MADEBYBRAIN · ZUM VIDEO

Hermes Agent im echten Business: 9 Jobs als Vorlage

Inoffiziell, nicht von Nous Research gesponsert · madebybrain.com/hermes-jobs

Stand: 27.10.2026, alle Befehle auf Hermes Agent v0.22.5 geprüft (Jobs in einer Demo-Installation angelegt, Skripte getestet). Kosten: Schätzung von Hermes für unseren Betrieb vom 12. bis 25.10.2026, auf 30 Tage hochgerechnet, ohne Server und Posting-Dienst.

Drei Sorten Jobs

Typ	Was passiert	Schalter
Skript	Hermes startet ein Programm, ganz ohne KI. Kostet nichts.	`--script … --no-agent`
Wächter	Ein Skript schaut nach, die KI wacht nur bei einer Änderung auf.	`--monitor-script …`
Agent	Die KI liest, denkt und schreibt. Nur das kostet Geld.	`--model … --provider …`

Im Beispiel ein günstiges Modell über OpenRouter. Jedes andere Modell geht auch, wichtig ist nur: Jobs bekommen ein festes, günstiges Modell und erben nie dein Hauptmodell. Großgeschriebene Platzhalter (WEBSITE, DATENQUELLE, …) ersetzt du durch deine Werte. Skripte gehören nach `~/.hermes/scripts/`.

Übersicht

#	Job	Wann	Typ	KI-Kosten/Monat
1	Gedächtnis	täglich 04:00	AGENT	1,18 $
2	Posteingang sortieren	alle 30 Minuten	SKRIPT	0 $
3	Formular-Wächter	stündlich	SKRIPT	0 $
4	Kosten-Report	täglich 09:00	SKRIPT	0 $
5	Antworten verarbeiten	täglich 09:00	AGENT	1,07 $
6	Blog in drei Sprachen	dienstags 06:00	AGENT	0,55 $ (davon 0,39 $ Bild)
7	LinkedIn-Wochenplan	sonntags 20:00	AGENT	0,14 $
8	SEO-Check und Auto-Index	montags 09:00 und täglich	AGENT + SKRIPT	0,05 $
9	Update-Wächter	alle 2 Stunden	WÄCHTER	ca. 5 Cent pro Release
	Summe			**3,38 $**

Die 9 Jobs

1. Gedächtnis

AGENT täglich 04:00 · 1,18 $ im Monat

Liest die Chats der letzten 24 Stunden und schlägt neue, dauerhafte Fakten für deine AGENTS.md vor. Ändert die Datei nie selbst: Hermes schützt seine Anweisungsdateien, übernommen wird nur, was du freigibst.

Auftrag an die KI

> Lies mit session_search die Chats der letzten 24 Stunden. Suche nur neue, dauerhafte Fakten: neue Werkzeuge und Zugänge (nie Passwörter oder Schlüssel), neue oder geänderte Projekte, geänderte Abläufe und Zuständigkeiten. Vergleiche mit ~/AGENTS.md und lass weg, was dort schon steht. Schreib die Vorschläge in die Datei ~/gedaechtnis-vorschlaege.md: oben das Datum, dann je Vorschlag eine Zeile mit Abschnitt, neuem Text und Quelle. Ergänze diese eine Datei und lege keine neue pro Tag an. Ändere AGENTS.md nicht selbst. Gibt es nichts Neues, antworte nur [SILENT].

Befehl zum Anlegen

```
$ hermes cron create '0 4 * * *' 'Lies mit session_search die Chats der letzten
24 Stunden. Suche nur neue, dauerhafte Fakten: neue Werkzeuge und Zugänge (nie
Passwörter oder Schlüssel), neue oder geänderte Projekte, geänderte Abläufe und
Zuständigkeiten. Vergleiche mit ~/AGENTS.md und lass weg, was dort schon steht.
Schreib die Vorschläge in die Datei ~/gedaechtnis-vorschlaege.md: oben das Datum,
dann je Vorschlag eine Zeile mit Abschnitt, neuem Text und Quelle. Ergänze diese
eine Datei und lege keine neue pro Tag an. Ändere AGENTS.md nicht selbst. Gibt es
nichts Neues, antworte nur [SILENT].' --name 'Gedächtnis' --model deepseek/
deepseek-v4-flash --provider openrouter
```

Tipp: Übernimm die Vorschläge einmal pro Woche in einem normalen Chat: "Lies ~/gedaechtnis-vorschlaege.md, zeig mir jeden Vorschlag und trag ein, was ich bestätige."

