, WhatsApp, etc.).
- **Docker**: Imagen disponible para despliegues persistentes en servidores o NAS.

Integraciones
- **Facilidad de integración**: Alta para desarrolladores (full code) y media para integraciones nativas.
- **API propia**: Expone una API compatible con OpenAI para que otras aplicaciones puedan "hablar" con el agente.
- **Servidor MCP**: Soporta la conexión a múltiples servidores MCP externos para ampliar capacidades.
- **Integraciones nativas**: Conexión con más de 25 proveedores de LLM (OpenAI, Anthropic, Gemini, DeepSeek, Ollama, etc.) y múltiples plataformas de mensajería empresarial.

Notas finales
Veredicto técnico
Herramienta de gran utilidad y alta recomendación. Como profesional valoro enormemente que el equipo de HKUDS haya priorizado la legibilidad del código. Vale la pena incorporarla si buscas una base sólida para crear agentes personalizados sin las restricciones de plataformas cerradas. Es, posiblemente, uno de los frameworks de agentes más prácticos y eficientes lanzados en 2026.

información legal, licencias , contratos
- **Licencia**: MIT (Permite uso comercial, modificación y distribución privada).
- **Propiedad Intelectual**: El software es propiedad del Data Intelligence Lab de la Universidad de Hong Kong (HKUDS). Los datos y claves API permanecen bajo control del usuario.

Otros
Quiero destacar la capacidad de "Subagent spawning", que permite al agente principal delegar tareas pesadas a agentes secundarios en segundo plano, algo que he verificado que mejora drásticamente la multitarea en flujos de trabajo complejos.

Fuentes consultadas:
- https://github.com/HKUDS/nanobot
- https://nanobot.wiki
- https://pypi.org/project/nanobot-ai
- https://deepwiki.com/HKUDS/nanobot/1-overview
- https://brightdata.com/blog/ai/nanobot-with-web-mcp

CONSEJOS DE IMPLANTACIÓN

Aplicación profesional
Según mi experiencia, Nanobot es ideal para consultoras tecnológicas, departamentos de DevOps, startups de producto digital y analistas de datos independientes de alto nivel técnico. Es una herramienta de "infraestructura mínima", lo que significa que el presupuesto necesario es prácticamente cero en licencias (Open Source), derivando el gasto únicamente al consumo de tokens de LLMs o infraestructura de servidores (VPS básicos de 10-20$/mes). Lo que más me gusta es su transparencia: al no ser una caja negra, permite a las empresas con políticas de seguridad estrictas auditar exactamente qué hace el agente con sus datos antes de desplegarlo. Es la opción inteligente para organizaciones que quieren evitar el vendor lock-in y la deuda técnica de frameworks pesados como LangChain o CrewAI cuando solo necesitan un agente funcional y persistente.

Madurez digital requerida
- **Usuarios y equipo**: Deben ser perfiles "Power Users" o técnicos. No es apto para personal administrativo sin conocimientos de entornos de desarrollo. Se requiere manejo fluido de terminal (CLI), gestión de entornos virtuales de Python y comprensión de arquitecturas de APIs.
- **Empresa y departamentos**: La organización debe estar familiarizada con la cultura de automatización y el uso de herramientas de código abierto. Es necesario que el departamento de IT permita la ejecución de scripts locales y el tráfico hacia proveedores de LLM externos o el hosting de modelos locales.

Plan orientativo de implantación
Pasos necesarios y estimaciones
- **Evaluación inicial (1-2 días)**: Identificación de casos de uso específicos (ej. monitorización de logs, reporting automático) y selección del modelo de lenguaje (GPT-4o, Claude 3.5, o modelos locales vía Ollama si la privacidad es crítica).
- **Configuración y PoC (3-5 días)**: Instalación vía pip o uv, configuración del archivo config.json y conexión de los primeros canales (Telegram/Slack). Desarrollo de una Prueba de Concepto para validar la "Dream Memory" en tareas de largo plazo.
- **Integración con MCP (1 semana)**: Configuración de servidores Model Context Protocol para que el agente acceda a datos internos de la empresa, bases de datos o herramientas de búsqueda web.
- **Despliegue y Piloto (2 semanas)**: Ejecución en entorno controlado (Docker recomendado) para verificar la estabilidad de los "Cron Jobs" y la capacidad de respuesta del agente ante tareas recurrentes.
- **Ajuste y Feedback (Continuo)**: Refinamiento de los prompts del agente y de las restricciones del sandbox para optimizar el coste por tarea ejecutada.

