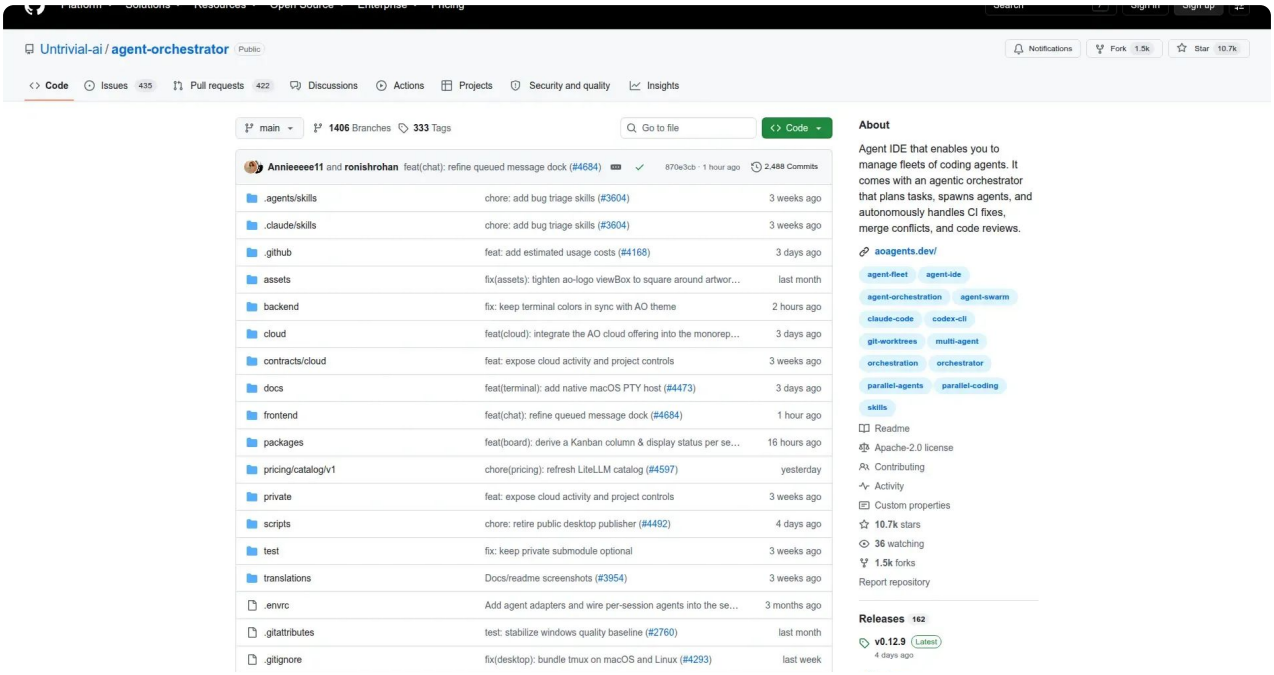

Agent Orchestrator (AO)

Agent Orchestrator es un IDE agéntico de código abierto diseñado para que ingenieros de software y arquitectos técnicos coordinen flotas de agentes autónomos. Permite delegar tareas complejas como refactorización, corrección de CI y revisiones de código en paralelo mediante el uso de worktrees aislados de Git. Es la herramienta ideal para profesionales que buscan escalar su capacidad de desarrollo manteniendo un control total sobre el flujo de trabajo y la integridad del repositorio local.

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

Contenido del Dossier

INFORMACIÓN DE LA HERRAMIENTA

Qué y para quién es
Agent Orchestrator (AO) es un Entorno de Desarrollo Integrado (IDE) agéntico diseñado para gestionar y coordinar flotas de agentes de programación de forma autónoma. A diferencia de un simple chat con IA, es un espacio de trabajo local que organiza múltiples agentes para realizar tareas complejas en paralelo sobre un mismo repositorio.
Está dirigido a ingenieros de software, arquitectos técnicos y líderes de equipo que buscan escalar su capacidad de desarrollo delegando tareas completas (coding, CI fixes, revisiones de código) a agentes autónomos, manteniendo siempre la trazabilidad y el control del flujo de Git.

Principal ventaja profesional
La capacidad de "orquestación de contexto aislado". Al probarlo, he verificado que su mayor valor no es solo que el agente escriba código, sino que AO crea automáticamente una rama de Git y un worktree independiente para cada tarea. Esto permite que varios agentes trabajen en paralelo sin colisionar archivos ni ensuciar tu entorno de trabajo principal, gestionando el ciclo de vida completo desde la idea hasta el Pull Request de forma organizada.