2. Posteingang sortieren

SKRIPT alle 30 Minuten · 0 $ im Monat

Verschiebt neue Mails nach festen Regeln in Ordner: Rechnungen, Newsletter, Projekte, Spam. Keine KI, keine Tokens.

Befehl zum Anlegen

```
$ hermes cron create '*/30 * * * *' --name 'Posteingang sortieren' --script
postfach_sortieren.py --no-agent
```

Tipp: Zugangsdaten als IMAP_HOST, IMAP_USER, IMAP_PASS in ~/.hermes/.env. Zuerst mit --probe testen: zählt nur, verschiebt nichts. Die Regeln stehen oben im Skript.

3. Formular-Wächter

SKRIPT stündlich · 0 $ im Monat

Prüft, ob das Backend deines Kontaktformulars antwortet. Wenn nicht: Neustart und eine Meldung an dich. Sonst schweigt er.

Befehl zum Anlegen

```
$ hermes cron create '0 * * * *' --name 'Formular-Wächter' --script
formular_waechter.sh --no-agent --deliver telegram
```

Tipp: URL, erwarteten Text und Neustart-Befehl oben im Skript eintragen. --deliver auf deinen Kanal setzen, damit die Meldung dich erreicht.

4. Kosten-Report

SKRIPT täglich 09:00 · 0 $ im Monat

Liest die Kostenschätzung, die Hermes zu jedem Lauf speichert, und schickt dir die KI-Kosten von gestern nach Modell.

Befehl zum Anlegen

```
$ hermes cron create '0 9 * * *' --name Kosten-Report --script kosten_report.py
--no-agent --deliver telegram
```

Tipp: Die Schätzung von Hermes lag bei uns etwas über der echten Abbuchung. Einmal im Monat mit der Rechnung deines Anbieters vergleichen.

5. Antworten verarbeiten

AGENT täglich 09:00 · 1,07 $ im Monat

Liest Antworten auf deine Anfragen (Partner, Kunden, Verzeichnis-Einträge), setzt Zusagen und Korrekturen um und schreibt Entwürfe für Fragen. Prüft jede Aktion gegen die echte Datenquelle.

Auftrag an die KI

Lies die neuen Mails im Ordner ANTWORTEN (Antworten auf unsere Anfragen). Ordne jede einer Kategorie zu: Zusage, Korrektur, neue Datei (zum Beispiel ein Logo), Frage, Absage, Sonstiges. Setz Zusagen und Korrekturen in DATENQUELLE um und prüfe danach in DATENQUELLE, ob die Änderung wirklich da ist. Für Fragen schreibst du einen Antwortentwurf, gesendet wird nichts ohne meine Freigabe. Bericht: je Mail eine Zeile mit Kategorie, Aktion und Ergebnis der Gegenprobe. Gibt es keine neuen Mails, antworte nur [SILENT].

Befehl zum Anlegen

```
$ hermes cron create '0 9 * * *' 'Lies die neuen Mails im Ordner ANTWORTEN
(Antworten auf unsere Anfragen). Ordne jede einer Kategorie zu: Zusage,
Korrektur, neue Datei (zum Beispiel ein Logo), Frage, Absage, Sonstiges. Setz
Zusagen und Korrekturen in DATENQUELLE um und prüfe danach in DATENQUELLE, ob die
Änderung wirklich da ist. Für Fragen schreibst du einen Antwortentwurf, gesendet
wird nichts ohne meine Freigabe. Bericht: je Mail eine Zeile mit Kategorie,
Aktion und Ergebnis der Gegenprobe. Gibt es keine neuen Mails, antworte nur
[SILENT].' --name 'Antworten verarbeiten' --model deepseek/deepseek-v4-flash --
provider openrouter --skill <dein-mail-skill>
```

Tipp: ANTWORTEN und DATENQUELLE ersetzen (zum Beispiel deinen CRM-Ordner oder deine Datenbank). Die Gegenprobe ist der wichtigste Satz: Verlass dich nie auf den eigenen Bericht des Agenten.

6. Blog in drei Sprachen

AGENT dienstags 06:00 · 0,55 $ (davon 0,39 $ Bild) im Monat

Wählt ein Thema, recherchiert Quellen, schreibt den Artikel in drei Sprachen, baut die Website, prüft jeden Link und veröffentlicht nur, wenn alles grün ist.

