

GUÍA NODESTUDIO

Leads automáticos con ScrapeGraphAI

El sistema completo para buscar empresas de un nicho, extraer sus datos públicos y dejarlos calificados en una planilla — todas las noches, solo.

⏱ Setup 20–30 min · Lectura 12 min

1 Quick Start

Qué vamos a construir

Un pipeline que arranca con el nombre de un nicho y termina con una planilla de empresas reales, con su contacto público y una línea que explica por qué te sirve cada una. Después lo dejás corriendo solo de noche.

Qué necesitamos

Python 3.12+, la librería open source `scrapegraphai`, un navegador (Playwright), un modelo de IA (por API o local con Ollama), un CSV y un scheduler.

Qué vamos a obtener

Un archivo `leads.csv` que crece todas las noches, sin duplicados, con una columna *Motivo* y un *Score* por empresa.

```
NICHO
  |
  ▼
BUSCAR EMPRESAS      SearchGraph + DuckDuckGo
  |
  ▼
ENTRAR A SUS WEBS    Playwright / Chromium
  |
  ▼
EXTRAER DATOS        LLM + schema Pydantic
  |
  ▼
CALIFICAR            Prompt 2 → motivo + score
  |
  ▼
GUARDAR EN PLANILLA  CSV / Google Sheets
  |
  ▼
REPETIR SOLO         cron, 03:00
```

LAS 3 PIEZAS

ScrapeGraphAI busca y extrae · **el modelo** entiende lo que hay en cada página · **tu planilla** es la memoria que evita repetir trabajo.

2 El repositorio oficial

REPO https://github.com/ScrapeGraphAI/Scrapegraph-ai

Licencia MIT, ~29,5k estrellas y 2,9k forks al momento de escribir esto. La documentación está en `docs.scrapegraphai.com/introduction`.

Open source vs. API cloud: no son lo mismo

ScrapeGraphAI existe en dos versiones con modelos de costo distintos. Esta guía usa **solo la primera**.

	Open source (`scrapegraphai`)	API cloud (`scrapegraph-py`)
Qué es	Librería Python que corrés vos	Servicio hosteado que llamás por SDK
Dónde corre	Tu máquina o tu VPS	La nube de ScrapeGraphAI
Modelo	El tuyo: OpenAI, Groq, Gemini, Azure o local vía Ollama	Gestionado por ellos
Navegador y proxies	Los configurás vos (Playwright)	Incluidos
Costo	Tokens de tu modelo + tu infra	Créditos pagos (hay tier gratuito limitado)
Licencia	MIT	SDK MIT, el servicio es pago

NO LOS MEZCLES

`pip install scrapegraphai` no te pide ninguna cuenta ni `SGAI_API_KEY`. Si un tutorial te manda a sacar una API key de scrapegraphai.com, está usando la versión cloud — que es otra cosa, y es paga.

3 Qué necesito instalar

Obligatorio

Qué	Por qué
Python 3.12 o superior	El paquete lo exige explícitamente. Con 3.11 el `pip install` falla.
scrapegraphai	La librería. Trae LangChain, Playwright y `ddgs` como dependencias.
Navegador de Playwright	Viene la librería, *no* el navegador: hay que bajarlo aparte con `playwright install`.
Un modelo de IA	API (OpenAI, Groq, Gemini, Azure) o local con Ollama. Sin modelo no hay extracción.
Un CSV	Donde caen los leads. El módulo `csv` ya viene con Python.

Opcional

Qué	Cuándo lo querés
Ollama + un modelo local	Si no querés pagar tokens. Camino B.
cron / Task Scheduler	Para la rutina nocturna. Si no automatizás, no hace falta.
Serper o SearXNG	Solo si DuckDuckGo te empieza a cortar las búsquedas.
Google Sheets	Si querés ver la planilla desde el celular en vez de un CSV local.
VPS	Si querés que corra de noche con la compu apagada.
Proxies	Solo a escala. Para 20–50 leads por noche no los necesitás.

> **LO QUE NO NECESITÁS**
> Ni cuenta en scrapegraphai.com, ni `scrapegraph-py`, ni base de datos, ni Docker, ni un servidor. Arranca todo en tu propia máquina.

