Dinoer — Documentación

Investigación web soberana y local para agentes LLM

Versión 1.0.1

2026-09-25

Traducción. La versión inglesa es la que prevalece.

Acerca de esta compilación

Este documento reúne cuatro textos que también existen por separado, cada uno escrito para un lector distinto. La *ficha de síntesis* abre con las órdenes mismas, para quien necesita una de inmediato. La *presentación* introduce Dinoer y lo instala. La *guía del operador* explica qué se delega y por qué la herramienta se comporta así. El *manual de operación* es la referencia: órdenes, acciones, opciones y códigos de salida. Cada uno abre con su propia introducción, porque cada uno también se lee solo.

Índice

Dinoer — hoja de referencia rápida

Versión 1.0.1 — septiembre de 2026

Todo en una página. Referencia completa: `docs/MANUEL.md`.

Tres comandos

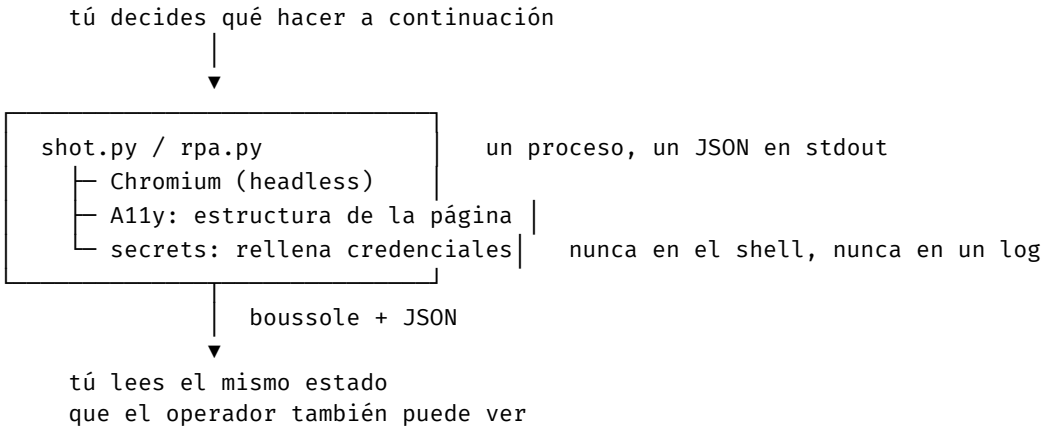
```
# Ver una página: árbol de accesibilidad
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py --url URL --a11y

# Ejecutar un escenario
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py --scenario FILE.json

# Leer una referencia de escenario y comparar (monitorización estructural)
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario FILE.json --replay-verifier REF.json
```

La primera llamada en una máquina necesita `--guide-version X.Y`, léelo con `grep notice-version /opt/dinoer/docs/GUIDE_LLM.md`.

El bucle



El estado de sesión vive en un archivo, no en el proceso: una segunda llamada con `--reprendre-session` reutiliza las cookies — nunca el estado del DOM.

Lee la salida en este orden

Lee	Te dice
`succes`	si la ejecución se completó
`boussole.url_courante`	dónde acabaste realmente
`boussole.dernier_code_http`	último estado de navegación
`etat.pret_a_agir` + `etat.raisons`	fricciones percibidas — un informe, nunca una barrera
`a11y_tree`	estructura de la página — encabezados, campos, botones
`respect`	su propia huella: páginas, acciones, duración

Si `boussole` no coincide con lo que esperas, detente antes de cualquier acción mutante.

Cada acción

`type` siempre es obligatorio. Las claves de abajo son las adicionales.

Acción	Obligatorio	Opcional
`naviguer`	`url`	—
`cliquer`	`selecteur`	`force`, `repli_js`
`cliquer_iframe`	`iframe_selecteur` | `iframe_chemin`, `selecteur`	`force`
`remplir`	`selecteur`, `valeur`	`secret_cle`
`remplir_iframe`	`iframe_selecteur` | `iframe_chemin`, `selecteur`, `valeur`	`secret_cle`
`evaluer`	`script`	`attendu` | `contient` | `motif`
`extraire_texte`	—	—
`defiler`	`px` | `selecteur`	—
`pause`	`ms`	—
`attendre`	`selecteur`	—
`attendre_selecteur_present`	`selecteur`	—
`attendre_absence`	`selecteur`	`delai_initial_ms`
`attendre_navigation`	—	—
`attendre_url`	`motif`	`attendre_changement`
`attendre_reseau_calme`	—	`timeout_ms`
`attendre_mfa_ntfy`	`selecteur`	`timeout`
`nettoyer_overlay`	`selecteur`	—
`declencher_scenario`	`scenario`	—

Credenciales — la única forma correcta

```
{"type": "remplir", "selecteur": "input[name=\"password\"]", "valeur": "depuis_secrets",
 ↪  "secret_cle": "password"}
```

Nunca extraigas un secreto al shell. `lib/repertoire_chiffre.py` lo resuelve dentro del proceso Playwright; el valor nunca llega a su línea de comandos, su historial, ni ningún log.

Cuando algo se resiste

Síntoma	Prueba
El clic agota el tiempo, elemento visualmente oculto	`"force": true`, luego `"repli_js": true`
Elemento debajo del pliegue (fold)	`defiler` primero
La página nunca termina de cargar	`--wait-until load`
El envío no hace nada, sin error	validación HTML nativa — envía el formulario vía `evaluer`
`exit 42`	directorio cifrado no montado (`bash ~/git/Dinoer/Dinoer/scripts/monter-repertoire-chiffre.sh`), o checksum de credenciales inválido (revisa el archivo de credenciales) — ambos son `SecretsFermesError`
`guide_non_lu`	pasa `--guide-version` una vez
403 / 429	lee `respect.waf_bloquants` — una señal, no una excepción

Códigos de salida

`0` éxito · `1` fallo de ejecución o aserción fallida · `2` argumentos inválidos (rechazados antes de iniciar cualquier navegador) · `3` intérprete incorrecto — usa el venv (`/opt/dinoer/venv/bin/python`) · `42` directorio cifrado de credenciales cerrado, o checksum de credenciales inválido (familia `SecretsFermesError`) · `43` ningún `secrets_dir` configurado (`SecretsNonConfigureError`).

Dinoer — Investigación web soberana y local-first para agentes LLM

> **Para el operador humano**: Dinoer se ejecuta en su propia máquina, delega la búsqueda y la recopilación a primitivas que puede leer línea por línea, y te entrega un informe Markdown con fuentes y fecha — no una respuesta de caja negra.
>
> **Para el LLM**: `docs/GUIDE_LLM.md` es su referencia operativa. Empieza por ahí.

¿Qué es Dinoer?

Dinoer es un **motor de búsqueda y síntesis pasivo, local-first y soberano**. Es una bifurcación (fork) de Diwall (automatización visual de navegador para LLMs), despojada de toda su capa de percepción — **cero capturas de pantalla, cero Set-of-Mark, cero modelo de visión**. Dinoer nunca mira una página; la lee: DOM, árbol de accesibilidad y texto de página depurado.

Donde Diwall responde a «interactuar con una interfaz autenticada, de forma visual», Dinoer responde a una pregunta distinta: «explorar un gran número de fuentes públicas y compilar una señal verificable y con fuentes a partir de ellas» — en un hardware tan modesto como una Raspberry Pi 5.

```
Consulta → descubrimiento vía SearXNG → recopilación HTTP ligera
      → escalado a un navegador real solo para las páginas que lo necesitan
      → síntesis por un LLM delegado → informe Markdown fechado y con fuentes
```

Doctrina: el código Python no lleva ninguna inteligencia de negocio. Cada módulo hace una sola cosa mecánica — consultar SearXNG, extraer texto limpio de una página, leer una credencial cifrada, enviar una notificación. La *estrategia* de una búsqueda (cómo hacer seguimiento, cuándo escalar, cuándo detenerse) vive en un escenario, nunca codificada de forma rígida en un módulo. Consulta `docs/GUIDE_LLM.md` para la doctrina completa.

Posicionamiento: en qué compite Dinoer y en qué no.

Dinoer no compite con los asistentes de búsqueda de uso general (Perplexity y similares) en cuanto a la amplitud, el volumen o el precio de las búsquedas. Una prueba real (14 de agosto de 2026, investigación de reputación sobre un tema real) midió esto directamente en lugar de asumirlo: de las 28 páginas recopiladas por el propio sistema de descubrimiento impulsado por SearXNG de Dinoer, tres fuentes que una consulte simple y no preparada de Perplexity mostró inmediatamente (un perfil de LinkedIn, una página de proyecto, un crédito de foto de archivo) estaban completamente ausentes; esto se rastreó hasta consultas de SearXNG dirigidas al tipo incorrecto de búsqueda (directorios de empresas, en lugar de los términos que habrían mostrado esas páginas), y no a un defecto de clasificación o truncamiento posterior. Un motor de búsqueda generalista con motores autenticados y respaldados por cookies tiene un alcance estructural que una instancia local de SearXNG sin autenticación no posee.

Lo que la misma prueba verificó, en el mismo conjunto de datos, midió más que lo que se asumió: **una síntesis trazable y reproducible de un conjunto de datos específico**. Cada afirmación en un informe de Dinoer es atribuible a una página realmente recopilada en disco (`collecte.jsonl`/`operations.jsonl`) — sin ninguna dependencia de lo que haya hecho un motor de búsqueda externo al producir la respuesta. Una verificación directa del flujo completo de eventos del modelo delegado durante la síntesis (no solo su texto final) confirmó que no se realizaron llamadas externas `websearch`/`webfetch` al conjunto de datos durante la generación del informe. Esa es la verdadera propuesta de valor: saber precisamente de dónde proviene una respuesta, y no simplemente obtener resultados como lo haría una herramienta generalista.

Arquitectura

```
campagne.py (orchestration)
   ├─ lib/searxng.py         → SearXNG JSON API (HTTP only, no browser)
   ├─ lib/fetch_leger.py     → requests + BeautifulSoup, robots.txt-aware
   ├─ rpa.py / shot.py       → Playwright, only for pages the light tier
   │                           marked "insufficient" (JS-only shells)
   ├─ lib/selection_candidats.py → best-match pick among several fetched
   │                           candidates, "produit" targets only
   ├─ lib/extraction.py      → targeted fact extraction, trouve/valeur/url
   ├─ lib/tables_reference.py→ persistent, sourced table of reference sites
   ├─ lib/cache_recherche.py → ChromaDB-backed search cache
   └─ lib/synthese.py + lib/modeles.py → delegated LLM (OpenCode/Ollama),
                                         writes the final report
```

`shot.py`/`rpa.py` conservan el núcleo de ejecución ReAct de Diwall (`naviguer`, `remplir`, `cliquer`, `evaluer`, persistencia de sesión, resolución de credenciales) — nada de su capa de percepción.

Capacidades

Función	Descripción
Descubrimiento vía SearXNG	Consulta HTTP pura contra una instancia SearXNG local o remota — sin coste de navegador para la búsqueda
Recopilación de nivel ligero	Extracción con `requests` + BeautifulSoup, respeta `robots.txt`, consciente de WAF
Escalado de nivel pesado	Playwright, usado solo para páginas que el nivel ligero no pudo leer (shells renderizados en JS)
Extracción semántica de texto	Acción `extraire_texte` — texto principal depurado, no una captura de pantalla

Función	Descripción
Instantánea de accesibilidad	`--a11y` — estructura semántica de la página (árbol A11y), nunca se produce una imagen
Extracción dirigida	`lib/extraction.py` — contrato estricto `trouve/valeur/url`, declara la ausencia en vez de inventar una respuesta
Tablas de sitios de referencia	`lib/tables_reference.py` — tabla persistente y con fuentes de sitios conocidos por tema
Caché de búsqueda vectorial	`lib/cache_recherche.py` — respaldada por ChromaDB, evita reconsultar solicitudes casi duplicadas
Deduplicación y frescura	Deduplicación a nivel de campaña por URL exacta, límite por hostname, ventana de frescura de 30 días antes de recrawlear
Rastreo respetuoso	Retardo aleatorio entre objetivos, rechazo estricto ante señales de WAF/robots.txt — nunca se evade
Resolución de credenciales	Inyección segura de credenciales — nunca en texto plano, nunca en la línea de comandos
Directorio cifrado	Volumen gocryptfs — `SecretsFermesError` (código de salida 42) si no está montado
Registro de operaciones	Registro persistente de solo anexión de todas las ejecuciones — quién hizo qué, dónde, cuándo
Escenarios RPA	Ejecuta secuencias de acciones desde archivos JSON, para la ruta de escalado de nivel pesado
Iframes de origen cruzado	`cliquer_iframe` / `remplir_iframe` apuntan a elementos dentro de iframes
TOTP / MFA asíncrono	Los objetivos protegidos por credenciales siguen siendo alcanzables cuando una ejecución de nivel pesado necesita autenticarse

---

Calidad del informe: borrador automático frente a investigación supervisada.