Auftrag an die KI

> Schreib einen Blogartikel für WEBSITE auf Deutsch, Englisch und Spanisch. Wähle ein Thema aus THEMENLISTE, das noch nicht im Blog steht. Recherchiere zwei bis drei belastbare Quellen und verlinke sie. Lies vorher die letzten drei Artikel und halte dich an ihre Ansprache und Länge. Speichere die Artikel im Blog-Ordner, baue die Website und prüfe jeden Link auf HTTP 200. Veröffentliche nur, wenn Build und Linkprüfung fehlerfrei sind. Sonst leg den Artikel als Entwurf ab und melde den Fehler. Bericht: Titel, URLs, Zahl der geprüften Links.

Befehl zum Anlegen

```
$ hermes cron create '0 6 * * 2' 'Schreib einen Blogartikel für WEBSITE auf
Deutsch, Englisch und Spanisch. Wähle ein Thema aus THEMENLISTE, das noch nicht
im Blog steht. Recherchiere zwei bis drei belastbare Quellen und verlinke sie.
Lies vorher die letzten drei Artikel und halte dich an ihre Ansprache und Länge.
Speichere die Artikel im Blog-Ordner, baue die Website und prüfe jeden Link auf
HTTP 200. Veröffentliche nur, wenn Build und Linkprüfung fehlerfrei sind. Sonst
leg den Artikel als Entwurf ab und melde den Fehler. Bericht: Titel, URLs, Zahl
der geprüften Links.' --name 'Blog in drei Sprachen' --model deepseek/deepseek-
v4-flash --provider openrouter --workdir /pfad/zu/deiner/website
```

Tipp: --workdir auf dein Website-Projekt setzen, dann liest der Job dessen AGENTS.md. Die Linkprüfung findet auch veraltete Anweisungen, bei uns zwei tote Linkziele.

7. LinkedIn-Wochenplan

AGENT sonntags 20:00 · 0,14 $ im Monat

Schaut, was in deinen Projekten in der Woche passiert ist, und schreibt daraus sechs Posts für Montag bis Samstag.

Auftrag an die KI

> Schau mit session_search, was in den letzten sieben Tagen in meinen Projekten passiert ist. Schreib daraus sechs LinkedIn-Posts für Montag bis Samstag, je 800 bis 1.300 Zeichen, mit einem Hook in der ersten Zeile und einer echten Erfahrung pro Post. Keine Kundennamen, Links nur im ersten Kommentar. Speichere sie in ~/linkedin/woche.md mit Tag und Uhrzeit. Bericht: die sechs Hooks.

Befehl zum Anlegen

```
$ hermes cron create '0 20 * * 0' 'Schau mit session_search, was in den letzten
sieben Tagen in meinen Projekten passiert ist. Schreib daraus sechs LinkedIn-
Posts für Montag bis Samstag, je 800 bis 1.300 Zeichen, mit einem Hook in der
ersten Zeile und einer echten Erfahrung pro Post. Keine Kundennamen, Links nur im
ersten Kommentar. Speichere sie in ~/linkedin/woche.md mit Tag und Uhrzeit.
Bericht: die sechs Hooks.' --name LinkedIn-Wochenplan --model deepseek/deepseek-
v4-flash --provider openrouter
```

Tipp: Fürs Einplanen nimmst du deinen Posting-Dienst oder postest selbst. Ein kleiner Skript-Wächter kann morgens prüfen, ob der Post wirklich rausging.

8. SEO-Check und Auto-Index

AGENT + SKRIPT montags 09:00 und täglich · 0,05 $ im Monat

Der Agent vergleicht die Search-Console-Zahlen mit der Vorwoche. Ein Skript meldet neue Seiten aus deiner Sitemap jeden Tag sofort per IndexNow.

Auftrag an die KI

> Hol aus der Google Search Console für PROPERTY die Zahlen der letzten sieben Tage und der Vorwoche: Klicks, Impressionen, CTR, Position, die zehn wichtigsten Suchanfragen und Seiten. Bericht in höchstens acht Zeilen: Gesamtzahlen mit Veränderung, drei Gewinner, drei Verlierer und eine konkrete Empfehlung für diese Woche.