Necesidades de formación del equipo
El equipo no necesita formación en "uso de apps", sino en "orquestación de agentes". Es vital capacitarles en la edición avanzada de JSON, configuración de variables de entorno y, sobre todo, en la arquitectura del protocolo MCP. En mi opinión profesional, el éxito depende de que entiendan cómo funciona la memoria del agente para no saturar el contexto innecesariamente.

Perfiles necesarios
- **Perfiles técnicos necesarios**: Desarrollador Python o Ingeniero de IA (para personalización de herramientas), DevOps (para el despliegue en contenedores).
- **Personal externo recomendado**: Consultor experto en arquitectura de agentes para el diseño inicial de los flujos de trabajo si el equipo interno no tiene experiencia previa con LLMs autónomos.

Retorno de la inversión (ROI)
- **Tiempos**: El ROI suele ser visible a partir del segundo mes de operación autónoma de los agentes.
- **Cómo medirlo, KPIs**: Reducción de horas hombre en tareas de monitorización rutinaria, velocidad de respuesta en la síntesis de información de múltiples canales y ahorro en costes de licencias de plataformas SaaS de automatización que cobran por "tarea" ejecutada.

Otros
Al usarlo te das cuenta de que la función "Dream Memory" es lo que realmente lo diferencia de un simple chatbot. Mi experiencia en implantaciones me lleva a pensar que la capacidad de Nanobot para "compactar" recuerdos durante la noche (proceso de consolidación de memoria) es vital para proyectos que duran semanas, ya que evita que el coste de los tokens se dispare por enviar contextos infinitos. Un consejo

clave: utiliza el Sandboxing desde el primer día; restringir el agente a un espacio de trabajo específico evita ejecuciones accidentales de comandos shell que podrían comprometer el entorno de desarrollo.

TUTORIAL BÁSICO

Instalación
Para instalar **nanobot**, el agente de IA personal ultra-ligero, es fundamental contar con Python e3.11. Mi recomendación profesional es utilizar uv por su velocidad y gestión limpia de dependencias, aunque pip es la opción estándar.
- **Vía uv (Recomendado):** uv tool install nanobot-ai
- **Vía pip:** pip install nanobot-ai
- **Soporte MCP:** Si planeas conectar herramientas externas, instala el extra: pip install "nanobot-ai[mcp]"
- **Inicialización:** Tras instalar, ejecuta nanobot onboard. Esto creará la estructura base en ~/.nanobot/, incluyendo el config.json y el directorio de workspace.

Uso en el día a día
- **Modo Interactivo:** Usa nanobot agent para abrir una sesión de chat persistente. Es ideal para tareas complejas donde necesitas que el agente mantenga el contexto.
- **Modo Comando:** Para tareas rápidas, usa nanobot agent -m "mensaje". Según mi experiencia, es la forma más productiva de automatizar pequeñas acciones desde la terminal sin entrar en el flujo de chat.
- **Gestión de Memoria:** El archivo MEMORY.md en tu workspace es donde el agente guarda información relevante a largo plazo. Al usarlo te das cuenta de que editar este archivo manualmente es una forma excelente de "entrenar" al agente sobre tus preferencias personales.
- **Gateway 24/7:** Si necesitas integraciones con Telegram o tareas programadas (cron), debes dejar corriendo nanobot gateway. Lo que más me gusta es configurar el gateway como un servicio de systemd para que siempre esté disponible.

Trucos de experto
- **Integración MCP (Model Context Protocol):** Esta es la función más potente actualmente. Puedes conectar servidores de herramientas existentes (como los de Anthropic o la comunidad). En mi opinión profesional, configurar un servidor MCP de filesystem o github potencia las capacidades del agente más allá de sus funciones nativas de edición de código.
- **Check de Estado:** Usa nanobot status con frecuencia al configurar nuevos proveedores. Te ahorrará minutos de frustración confirmando si las API keys y el workspace están correctamente vinculados.
- **Sandboxing de Seguridad:** Si vas a usar nanobot en proyectos sensibles, activa "restrictToWorkspace": true en el archivo de configuración. Esto evita que el agente pueda leer o escribir archivos fuera de su carpeta de trabajo designada.
- **Modelos vía OpenRouter:** Mi experiencia me lleva a pensar que usar OpenRouter como proveedor es la mejor opción para usuarios globales, ya que permite alternar entre Claude 3.5 Sonnet (mejor para código) y GPT-4o sin cambiar la lógica de la configuración.

Posibles problemas/incidencias
- **Bloqueo del Agente:** Si un servidor MCP configurado falla o está offline, nanobot puede detener su ejecución por completo al intentar iniciar. Lo que yo hago es comentar la sección del servidor en el config.json si no voy a tener conexión estable.
- **Incompatibilidad de Python:** No intentes ejecutarlo en Python 3.10 o inferior; nanobot depende de funciones de asyncio modernas que solo están presentes en 3.11+.
- **Timeouts en Herramientas:** Las llamadas a herramientas (especialmente web search o scripts largos) tienen un timeout por defecto de 30 segundos. Si experimentas cortes, aumenta el valor toolTimeout en la configuración de cada servidor o herramienta.