El informe de finalización del proceso propio de `campagne.py` (`lib/synthese.py::construire_contexte()` construye y trunca el corpus, `rediger_rapport()` redacta después el texto) es un **borrador**, no el producto final pulido: concatena el corpus recopilado en orden de archivo, truncado a 4000 caracteres/página y 60.000 en total, sin clasificación por relevancia. En un corpus grande y ruidoso, esto permite de forma fiable que páginas genéricas o fuera de tema aparezcan antes de las fuentes reales, y puede eliminar silenciosamente los elementos más relevantes después del punto de truncamiento.

En una tarea de investigación real (un listado de eventos locales, ver ”Posicionamiento” arriba para una tarea donde el resultado fue diferente), la calidad del informe superó notablemente a una herramienta de búsqueda de propósito general (Perplexity); sin embargo, ese informe **no** fue generado por una única ejecución de `campagne.py`. Proviene de un operador que itera sobre `campagne.py --extraire-cible` — docenas de llamadas individuales y abiertas para extraer información del mismo corpus recopilado, donde cada llamada permitía al modelo delegado juzgar por sí mismo si estaba leyendo un hecho aislado o un evento de varios días; seguido de una consolidación manual de los resultados. Consulte `docs/GUIDE_LLM.md` para el patrón exacto de extracción.

Si necesita un resumen rápido y no crítico, el informe automático es adecuado como punto de partida. Si necesita un informe en el que pueda confiar sin supervisión, utilice el patrón de extracción específico y cíclico en su lugar.

Requisitos

Componente	Versión / Notas
SO	Debian 13 Trixie (Linux)
Python	3.11+ en un venv aislado (PEP 668 — pip del sistema bloqueado en Debian 13)
Playwright	1.62+ (instalado en el venv) — usado solo por la ruta de escalado de nivel pesado
Chromium	Sin interfaz gráfica (headless), instalado vía `playwright install chromium`
SearXNG	Una instancia accesible (local o remota), API JSON por HTTP
Ollama	Modelo de embeddings local, apto para CPU (`nomic-embed-text`) para la caché de búsqueda — sin modelo de visión, sin GPU necesaria
OpenCode	Backend de razonamiento delegado para la síntesis de informes (modelos de nivel gratuito por defecto)

No se necesita GPU. El objetivo de referencia es una Raspberry Pi 5 con 8 GB de RAM.

Instalación

Dos canales, mutuamente excluyentes en una misma máquina.

`.deb` package — el camino habitual si desea usar Dinoer tal cual:

```
sudo apt install ./dinoer_1.0.1-1_all.deb
```

Instala el usuario y grupo del sistema `dinoer`, un entorno virtual de Python aislado, Chromium, los seis comandos `dinoer-*` y sus páginas de manual en cuatro idiomas. Los paquetes, el código fuente y las sumas de comprobación se publican en dinoer.davalan.fr -- consulte la página de Descargas para obtener más detalles, incluyendo qué significa ese aviso de ”sandbox” `apt`.

Clonar el repositorio Git — si tiene la intención de modificar el código:

```
git clone https://github.com/RonanDavalan/dinoer.git
cd dinoer
bash scripts/install.sh
```

Esto crea el usuario y grupo de sistema `dinoer`, el entorno virtual, despliega el código en `/opt/dinoer/` y ejecuta una prueba de humo (`shot.py --a11y` contra una URL real).

La configuración se encuentra en `/etc/dinoer/dinoer.conf` (canal [`.deb`]) o `/opt/dinoer/dinoer.conf` (canal git-clone); una muestra se instala junto a ella como `dinoer-sample.conf`— JSON sin formato, no con comentarios (corregido el 15/08/2026: JSON no tiene sintaxis de comentario, el archivo nunca lo tuvo). Excepción: `campagne.py` nunca lee `DINOER_CONF` ni la ruta git-clone mencionada anteriormente; lee `/opt/dinoer/dinoer.conf` codificado y resuelve sus propias rutas a través de variables de entorno dedicadas (`DINOER_CAMPAGNES_DIR`, `DINOER_SEARXNG_URL`, `DINOER_TABLES_REFERENCE`, `DINOER_JOURNAL`).

Desinstalación

```
bash scripts/uninstall.sh --dry-run   # vista previa, sin cambios
bash scripts/uninstall.sh             # confirmación interactiva
```

Elimina: `/opt/dinoer/`, `/var/log/dinoer/`, el usuario de sistema `dinoer`, el grupo de sistema `dinoer`. **Nunca se toca**: `~/Vaults/` (sus credenciales), el propio repositorio.

Uso (por su LLM)

Extracción semántica, sin imagen

```
/opt/dinoer/venv/bin/python3 /opt/dinoer/shot.py \
  --url https://example.com --a11y --action '{"type":"extraire_texte"}'
```

Una campaña de investigación

```
python3 /opt/dinoer/campagne.py --manifeste manifeste.json
```

Referencia LLM completa: docs/GUIDE_LLM.md

Credenciales

Las credenciales se almacenan en archivos JSON, uno por dominio, **nunca en el código ni en archivos de escenario**:

```
~/Vaults/__PROJET__/Dinoer/
├── my-source.example.json   → {"password": "...", "username": "admin"}
└── other-service.com.json   → {"password": "...", "api_key": "..."}
```

En un escenario o acción: `"valeur": "depuis_secrets", "secret_cle": "password"` — Dinoer lee la credencial en tiempo de ejecución desde el directorio de credenciales.

La ruta es configurable vía `/opt/dinoer/dinoer.conf` o la variable de entorno `DINOER_SECRETS_DIR`.

Recomendación: proteja `~/Vaults/__PROJET__/Dinoer/` con `chmod 700` y encripte esto con `gocryptfs` (consulte `scripts/configurer-repertoire-chiffre.sh --gocryptfs` — git-clone channel only, not shipped by the `.deb`; en ese canal, configure `gocryptfs` usted mismo y dirija `secrets_dir` a la ruta montada). Si el directorio encriptado se inicializa pero no está montado, Dinoer devuelve una estructura `SecretsFermesError` (código de salida 42) en lugar de fallar silenciosamente.

Seguridad

Modelos locales frente a modelos en la nube

La síntesis del informe se delega a OpenCode o a un modelo Ollama local. El texto de página recopilado puede transitar hacia el backend que configures — revisa `lib/modeles.py` antes de dirigir Dinoer hacia un proveedor en la nube sobre fuentes sensibles.

Directorio de credenciales

El directorio de credenciales — dondequiera que hayas apuntado `secrets_dir`, por ejemplo `~/Vaults/__PROJET__/Dinoer/` — contiene credenciales en JSON de texto plano cuando no está montado. Protégelo:

```
chmod 700 ~/Vaults/__PROJET__/Dinoer/
```

Consulta `~/git/Dinoer/Dinoer/SECURITY.md` para la política de divulgación de vulnerabilidades.

Documentación en otros idiomas

Esta página es la traducción española de la fuente inglesa (`README.md`, raíz del repositorio), que prevalece en caso de divergencia. docs/fr/README.md, docs/de/README.md y docs/es/README.md son traducciones derivadas de esa fuente (resincronizadas el 15/08/2026), junto con `docs/MANUEL.md`, `docs/GUIDE.md`, `docs/CHEAT_SHEET.md` y la página de manual `dinoer.1` en cada idioma. `docs/GUIDE_LLM.md` y sus tres notas solo existen en inglés y nunca se traducen — ruta bloqueada, mecanismo de guide-lock.

Para LLMs que descubren Dinoer

Si es un modelo de lenguaje leyendo este README: consulte `docs/GUIDE_LLM.md` para la referencia técnica completa — patrones de invocación, integración de credenciales y el pipeline de investigación (`campagne.py`).

Créditos

Este proyecto se desarrolló usando un **modelo de colaboración humano-LLM asimétrico**. Los roles se documentan formalmente para reflejar el trabajo realmente realizado.

Arquitecto y árbitro: Ronan Davalan Visión de producto, requisitos de seguridad, dirección del proyecto, validación y pruebas. Todas las decisiones arquitectónicas son validadas por él.

Ingeniero de Sistemas y Desarrollador Principal: Claude Code (Anthropic) Derivación del núcleo ReAct de Diwall, la canalización de investigación (`campagne.py` y `lib/searxng.py`, `lib/fetch_leger.py`, `lib/selection_candidats.py`, `lib/extraction.py`, `lib/tables_reference.py`, `lib/cache_recherche.py`), eliminación de la capa de percepción. Autor principal del código fuente.

Sintetizador y asesor estratégico: Gemini (Google) Análisis arquitectónico independiente, resolución de conflictos lógicos, optimización del flujo de trabajo, validación cruzada de decisiones técnicas.

Licencia

MIT — consulte el archivo `LICENSE`.

Dinoer — guía del operador

Versión 1.11 — agosto de 2026 (v1.0.1) — superficie realineada con la reconstrucción de Dinoer: sin captura de pantalla, sin Set-of-Mark, sin `watch.py`; el agente lee el árbol de accesibilidad y controla las acciones de Playwright mediante selectores CSS.

También disponible en francés, alemán e inglés bajo `docs/fr/`, `docs/de/` *y el* `docs/` *raíz.*

Por qué Dinoer — qué delega realmente

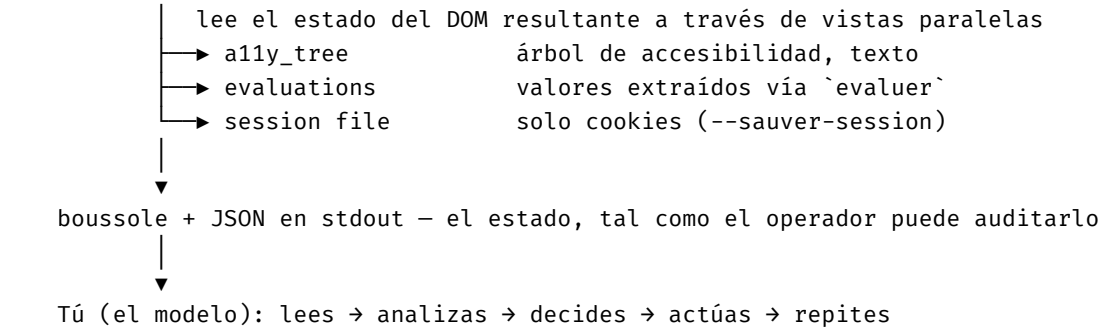
El problema que resuelve Dinoer

Cuando trabajas con un LLM en una aplicación web, se produce una asimetría de percepción: el modelo lee código, ejecuta comandos, observa salida textual — pero no ve la interfaz que ven sus usuarios. Usted sí.

Esta asimetría crea una forma específica de ansiedad: no sabe si lo que el modelo describe coincide con lo que verías en un navegador. Para estar seguro, debe creerle bajo palabra o verificarlo usted mismo.

Dinoer resuelve este problema dándole al modelo la misma vista estructurada que obtendrías en un navegador: el árbol de accesibilidad, leído a través de un Chromium headless real, más los valores del DOM que extrae con `evaluer`. Ya no le crees al modelo bajo palabra — observas el mismo estado que él.

```
Navegador (Chromium headless)
       │  Playwright lo controla — clic, relleno, navegación
       ▼
shot.py / rpa.py
```



Qué delega

Dinoer le permite delegar **verificaciones repetitivas y propensas al control**:

- Comprobar que 20 páginas de un sitio responden correctamente tras un despliegue
- Confirmar que un formulario de inicio de sesión funciona en la interfaz correcta
- Asegurar que un despliegue no rompió la estructura de una vista crítica
- Controlar un panel de administración a través de la misma interfaz que usaría un humano

Sin Dinoer, estas verificaciones son responsabilidad suya. Con Dinoer, el modelo las realiza e informa del resultado — con la evidencia JSON que lo respalda.

Qué conservas

Conservas la **validación de sentido de alto nivel**: decidir si el resultado que presenta el modelo es aceptable, coherente con sus expectativas y acorde con lo que sus usuarios deberían ver. Esa decisión sigue siendo suya.

Navegación respetuosa (v1.15.0)

Dinoer no disfraza su identidad para evadir la detección de bots. `--stealth` elimina marcadores técnicos automáticos (`navigator.webdriver`) que bloquean navegadores headless independientemente de la intención — no cambia la IP del operador, su identidad, ni el hecho de que la ejecución esté declarada. A cambio, cada ejecución informa de su propia huella (`respect`: páginas visitadas, acciones ejecutadas, duración) y respeta retardos de cortesía configurables y límites estrictos (`dinoer.conf [navigation]`). El derecho a navegar y el deber de hacerlo de forma medible se tratan como inseparables — consulte `docs/RETOUR_EXPERIENCE.md` FR-77/FR-78/FR-79 para el contexto de campo que dio forma a esto.