Befehl zum Anlegen

```
$ hermes cron create '0 9 * * 1' 'Hol aus der Google Search Console für PROPERTY die Zahlen der letzten sieben Tage und der Vorwoche: Klicks, Impressionen, CTR, Position, die zehn wichtigsten Suchanfragen und Seiten. Bericht in höchstens acht Zeilen: Gesamtzahlen mit Veränderung, drei Gewinner, drei Verlierer und eine konkrete Empfehlung für diese Woche.' --name 'SEO-Check und Auto-Index' --model deepseek/deepseek-v4-flash --provider openrouter
```

```
$ hermes cron create '40 10 * * *' --name Auto-Index --script auto_index.py --no-agent
```

Tipp: Für den Agenten brauchst du einen Search-Console-Zugang in Hermes (MCP-Server). Fürs Skript SITEMAP_URL und INDEXNOW_KEY setzen, erster Lauf merkt sich nur den Bestand.

9. Update-Wächter

WÄCHTER alle 2 Stunden · ca. 5 Cent pro Release im Monat

Ein Skript schaut bei GitHub nach einer neuen Version. Nur wenn sich etwas geändert hat, wacht die KI auf und fasst die Release Notes zusammen.

Auftrag an die KI

> Der Monitor hat eine neue Version gemeldet (Unterschied oben). Hol die Release Notes der neuen Version von https://api.github.com/repos/NousResearch/hermes-agent/releases/tags/<tag>. Fasse in fünf Zeilen zusammen, was sich für uns ändert, und gib ein Urteil: sofort handeln, testen oder ignorieren.

Befehl zum Anlegen

```
$ hermes cron create '0 */2 * * *' 'Der Monitor hat eine neue Version gemeldet (Unterschied oben). Hol die Release Notes der neuen Version von https://api.github.com/repos/NousResearch/hermes-agent/releases/tags/<tag>. Fasse in fünf Zeilen zusammen, was sich für uns ändert, und gib ein Urteil: sofort handeln, testen oder ignorieren.' --name 'Update-Wächter' --monitor-script release_monitor.sh --model deepseek/deepseek-v4-flash --provider openrouter
```

Tipp: Das Muster passt überall: Lieferantenpreise, Ausschreibungen, die Website der Konkurrenz. Wichtig: Das Monitor-Skript darf keine Uhrzeit ausgeben, sonst sieht es jedes Mal neu aus.

Der Schalter, der uns am meisten gespart hat

Jobs, die nur ein Skript starten, nie als KI-Auftrag anlegen. Bei uns liefen neun solche Jobs als Agent: 1.700 Läufe und 99 Millionen Tokens in einer Woche. Umstellen mit:

```
$ hermes cron list
$ hermes cron edit <job-id> --script <datei> --no-agent
```

Anhang: die Skripte

Auch zum Herunterladen unter madebybrain.com/downloads/hermes-jobs/ (MIT-Lizenz).

postfach_sortieren.py

```
#!/usr/bin/env python3
"""Posteingang sortieren mit festen Regeln, ganz ohne KI.
Liest neue (ungelesene) Mails per IMAP und verschiebt sie nach Regeln in Ordner.
Still, wenn nichts neu ist.
Vorlage von madebybrain.com/hermes-jobs (MIT). Läuft als Hermes-Job ohne KI (--no-
agent), z. B. alle 30 Minuten.
Zugangsdaten nur als Umgebungsvariablen (z. B. in ~/.hermes/.env): IMAP_HOST,
IMAP_USER, IMAP_PASS.
Testen ohne Verschieben: postfach_sortieren.py --probe
"""
import email
import email.header
import imaplib
import os
import re
import sys

# Regeln: Ordnername -> Muster für "Absender Betreff". Reihenfolge zählt, die erste
passende Regel gewinnt.
REGELN = [
    ("Spam",        r"(?i)\b(casino|crypto ?bonus|viagra|gewinnspiel)\b"),
    ("Rechnungen",  r"(?i)\b(rechnung|invoice|receipt|beleg|zahlungseingang)\b"),
    ("Newsletter",  r"(?i)\b(newsletter|weekly digest|unsubscribe|abmelden)\b|
noreply@|no-reply@"),
    ("Projekte",    r"(?i)\b(angebot|projekt|auftrag|termin)\b"),
]
PROBE = "--probe" in sys.argv


def dekodiere(wert):
    return "".join(t.decode(c or "utf-8", "ignore") if isinstance(t, bytes) else t
                   for t, c in email.header.decode_header(wert or ""))


imap = imaplib.IMAP4_SSL(os.environ["IMAP_HOST"])
imap.login(os.environ["IMAP_USER"], os.environ["IMAP_PASS"])
imap.select("INBOX")
_, ids = imap.search(None, "UNSEEN")
zaehler = {}
for mid in ids[0].split():
    _, daten = imap.fetch(mid, "(BODY.PEEK[HEADER.FIELDS (FROM SUBJECT)])")
    kopf = email.message_from_bytes(daten[0][1])
    text = dekodiere(kopf["From"]) + " " + dekodiere(kopf["Subject"])
    ziel = next((ordner for ordner, muster in REGELN if re.search(muster, text)),
None)
    zaehler[ziel or "bleibt im Posteingang"] = zaehler.get(ziel or "bleibt im
Posteingang", 0) + 1
    if ziel and not PROBE:
```

```
        imap.create(ziel)  # existiert der Ordner schon, passiert nichts
        if imap.copy(mid, ziel)[0] == "OK":
            imap.store(mid, "+FLAGS", "\\Deleted")
if not PROBE:
    imap.expunge()
imap.logout()
if zaehler:
    print(f"Posteingang: {sum(zaehler.values())} neue Mails" + (" (Probelauf)" if
PROBE else ""))
    for k, v in sorted(zaehler.items(), key=lambda x: -x[1]):
        print(f"  {k}: {v}")
```