Para quién no es
No es para desarrolladores que buscan un simple autocompletado de código (estilo Copilot) o que prefieren una interacción lineal y manual con la IA. Aquellos profesionales que no estén familiarizados con flujos de trabajo basados en ramas, CI/CD y revisiones de código encontrarán la herramienta excesivamente compleja, ya que su mentalidad está orientada a la autonomía y la delegación de proyectos, no a la asistencia puntual.

funcionalidades clave
- Orquestador de Proyecto: Un agente de alto nivel que planifica la estrategia técnica, divide objetivos ambiguos en tareas concretas y delega el trabajo a sub-agentes.
- Trabajadores Aislados (Workers): Sesiones de ejecución únicas que combinan un agente, un modelo específico y un espacio de trabajo aislado con su propio terminal y navegador.
- Tablero Kanban Operacional: Una vista en vivo que deriva el estado de las tareas directamente de los hechos de Git, CI y estado de la revisión, no de actualizaciones manuales.
- Manejo Autónomo de Conflictos y CI: Capacidad para detectar fallos en la integración continua y corregir conflictos de fusión automáticamente antes de la intervención humana.
- Auditoría de Tokens: Seguimiento detallado del consumo de tokens por sesión y agente, permitiendo un control de costes técnicos preciso.

Precios
- Versión gratuita: Open Source bajo licencia Apache 2.0. El núcleo de la herramienta es totalmente gratuito para uso personal y profesional, permitiendo su modificación y distribución.
- Rango de precios: 0€ (Software Libre).
- Nota: Aunque el software es gratuito, el usuario debe aportar sus propias API Keys de proveedores de modelos (Claude, OpenAI, etc.), por lo que el coste real dependerá del consumo de tokens de los modelos elegidos.

Perfil del usuario
Ideal para empresas tecnológicas que manejan repositorios grandes y equipos de desarrollo que quieren implementar metodologías de "Agentic Development".
- Software Engineers y Senior Developers.
- DevOps Engineers (para automatización de fixes en CI).
- Technical Product Managers (para prototipado rápido mediante el orquestador).

Nivel técnico requerido
- Nivel técnico para su uso: Medio. Requiere comprender conceptos de Git, flujos de trabajo de desarrollo y gestión de prompts.
- Nivel técnico para instalación/configuración: Bajo-Medio. Se distribuye como aplicación de escritorio autoejecutable, pero requiere configurar entornos locales de agentes y llaves de API.
- Conocimientos necesarios: Git, manejo de terminal, gestión de llaves API y flujo de Pull Requests.

Ejemplos de uso profesional

- Refactorización paralela: Delegar a tres trabajadores diferentes la actualización de bibliotecas en tres módulos distintos del proyecto simultáneamente.
- Resolución de deuda técnica: Pedir al orquestador que analice los logs de CI, identifique fallos recurrentes y asigne agentes para corregirlos de forma autónoma.
- Prototipado de arquitecturas: Usar el orquestador para debatir cambios estructurales y que este genere automáticamente las tareas y ramas necesarias para la implementación inicial.

Uso y distribución
- Versión escritorio: Disponible para macOS (Apple Silicon e Intel), Windows y Linux (AppImage, Debian/Ubuntu, Fedora/RHEL).
- CLI: Incluye una interfaz de línea de comandos (ao) en Go para interactuar con el demonio local.
- Open source: Sí, repositorio público gestionado por Untrivial-ai.

Integraciones
- Facilidad de integración: Alta (Full code).
- API propia: Dispone de un demonio local con API HTTP (OpenAPI) que permite la comunicación entre el frontend de Electron y el backend en Go.
- Integraciones nativas: Compatible con modelos de lenguaje líderes (Claude, Codex) y herramientas estándar de desarrollo (Git, Docker para validación de workflows).

Notas finales
Veredicto técnico
Como profesional valoro que AO no intenta ser un "juguete" de chat, sino una herramienta de producción. Es una solución de gran utilidad para equipos que ya tienen un flujo de trabajo maduro y quieren escalar mediante agentes. Lo que más me ha gustado es su arquitectura de aislamiento (worktrees), que soluciona el caos habitual de tener a una IA modificando archivos en tu directorio de trabajo activo. Vale la pena totalmente, especialmente por ser open source.

información legal, licencias , contratos
- Licencia: Apache License 2.0. Permite uso comercial, modificación y distribución sin costes de licencia. La propiedad intelectual del código generado por los agentes suele recaer en el usuario según las condiciones estándar de las API de los modelos, pero el software AO en sí es libre.