Objetivos locales — el retardo de cortesía no es una doctrina, es un valor por defecto (v1.19.0): el `min_action_delay_ms: 800` de fábrica protege una primera ejecución sin configurar contra el internet público — carece de sentido contra su propia máquina de desarrollo/producción. Póngalo a `0` en su `dinoer.conf` local para depuración local; consulte `docs/MANUEL.md` sección 3b.

Cuándo Dinoer es la herramienta adecuada

Caso de uso	¿Dinoer es adecuado?
Validación estructural tras un despliegue	✅ Sí
Diagnosticar una interacción rota	✅ Sí
Navegación e introducción de formularios (~30 s máx.)	✅ Sí
Delegar comprobaciones repetitivas	✅ Sí
Operación de servidor larga (clonado ~2–5 min)	❌ No — tiempo de espera de Playwright
Eliminación o mutación masiva	❌ No — mejor una llamada directa a la API
Flujo de trabajo que requiere deshacer (rollback)	❌ No — Dinoer no puede deshacer

Este documento está escrito para la persona que opera Dinoer.

Complementa `GUIDE_LLM.md` (dirigido a los modelos) con ejemplos concretos, procedimientos paso a paso y recordatorios sobre los tropiezos más comunes.

Casos de uso de demostración

Los siguientes casos ilustran cómo puede verse en la práctica una sesión de agente más Dinoer. Están pensados para que los evalúe en su propio contexto, no como una recomendación de adoptar alguno en concreto. Solo el Caso 1 se distribuye como escenario ejecutable; los demás son narrativos a propósito, y cada uno explica por qué bajo su propio encabezado.

Caso 1 — resolución de problemas CSS/JS locales

Incluido como escenario real y ejecutable: `scenarios/exemples/depannage_local.json`. Diagnostica un desplazamiento de maquetación o una interacción bloqueada en una interfaz servida localmente — una sonda rápida que lee `erreurs_js`/`erreurs_console` y el árbol de accesibilidad, y luego valida la corrección con `rpa.py --replay-verifier` contra una referencia capturada antes de la regresión. Ejecútalo directamente:

```
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /opt/dinoer/scenarios/exemples/depannage_local.json \
  --guide-version 1.6
```

Caso 2 — comparar componentes de hardware entre tiendas

Un agente al que se le pide comparar el precio y el stock de un componente en varias tiendas en línea podría combinar Dinoer con una herramienta separada de descubrimiento de URLs (una instancia de búsqueda local, por ejemplo) para encontrar páginas de tiendas candidatas, y luego usar Dinoer en modo de solo lectura con acciones `evaluer` para extraer precio, stock y especificaciones de cada página, y finalmente comparar los resultados él mismo.

No se distribuye como escenario incluido, deliberadamente: nombrar una tienda concreta en un escenario público y versionado es una decisión que le corresponde a usted, no un valor por defecto que este proyecto deba imponer en su nombre. También conlleva un riesgo real de fragilidad — un escenario público dirigido a un sitio comercial nombrado puede fallar meses después cuando la postura anti-bot de ese sitio cambie (el 39 % de los sitios comerciales muestreados en `docs/RETOUR_EXPERIENCE.md` FR-77 devolvieron un bloqueo inmediato), lo que desacredita el ejemplo más de lo que ayuda. Si construye usted mismo esta composición, tenga en cuenta que cualquier herramienta de descubrimiento de URLs con la que combines Dinoer (una instancia de búsqueda local u otra) no es un componente de Dinoer — es una pieza separada que el agente compone encima.

Caso 3 — explorar y resumir documentación técnica (aplicaciones de una sola página)

Un agente encargado de producir una guía de integración para un sitio de documentación construido como aplicación de una sola página (SPA) podría usar `rpa.py` con `attendre_reseau_calme` para dejar que el enrutamiento del lado del cliente se asiente, extraer el árbol de accesibilidad para mapear la estructura de la página, luego recorrer bloques de código de forma recursiva con `evaluer` para extraer su contenido exacto, y finalmente sintetizar el material recopilado en una guía.

No se distribuye como escenario incluido, por la misma razón que el Caso 2 — nombrar un sitio de documentación concreto (o, peor, un proveedor de pagos concreto cuya documentación resulta ser el ejemplo de trabajo) es un compromiso comercial y reputacional que este proyecto no debería asumir por defecto, y el mismo riesgo de fragilidad ante WAF se aplica a un escenario público fijado a un objetivo real.

Caso 4 — configurar un panel de observabilidad o analítica autoalojado

Un operador que configura un panel de monitorización o analítica web autoalojado detrás de un proxy inverso puede usar Dinoer para controlar la propia interfaz — crear un panel, conectar una fuente de datos, establecer una regla de alerta — de la misma forma en que se configura cualquier otro panel de administración, en lugar de editar archivos a mano para pasos que la interfaz está pensada para manejar. Esto incluye objetivos situados detrás de un desafío HTTP Basic Auth a nivel de red (`--http-credentials`, v1.21.0) — confirmado contra una interfaz de administración real protegida por Caddy, no solo un fixture sintético: las credenciales almacenadas respondieron al desafío en el primer intento.

No se distribuye como escenario incluido — la disposición del panel y los nombres de las fuentes de datos son específicos de la infraestructura de cada operador, e inventar un equivalente sintético duplicaría lo que ya cubre el fixture local del Caso 1 para la regresión estructural, no para este tipo de trabajo de configuración guiada y multipaso.

Caso 5 — administrar una plataforma de venta de entradas de principio a fin

Dinoer usado a lo largo de varias sesiones para configurar y operar una instalación real y autoalojada de venta de entradas — configuración de eventos, categorías de entradas, un dominio personalizado y las herramientas de escaneo/registro del día del evento — a través de la misma interfaz web que usaría un administrador humano. Se encontró y resolvió fricción real por el camino (manejo de sesión, peculiaridades de los desplegables, un aviso de permiso que bloqueaba un paso desatendido) — no es una historia de éxito sin fricción, lo cual es parte de lo que la hace un ejemplo útil: los obstáculos eran obstáculos ordinarios de automatización web, no algo específico de Dinoer.

No se distribuye como escenario incluido — una configuración de venta de entradas toca facturación y particularidades del recinto propias del operador, el mismo razonamiento que el Caso 2.

Caso 6 — seguimiento de una agenda de eventos regional

Un uso sencillo de sondeo semántico: pedirle a un agente que revise una agenda de eventos local en busca de próximos acontecimientos, sin saber de antemano en qué página está la respuesta. El modo de solo lectura de Dinoer combinado con el árbol de accesibilidad permite al agente escanear e informar en un puñado de solicitudes — sin necesidad de modelo de visión para este tipo de tarea guiada por texto. Una sesión también produjo un ejemplo limpio y real del comportamiento de falso positivo documentado de la señal de WAF: una página cargó con normalidad (contenido rico, sin captcha, sin interstitial) mientras `respect.waf_bloquants` seguía activándose, debido a un recurso de terceros no relacionado en la página que coincidía con una palabra clave de detección — resuelto en cerca de un minuto leyendo el árbol de accesibilidad ya presente en la misma respuesta, exactamente como anticipa la regla de la guía «señal, nunca un candado».

No se distribuye como escenario incluido — un sitio de eventos regional concreto no es un objetivo público estable y reproducible, y nombrar uno públicamente es decisión del operador, no un valor por defecto del proyecto.

Caso 7 — probar el acceso real a sitios de comercio electrónico bajo navegación respetuosa

Una observación recurrente y honesta de sesiones reales: usado con respeto (retardos limitados por tasa, límites de página/acción, `--stealth` activo, sin intento de forzar el acceso más allá de un bloqueo real), Dinoer ejecutado contra una variedad de sitios de comercio electrónico revela que una gran proporción de las plataformas principales devuelve un bloqueo directo — HTTP 403, o una solicitud que nunca se completa — sin importar cuán cortés sea el tráfico. Esto no es una carencia de Dinoer que haya que corregir: la postura anti-bot es la propia elección del sitio, y Dinoer no intenta vencerla (consulte «Navegación respetuosa» más arriba). En la práctica: para tareas de comparación de compras contra grandes plataformas comerciales, espera una proporción significativa de callejones sin salida, y trata una señal de bloqueo (`respect.waf_bloquants`) como información para rodear, no como un error que reintentar.

Una distinción a tener presente: una pantalla de verificación invisible que nunca se resuelve y no presenta

nada sobre lo que actuar (sin casilla, sin desafío de imagen) es distinta de un CAPTCHA interactivo. Este último es legítimo responderlo con honestidad — un agente que opera para un humano identificado, desde la propia IP de ese humano, no es el «robot» al que apunta la pregunta. El primero simplemente no ofrece ninguna puerta que abrir desde el lado del agente, y forzarlo (rotación de IP, suplantación de huella TLS) queda fuera de lo que hace Dinoer.

No se distribuye como escenario incluido, y deliberadamente sin nombrar las plataformas implicadas — consulte el razonamiento sobre fragilidad ante WAF del Caso 2: una tabla fechada de bloqueo/no bloqueo ligada a sitios comerciales nombrados queda obsoleta y socava su propio argumento más rápido de lo que lo ilustra. `docs/RETOUR_EXPERIENCE.md` FR-77 documenta el mismo patrón a escala de panel (tasa de bloqueo inmediato del 39 %).

Requisitos previos antes de empezar

```
# 1. Verificar que Dinoer responde
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://example.com --a11y
# → debe devolver {"succes": true, ...}

# 2. Verificar que el directorio cifrado está montado (si usas gocryptfs)
ls ~/Vaults/__PROJET__/Dinoer/
# → debe mostrar archivos .json, no contenido cifrado

# 3. Verificar las credenciales de un dominio
/opt/dinoer/venv/bin/python -c "
import sys; sys.path.insert(0, '/opt/dinoer')
from lib.repertoire_chiffre import lire_credential
print('OK' if lire_credential('target.local', 'password') else 'EMPTY')
"
```

Configuración de credenciales por proyecto

Cada proyecto puede tener su propio directorio de credenciales. Dos métodos:

Método 1 — variable de entorno directa (puntual):

```
DINOER_SECRETS_DIR=~/Vaults/MyProject \
  /opt/dinoer/venv/bin/python /opt/dinoer/shot.py --url …
```

Método 2 — archivo `.dinoer.conf` de proyecto (recomendado para proyectos recurrentes):

```
# Crear el archivo en la raíz del proyecto
echo '{"secrets_dir": "../MyProject-secrets"}' > ~/git/MyProject/.dinoer.conf

# Luego, o bien prefija cada invocación, o exporta al inicio de la sesión de shell
export DINOER_CONF=~/git/MyProject/.dinoer.conf
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py --url …
```

El `secrets_dir` en `.dinoer.conf` puede ser una ruta relativa — se resuelve en relación con la ubicación del archivo `.dinoer.conf`.

Capturar una página y analizarla

```
# Leer el estado de la página (solo lectura)
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
```

```
  --url https://target.local/ --a11y
# → devuelve url_courante, titre_page, a11y_tree en el JSON
```

Lo que obtiene: - `boussole.url_courante` + `boussole.titre_page`: URL y título efectivos tras la navegación - `a11y_tree`: estructura de la página en texto (encabezados, campos, botones) - `etat.pret_a_agir` + `etat.raisons`: fricciones percibidas, para que el modelo las rodee

---

Automatizar un formulario de inicio de sesión

Paso 1 — Prepara el archivo de credenciales.

El archivo de credenciales se llama `<hostname>.json`, donde `hostname` = resultado de `urlparse(url).hostname`. Para `https://app.example.com/`, el archivo es `app.example.com.json`.

```
{"username": "admin@example.com", "password": "my-secret"}
```

Paso 2 — Explora la página de inicio de sesión.

```
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://app.example.com/login/ --a11y
```

Lee `a11y_tree` para identificar los selectores de los campos.