4 Instalación con Claude Code

No hace falta que sepas de terminal. Abrí Claude Code en una carpeta vacía y pegale esto — el link del repo va adentro, por si el prompt viaja solo:

REPO https://github.com/ScrapeGraphAI/Scrapegraph-ai

PROMPT DE INSTALACIÓN

```
Quiero instalar y dejar andando la librería open source ScrapeGraphAI (repo:
https://github.com/ScrapeGraphAI/Scrapegraph-ai) en esta carpeta, para un proyecto
de búsqueda de leads. Paso 1: verificá que tengo Python 3.12 o superior — si no,
pará y decime cómo instalarlo antes de seguir. Paso 2: creá un entorno virtual
.venv. Paso 3: instalá scrapegraphai con pip dentro de ese entorno. Paso 4: corré
`playwright install chromium` y confirmá leyendo la salida que el navegador quedó
realmente descargado. Paso 5: creá un .env para mi clave de modelo y un .gitignore
que lo excluya. Paso 6: escribí un script test_scrape.py que use SmartScraperGraph
sobre una sola URL pública, corrélo, y mostrame el JSON. Antes de instalar nada
leé el README del repo por si cambió el método de instalación, y decime qué
comandos vas a correr.
```

Elegí el camino antes de pegarlo

Camino A — el simple. Agregale al final: *"configurá todo para usar OpenAI gpt-4o-mini con mi API key"*. Arranca en minutos y los resultados son buenos desde la primera corrida. Pagás por token.

Camino B — el económico. Agregale: *"configurá todo para usar Ollama con llama3.2, sin API keys, e instalá Ollama si no lo tengo"*. Costo en dólares cero, pero el modelo es más chico y necesita prompts más explícitos.

CLAUDE CODE NO HACE EL 100% SOLO

Hay dos puntos donde vas a tener que meter mano: **la clave del modelo** (o correr `ollama pull` vos mismo) y **la descarga del navegador** de Playwright, que en algunas redes queda bloqueada y falla en silencio. Quedate mirando esos dos pasos.

5 Qué corre por detrás (y cómo verificarlo)

Estos son los comandos que Claude Code ejecuta por vos. Sirven como referencia y para chequear que no se salteó nada.

```
python3 -m venv .venv                # entorno aislado
source .venv/bin/activate            # en Windows: .venv\Scripts\activate
pip install scrapegraphai            # la librería (verificado: v2.2.4)
playwright install chromium          # baja el navegador, ~150 MB
```

Solo para el Camino B, un comando más:

```
ollama pull llama3.2                 # modelo local, ~2 GB
```

El test que confirma que quedó bien

```
from scrapegraphai.graphs import SmartScraperGraph

config = {
    "llm": {"api_key": "TU_API_KEY", "model": "openai/gpt-4o-mini"},
    "headless": True,
}

g = SmartScraperGraph(
    prompt="Extraé el nombre del sitio y qué ofrece.",
    source="https://scrapegraphai.com/",
    config=config,
)
print(g.run())
```

Para el Camino B cambiás **solo** el bloque `llm`:

```
config = {
    "llm": {"model": "ollama/llama3.2", "model_tokens": 8192, "format": "json"},
    "headless": True,
}
```

Si eso imprime un diccionario con datos del sitio, ya está todo instalado.

6 Tu primera búsqueda

Objetivo: **centros de estética en Montevideo**. Guardá esto como `buscar_leads.py`.

```
from typing import List, Optional
from pydantic import BaseModel
from scrapegraphai.graphs import SearchGraph

class Negocio(BaseModel):
    nombre: str
    web: Optional[str] = None
    telefono: Optional[str] = None
    email: Optional[str] = None
    instagram: Optional[str] = None
    ubicacion: Optional[str] = None
    reservas_online: Optional[bool] = None

class Resultado(BaseModel):
    negocios: List[Negocio]

config = {
    "llm": {"api_key": "TU_API_KEY", "model": "openai/gpt-4o-mini"},
    "max_results": 8,               # cuántas páginas visita
    "search_engine": "duckduckgo",  # es el default, sin API key
    "headless": True,
    "verbose": True,
}

graph = SearchGraph(prompt=PROMPT, config=config, schema=Resultado)
data = graph.run()

print(data)
print(graph.get_considered_urls())   # qué páginas visitó
```

PROMPT (PEGALO ARRIBA, COMO VARIABLE)

```
Buscá centros de estética en Montevideo, Uruguay. Para cada centro, entrá a su
sitio web y extraé únicamente información pública de contacto del negocio: nombre
comercial, URL del sitio, teléfono comercial, email público, usuario de Instagram,
barrio o dirección, y si tiene sistema de reservas online (true/false). Si un dato
no está publicado en el sitio, dejalo en null: no lo inventes ni lo deduzcas.
Devolvé todos los centros encontrados dentro de la lista 'negocios'.
```

POR QUÉ EL SCHEMA ES OBLIGATORIO

`SearchGraph` fusiona lo que encontró en todas las páginas en *una sola* respuesta. Sin un schema con una lista adentro, te devuelve un resumen en prosa en vez de varios negocios.