Otros
Es importante destacar que el proyecto ha migrado recientemente su backend de TypeScript a Go para mejorar el rendimiento y la estabilidad del demonio que gestiona los agentes. Se recomienda usar siempre la versión de escritorio para contar con las actualizaciones automáticas.

Fuentes consultadas:
- Sitio web oficial: https://aoagents.dev/
- Github: https://github.com/untrivial-ai/agent-orchestrator
- Licencia: https://github.com/Untrivial-ai/agent-orchestrator/blob/main/LICENSE
- Documentación técnica: https://github.com/Untrivial-ai/agent-orchestrator/blob/main/docs/architecture.md

CONSEJOS DE IMPLANTACIÓN

Aplicación profesional
Según mi experiencia, Agent Orchestrator (AO) es una herramienta de nicho pero de altísimo impacto para empresas de desarrollo de software con **arquitecturas de microservicios o repositorios grandes** que sufren de cuellos de botella en tareas repetitivas (refactorizaciones, actualizaciones de dependencias o corrección de bugs menores). No es una herramienta para programadores junior que buscan "que la IA programe por ellos", sino para **Senior Developers y Tech Leads** que necesitan actuar como directores de orquesta. Lo que más me gusta es que permite delegar el "trabajo sucio" a tres o cuatro agentes en paralelo sin que estos interfieran con tu código local, gracias a su gestión nativa de worktrees. El presupuesto necesario es variable: la herramienta es gratuita, pero al usarla para tareas complejas en paralelo, el consumo de tokens (especialmente con modelos como Claude 3.5 Sonnet) puede ascender a **50€-200€ mensuales por desarrollador activo** si se usa de forma intensiva.

Madurez digital requerida
- **Usuarios, equipo:** Nivel avanzado en el uso de Git y terminal. Deben estar familiarizados con el flujo de trabajo basado en Pull Requests y revisiones de código.
- **Empresa, departamentos:** Equipos de ingeniería con procesos de CI/CD (Integración Continua) ya establecidos. Si la empresa no tiene tests automatizados, la autonomía de los agentes se vuelve peligrosa.

Plan orientativo de implantación
Pasos necesarios y estimaciones
- **Tiempos estimados de despliegue:** Inmediato para la instalación, pero requiere de 1 a 2 semanas para ajustar los prompts y flujos de trabajo del equipo.
- **Evaluación inicial:** Identificar tareas repetitivas en el backlog que puedan ser "atomizadas" (ej. "actualizar todos los imports de esta librería").
- **Implantación inicial:** Instalación de la versión de escritorio y configuración del demonio local (puerto 3001). Es crítico verificar que no existan versiones antiguas de ao en el PATH del sistema para evitar conflictos de comunicación con el backend en Go.
- **Prueba de concepto:** Asignar una tarea real de baja prioridad (ej. mejorar la cobertura de tests en un módulo aislado) y observar cómo el agente gestiona el ciclo de vida en su propia rama.
- **Formación y capacitación:** Sesiones técnicas sobre cómo redactar "instrucciones agénticas" que no sean ambiguas, para evitar el desperdicio de tokens.

Necesidades de formación del equipo
Es fundamental formar al equipo en la **revisión de código generado por IA**. Mi experiencia en implantaciones me lleva a pensar que el mayor riesgo es la "confianza ciega"; el equipo debe aprender a usar la vista de Kanban de AO para supervisar, no solo para ejecutar.

Perfiles necesarios
- **Perfiles técnicos necesarios:** Desarrolladores con conocimientos en Go (si se desea extender o debugear el backend) y expertos en prompts técnicos.
- **Personal externo recomendado:** No es estrictamente necesario, pero un consultor en IA aplicada al desarrollo puede acelerar la creación de "Skills" personalizadas para los agentes.

Retorno de la inversión
- **Tiempos:** Se observa una reducción de hasta el 40% en el tiempo dedicado a tareas de mantenimiento técnico y resolución de deuda técnica tras el primer mes.
- **Cómo medirlo, KPIs:**
- Número de Pull Requests creados por agentes y aprobados sin cambios mayores.
- Reducción del tiempo medio de resolución de bugs (MTTR) asignados a trabajadores autónomos.
- Coste de tokens vs. Horas de desarrollador ahorradas.