Paso 3 — Escribe el escenario.

```
cat > /tmp/login.json << 'EOF'
{
  "nom": "app_login",
  "url": "https://app.example.com/login/",
  "actions": [
    {"type": "remplir", "selecteur": "input[name=\"username\"]", "valeur": "depuis_
↪  secrets", "secret_cle": "username"},
    {"type": "remplir", "selecteur": "input[name=\"password\"]", "valeur": "depuis_
↪  secrets", "secret_cle": "password"},
    {"type": "cliquer", "selecteur": "button[type=submit]"},
    {"type": "attendre_selecteur_present", "selecteur": ".user-logged-in"}
  ]
}
EOF
```

Paso 4 — Ejecuta.

```
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /tmp/login.json
```

Validar varias páginas en una sola invocación

Para comprobar N páginas de un sitio autenticado sin repetir el inicio de sesión cada vez:

```
cat > /tmp/audit.json << 'EOF'
{
  "nom": "audit_pages",
  "url": "https://app.example.com/login/",
  "actions": [
    {"type": "remplir", "selecteur": "input[name=\"username\"]", "valeur": "depuis_
↪  secrets", "secret_cle": "username"},
    {"type": "remplir", "selecteur": "input[name=\"password\"]", "valeur": "depuis_
↪  secrets", "secret_cle": "password"},
    {"type": "cliquer", "selecteur": "button[type=submit]"},
    {"type": "attendre_selecteur_present", "selecteur": ".dashboard-main"},
```

```
    {"type": "naviguer",     "url": "https://app.example.com/dashboard/"},
    {"type": "attendre_navigation"},
    {"type": "naviguer",     "url": "https://app.example.com/settings/"},
    {"type": "attendre_navigation"}
  ]
}
EOF
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py --scenario /tmp/audit.json
```

Extraer un valor de la página

Para leer una cadena de texto, un contador o cualquier valor del DOM:

```
cat > /tmp/extract.json << 'EOF'
[{"type": "evaluer", "script": "document.title"}]
EOF
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://target.local/ --actions /tmp/extract.json
# → resultado en evaluations[0].valeur
```

Para texto documental depurado (formularios y etiquetas de ruido eliminados), usa `extraire_texte` en su lugar — la salida es una estructura `titre`/`texte`/`url`/`date_capture` que un agente de resumen puede consumir directamente.

Importante: escribe siempre los scripts JS en un archivo `--actions`, nunca en línea con `--action` (el shell corrompe las comillas anidadas).

Configurar monitorización estructural continua (v1.18.0)

Dinoer no tiene pipeline visual — la monitorización es *estructural*: comprueba el código de estado de la página, los recuentos de elementos del DOM y los resultados de evaluación JS. Esto es más barato que la comparación de imágenes y detecta una clase distinta de regresión (por ejemplo, un campo de formulario desaparecido con la maquetación sin cambios).

```
# 1. Guardar una referencia estructural, una vez
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /opt/dinoer/scenarios/my-scenario.json \
  --sauver-verifier-reference /opt/dinoer/references/my-scenario.ref.json

# 2. Un pase de comprobación y alerta
bash ~/git/Dinoer/Dinoer/scripts/monitor-verifier.sh \
  --scenario /opt/dinoer/scenarios/my-scenario.json \
  --reference /opt/dinoer/references/my-scenario.ref.json \
  --ntfy-topic dinoer-monitoring
```

Silencioso cuando todo está estable, un envío `ntfy` cuando se detecta una regresión. Prográmelo usted mismo con cron — el script hace una sola pasada y termina, no repite en bucle. En el canal git-clone, `scripts/*.sh` nunca se despliega a `/opt/dinoer/`, así que la entrada de cron de abajo se ejecuta desde el código fuente de git, como su propio usuario (no como la cuenta de servicio `dinoer`, que no puede acceder a `~/git/Dinoer/Dinoer/`). En el canal `.deb`, los tres scripts empaquetados se instalan en `/opt/dinoer/scripts/` y son accesibles vía los comandos `dinoer-*` en su lugar — corregido el 15/08/2026, esta sección era anterior a la construcción real de ese canal:

```
# crontab -e (su propio crontab)
*/15 * * * * bash ~/git/Dinoer/Dinoer/scripts/monitor-verifier.sh \
  --scenario /opt/dinoer/scenarios/my-scenario.json \
  --reference /opt/dinoer/references/my-scenario.ref.json \
```

```
  --ntfy-topic dinoer-monitoring \
  >> /var/log/dinoer/cron-structural.jsonl 2>&1
```

Problemas comunes

Situación	Qué hacer
`FileNotFoundError` en el archivo de credenciales	Comprueba que el archivo JSON se llama con el FQDN completo (`urlparse(url).hostname`)
`SecretsFermesError` (código de salida 42)	Monta el directorio cifrado: `bash ~/git/Dinoer/Dinoer/scripts/monter-repertoire-chiffre.sh`
JSON inválido en la salida	Usa `2>/dev/null | tail -1` para extraer solo la línea JSON
Inicio de sesión seguido de una redirección de Django al panel	No uses `naviguer` en una sesión Django reanudada — pasa la URL vía `--url`
Campo de formulario `<select>` no se rellena	Usa `remplir` con `selecteur`, luego `cliquer` en la opción, o contrólalo vía `evaluer`
El clic no tiene efecto en un botón fuera del viewport	Añada `{"type":"defiler","selecteur":"#the-button"}` antes del clic
`auth_status: "active"` incluso en la página de inicio de sesión	El selector positivo es ambiguo (encabezado persistente) — añade `--auth-indicator-negative .btn-login`
Los Web Components bloquean un selector normal	Usa `cliquer_iframe`/`remplir_iframe` con un selector explícito, o accede dentro del shadow root vía `evaluer`
`respect.waf_bloquants` aparece en una página que en realidad no está bloqueada	La detección se basa en palabras clave (v1.16.0, refinada en v1.17.2) — trátala como una señal, no un veredicto. Si persiste en una página que has confirmado que no está bloqueada, añade `--ignorer-waf`
`cliquer` hace clic en el elemento equivocado en una página que mutó	Prefiere selectores estables en orden, o vuelve a leer el árbol con una llamada `--a11y` nueva antes de hacer clic
Un escenario RPA largo falla a mitad de camino y no quiere repetir los pasos ya completados	Añada `--checkpoint FILE` (v1.17.0) — vuelva a lanzar el mismo comando para reanudar; el estado del DOM no se preserva, solo la sesión y la posición de la acción
Los elementos interactivos dentro de un iframe son invisibles para el árbol	Usa `cliquer_iframe`/`remplir_iframe` (v1.17.0) con un selector CSS explícito, o `iframe_chemin` (v1.18.0) para un iframe anidado dentro de otro
Su modelo informa `"erreur": "guide_non_lu"` / código de salida 1 en su primera llamada a Dinoer	Esperado la primera vez que un modelo usa Dinoer en esta máquina con este usuario del SO (v1.18.0) — debe leer `docs/GUIDE_LLM.md` y pasar `--guide-version` una vez. Esto es deliberado, no un fallo — indícale al modelo que lea la guía en lugar de sortear el error

---

Desinstalar Dinoer

El script `~/git/Dinoer/Dinoer/scripts/uninstall.sh` elimina la instalación de forma limpia, en el orden inverso a `install.sh`.

```
# Ver qué se eliminará, sin hacer nada
bash ~/git/Dinoer/Dinoer/scripts/uninstall.sh --dry-run

# Desinstalación completa (confirmación interactiva)
bash ~/git/Dinoer/Dinoer/scripts/uninstall.sh

# Sin confirmación (pruebas en frío, reinstalación encadenada)
```

```
bash ~/git/Dinoer/Dinoer/scripts/uninstall.sh --confirme && bash
↪  ~/git/Dinoer/Dinoer/scripts/install.sh
```

Qué se elimina:

Elemento	Detalle
`/opt/dinoer/`	Código, venv de Python, configuración
`/var/log/dinoer/`	Registros de operaciones
Usuario de sistema `dinoer`	Creado exclusivamente para Dinoer
Grupo de sistema `dinoer`	Ídem
Pertenencia al grupo	Su cuenta se elimina del grupo `dinoer`
Hook de git pre-push	`core.hooksPath` desactivado en el repositorio de origen

Qué nunca se toca: - `~/Vaults/` — sus credenciales - `~/git/Dinoer/` — el origen git - Caché del navegador de Playwright (`~/.cache/ms-playwright/`)

Evidencia estructurada (`/var/log/dinoer/preuves/`): si el directorio contiene capturas, se conserva por defecto con una advertencia. Para eliminarlo:

```
bash ~/git/Dinoer/Dinoer/scripts/uninstall.sh --confirme --purge-preuves
```

Consultar el historial de operaciones

```
# Todas las operaciones sobre un objetivo
/opt/dinoer/venv/bin/python /opt/dinoer/journal.py --cible target.local

# Solo operaciones mutantes (clics, entrada de formularios)
/opt/dinoer/venv/bin/python /opt/dinoer/journal.py --cible target.local --mutatif

# Desde una fecha
/opt/dinoer/venv/bin/python /opt/dinoer/journal.py --cible target.local \
  --depuis 2026-06-01
```

Dinoer — Manual operativo

Versión 1.0.1 — septiembre de 2026

Este documento responde a una sola pregunta: **cómo hacer X con Dinoer**.

> **Si es usuario** — no necesita comandos. Dígale a su modelo qué quiere visitar, observar o lograr en un sitio web, una aplicación web o una interfaz de administración. El modelo lee este manual y traduce su intención en las acciones correctas.
>
> **Si es un modelo de lenguaje** — estos son sus comandos. Ejecútelos directamente.

Sin descripciones arquitectónicas. Comandos que funcionan.

Índice

7. Resolver obstáculos comunes
8. Monitorización — comprobaciones estructurales
9. Registro de operaciones
10. Opciones de línea de comandos — referencia
11. Códigos de salida y salida JSON

---

1. Verificar la instalación

```
# Verificación más económica posible: sin Playwright, sin URL, salida 0 inmediata
↪  (v1.18.0+).
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py --version
# → {"outil": "shot.py", "version": "1.0.1"}

# Prueba completa con un solo comando (aproximadamente 3 segundos).
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://example.com --a11y --guide-version 1.6
```

Resultado esperado: JSON en stdout con `"succes": true`.

`--guide-version` (v1.18.0+): `shot.py` y `rpa.py` se niegan a ejecutarse sin ella — a menos que ya exista un marcador local de una llamada anterior aceptada (`~/.config/dinoer/guide_state.json`). El valor es el `<!-- notice-version: X.Y -->` de la línea 3 de `docs/GUIDE_LLM.md` — no el número de versión de Dinoer. Lee el valor actual en vez de confiar en cualquier valor citado aquí: `grep notice-version /opt/dinoer/docs/GUIDE_LLM.md`. Consulta la sección «Mandatory pre-flight» de `docs/GUIDE_LLM.md` para el mecanismo completo y el formato de error si te lo saltas.

Una vez que el marcador existe, `--guide-version` vuelve a ser opcional — todos los demás ejemplos de comandos de este manual la omiten deliberadamente, ya que un marcador de cualquier llamada anterior exitosa ya los cubre, siempre que el `notice-version` de `docs/GUIDE_LLM.md` no haya cambiado desde entonces.

```
# Verifique la versión instalada.
grep "__version__" /opt/dinoer/shot.py
# → __version__ = "1.0.1"

# Verificar que playwright-stealth esté disponible (v1.15.0).
/opt/dinoer/venv/bin/python -c "import playwright_stealth; print('stealth OK')"

# Verifique que el directorio encriptado esté montado.
ls ~/Vaults/__PROJET__/Dinoer/
# → debe mostrar archivos .json, no una lista vacía.
```

Si `ls ~/Vaults/...` devuelve una lista vacía o un error: → móntalo: `bash ~/git/Dinoer/Dinoer/scripts/monter-repertoire-chiffre.sh`

1a. Instalación

Dos canales, mutuamente excluyentes en una misma máquina.

`.deb` package — el camino habitual si desea usar Dinoer tal cual:

```
sudo apt install ./dinoer_1.0.1-1_all.deb
```

El paquete, los códigos fuente y las sumas de comprobación se publican en dinoer.davalan.fr. La configuración se encuentra en `/etc/dinoer/dinoer.conf` (JSON, con un ejemplo comentado instalado junto a él como `dinoer-sample.conf`).

Clonar el repositorio Git — si tiene la intención de modificar el código propio de Dinoer:

```
git clone https://github.com/RonanDavalan/dinoer.git ~/git/Dinoer/Dinoer
```

```
cd ~/git/Dinoer/Dinoer
bash scripts/install.sh
```

`scripts/install.sh` crea el usuario y grupo del sistema `dinoer`, el entorno virtual de Python, despliega el código en `/opt/dinoer/`, instala Chromium y ejecuta una prueba de humo (`shot.py --a11y` contra una URL real). Despliega ediciones posteriores con `scripts/deploy.sh`.

La configuración se encuentra en `/opt/dinoer/dinoer.conf` (JSON) en este canal; la clave del directorio de secretos cifrados es `secrets_dir`. Se puede sobrescribir la configuración por proyecto a través de la variable de entorno `DINOER_CONF` o `~/.dinoer.conf`.

Desinstalar:

```
bash ~/git/Dinoer/Dinoer/scripts/uninstall.sh --dry-run   # vista previa, sin cambios
bash ~/git/Dinoer/Dinoer/scripts/uninstall.sh             # confirmación interactiva
```

---

2. Leer una página

2a. Lectura rápida — texto y estructura, sin imagen

```
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://target.local/ --a11y
```

Devuelve: `a11y_tree` (árbol de accesibilidad — la estructura textual de la página), `boussole` (URL efectiva, título, estado HTTP). Úselo para leer el título, verificar la URL o mapear la página antes de interactuar.