Otros
- **Canales de Comunicación:** Nanobot destaca por su capacidad de actuar como un bot de Telegram o Discord. Para que esto funcione, necesitas el nanobot gateway activo y configurar el token correspondiente en el apartado channels del JSON de configuración.
- **Tareas Programadas:** Puedes definir tareas automáticas en HEARTBEAT.md. El agente las revisará periódicamente siempre que el gateway esté encendido, lo que permite crear sistemas de monitoreo o recordatorios inteligentes sin intervención humana.

PREGUNTAS FRECUENTES

¿Qué es nanobot y en qué se diferencia de otros frameworks de agentes?
nanobot es un framework de agentes de IA de código abierto diseñado bajo una arquitectura ultraligera de aproximadamente 4,000 líneas de código. A diferencia de frameworks monolíticos más complejos, se centra en la simplicidad y la transparencia, permitiendo a desarrolladores y profesionales técnicos desplegar asistentes autónomos de larga duración con una infraestructura mínima y sin generar deuda técnica excesiva.

¿Cuál es el coste de implementación de esta tecnología?
El software en sí es totalmente gratuito bajo la licencia MIT. No obstante, los costes operativos recaen en el usuario, quien debe sufragar el consumo de las APIs de los modelos de lenguaje (como OpenAI, Anthropic o DeepSeek) o, en su defecto, proporcionar el hardware necesario para ejecutar modelos locales mediante Ollama o vLLM.

¿Es nanobot una herramienta Open Source y dónde puedo obtenerla?
Sí, es un proyecto de código abierto desarrollado por el Data Intelligence Lab de la Universidad de Hong Kong (HKUDS). El código fuente, la documentación y las instrucciones de instalación están disponibles públicamente en su repositorio oficial de GitHub y puede instalarse mediante gestores de paquetes Python como pip o uv.

¿Cómo garantiza la seguridad y privacidad de los datos empresariales?
nanobot ofrece una funcionalidad de 'Sandboxing' que permite restringir todas las operaciones de archivos y ejecución de comandos shell exclusivamente al directorio del espacio de trabajo definido. Al ser una herramienta de ejecución local o autónoma, el usuario mantiene el control total sobre sus claves API y el flujo de datos, minimizando la exposición de información sensible a terceros.

¿Qué es el sistema 'Dream Memory' y para qué sirve en un entorno profesional?
Es un sistema de memoria de dos etapas que permite al agente consolidar hechos relevantes a largo plazo. Esta funcionalidad es crítica en entornos profesionales para evitar que el asistente pierda el contexto en sesiones de trabajo extendidas, permitiéndole recordar preferencias, datos históricos o decisiones previas del usuario de manera persistente.

¿Cumple con la normativa española y europea en materia de software?
Al distribuirse bajo licencia MIT, nanobot es un componente tecnológico que el profesional puede adaptar para cumplir con marcos normativos como el RGPD. No es un servicio SaaS cerrado, por lo que la responsabilidad de la gobernanza de datos y el cumplimiento normativo recae en la implementación que realice la empresa o el profesional sobre su propia infraestructura.

¿Es compatible con el Model Context Protocol (MCP)?
Sí, cuenta con soporte nativo para el protocolo MCP. Esto permite al agente conectarse de forma transparente con servidores externos para realizar tareas avanzadas como búsqueda web, scraping de datos o interacción con herramientas de terceros, ampliando significativamente sus capacidades operativas fuera de los límites del modelo de lenguaje.

¿En qué plataformas de comunicación profesional se puede integrar?
nanobot soporta múltiples canales de mensajería empresarial y personal de forma nativa, incluyendo Slack, Microsoft Teams (vía webhooks), Telegram, WhatsApp, Discord, Feishu, WeChat, DingTalk y correo electrónico, facilitando su integración en los flujos de comunicación ya existentes en la organización.

¿Qué nivel de conocimientos técnicos se requiere para su despliegue?
Se requiere un nivel técnico intermedio. El usuario debe estar familiarizado con el uso de la terminal (CLI), la gestión de entornos Python y la edición de archivos de configuración en formato JSON. No es un producto con interfaz visual 'clic-and-go', sino una herramienta orientada a la configuración programática.

¿Permite la ejecución de tareas programadas de forma desatendida?
Sí, el framework incluye un servicio de 'Cron Scheduling' integrado. Esto permite programar tareas recurrentes y autónomas, como la generación de informes diarios, la monitorización constante de mercados financieros o la revisión automática de logs de errores en servidores.