7 Prompt 1 — reutilizable para cualquier nicho

PROMPT 1 · ENCONTRAR Y EXTRAER

```
Buscá [TIPO DE EMPRESA] en [CIUDAD, PAÍS]. Para cada empresa, entrá a su sitio web y extraé únicamente información pública de contacto del negocio: nombre comercial, URL del sitio, teléfono comercial, email público, usuario de Instagram, barrio o dirección, y [SEÑAL QUE TE IMPORTA] (true/false). Si un dato no está publicado en el sitio, dejalo en null: no lo inventes ni lo deduzcas. Devolvé todas las empresas encontradas dentro de la lista 'negocios'.
```

Tres variables y ya cambiaste de nicho:

- `[TIPO DE EMPRESA]` — "gimnasios", "estudios contables", "veterinarias".
- `[CIUDAD, PAÍS]` — cuanto más específico, mejores resultados de búsqueda.
- `[SEÑAL QUE TE IMPORTA]` — el dato que después define el score: "si tiene reservas online", "si tiene tienda online", "si tiene WhatsApp Business".

Qué campos conviene pedir

Pedí esto	No pidas esto
Nombre comercial, sitio web	Nombre o celular de empleados
Teléfono comercial publicado	Emails personales de personas
Email genérico (info@, contacto@)	Facturación o cantidad de empleados
Redes del negocio	Datos de sus clientes
Señales visibles en la web (reservas, delivery, precios, idiomas)	Cualquier cosa que no esté escrita en la página

LA REGLA DE ORO

Todo campo que el sitio no publique, el modelo lo va a *completar* si lo dejás. Por eso la frase "dejalo en null, no lo inventes" no es decorativa: es lo que separa una planilla usable de una planilla con teléfonos falsos.

8 La planilla

Columna	Para qué sirve
Empresa	Nombre comercial, como aparece en su web.
Web	URL. Es también la clave de deduplicación.
Teléfono	Comercial, publicado.
Email	Genérico del negocio.
Instagram	Para ver actividad real antes de escribir.
Ubicación	Barrio o zona: filtra por cercanía.
Reservas online	La señal del nicho. Es la que dispara el score.
Fuente	La URL desde donde salió. Te deja auditar un dato raro.
Motivo	Una línea: por qué este negocio te sirve. Sale del Prompt 2.
Score	1 a 5. Define a quién contactás primero.
Fecha encontrada	Para saber qué trajo cada corrida y detectar si se secó el nicho.
Estado	nuevo / contactado / respondió / descartado. Lo movés vos a mano.

Encabezado listo para copiar

```
empresa,web,telefono,email,instagram,ubicacion,reservas_online,fuente,motivo,sco
```

SHEETS

Pegá esa línea en A1 de una hoja nueva, Datos → Dividir texto en columnas, y ya tenés la planilla. Después subís el CSV con Archivo → Importar → Agregar a la hoja actual.

9 El diferencial: calificar los leads

Una lista de teléfonos no vale nada. Lo que vale es saber *a quién escribirle primero y por qué*. Esta es la segunda pasada: el JSON de cada lead entra, y sale con un motivo y un número.

PROMPT 2 · CALIFICAR

```
Sos analista de prospección. Te paso los datos públicos de un negocio, extraídos de su sitio web. Usando SOLO esos datos: 1) escribí en una línea por qué podría ser un buen prospecto para [TU SERVICIO]; 2) asignale un score del 1 al 5 según esta escala: 5 = tiene web activa, contacto público y NO tiene [SEÑAL]; 4 = no tiene [SEÑAL] pero le falta algún dato de contacto; 3 = está completo pero YA tiene [SEÑAL]; 2 = contacto incompleto o web sin información suficiente; 1 = no se puede contactar con datos públicos o no es del rubro. Si falta un dato, no lo asumas: bajá el score y decí qué falta. Devolvé JSON con las claves motivo, score y falta. Datos del negocio: [PEGAR JSON DEL LEAD]
```

La escala, en criterios concretos

Score	Qué significa
5	Web activa + contacto público + le falta justo lo que vos vendés.
4	Le falta lo que vendés, pero el contacto está incompleto.
3	Todo completo, pero ya resolvió eso. Sirve para otro servicio.
2	Datos pobres: no alcanza para decidir.
1	Sin forma de contacto pública, o no es del rubro.