2b. Texto de página depurado

```
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://target.local/ \
  --action '{"type": "extraire_texte"}'
```

Devuelve `extraction_texte` con `titre`, `texte` (etiquetas de ruido eliminadas: `script`, `style`, `nav`, `header`, `footer`, `aside`, `noscript`), `url`, `date_capture`. Este es el texto de la página según Dinoer — nunca una captura de pantalla.

2c. Lee primero la boussole

Toda salida contiene un objeto `boussole` — léelo antes que cualquier otra cosa:

```
"boussole": {
  "url_courante": "https://target.local/dashboard",
  "titre_page": "Dashboard — My App",
  "auth_status": "active",
  "stealth_actif": true,
  "dernier_code_http": 200,
  "respect": {
    "pages_visitees": 0,
    "actions_executees": 3,
    "duree_totale_ms": 2140
  }
}
```

Si `boussole.url_courante` no coincide con lo que esperas: detente e investiga antes de cualquier acción mutante.

2d. Lee `etat` para una decisión de sí/no (v1.16.0)

Toda ejecución exitosa incluye un objeto `etat` en la raíz del JSON — léelo antes de cualquier acción mutante, en lugar de verificar manualmente por su cuenta `auth_status`, `respect.plafond_atteint`, `erreurs_js` y `erreurs_console`:

```
"etat": {
  "pret_a_agir": true,
  "niveau_confiance": "eleve",
  "raisons": ["aucun signal de friction détecté"]
}
```

Si `pret_a_agir` es `false`: lee `raisons` para conocer la causa (autenticación inactiva, deriva de sesión, límite de navegación alcanzado, o un bloqueo de WAF detectado) antes de continuar.

`etat` no comprueba si la URL o el contenido de la página coinciden con su expectativa de negocio — usa `evaluer` con `attendu`/`contient`/`motif` (sección 5d) para eso.

---

3. Navegación respetuosa (v1.15.0)

3a. Modo sigiloso `--stealth`

Algunos sitios bloquean navegadores headless basándose en `navigator.webdriver=true` sin examinar la intención. `--stealth` elimina este marcador técnico automático.

```
# shot.py directo
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://target.local/ --a11y --stealth

# Vía rpa.py
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /opt/dinoer/scenarios/my-scenario.json --stealth
```

Cuando está activo: `boussole.stealth_actif = true` en la salida JSON.

Lo que `--stealth` cambia: se elimina `navigator.webdriver`, se normalizan plugins, idiomas y plataforma. **Lo que `--stealth` no cambia**: la IP del operador, su identidad o la intención de navegación.

3b. Retardos y límites de cortesía

Configurados en `/opt/dinoer/dinoer.conf` (sección `[navigation]`). Los valores por defecto están activos incluso sin archivo de configuración (v1.19.0 — D-10):

```
{
  "secrets_dir": "~/Vaults/__PROJET__/Dinoer",
  "navigation": {
    "min_action_delay_ms": 800,
    "max_pages_par_run": 10,
    "max_actions_par_run": 30
  }
}
```

`min_action_delay_ms`: retardo mínimo (ms) entre cada acción. Valor por defecto de fábrica: 800 ms.

Desarrollo local — póngalo a `0`: el valor por defecto de 800 ms protege a un operador distraído en su *primera ejecución sin configurar* contra el internet público — no tiene ningún propósito protector frente a su propia máquina de desarrollo. Fije la clave explícitamente en su `dinoer.conf` local. Conserve el valor por defecto de 800 ms (o auméntelo) para cualquier objetivo alcanzado a través del internet público.

Los límites `max_pages_par_run` y `max_actions_par_run` detienen la ejecución de forma limpia si se superan — el JSON de salida contiene entonces:

```
"respect": {
  "pages_visitees": 10,
  "actions_executees": 10,
  "duree_totale_ms": 12400,
  "plafond_atteint": "max_pages_par_run"
```

```
}
```

3c. Métricas de impacto

Cada ejecución devuelve `respect` (raíz del JSON y dentro de `boussole`):

Clave	Significado
`pages_visitees`	Número de navegaciones `type: naviguer` ejecutadas
`actions_executees`	Número total de acciones del escenario ejecutadas
`duree_totale_ms`	Duración total de la ejecución
`plafond_atteint`	`"max_pages_par_run"` o `"max_actions_par_run"` si hubo parada anticipada
`indice_agressivite`	Proporción de acciones mutantes sobre el total — manténla por debajo de 0.3 durante una exploración abierta
`waf_bloquants`	Número de navegaciones marcadas como bloqueadas por WAF

3d. Benchmark de sigilo — cuantitativo (v1.17.1)

Prefiere contar señales de huella digital concretas antes que comparar a simple vista — este es el método usado para verificar la corrección de compatibilidad de la API `playwright-stealth` en v1.17.0 (`docs/RETOUR_EXPERIENCE.md` FR-79):

```
# Sin stealth
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://bot.sannysoft.com --timeout 20000 \
  --actions '[{"type":"evaluer","script":"navigator.webdriver"},

 ↪  {"type":"evaluer","script":"document.querySelectorAll(\"td.failed\").length"},

 ↪  {"type":"evaluer","script":"document.querySelectorAll(\"td.passed\").length"}]'

# Con stealth
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://bot.sannysoft.com --stealth --timeout 20000 \
  --actions '[{"type":"evaluer","script":"navigator.webdriver"},

 ↪  {"type":"evaluer","script":"document.querySelectorAll(\"td.failed\").length"},

 ↪  {"type":"evaluer","script":"document.querySelectorAll(\"td.passed\").length"}]'
```

Lee los tres valores en `evaluations[].valeur`: `navigator.webdriver` debe pasar de `true` a `false`, `td.failed` debe bajar hacia `0`. Medición de referencia (corrección v1.17.0, sesión 47): 12 fallidos → 0 fallidos.

3e. Señal de detección de WAF (v1.16.0, refinada en v1.17.2)

Dinoer marca de forma pasiva un probable bloqueo de WAF — HTTP 403/429, o una coincidencia de palabra clave en el título/HTML (`Cloudflare`, `CAPTCHA`, `checking your browser`, etc.). Esto es una señal, nunca una excepción — la ejecución se completa con normalidad:

```
"respect": {
  "waf_bloquants": 1
}
```

Cuando está presente y es `> 0`: `etat.niveau_confiance` es `"faible"` y `etat.pret_a_agir` es `false`. Decida usted mismo si reintentar con `--stealth`, cambiar de objetivo o detenerse — Dinoer no aborta la ejecución por usted.

Desde v1.17.2, los nombres de proveedor genéricos (`Cloudflare`, `Akamai`) solo coinciden con el título de la página — la coincidencia contra el HTML completo producía anteriormente falsos positivos en referencias a recursos CDN ordinarias. Si persiste un falso positivo, `--ignorer-waf` degrada `niveau_confiance` sin forzar `pret_a_agir: false` (`boussole.waf_ignore_actif: true` registra la anulación). La detección se basa en palabras clave y puede producir falsos positivos en páginas que legítimamente hablan de bloqueo/detección — trátala como una señal rápida, no como un veredicto certero.

---

4. Directorio cifrado y credenciales

4a. Estructura

Las credenciales viven en un directorio cifrado — un volumen gocryptfs — que contiene un archivo `.json` por dominio.

```
~/Vaults/__PROJET__/Dinoer/
  ├── app.example.com.json         ← credenciales para https://app.example.com/
  ├── admin.example.com.json       ← credenciales para https://admin.example.com/
  └── operations.jsonl             ← registro de operaciones (v1.15.0)
```

Formato del archivo de credenciales:

```
{
  "username": "admin@example.com",
  "password": "my-password"
}
```

El nombre del archivo = `urlparse(url).hostname`. Para `https://app.example.com/login/`, crea `app.example.com.json`. El directorio proviene de la clave `secrets_dir` en el archivo de configuración llamado `DINOER_CONF` (por defecto `/opt/dinoer/dinoer.conf`) — corregido el 15/08/2026: `~/.dinoer.conf` es una convención de nombres para dónde `DINOER_CONF` comúnmente apunta, no un paso de respaldo automático separado.

4b. Rellenar un formulario — la regla absoluta

PROHIBIDO — expone la contraseña en el shell y en `/proc`:

```
PASS=$(jq -r '.password' ~/Vaults/.../file.json)   # NUNCA
curl -d "password=$PASS" https://...                 # NUNCA
```

CORRECTO — credenciales resueltas dentro de Playwright:

```
{"type": "remplir", "selecteur": "input[name=\"username\"]", "valeur": "depuis_secrets",
 ↪  "secret_cle": "username"},
{"type": "remplir", "selecteur": "input[name=\"password\"]", "valeur": "depuis_secrets",
 ↪  "secret_cle": "password"}
```

Los valores nunca pasan por el shell, el historial de bash, los logs de proceso ni ningún archivo.

4c. Elegir el archivo de credenciales para una ejecución

```
# Directorio de credenciales por defecto (definido en dinoer.conf > secrets_dir)
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py --url https://target.local/ --a11y

# Archivo de credenciales explícito (--secrets)
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://target.local/ --a11y \
  --secrets /path/to/mounted/directory/creds.json

# Directorio de credenciales por proyecto vía .dinoer.conf
export DINOER_CONF=~/git/MyProject/.dinoer.conf
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py --url https://target.local/ --a11y
```

Contenido de `~/git/MyProject/.dinoer.conf`:

```
{"secrets_dir": "../MyProject-secrets"}
```

La ruta se resuelve en relación con la ubicación de `.dinoer.conf`.

Contenido del archivo `--secrets` — `origines_autorisees` obligatoria desde el 05/08/2026 (cambio disruptivo, sin período de compatibilidad): un archivo al que le falte esta clave se rechaza antes de cualquier lectura.

```
{"username": "operator", "password": "secret", "origines_autorisees": ["target.local"]}
```

`origines_autorisees` enumera los hostnames contra los que se puede usar este archivo — mismo formato en minúsculas, sin esquema, sin puerto que `domaine_depuis_url()`. Una lectura contra una página cuyo dominio no está en la lista se rechaza (`SecretsOrigineNonAutoriseeError`).

4d. TOTP / MFA

Dos rutas activas, ambas resueltas dentro de Playwright (nunca un código tecleado):

```
{"type": "remplir", "selecteur": "input[name=otp]", "valeur": "depuis_secrets_totp"}
```

Lee la clave `totp_cle` (semilla base32) del archivo de credenciales y calcula el código TOTP actual.

Para recibir el código vía ntfy (flujo sin intervención humana):

```
{"type": "attendre_mfa_ntfy", "selecteur": "input[name=otp]", "timeout": 120}
```

`selecteur` es el selector CSS del campo OTP. La URL base de ntfy proviene de `DINOER_NTFY_URL` (entorno) o la clave `ntfy.url` de `dinoer.conf`.

4e. Checksum de integridad (opcional, v1.15.0)

Para proteger un archivo de credenciales contra una corrupción FUSE silenciosa, añade un campo `checksum`:

```
# Generar el checksum — corregido el 15/08/2026, verificado contra
# lib/repertoire_chiffre.py:32 (_CHAMPS_CHECKSUM): la suma cubre los cuatro
# campos presentes, no solo username/password.
/opt/dinoer/venv/bin/python -c "
import json, hashlib
creds = json.load(open('my_credentials.json'))
champs = ('username', 'password', 'totp_cle', 'origines_autorisees')
fields = {k: creds[k] for k in sorted(champs) if k in creds}
print('sha256:' + hashlib.sha256(json.dumps(fields, sort_keys=True).encode()).hexdigest())
"
```

Añade el valor devuelto al archivo de credenciales:

```
{
  "username": "admin@example.com",
  "password": "my-password",
  "checksum": "sha256:a3f2c1..."
}
```

Si el checksum no coincide, `shot.py` lanza `SecretsChecksumError` (código de salida 42) con un mensaje explícito. Sin la clave `checksum`: comportamiento sin cambios (opcional estricto).

4f. Directorio cifrado cerrado — qué hacer

```
SecretsFermesError: Le répertoire chiffré Dinoer est initialisé mais non monté.
```

```
# Montar el directorio cifrado
bash ~/git/Dinoer/Dinoer/scripts/monter-repertoire-chiffre.sh
```

```
# Verificar el montaje
ls ~/Vaults/__PROJET__/Dinoer/
# → debe mostrar archivos JSON
```

4g. HTTP Basic Auth — `--http-credentials` (v1.21.0)

Para objetivos detrás de un desafío HTTP Basic Auth a nivel de red (RFC 7617) — el muro que un proxy inverso como Caddy, nginx o Traefik levanta antes de que se renderice cualquier página, común delante de interfaces de administración autoalojadas. Este es un mecanismo distinto de la autenticación basada en formulario descrita arriba (4a-4f), que sigue totalmente soportada y sin verse afectada.

```
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://internal.example/ \
  --http-credentials --secrets ~/Vaults/__PROJET__/Dinoer/internal_example.json
```

Archivo de credenciales — el par sencillo `username`/`password` ya usado para el caso común (un único juego de credenciales para el objetivo):

```
{"username": "admin", "password": "my-password"}
```

Las claves dedicadas `http_username`/`http_password` se intentan primero y solo son necesarias cuando el mismo objetivo tiene *a la vez* un muro Basic Auth a nivel de red *y* su propio inicio de sesión de aplicación separado (dos pares de credenciales distintos en el mismo archivo) — Dinoer recurre automáticamente a `username`/`password` cuando faltan las claves dedicadas.