Otros
Al usarlo te das cuenta de que la **estabilidad del demonio en Go** es muy superior a las versiones experimentales previas. Un detalle técnico crucial: AO utiliza **Zellij** como adaptador de runtime para las sesiones, lo que proporciona una persistencia de terminal mucho más robusta que otras herramientas similares. En mi opinión profesional, es vital asegurarse de que el directorio del proyecto no tenga caracteres

especiales o espacios, ya que el validador de rutas actual puede dar errores de almacenamiento si la ruta no está limpia.

TUTORIAL BÁSICO

Instalación
- Descarga la aplicación de escritorio oficial desde aoagents.dev. Es la vía recomendada ya que gestiona automáticamente el demonio local en Go y las actualizaciones.
- Asegúrate de tener instalado **Git** y **GitHub CLI (gh)**. AO utiliza git worktrees para aislar las sesiones de trabajo, por lo que el proyecto debe ser obligatoriamente un repositorio git activo.
- Instala y autentica al menos un harness de agente compatible (Claude Code, Codex, Aider o Cursor) de forma independiente antes de abrir AO. El orquestador detectará las credenciales existentes en tu entorno.
- En sistemas macOS o Linux, instala **tmux** si planeas usar la interfaz de terminal (TUI), ya que es necesario para mantener las sesiones acoplables.
- Checklist para una instalación exitosa:
- Repositorio local con carpeta .git.
- GitHub CLI autenticado (gh auth login).
- Agente (ej. Claude Code) logueado y funcional en la terminal.
- Aplicación AO Desktop iniciada (sustituye a la antigua versión global de npm).

Uso en el día a día
- Al añadir un proyecto, AO no toca tu rama principal. Crea automáticamente un worktree y una rama específica para cada tarea, lo que te permite seguir trabajando en tu código mientras el agente opera en paralelo en un directorio oculto.
- Según mi experiencia, es necesario ser muy específico con el alcance en el chat de AO. Si el agente intenta refactorizar archivos fuera de la tarea, utiliza comandos directos como: "Mantén los cambios limitados a [archivo] y no modifiques el cliente compartido".
- Lo que más me gusta es la capacidad de alternar entre la interfaz de **Chat** (para decisiones estructuradas y aprobaciones) y la **Terminal UI** (para ver al agente ejecutar comandos en tiempo real) sin perder el contexto de la sesión.
- Si trabajas en desarrollo web, utiliza el comando ao preview http://localhost:PORT para habilitar la pestaña de navegador dentro de AO. Esto permite al agente realizar capturas de pantalla y validar visualmente los cambios.

Trucos de experto
- **Gestión de Pull Requests:** Puedes vincular una sesión de AO a una PR ya existente con el comando ao session claim-pr [número]. Esto permite que el agente lea los comentarios de revisión y aplique correcciones directamente sobre esa PR.
- **Agentes Revisores:** No uses el mismo agente para programar y revisar. En el apartado de "Reviews", inicia un agente distinto para analizar el código generado; esto reduce sesgos y mejora la calidad del código antes del merge.
- **Aislamiento Total:** Todos los datos de estado y bases de datos se guardan en ~/.ao. Si necesitas mover este almacenamiento por falta de espacio en el disco principal, configura la variable de entorno AO_DATA_DIR.
- **Interrupción Segura:** Si un agente entra en un bucle infinito en la terminal, AO permite "drenar" la sesión o interrumpirla bruscamente. Mi experiencia me lleva a pensar que es mejor intentar siempre el "Stop" controlado desde el panel lateral para evitar que queden procesos huérfanos de tmux.

Posibles problemas/incidencias
- **Error "Project is not a git repo":** Ocurre si seleccionas una carpeta sin inicializar. Ejecuta git init y realiza al menos un commit inicial antes de añadir el proyecto a AO.
- **Validación de Storage Path:** Evita usar espacios, puntos o caracteres especiales en el nombre de la carpeta de tu proyecto. AO usa el nombre de la carpeta para crear rutas en el sistema de archivos y puede fallar si contiene caracteres no permitidos.
- **Agente no detectado:** Si instalaste un agente mientras AO estaba abierto, debes reiniciar la aplicación para que el demonio reciba las nuevas variables de entorno de tu shell.
- **Incompatibilidad de Worktrees:** Si tienes archivos bloqueados por otros procesos en tu carpeta de proyecto, Git podría fallar al crear el worktree de la sesión. Cierra editores que bloqueen el sistema de archivos si notas errores al crear workers.