CUIDADO CON LAS SEÑALES INVENTADAS

El score solo puede apoyarse en lo que el scraping trajo. "Parece que factura poco" o "seguro no tiene marketing" no son señales: son alucinaciones caras. Si un criterio no se puede leer de la web, no lo pongas en la escala.

10 Prompt 3 — actualizar sin duplicar

Si corrés esto todas las noches sobre el mismo nicho, el 70% de lo que encuentres ya va a estar en tu planilla. Esta es la pasada que lo resuelve.

PROMPT 3 · DEDUPLICAR Y ACTUALIZAR

```
Te paso dos listas en JSON: leads_existentes (lo que ya tengo en la planilla) y leads_nuevos (lo que encontré hoy). Compará por dominio del campo web, ignorando www y http/https. Si un lead no tiene web, compará por nombre normalizado (minúsculas, sin tildes, sin S.A. ni SRL) más teléfono. Devolvé JSON con tres claves: 'agregar' (los que no existen), 'actualizar' (los que ya existen pero traen un dato que antes estaba vacío — incluí solo los campos a completar) y 'duplicados' (los que ya estaban completos). No toques las columnas estado ni fecha_encontrada de los que ya existían.
```

La versión en código, por si preferís no gastar tokens acá

```python
from urllib.parse import urlparse

def clave(lead):
    web = (lead.get("web") or "").strip()
    if web:
        return urlparse(web).netloc.lower().replace("www.", "")
    return (lead.get("nombre") or "").strip().lower()

existentes = {clave(l) for l in leads_existentes}
nuevos = [l for l in leads_nuevos if clave(l) not in existentes]
```

POR QUÉ EL DOMINIO Y NO EL NOMBRE

"Estética Lumen", "Lumen Estética" y "LUMEN" son el mismo negocio y tres strings distintos. El dominio no cambia. Usá el nombre solo como plan B cuando no hay web.

11 Cómo hacer que corra solo todas las noches

Guardá esto como `run_leads.sh` al lado de tu script:

```
#!/usr/bin/env bash
set -euo pipefail
cd /home/TU_USUARIO/leads
mkdir -p logs
source .venv/bin/activate
python buscar_leads.py >> "logs/leads-$(date +%F).log" 2>&1 \
  || echo "FALLO $(date)" >> logs/errores.log
```

Dale permiso de ejecución y programalo:

```
chmod +x run_leads.sh
crontab -e
```

```
0 3 * * * /home/TU_USUARIO/leads/run_leads.sh   # todas las noches, 03:00
```

Cuestión	Resolución
Dónde quedan los resultados	`leads.csv` en la carpeta del proyecto, en modo append.
Cómo evita duplicados	El script lee el CSV antes de escribir y aplica la clave por dominio.
Dónde quedan los logs	Un archivo por día en `logs/`, con toda la salida.
Qué pasa si falla	`set -euo pipefail` corta la corrida, el error queda en el log del día y se anota en `errores.log`.
En Windows	Task Scheduler: tarea diaria que ejecuta el `.bat` equivalente.

> **UN NICHO POR NOCHE**
> Si querés cubrir varios, una línea de cron por nicho separadas media hora — no diez procesos en paralelo a las 3 AM. El cuello de botella es el buscador, no tu CPU.