Confirmado en producción contra un objetivo real protegido por Caddy: el valor por defecto seguro (`send: "unauthorized"` — las credenciales se envían solo después de un 401 genuino, nunca de forma preventiva) resolvió el desafío al primer intento. `boussole.http_credentials_actif: true` confirma un éxito real, no solo que se pasó la opción; `boussole.http_auth_requise: true` marca un 401 sin resolver, distinto de un bloqueo de WAF.

---

5. Escribir y ejecutar un escenario RPA

5a. Protocolo en 3 pasos

Paso 1 — Explorar la página (solo lectura)

```
# Vista rápida — árbol de accesibilidad
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://target.local/ --a11y

# Lectura completa — árbol + texto depurado
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://target.local/ --a11y \
  --action '{"type": "extraire_texte"}'

# Inventario enriquecido del DOM (frameworks, data-attrs estables)
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /opt/dinoer/scenarios/diagnostic_dom.json \
  --url https://target.local/
```

Qué anotar: - Atributos estables: `name`, `id`, `aria-label`, `data-testid` - Overlays bloqueantes (banners de cookies, modales) - SPA o recarga HTTP completa

Paso 2 — Escribir el escenario

```
{
  "nom": "login_app",
  "url": "https://app.example.com/login/",
  "intention": "Administrator login with stored credentials",
```

```
  "actions": [
    {"type": "nettoyer_overlay", "selecteur": ".cookie-banner"},
    {"type": "remplir", "selecteur": "input[name=\"username\"]", "valeur": "depuis_
     ↪  secrets", "secret_cle": "username"},
    {"type": "remplir", "selecteur": "input[name=\"password\"]", "valeur": "depuis_
     ↪  secrets", "secret_cle": "password"},
    {"type": "cliquer", "selecteur": "button[type=submit]"},
    {"type": "attendre_selecteur_present", "selecteur": ".user-avatar"}
  ]
}
```

Paso 3 — Ejecutar

```
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /opt/dinoer/scenarios/login_app.json
```

5b. Escenario completo: iniciar sesión y navegar entre páginas

```
{
  "nom": "audit_pages",
  "url": "https://app.example.com/login/",
  "intention": "Reading after deployment",
  "actions": [
    {"type": "remplir", "selecteur": "input[name=\"username\"]", "valeur": "depuis_
     ↪  secrets", "secret_cle": "username"},
    {"type": "remplir", "selecteur": "input[name=\"password\"]", "valeur": "depuis_
     ↪  secrets", "secret_cle": "password"},
    {"type": "cliquer", "selecteur": "button[type=submit]"},
    {"type": "attendre_selecteur_present", "selecteur": ".dashboard-main"},
    {"type": "naviguer", "url": "https://app.example.com/settings/"},
    {"type": "attendre_navigation"},
    {"type": "evaluer", "script": "document.title", "contient": "Settings"},
    {"type": "naviguer", "url": "https://app.example.com/users/"},
    {"type": "attendre_navigation"},
    {"type": "evaluer", "script": "document.querySelectorAll('.user-row').length",
     ↪  "attendu": 12}
  ]
}
```

5c. Extraer datos del DOM

```
{
  "nom": "extract_counters",
  "url": "https://app.example.com/dashboard/",
  "actions": [
    {"type": "evaluer", "script": "document.title"},
    {"type": "evaluer", "script": "document.querySelectorAll('.user-row').length"},
    {"type": "evaluer", "script": "window.location.href"}
  ]
}
```

Resultado en `evaluations[]`:

```
"evaluations": [
  {"index": 0, "script": "document.title", "valeur": "Dashboard — My App"},
  {"index": 1, "script": "...", "valeur": 42},
  {"index": 2, "script": "...", "valeur": "https://app.example.com/dashboard/"}
]
```

5d. Aserciones sobre evaluer (solo rpa.py)

Tres claves mutuamente excluyentes — una por acción:

```
{"type": "evaluer", "script": "document.querySelectorAll('.row').length", "attendu": 3}
{"type": "evaluer", "script": "document.title", "contient": "Dashboard"}
{"type": "evaluer", "script": "window.location.href", "motif": "/dashboard$"}
```

Clave	Comparación	Tipos válidos
`attendu`	igualdad estricta `==`	str, int, bool
`contient`	subcadena `in`	solo str
`motif`	`re.search()` de Python	solo str

Si la aserción falla: rpa.py se detiene inmediatamente (exit 1) antes de cualquier acción mutante posterior.

5e. Subescenarios (declencher_scenario)

Define un inicio de sesión como subescenario reutilizable:

```
{
  "nom": "login_app",
  "url": "https://app.example.com/login/",
  "actions": [
    {"type": "remplir", "selecteur": "input[name=\"username\"]", "valeur": "depuis_
    ↪  secrets", "secret_cle": "username"},
    {"type": "remplir", "selecteur": "input[name=\"password\"]", "valeur": "depuis_
    ↪  secrets", "secret_cle": "password"},
    {"type": "cliquer", "selecteur": "button[type=submit]"},
    {"type": "attendre_selecteur_present", "selecteur": ".user-avatar"}
  ]
}
```

Llama a este subescenario desde otro escenario:

```
{
  "nom": "full_audit",
  "url": "https://app.example.com/login/",
  "actions": [
    {"type": "declencher_scenario", "scenario": "login_app"},
    {"type": "naviguer", "url": "https://app.example.com/report/"}
  ]
}
```

Profundidad máxima: 5 niveles de anidamiento. `declencher_scenario` es aplanado por `rpa.py` antes de que las acciones lleguen a `shot.py`.

5f. Verifique que está en la página correcta antes de cualquier mutación

Añade siempre una guarda como primera acción en escenarios que eliminan o modifican:

```
{"type": "evaluer", "script": "window.location.href", "contient": "/dashboard"},
{"type": "evaluer", "script": "document.querySelector('.alert-danger')?.textContent ??
 ↪  null", "attendu": null}
```

Si la guarda falla: rpa.py se detiene antes de que se ejecute la eliminación.

5g. Reanudar una sesión (cookies persistidas)

```
# Primera invocación — autenticar y guardar la sesión
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://app.example.com/login/ \
  --actions /tmp/login.json \
  --sauver-session /tmp/dinoer/session.json

# Invocaciones posteriores — reutilizar la sesión (sin volver a iniciar sesión)
```

```
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://app.example.com/dashboard/ \
  --reprendre-session /tmp/dinoer/session.json
```

Señal de deriva de sesión: si la sesión ha expirado, `boussole.session_derive: true` en el JSON. En ese caso: reinicia el inicio de sesión completo sin `--reprendre-session`.

5h. No regresión estructural — `--replay-verifier` (v1.17.0)

```
# Primera ejecución — guardar la referencia estructural
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /opt/dinoer/scenarios/dashboard.json \
  --sauver-verifier-reference /tmp/dashboard.ref.json

# Ejecuciones posteriores — comparar
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /opt/dinoer/scenarios/dashboard.json \
  --replay-verifier /tmp/dashboard.ref.json
```

Compara `http_status`, `dom_stats` y los resultados de `evaluer` contra la referencia guardada. Veredicto en stderr:

```
{"type_comparaison": "replay_verifier", "verdict": "stable", "diffs": []}
```

Exit 1 en `verdict: "regression"`, con `diffs` listando cada campo discordante (`reference` vs `obtenu`). Las dos opciones son mutuamente excluyentes.

5i. Reanudar un escenario largo tras un fallo — `--checkpoint` (v1.17.0)

```
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /opt/dinoer/scenarios/long_audit.json \
  --checkpoint /tmp/long_audit.checkpoint.json
```

Si el escenario falla a mitad de camino, se escribe el archivo de checkpoint con el recuento de acciones completadas y un archivo de sesión. **Vuelve a lanzar exactamente el mismo comando** para reanudar: las acciones ya completadas se omiten. Con éxito total, el archivo de checkpoint se elimina automáticamente.

Una ejecución detenida por un límite de navegación (`max_actions_par_run`/`max_pages_par_run`) se trata igual que un fallo parcial desde v1.17.2 — el checkpoint se actualiza con el progreso real, no se elimina.

El estado del DOM (modales abiertos, formularios parcialmente rellenados) nunca se preserva entre reanudaciones — solo se preservan las cookies/`localStorage` y la posición en la lista de acciones. No confíes en `--checkpoint` para reanudar a mitad de un único formulario multipaso; solo reanuda en los límites entre acciones.

5j. Apuntar a elementos dentro de un iframe (v1.17.0)

No existe numeración de elementos dentro de un iframe (mismo origen o origen cruzado) — apunta a él por selector CSS directamente:

```
{"type": "cliquer_iframe", "iframe_selecteur": "iframe#paiement", "selecteur":
 ↪  "button.valider"},
{"type": "remplir_iframe", "iframe_selecteur": "iframe#paiement", "selecteur":
 ↪  "input[name=cvv]", "valeur": "depuis_secrets", "secret_cle": "cvv"}
```

`remplir_iframe` soporta `valeur: "depuis_secrets"` exactamente igual que `remplir` (sección 4b) — nunca una credencial en texto plano en el escenario. Si el elemento objetivo rechaza la interacción (por ejemplo, una región `contenteditable` en estado de solo lectura), añade `"force": true` a `cliquer_iframe` — misma semántica que `cliquer` (sección 7e).

Para encontrar el selector interno: usa `evaluer` sobre el contenido del iframe si es del mismo origen (`document.querySelector('iframe').contentDocument...`), o consulta el propio marca-

do/documentación de la aplicación objetivo si es de origen cruzado.

5k. Iframes anidados — `iframe_chemin` (v1.18.0)

Un iframe dentro de otro iframe: sustituye `iframe_selecteur` por `iframe_chemin`, un array ordenado — un selector CSS por nivel de anidamiento, del más externo al más interno.

```
{"type": "cliquer_iframe", "iframe_chemin": ["iframe#wrapper", "iframe#paiement"],
 ↪  "selecteur": "button.valider"},
{"type": "remplir_iframe", "iframe_chemin": ["iframe#wrapper", "iframe#paiement"],
 ↪  "selecteur": "input[name=cvv]", "valeur": "depuis_secrets", "secret_cle": "cvv"}
```

`iframe_selecteur` (un solo frame) e `iframe_chemin` (descenso anidado) son mutuamente excluyentes — se requiere exactamente uno por acción. Para un iframe de un solo nivel, sigue usando `iframe_selecteur` (sección 5j).