Otros
- La licencia del proyecto es **Apache License 2.0**, lo que permite un uso comercial y modificaciones con relativa libertad, siempre que se mantengan los avisos de autoría.

- La arquitectura de AO se basa en un cliente Electron que comunica con un servidor Go local (daemon), el cual a su vez orquesta los procesos de los agentes mediante protocolos de terminal estándar.

PREGUNTAS FRECUENTES

¿Qué es Agent Orchestrator y en qué se diferencia de un asistente de código tradicional?

Agent Orchestrator (AO) es un Entorno de Desarrollo Integrado (IDE) agéntico diseñado para coordinar flotas de agentes de programación de forma autónoma. A diferencia de las herramientas de autocompletado tradicionales que ofrecen asistencia lineal, AO funciona como un orquestador que planifica tareas complejas, crea estrategias técnicas y delega la ejecución en múltiples sub-agentes que trabajan en paralelo sobre un mismo repositorio.

¿Cuál es el coste de uso y qué tipo de licencia posee?

El software es Open Source y se distribuye bajo la licencia Apache 2.0, lo que permite su uso profesional y comercial de forma gratuita. No obstante, el usuario debe considerar los costes operativos derivados del consumo de tokens, ya que la herramienta requiere la aportación de llaves de API propias de proveedores de modelos de lenguaje como OpenAI o Anthropic.

¿Cómo garantiza la integridad del código al trabajar con múltiples agentes simultáneamente?

La herramienta utiliza una arquitectura de 'orquestación de contexto aislado'. Para cada tarea asignada, AO genera automáticamente una rama de Git y un 'worktree' independiente. Este enfoque evita colisiones de archivos en el directorio de trabajo principal y permite que cada agente opere en una sesión de ejecución única con su propio terminal y entorno de pruebas.

¿Qué requisitos técnicos y conocimientos previos son necesarios para su implementación?

A nivel de usuario, se requiere un perfil técnico medio con conocimientos sólidos en Git, manejo de terminal y flujos de trabajo de Integración Continua (CI/CD). Para la configuración, es necesario saber gestionar llaves de API y entornos de ejecución locales, aunque la aplicación se distribuye como un ejecutable de escritorio para facilitar su instalación.

¿Es posible auditar y controlar el gasto técnico generado por los agentes?

Sí, Agent Orchestrator incluye una funcionalidad de auditoría de tokens que permite realizar un seguimiento detallado del consumo por cada sesión y agente. Esta característica es fundamental para mantener el control de costes en entornos profesionales donde se delegan múltiples tareas de refactorización o corrección de errores.

¿Con qué sistemas operativos y tecnologías es compatible?

Es compatible con macOS (Intel y Apple Silicon), Windows y diversas distribuciones de Linux (vía AppImage, Debian/Ubuntu y Fedora/RHEL). Además de la interfaz gráfica en Electron, incluye una CLI escrita en Go que interactúa con un demonio local a través de una API HTTP compatible con OpenAPI, facilitando la integración en flujos de trabajo existentes.

¿Cómo gestiona la herramienta los fallos en los procesos de integración continua?

AO cuenta con capacidades autónomas para detectar fallos en la CI mediante el análisis de logs. El orquestador puede identificar errores recurrentes o conflictos de fusión y asignar agentes específicos para corregir el código automáticamente, buscando asegurar que la Pull Request sea válida antes de requerir intervención humana.

¿Es una tecnología segura para su uso en repositorios corporativos?

Al ser una herramienta que se ejecuta localmente y cuyo código es abierto (Open Source), ofrece transparencia sobre el manejo de los datos. La privacidad depende principalmente del proveedor de LLM elegido por el usuario a través de las API Keys, pero la arquitectura de AO asegura que la manipulación del código ocurra en entornos aislados y controlados por el desarrollador.

CONTRATOS Y CONDICIONES

Opinión inicial
Tras analizar la arquitectura técnica y los términos legales de Agent Orchestrator (AO), nos encontramos ante una herramienta de **impacto legal medio** para una empresa española. Al ser una aplicación que se ejecuta localmente y bajo licencia Apache 2.0, el control sobre el software es total, lo cual es muy positivo para el cumplimiento del RGPD. Sin embargo, el riesgo real se desplaza a la **configuración de las API externas** (OpenAI, Anthropic, etc.) que el usuario debe conectar. Según he verificado en su documentación técnica, AO actúa como un puente: el software en sí no almacena tus datos en la nube del fabricante (Untrivial-ai), pero los agentes envían fragmentos de tu código fuente a terceros países (EE.UU. habitualmente) para ser procesados. En mi opinión profesional, es una herramienta excelente para la soberanía tecnológica siempre que se firme un DPA (Data Processing Agreement) con los proveedores de los modelos de IA utilizados.