12 Ejemplo completo

Objetivo: encontrar centros de estética de Montevideo que podrían necesitar mejorar sus reservas online.

```
Búsqueda      Prompt 1 + SearchGraph, max_results = 8
   ▼
Extracción    7 sitios visitados, 5 negocios con datos utilizables
   ▼
Planilla      5 filas nuevas en leads.csv
   ▼
Calificación  Prompt 2 sobre cada fila
   ▼
Score         ordenás de 5 a 1
   ▼
Resultado     2 leads para contactar hoy, el resto espera
```

Empresa	Web	Reservas	Motivo	Score
Estética Lumen	esteticalumen.example	No	Publica precios y agenda solo por WhatsApp manual: candidata clara para reservas online.	5
Centro Aura	centroaura.example	No	Sin reservas y con Instagram activo, pero no publica email: hay que entrar por teléfono.	4
Bella Piel	bellapiel.example	Sí	Ya tiene sistema de reservas; sirve para otro servicio, no para este.	3

DATOS FICTICIOS

Los tres negocios de arriba son inventados y los dominios usan `.example`, que no existe. Sirven para mostrar el formato de salida, no como leads reales.

Lo que hacés con esto: escribís a Lumen y a Aura hoy, revisando antes su web y su Instagram con tus propios ojos. Bella Piel queda archivada con estado `descartado` para este servicio.

13 ¿Esto realmente es gratis?

Depende de la configuración. Esta es la separación honesta:

Componente	¿Gratis?	Detalle
Librería `scrapegraphai`	Sí	MIT, corre en tu máquina.
Playwright + Chromium	Sí	Descarga libre.
Búsqueda (DuckDuckGo)	Sí	Sin API key. Tiene sus propios límites de uso.
Modelo local (Ollama)	Sí	Pagás en RAM, CPU/GPU y tiempo, no en dólares.
Modelo por API	No	Por token. Cada página escrapeada es al menos una llamada.
API cloud de ScrapeGraph	No	Créditos, con un tier gratuito acotado. **No se usa en esta guía.**
Hosting	Depende	Tu PC: cero. Un VPS: unos pocos dólares al mes.
Proxies	Depende	Solo si scrapeás a escala y te empiezan a bloquear.

La configuración que no cuesta un peso

Librería open source + DuckDuckGo + Ollama local + CSV + cron, todo en tu propia máquina. Ahí no pagás servicios: el costo es electricidad y tiempo de tu compu.

> **SOBRE "MILES DE LEADS"**
> Con modelo local el costo en dólares es cero, pero el techo real es otro: el tiempo de tu máquina y los límites del buscador. Miles de leads no salen en una noche con un modelo chico — salen de correr el sistema todas las noches durante semanas, en varios nichos.

> **MEDILO ANTES DE ESCALAR**
> `graph.get_execution_info()` te devuelve tokens y costo estimado de la corrida. Miralo con 5 leads antes de lanzarte a 500.

14 Límites y uso responsable

Qué datos tocar

Información **pública y comercial** de empresas: nombre, web, teléfono de contacto, email genérico, redes del negocio. Eso es lo que la empresa publicó justamente para que la contacten.

Lo que no: nombres, celulares o correos personales de empleados, perfiles personales, datos de sus clientes. Si un dato no te sirve para contactar a la empresa, no lo pidas. Cada campo de más es riesgo de más.

Respetar los sitios

- Mirá los términos y el `robots.txt`. Si un sitio prohíbe el scraping, sacalo de la lista y seguí.
- Mantené `max_results` bajo (5–10) y una corrida por noche. La librería trae un límite interno en la búsqueda, pero el criterio es tuyo.
- Si un sitio empieza a devolver errores, no insistas con reintentos: es su forma de decirte que pares.

Esto genera prospectos, no mensajes

El sistema termina en la planilla. A propósito. **No lo conectes a un envío automático**: mandar mensajes en masa a negocios que nunca te pidieron nada es spam, te quema la marca y en varios lados es ilegal.

Revisá cada lead antes de escribirle. El modelo se equivoca, y un mensaje mal dirigido cuesta mucho más que los diez segundos que ahorraste.

> **NOTA LEGAL**
> Si estás en Uruguay, la Ley 18.331 regula el tratamiento de datos personales. Los datos de contacto comercial publicados por una empresa son otra categoría, pero no los mezcles con datos de personas. Esto no es asesoramiento legal: si vas a operar a escala, consultá.

15 Troubleshooting

1 · PIP DICE "REQUIRES A DIFFERENT PYTHON"

Causa: el paquete exige Python 3.12 o superior y estás en 3.10/3.11.
Solución: instalá 3.12 y creá el entorno con `python3.12 -m venv .venv`.