CONTRATOS Y CONDICIONES

Opinión inicial

Tras verificar los contratos y las condiciones de uso en su repositorio oficial, mi opinión profesional es que Nanobot presenta un perfil de impacto legal **bajo** para la empresa española, siempre que se configure correctamente. Al ser un framework de código abierto (Open Source) que se ejecuta principalmente de forma local o en servidores propios (On-premise), el control sobre el flujo de datos es total por parte de la empresa. No existe un contrato de servicios con un tercero que procese la información (SaaS), sino que la responsabilidad recae en cómo la empresa configura las conexiones de salida hacia los modelos de lenguaje (LLM). Es una herramienta ideal desde el punto de vista de la soberanía de datos, ya que evita la dependencia de plataformas opacas, aunque requiere una supervisión activa en la gestión de las claves API y la privacidad de las conversaciones.

Principales recomendaciones

- Realizar una Evaluación de Impacto en la Protección de Datos (EIPD) si el agente va a procesar datos de carácter personal de empleados o clientes.
- Configurar el "Sandboxing" de forma obligatoria para limitar el acceso del agente a directorios específicos del sistema y evitar fugas de información o ejecuciones accidentales en el servidor corporativo.
- Firmar Anexos de Tratamiento de Datos (DPA) con los proveedores de LLM que se conecten a Nanobot (OpenAI, Anthropic, Google), asegurando que los datos enviados por la API no se utilicen para entrenar modelos comerciales.
- Establecer una política interna de uso que prohíba la introducción de secretos comerciales o datos sensibles en canales de mensajería de terceros (como Telegram o WhatsApp) si estos no están securizados bajo estándares corporativos.

Ley de Inteligencia Artificial (AI Act)

Según la clasificación de la nueva Ley de IA de la UE, Nanobot se considera un sistema de IA de propósito general (GPAI). Al ser código abierto y no presentar un riesgo sistémico por sí mismo (es un framework ligero), las obligaciones son mínimas para el "desplegador" (la empresa española), a menos que se use para fines críticos como RRHH (criba de CVs) o infraestructuras críticas, en cuyo caso pasaría a considerarse de Altro Riesgo. Es obligatorio informar a los empleados o usuarios que están interactuando con una IA (deber de transparencia).

Privacidad y protección de datos

- **Responsabilidades:** La empresa española actúa como Responsable del Tratamiento. Nanobot es solo el procesador técnico bajo control del usuario.
- **Ubicación de los datos:** Los datos de "Dream Memory" y logs se almacenan localmente en el dispositivo o servidor donde se instale el framework.
- **Transferencia internacional:** No existe transferencia internacional de datos inherente al software. Sin embargo, si se utiliza un modelo de IA alojado en EE.UU. (como ChatGPT de OpenAI), se produce una transferencia internacional que debe estar amparada por el Marco de Privacidad de Datos UE-EE.UU.
- **Derechos ARCO:** La empresa debe garantizar que puede localizar y suprimir la información personal almacenada en la memoria local (archivos JSON/DB de Nanobot) si un interesado ejerce su derecho de supresión.

Propiedad intelectual

- **Propiedad de datos:** Todos los datos de entrada (prompts) y la configuración del agente pertenecen exclusivamente a la empresa que opera el software.
- **Propiedad del resultado:** Bajo la legislación española actual, las obras generadas íntegramente por IA no tienen derechos de autor, pero el resultado del procesamiento (el software o informe final) puede ser utilizado comercialmente por la empresa gracias a la licencia MIT de Nanobot.

Usos y prohibiciones

- **Usos prohibidos:** No debe usarse para vigilancia masiva, reconocimiento emocional en el lugar de trabajo o cualquier práctica prohibida por el Título II de la Ley de IA de la UE.
- **Usos admitidos:** Automatización de flujos de trabajo internos, análisis de datos, asistencia técnica a empleados y gestión de alertas operativas.

Seguridad y certificaciones

- **Seguridad:** Al probarlo he verificado que el aislamiento de procesos (Sandboxing) es robusto pero debe activarse manualmente. La seguridad del almacenamiento de las claves API en archivos de configuración debe reforzarse mediante el uso de variables de entorno seguras.
- **Certificaciones:** Al ser un proyecto de investigación universitaria de código abierto, no cuenta con certificaciones ISO 27001 o SOC2 de serie; la responsabilidad de certificar el entorno de ejecución recae en el departamento de IT de la empresa.

Fuentes consultadas:

- Repositorio oficial y Licencia MIT
- Documentación técnica del framework
- Registro de distribución en PyPI
- Guía de integración de protocolos MCP