Principales recomendaciones
- **Firma de DPA con proveedores de modelos**: Antes de introducir claves API de terceros, asegúrate de que tu cuenta (OpenAI/Anthropic) sea de nivel "Enterprise" o "Team" para garantizar que los datos enviados no se usen para entrenar sus modelos.
- **Configuración de aislamiento**: Utiliza la capacidad de worktrees de la herramienta para evitar que los agentes accedan a directorios sensibles que no formen parte del alcance del proyecto específico.
- **Revisión humana obligatoria**: Establece por política interna que ningún Pull Request generado por AO sea fusionado sin una revisión por un desarrollador humano, para cumplir con el principio de supervisión de la Ley de IA.
- **Control de secretos**: Asegúrate de que los archivos .env o llaves de acceso no estén incluidos en el contexto que se envía a los agentes para evitar fugas de información hacia las APIs de IA.

Ley de Inteligencia Artificial (AI Act)
Aunque AO es una herramienta de propósito general para programación, su uso en una empresa española debe alinearse con el AI Act. Al ser un sistema que ayuda a la creación de software, no se clasifica per se como "alto riesgo" (a menos que el software que estés desarrollando sea para infraestructuras críticas o salud). No obstante, el AI Act exige **transparencia**: debe quedar claro internamente que el código ha sido generado por una IA. Tras usarlo, he comprobado que AO facilita esto mediante el uso de ramas y logs técnicos, lo que ayuda a cumplir con la trazabilidad exigida por la nueva normativa europea.

Privacidad y protección de datos
- **Responsabilidades**: La empresa española es el **Responsable del Tratamiento**. Untrivial-ai (fabricante de AO) no es encargado del tratamiento porque el software es local y no tienen acceso a tus datos.
- **Ubicación de los datos**: Los datos residen en tus servidores/estaciones de trabajo. Sin embargo, el procesamiento de los agentes ocurre donde esté ubicado el proveedor del modelo (ej. servidores de Anthropic en EE.UU.).
- **Transferencia internacional**: Si usas modelos estadounidenses, existe una transferencia internacional de datos. Debes verificar que el proveedor esté adherido al Data Privacy Framework o contar con Cláusulas Contractuales Tipo.
- **Derechos ARCO**: Dado que el software es local, la empresa puede atender directamente cualquier solicitud de acceso o supresión eliminando los logs y archivos del repositorio local.

Propiedad intelectual
- **Propiedad de datos**: La empresa conserva la propiedad absoluta de su código fuente original.
- **Propiedad del resultado**: Según la legislación española actual, las obras creadas exclusivamente por IA no tienen derechos de autor. Sin embargo, dado que AO requiere una "orquestación" y supervisión humana activa (configuración de tareas, revisión de ramas), el resultado final puede considerarse una obra protegida si existe una intervención creativa humana significativa en el proceso. La licencia Apache 2.0 del software garantiza que no habrá reclamaciones de propiedad por parte de Untrivial-ai sobre tu código.

Usos y prohibiciones
- **Usos admitidos**: Refactorización de código, corrección de errores en entornos de CI, generación de documentación técnica y prototipado rápido.
- **Usos prohibidos**: No debe utilizarse para procesar bases de datos que contengan datos personales reales (nombres, DNI, salud) a menos que se usen modelos de IA locales (vía Ollama o similar) que no envíen datos fuera de la red de la empresa.

Seguridad y certificaciones
- **Seguridad**: Al ser Open Source, el código es auditable. El uso de worktrees aislados proporciona una capa de seguridad operativa que evita corrupciones accidentales del repositorio principal.
- **Certificaciones**: Al ser un proyecto comunitario/open source bajo Apache 2.0, no cuenta con certificaciones ISO 27001 propias, por lo que la seguridad recae en la infraestructura de la empresa española que lo instala.

Otros
Es vital destacar que AO permite el uso de modelos locales (como Llama 3 a través de proveedores compatibles con la API de OpenAI). Para empresas con datos altamente confidenciales o secretos industriales, recomiendo configurar AO exclusivamente con modelos que corran en servidores propios para evitar cualquier salida de datos a la nube.

Fuentes consultadas:
- Sitio web oficial
- Repositorio oficial en Github
- Licencia Apache 2.0
- Arquitectura técnica y documentación