2 · "EXECUTABLE DOESN'T EXIST AT .../CHROME-HEADLESS-SHELL"

Causa: el navegador de Playwright nunca se descargó. Pasa cuando la red bloquea el CDN: `playwright install` termina sin error visible y no baja nada.
Solución: volvé a correr `playwright install chromium` leyendo toda la salida. Si tu red lo bloquea, probá desde otra conexión o apuntá `PLAYWRIGHT_BROWSERS_PATH` a un Chromium ya instalado.

3 · "ZERO RESULTS FOUND FOR THE SEARCH QUERY"

Causa: la búsqueda no devolvió ninguna URL: consulta demasiado específica, o el buscador te cortó por volumen.
Solución: simplificá el prompt de búsqueda, bajá `max_results`, esperá unos minutos, o cambiá `search_engine` a `"bing"`.

4 · "COULD NOT IMPORT THE 'DDGS' PACKAGE"

Causa: falta o quedó vieja la dependencia de búsqueda. El paquete `duckduckgo-search` se renombró a `ddgs`.
Solución: `pip install -U ddgs`.

5 · VUELVE UN SOLO NEGOCIO, O CON CAMPOS INVENTADOS

Causa: sin schema, el grafo fusiona todo en un resumen; y un modelo chico rellena los huecos que encuentra.
Solución: pasá siempre `schema=` con una lista adentro, y dejá en el prompt la frase "si el dato no está publicado, dejalo en null".

16 Mi setup recomendado

Para alguien que quiere arrancar sin complicarse:

```
ScrapeGraphAI (librería open source)
          +
Ollama + llama3.2        # o gpt-4o-mini si querés más precisión
          +
DuckDuckGo               # el buscador por defecto, sin API key
          +
CSV  →  Google Sheets
          +
Claude para la calificación  # Prompt 2, sobre un JSON chico
          +
cron 03:00               # un nicho por noche
```

Por qué esta combinación

La parte cara del sistema es el scraping: son muchas páginas y mucho texto. Esa la hace un modelo local, gratis. La parte que realmente necesita criterio es la calificación, y esa corre sobre un JSON de diez campos: ahí sí conviene un modelo bueno, y cuesta centavos.

> **SI RECIÉN EMPEZÁS**
>
> Hacé las primeras dos o tres corridas con el Camino A (gpt-4o-mini). Ver resultados buenos rápido te dice si el nicho vale la pena. Una vez que el pipeline funciona, migrás el bloque `llm` a Ollama y dejás de pagar.

17 Cheat Sheet

Dejá esta página abierta mientras armás el sistema.

Quiero...	Hacé esto
Buscar un nuevo nicho	Prompt 1: cambiá `[TIPO DE EMPRESA]` y `[CIUDAD, PAÍS]`
Calificar los resultados	Prompt 2 + la escala 1–5
Agregar leads sin duplicados	Prompt 3 — la clave es el dominio de la web
Ejecutarlo automáticamente	`0 3 * * * /ruta/run_leads.sh`
Cambiar de nicho	La variable `PROMPT` y el nombre del CSV de salida
Visitar más (o menos) páginas	`max_results` en el config
Dejar de pagar tokens	Cambiá el bloque `llm` a `ollama/llama3.2`
Cambiar de buscador	`search_engine` : duckduckgo · bing · searxng · serper
Ver cuánto costó la corrida	`graph.get_execution_info()`
Ver qué páginas visitó	`graph.get_considered_urls()`
Evitar datos inventados	Schema Pydantic + "si no está publicado, dejalo en null"
Revisar por qué falló anoche	`logs/leads-AAAA-MM-DD.log`

18 Links

Qué	Dónde
Repo open source (el de esta guía)	https://github.com/ScrapeGraphAI/Scrapegraph-ai
Documentación	https://docs.scrapegraphai.com/introduction
Paquete en PyPI	https://pypi.org/project/scrapegraphai/
Playwright — navegadores	https://playwright.dev/python/docs/browsers
Ollama (modelos locales)	https://ollama.com
Claude Code	https://code.claude.com
SDK cloud — *no* hace falta acá	https://github.com/ScrapeGraphAI/scrapegraph-py
Precios de la API cloud	https://scrapegraphai.com/pricing

SI ALGO NO COINCIDE

El proyecto se mueve rápido. Todo lo de esta guía está verificado contra la versión `2.2.4` de `scrapegraphai`. Si un comando cambió, el README del repo manda — y ese es justamente el link que le pasás a Claude Code.



NodeStudio

Automatizamos procesos de negocio con IA y sistemas a medida.

nodestudioai.com