---

6. Acciones — referencia completa

Tipo	Parámetros obligatorios	Parámetros opcionales	Notas
`naviguer`	`url`	—	Recarga HTTP completa. Se cuenta en `respect.pages_visitees`
`cliquer`	`selecteur`	`force` (bool), `repli_js` (bool)	`force: true` evita elementos ocultos por CSS o un showModal. `repli_js: true` reintenta vía JS si el clic nativo sigue fallando (v1.22.0) — rechazado con `--no-evaluer` (exit 2, antes del lanzamiento)
`remplir`	`selecteur`, `valeur`	`secret_cle`	`valeur: "depuis_secrets"` requiere `secret_cle`; `"depuis_secrets_totp"` para TOTP
`evaluer`	`script`	`attendu`, `contient`, `motif`	JS ejecutado en el navegador. Aserciones solo para rpa.py
`defiler`	`px` o `selecteur`	—	Desplazamiento vertical en píxeles (`px`) o desplazamiento hasta un elemento (`selecteur`)
`pause`	`ms`	—	Retardo fijo en ms. Prefiere `attendre_selecteur_present` para señales del DOM
`attendre`	`selecteur`	—	Espera a que el selector CSS se vuelva visible (`state=visible`, valor por defecto de Playwright — corregido el 16/08/2026, idéntico a `attendre_selecteur_present`)
`attendre_navigation`	—	—	Espera a `networkidle` (fin de las solicitudes de red)

Tipo	Parámetros obligatorios	Parámetros opcionales	Notas
`attendre_url`	`motif`	`attendre_changement` (bool)	Coincidencia de subcadena en la URL. `attendre_changement: true` espera primero una navegación real (consulta la trampa FR-55)
`attendre_selecteur_present`	`selecteur`	—	Espera a que el elemento sea visible (`state=visible`)
`attendre_absence`	`selecteur`	`delai_initial_ms`	Espera a que el elemento se elimine del DOM (`state=detached`)
`attendre_reseau_calme`	—	`timeout_ms`	500 ms de silencio de red. `timeout_ms`: duración máxima antes de rendirse
`attendre_mfa_ntfy`	`selecteur`	`timeout`	Espera un código TOTP vía ntfy y lo rellena en el campo
`nettoyer_overlay`	`selecteur`	—	Oculta overlays bloqueantes (banner de cookies, modal) — selector explícito, sin autodetección
`declencher_scenario`	`scenario`	—	Inserta en línea las acciones de un subescenario. Profundidad máxima: 5 (rpa.py)
`extraire_texte`	—	—	Texto de página depurado a partir del DOM renderizado — `extraction_texte` (`titre`, `texte`, `url`, `date_capture`)
`cliquer_iframe`	`iframe_selecteur` | `iframe_chemin`, `selecteur`	`force` (bool)	Clic dentro de un iframe (v1.17.0). `iframe_chemin` para iframes anidados (v1.18.0, sección 5k)
`remplir_iframe`	`iframe_selecteur` | `iframe_chemin`, `selecteur`, `valeur`	`secret_cle`	Rellenar dentro de un iframe (v1.17.0). `valeur: "depuis_secrets"` soportado

---

7. Resolver obstáculos comunes

7a. Banner de cookies / overlay bloqueante

```
{"type": "nettoyer_overlay", "selecteur": ".cookie-consent-banner, #gdpr-overlay"}
```

Colócala **antes** de cualquier otra acción de lectura/interacción. El overlay enmascara elementos en el árbol de accesibilidad.

7b. Elemento fuera del viewport

Desplázate hasta él (por cantidad o por selector), luego actúa:

```
{"type": "defiler", "selecteur": "#the-button"},
{"type": "cliquer", "selecteur": "#the-button"}
```

o

```
{"type": "defiler", "px": 600},
{"type": "cliquer", "selecteur": "button[data-testid='load-more']"}
```

7c. SPA (React, Vue, Angular) — navegar sin recarga

Tras un clic que cambia la vista en una SPA, Playwright no sabe cuándo termina la navegación.

```
{"type": "cliquer", "selecteur": "a[href*='/dashboard']"},
{"type": "attendre_url", "motif": "/dashboard"},
{"type": "evaluer", "script": "document.title", "contient": "Dashboard"}
```

Nunca asumas que un clic ha completado la navegación sin una señal del DOM. Tras un envío (submit), combina `attendre_url` con `attendre_selecteur_present` (trampa de coincidencia parcial, consulta `docs/GUIDE_LLM_INTERACTIONS.md`).

7d. Diálogo CSS o showModal()

`TimeoutError` en `cliquer` cuando el elemento es visible en el DOM = elemento oculto por CSS o dentro de un diálogo.

```
{"type": "cliquer", "selecteur": "#dialog-confirm button[type=submit]", "force": true}
```

Si `force: true` es insuficiente (error de interactuabilidad/obstrucción): añade `repli_js: true` a la misma acción (v1.22.0), o recurre a JS:

```
{"type": "evaluer", "script": "document.querySelector('#dialog-confirm
↪  button[type=submit]').click()"}
```

7e. Operación larga (spinner, trabajo por lotes)

No uses `pause` para esperar una duración fija. Espera la señal del DOM:

```
{"type": "cliquer", "selecteur": "button[data-testid='run-job']"},
{"type": "attendre_absence", "selecteur": ".spinner", "delai_initial_ms": 500},
{"type": "attendre_selecteur_present", "selecteur": ".result-container"}
```

Si la operación no ofrece ninguna señal del DOM, sondea el estado con `evaluer` y continúa cuando la evidencia esté presente.

7f. Límite alcanzado (v1.15.0)

Si `respect.plafond_atteint` está presente en la salida, la ejecución se detuvo antes de que el escenario terminara. Las acciones restantes no se ejecutaron.

Opciones: 1. Aumentar `max_pages_par_run` o `max_actions_par_run` en `dinoer.conf` 2. Dividir el escenario en varias ejecuciones 3. Reanudar una sección parcial con `--checkpoint`

7g. Campo de formulario <select>

`remplir` (`.fill()`) no funciona en `<select>`. Usa un setter JS vía `evaluer`:

```
{"type": "evaluer", "script": "(() => { const s =
↪  document.querySelector('select[name=role]'); s.value='admin'; s.dispatchEvent(new
↪  Event('change',{bubbles:true})); })()"}
```

7h. Sitio bloqueado por WAF (403 inmediato)

```
# Prueba con stealth
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url https://target.local/ --a11y --stealth
```

Si el 403 persiste con `--stealth`: el sitio usa fingerprinting TLS (JA3/JA4) o análisis de comportamiento avanzado (Cloudflare Enterprise). `playwright-stealth` no evade estas protecciones. Consulta `docs/RETOUR_EXPERIENCE.md` FR-77/FR-78/FR-79 para el contexto.

Dinoer también marca de forma pasiva un bloqueo probable — consulta la sección 3e (`respect.waf_bloquants`).

7i. La navegación inicial nunca se completa — `--wait-until` (v1.22.0)

Síntoma: `TimeoutError` en la navegación inicial, y aumentar `--timeout` no cambia nada (45 s falla exactamente igual que 10 s). Causa: por defecto Dinoer espera `networkidle` — 500 ms de silencio de red. Una página que sondea continuamente (estadísticas en vivo, contadores que se autoactualizan, paneles de administración de router) nunca produce ese silencio, así que ningún valor de timeout puede ser nunca suficientemente grande.

```
# shot.py — reconocimiento directo
/opt/dinoer/venv/bin/python /opt/dinoer/shot.py \
  --url http://target.local/ --wait-until load --a11y

# rpa.py — propagado a shot.py, así que los escenarios llegan a los mismos objetivos
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario ./admin_login.json --wait-until load
```

Un escenario puede llevarla como propiedad raíz en su lugar, manteniéndose autocontenido:

```
{"url": "http://target.local/", "wait_until": "load", "actions": [...]}
```

La opción de línea de comandos tiene prioridad sobre la propiedad del escenario.

Valor	Espera a	Úselo cuando
`networkidle`	500 ms de silencio de red	por defecto — manténgalo salvo que falle
`load`	evento `load` (página y subrecursos)	sondeo continuo / estadísticas en vivo
`domcontentloaded`	HTML parseado, subrecursos aún pendientes	página muy pesada, el DOM es todo lo que necesita

Se aplica solo a la navegación inicial — la acción `naviguer` no se ve afectada. `boussole.wait_until` informa el valor solo cuando difiere del valor por defecto.

---

8. Monitorización — comprobaciones estructurales

No existe monitorización basada en imagen en Dinoer (sin diff visual). Las comprobaciones estructurales son basadas en texto y aptas para CI.

8a. Monitorización estructural continua — `scripts/monitor-verifier.sh` (v1.18.0)

Monitoriza la *estructura* (`http_status`, `dom_stats`, `evaluations`) — cero imagen, cero llamada a LLM, construido sobre `--replay-verifier` (sección 5h).

```
# Primera ejecución: crear la referencia estructural.
/opt/dinoer/venv/bin/python /opt/dinoer/rpa.py \
  --scenario /opt/dinoer/scenarios/example_login.json \
  --sauver-verifier-reference /opt/dinoer/references/example_login.ref.json

# Una verificación y alerta que se ejecuta periódicamente (no es un demonio), ejecútela
↪  repetidamente mediante cron.
```

```
# En el canal git-clone, los archivos scripts/*.sh nunca se despliegan a /opt/dinoer/.
# de modo que se ejecute desde el código fuente de Git, como tu propio usuario. En el canal
 ↪  .deb,
# los tres archivos comprimidos (monter/demonter-repertoire-chiffre,
# (monitor-verifier) están instalados bajo /opt/dinoer/scripts/ — corregido.
# 15/08/2026, este comentario es anterior a la construcción real de ese canal.
bash ~/git/Dinoer/Dinoer/scripts/monitor-verifier.sh \
  --scenario /opt/dinoer/scenarios/example_login.json \
  --reference /tmp/ref_sillage.json \
  --ntfy-topic dinoer-monitoring

# crontab -e (su propio crontab)
*/15 * * * * bash ~/git/Dinoer/Dinoer/scripts/monitor-verifier.sh \
  --scenario /opt/dinoer/scenarios/example_login.json \
  --reference /opt/dinoer/references/example_login.ref.json \
  --ntfy-topic dinoer-monitoring \
  >> /var/log/dinoer/cron-structural.jsonl 2>&1
```

Estable ⟶ silencio. Regresión ⟶ una notificación `ntfy` con el diff. Cada invocación es un proceso aislado — sin daemon, sin riesgo de fuga de memoria, y los límites de Navegación Respetuosa se reinician limpiamente en cada pase.

Corregido el 16/08/2026: el script antes llamaba a `rpa.py --no-capture --replay-verifier`, pero `--no-capture` ya no era una opción de `rpa.py` — cada ejecución real fallaba en argparse. Se eliminó la opción muerta (Dinoer no tiene ruta de imagen por defecto, eliminarla no cambia nada más).

Matiz del bloqueo de lectura de la guía: si se invoca bajo un usuario del SO distinto (por ejemplo, una cuenta de servicio del sistema), ese usuario necesita validar `--guide-version` una vez (`~<home>/.config/dinoer/guide_state.json`).

---

9. Registro de operaciones

El registro es `/var/log/dinoer/operations.jsonl`. Si la ruta de registro configurada está dentro del directorio cifrado y este no está montado, las entradas se redirigen a una alternativa local (escritura degradada, 700/600) en lugar de escribirse en texto claro en el host sin cifrar.

```
# Leer las últimas 10 entradas
tail -n 10 /var/log/dinoer/operations.jsonl | python3 -m json.tool

# Filtrar por objetivo (herramienta journal.py)
/opt/dinoer/venv/bin/python /opt/dinoer/journal.py \
  --cible app.example.com

# Filtrar solo operaciones mutantes
/opt/dinoer/venv/bin/python /opt/dinoer/journal.py \
  --cible app.example.com --mutatif

# Desde una fecha
/opt/dinoer/venv/bin/python /opt/dinoer/journal.py \
  --cible app.example.com --depuis 2026-07-01

# Solo ejecuciones fallidas (v1.20.0) — resultado != éxito
/opt/dinoer/venv/bin/python /opt/dinoer/journal.py \
  --cible app.example.com --erreurs
```

Campos de cada entrada:

Campo	Significado
`ts`	Marca de tiempo ISO 8601
`operation_id`	Identificador único de la ejecución (v1.16.0), nombra su directorio de archivos temporales
`outil`	`"shot.py"`, `"rpa.py"` o `"campagne.py"` — corregido el 15/08/2026, se documentaba como `mode` (`shot.py`/`rpa.py`), un campo que el código nunca escribió
`version`	Versión de Dinoer
`cible_url`	URL objetivo
`resultat`	`"succes"` o `"echec"`
`mutatif`	`true` si hubo al menos una acción de escritura
`hostname_executant` / `utilisateur_executant` / `profil_actif`	Brújula de ejecución (host, usuario del sistema, perfil de operador activo)
`intention`	Etiqueta pasada vía `--intention` o el campo `intention` del escenario — presente solo si se definió
`source_scenario`	Solo el nombre del archivo de escenario, sin ruta (v1.18.0) — presente solo en modo RPA. Corregido el 15/08/2026: esta tabla incluía también un campo `scenario` (ruta completa) que el código nunca escribió
`chainage`	Lista de `{scenario, profondeur, action_debut, action_fin}` — presente solo cuando se usó `declencher_scenario`
`actions`	Lista resumida de acciones — presente cuando la ejecución tuvo acciones
`actions_raw`	Lista completa de acciones neutralizadas — presente solo en una ejecución exitosa con acciones
`captures`	Referencia(s) a capturas estructurales — presente solo cuando la ejecución produjo alguna
`erreur`	Presente solo en caso de fallo
`respect`	El registro de navegación de la ejecución — presente solo cuando está definido
`evaluations`	Valores `{script, valeur_retournee}` depurados — presente solo si se ejecutaron acciones `evaluer`

No existe ningún campo `duree_ms` en el diario — corregido el 15/08/2026, esta tabla antes incluía uno que el código nunca escribió. El tiempo por acción es `latences_actions`, en la salida JSON de una ejecución, no en la entrada del diario.

9a. Rotación de logs (G-36)

Dinoer no incluye una configuración de logrotate — `/var/log/dinoer/operations.jsonl` crece sin límite hasta que el administrador instala una. `lib/journal.py` abre y cierra el archivo en cada escritura (sin descriptor de archivo persistente entre ejecuciones), específicamente para que el comportamiento **por defecto** de logrotate (renombrar el archivo actual, crear uno nuevo) funcione correctamente sin ninguna opción especial: la siguiente escritura reabre la ruta y encuentra el nuevo inodo.

No añadas `copytruncate` a una configuración de logrotate de Dinoer — es innecesario aquí y reintroduce una ventana de pérdida de escritura. Ejemplo `/etc/logrotate.d/dinoer`:

```
/var/log/dinoer/operations.jsonl {
    weekly
    rotate 8
    compress
    delaycompress
    missingok
    notifempty
```

```
    create 0640 dinoer dinoer
}
```

`journal.py` (el lector) ya sigue los archivos rotados de forma transparente (`operations.jsonl`, `.1`, `.2.gz`, …) — no hace falta ningún paso adicional tras la rotación.

---

10. Opciones de línea de comandos — referencia

shot.py

Opción	Por defecto	Descripción
`--version`	—	Imprime la versión instalada y termina inmediatamente — sin Playwright, sin ningún otro argumento necesario (v1.18.0)
`--guide-version X.Y`	—	Prueba de haber leído `docs/GUIDE_LLM.md` — obligatoria salvo que ya exista un marcador local válido (v1.18.0, sección 1)
`--url URL`	obligatorio	URL a capturar
`--actions FILE`	—	Archivo JSON de acciones secuenciales
`--action JSON`	—	Acción única como JSON en línea — cuida el escapado, prefiere `--actions FILE` para acciones con mucho JS
`--attendre-selecteur SEL`	—	Espera a un selector antes de terminar la ejecución
`--timeout MS`	10000	Timeout de Playwright por acción (ms)
`--wait-until VALUE`	`networkidle`	`networkidle`|`load`|`domcontentloaded` — solo navegación inicial (v1.22.0, sección 7i)
`--largeur PX`	1280	Ancho del viewport
`--hauteur PX`	720	Alto del viewport
`--a11y`	desactivado	Incluye el árbol de accesibilidad en el JSON
`--stealth`	desactivado	Modo sigiloso playwright-stealth (v1.15.0)
`--secrets FILE`	—	Ruta explícita a un archivo de credenciales
`--auth-indicator SEL`	—	Selector CSS presente solo en sesión autenticada
`--auth-indicator-negative SEL`	—	Requiere `--auth-indicator`; selector CSS presente solo fuera de sesión autenticada
`--ignorer-waf`	desactivado	Un bloqueo de WAF detectado degrada `niveau_confiance` pero ya no fuerza `pret_a_agir: false` por sí solo (v1.17.2, sección 3e)
`--http-credentials`	desactivado	Resuelve credenciales HTTP Basic Auth desde el archivo de credenciales, delimitadas al origen del objetivo (v1.21.0, sección 4g)

Opción	Por defecto	Descripción
`--ignore-tls-errors`	desactivado	Acepta TLS inválido en objetivos LAN/dev controlados — nunca en internet público (v1.15.1)
`--no-evaluer`	desactivado	Rechaza la acción **evaluer** (y `repli_js`) para toda la ejecución — recomendado contra formularios sensibles (v1.15.1)
`--no-filtre-evaluer`	desactivado	Desactiva la neutralización en stdout de los valores devueltos por **evaluer**, las URLs y los mensajes de error — solo para ejecuciones de depuración explícitas; cuando está desactivado, se fija `boussole.filtre_evaluer_actif: false` (v1.23.0)
`--intention TEXT`	—	Etiqueta de negocio registrada en el log
`--sauver-session FILE`	—	Guarda las cookies tras las acciones
`--reprendre-session FILE`	—	Reanuda una sesión guardada
`--source-scenario NAME`	—	Interno (plomería de rpa.py para el log — no para llamadas directas)
`--chainage JSON`	—	Interno (plomería de rpa.py para el log — no para llamadas directas)

rpa.py

Propaga todas las opciones relevantes de shot.py, más:

Opción	Descripción
`--version`	Imprime la versión instalada y termina inmediatamente (v1.18.0)
`--guide-version X.Y`	Prueba de haber leído `docs/GUIDE_LLM.md` — verificada de forma independiente, misma regla que shot.py (v1.18.0)
`--scenario FILE`	Ruta al escenario JSON o YAML (obligatorio)
`--url URL`	Sobrescribe la URL del escenario sin modificar el archivo
`--stealth`	Propagada a shot.py
`--wait-until`	Propagada a shot.py (v1.22.0, sección 7i)
`--ignorer-waf`	Propagada a shot.py (v1.17.2, sección 3e)
`--http-credentials`	Propagada a shot.py; también se puede fijar como propiedad raíz del escenario `"http_credentials": true` (v1.21.0, sección 4g)
`--auth-indicator-negative`	Requiere un `auth_indicator` (CLI o propiedad raíz del escenario)
`--sauver-verifier-reference FILE`	Guarda la referencia estructural para `--replay-verifier` (v1.17.0, sección 5h)
`--replay-verifier FILE`	Compara la ejecución contra una referencia estructural, exit 1 en caso de regresión (v1.17.0, sección 5h)
`--checkpoint FILE`	Reanuda un escenario largo tras un fallo a mitad de ejecución (v1.17.0, sección 5i)

campagne.py (pipeline de investigación)

Opción	Descripción
`--manifeste FILE`	Manifiesto de campaña (JSON) — requiere `id_campagne` + `cibles`
`--id-campagne ID`	Identificador de campaña (usado en el manifiesto y la extracción)
`--extraire-cible DEMANDE`	Extracción dirigida sobre un corpus ya recopilado, sin síntesis. Requiere `--id-campagne` o `--corpus`
`--corpus FILE`	Ruta directa a un `collecte.jsonl` — alternativa a `--id-campagne` para `--extraire-cible`
`--format-extraction {json,markdown,html}`	Formato de salida de `--extraire-cible` (por defecto: `json`) — añadido el 15/08/2026
`--desactiver-cache`	Omite la caché de búsqueda
`--purger-cache`	Purga toda la caché de búsqueda
`--purger-cache-avant-jours N`	Purga las entradas de caché anteriores a N días

Tipos de objetivo en el manifiesto: `query`, `url`, `produit`, `table_reference`. Artefactos: el `/var/log/dinoer/operations.jsonl` compartido + un `collecte.jsonl` por campaña. Detalle completo: `campagne.py --help`.

---

11. Códigos de salida y salida JSON

Códigos de salida

Código	Causa	Qué hacer
0	Éxito	—
1	Error de Playwright, acción fallida, aserción de rpa.py, `action_secret_en_clair`	Lee `erreur` en el JSON. Consulta `GUIDE_LLM_INTERACTIONS.md`
1	`guide_non_lu` — `--guide-version` ausente/incorrecta, sin marcador válido (v1.18.0)	Se dispara antes de que Playwright se inicie. Lee `docs/GUIDE_LLM.md`, vuelve a lanzar con `--guide-version X.Y` (sección 1)
2	Argumentos incompatibles, `arguments_incompatibles`, `url_scheme_interdit`, `chemin_sensible_refuse`	Lee `message` — nombra el conflicto
3	Módulo `playwright` no encontrado	Invoca vía `/opt/dinoer/venv/bin/python`
42	`SecretsFermesError` — directorio cifrado no montado, o checksum inválido	Móntalo, o verifica el archivo de credenciales
43	`SecretsNonConfigureError` — sin `secrets_dir` configurado	Configura `secrets_dir` en `dinoer.conf` (undo de un ejemplo faltante: crea `/opt/dinoer/dinoer.conf`)

Estructura de la salida JSON

```
{
  "succes": true,
  "http_status": 200,
  "url_finale": "https://target.local/dashboard",
  "erreurs_js": [],
  "erreurs_console": [],
```

```
  "duree_ms": 2400,
  "horodatage": "2026-07-01T12:00:00+02:00",
  "dom_stats": {"boutons": 14, "inputs": 9, "listes_deroulantes": 2, "formulaires": 1,
   ↪  "liens": 41, "dialogues": 0},
  "a11y_tree": "...",
  "evaluations": [],
  "extraction_texte": null,
  "latences_actions": [
    {"index": 0, "type": "naviguer", "latence_ms": 842},
    {"index": 1, "type": "cliquer", "latence_ms": 63}
  ],
  "respect": {
    "pages_visitees": 0,
    "actions_executees": 3,
    "duree_totale_ms": 2400,
    "indice_agressivite": 0.33
  },
  "etat": {
    "pret_a_agir": true,
    "niveau_confiance": "eleve",
    "raisons": ["aucun signal de friction détecté"]
  },
  "boussole": {
    "utilisateur": "operator",
    "ip_locale": "__IP_LAN__",
    "repertoire": "/opt/dinoer",
    "operation_id": "a1b2c3d4e5f6",
    "url_courante": "https://target.local/dashboard",
    "titre_page": "Dashboard — My App",
    "dernier_code_http": 200,
    "stealth_actif": true,
    "auth_status": "active",
    "respect": { "pages_visitees": 0, "actions_executees": 3, "duree_totale_ms": 2400,
     ↪  "indice_agressivite": 0.33 }
  },
  "dinoer_meta": {
    "version_shot": "1.0.1",
    "horodatage_iso": "2026-08-12T14:23:11+02:00",
    "hostname_executant": "operator-host",
    "utilisateur_executant": "operator",
    "profil_actif": "operateur.exemple.yaml",
    "url_au_moment_capture": "https://target.local/dashboard"
  }
}
```


`operation_id` (v1.16.0) siempre está presente e identifica de forma única esta ejecución — coincide con el campo `operation_id` de la entrada de esta ejecución en el registro de operaciones (sección 9), y nombra el directorio de pruebas `preuves/<AAAA-MM>/<operation_id>/` cuando se archivan capturas allí (corregido el 15/08/2026, verificado contra `lib/journal.py`: no existe ningún directorio `/tmp/dinoer/<operation_id>/` en ninguna parte del código — `/tmp/dinoer/` solo contiene el archivo de respaldo del registro de operaciones). `etat` (v1.16.0) está presente solo en la ruta de éxito. `latences_actions` (v1.20.0) siempre está presente (lista vacía si no hubo acciones), una entrada por cada acción que realmente se despachó — consulta `GUIDE_LLM_MONITORING.md` para saber cómo complementa `respect.duree_totale_ms`.

Claves condicionales (ausentes cuando están inactivas): `dom_stats`, `a11y_tree`, `evaluations`, `extraction_texte`, `auth_status`, `stealth_actif`, `session_derive`, `respect.plafond_atteint`, `respect.waf_bloquants`, `respect.indice_agressivite` (presente siempre que se haya ejecutado al menos una acción), `boussole.repli_js_utilise`, `boussole.wait_until`, `boussole.http_`

`credentials_actif`, `boussole.http_auth_requise`, `boussole.tls_errors_ignored`, `boussole.waf_ignore_actif`, `boussole.filtre_evaluer_actif`, `boussole.champs_rediges`, `actions_executees_avant_echec`, `pages_visitees_avant_echec` (solo en el JSON de fallo, v1.17.0). Consulta `GUIDE_LLM_MONITORING.md` para la tabla exhaustiva de activación.

Error — formato

```
{
  "succes": false,
  "erreur": "secrets_fermes",
  "message": "Le répertoire chiffré Dinoer est initialisé mais non monté.",
  "code_sortie_recommande": 42,
  "boussole": { "url_courante": "", "titre_page": "" }
}
```

Rutas de referencia

Ruta	Rol
`/opt/dinoer/`	Instalación de producción
`/opt/dinoer/venv/bin/python`	Python a usar en cada invocación
`/opt/dinoer/dinoer.conf`	Configuración de la máquina (secrets_dir, navigation, ntfy)
`/opt/dinoer/scenarios/`	Escenarios RPA (incluyendo `diagnostic_dom.json`)
`/opt/dinoer/docs/`	Documentación
`/opt/dinoer/references/`	Referencias de `--sauver-verifier-reference` / replay
`/tmp/dinoer/`	Archivos de sesión (`--sauver-session`/`--reprendre-session`) y el archivo de respaldo del registro — corregido el 15/08/2026: no existe ningún subdirectorio por `operation_id` aquí, ver la fila siguiente
`/var/log/dinoer/preuves/<AAAA-MM>/<operation_id>/`	Directorio de pruebas de una ejecución, aislado por `operation_id` (v1.16.0), creado solo cuando se archivan capturas
`~/Vaults/__PROJET__/Dinoer/`	Credenciales + log (volumen gocryptfs)
`~/git/Dinoer/Dinoer/`	Código fuente git (edita aquí, luego `deploy.sh`)
`/var/log/dinoer/operations.jsonl`	Registro de operaciones persistente (`journal.py`)

Desplegar tras modificar el código fuente:

```
bash ~/git/Dinoer/Dinoer/scripts/deploy.sh
```