formular_waechter.sh

```
#!/usr/bin/env bash
# Wächter für ein Formular-Backend (oder jeden anderen Dienst). Still, solange alles
läuft.
# Vorlage von madebybrain.com/hermes-jobs (MIT). Läuft als Hermes-Job ohne KI (--no-
agent), z. B. stündlich.
# Anpassen: URL, ERWARTET (Text, der in einer gesunden Antwort steht), NEUSTART
(Befehl zum Neustart).
URL="${WAECHTER_URL:-https://example.com/api/health}"
ERWARTET="${WAECHTER_ERWARTET:-ok}"
NEUSTART="${WAECHTER_NEUSTART:-systemctl restart mein-formular.service}"

antwort=$(curl -s -m 15 "$URL")
if ! printf '%s' "$antwort" | grep -q "$ERWARTET"; then
  bash -c "$NEUSTART"; sleep 3
  danach=$(curl -s -m 15 "$URL")
  if printf '%s' "$danach" | grep -q "$ERWARTET"; then status="läuft wieder"; else
status="IMMER NOCH DOWN"; fi
  echo "Formular-Backend war nicht erreichbar (${URL}). Neustart ausgeführt: $
{status}."
fi
# Keine Ausgabe = alles gut = Hermes meldet nichts.
```

kosten_report.py

```
#!/usr/bin/env python3
"""Kosten-Report für Hermes Agent: KI-Kosten von gestern, nach Modell.
Liest die Kostenschätzung, die Hermes zu jeder Session in ~/.hermes/state.db
speichert.
Vorlage von madebybrain.com/hermes-jobs (MIT). Läuft als Hermes-Job ohne KI (--no-
agent).
"""
import os
import sqlite3
from datetime import datetime, timedelta

DB = os.path.expanduser(os.environ.get("HERMES_STATE_DB", "~/.hermes/state.db"))
heute = datetime.now().replace(hour=0, minute=0, second=0, microsecond=0)
if os.environ.get("KOSTEN_TAG"):  # anderer Tag, z. B. KOSTEN_TAG=2026-10-25
    heute = datetime.strptime(os.environ["KOSTEN_TAG"], "%Y-%m-%d") +
timedelta(days=1)
gestern = heute - timedelta(days=1)

con = sqlite3.connect(f"file:{DB}?mode=ro", uri=True)
rows = con.execute(
```

```
    """SELECT COALESCE(model, '?'), source, COUNT(*),
              SUM(COALESCE(input_tokens,0) + COALESCE(cache_read_tokens,0) +
COALESCE(output_tokens,0)),
              SUM(COALESCE(estimated_cost_usd, 0))
       FROM sessions WHERE started_at >= ? AND started_at < ?
       GROUP BY 1, 2 ORDER BY 5 DESC""",
    (gestern.timestamp(), heute.timestamp()),
).fetchall()

if not rows:
    print(f"Kosten-Report {gestern:%d.%m.%Y}: keine Sessions.")
    raise SystemExit
summe = sum(r[4] for r in rows)
print(f"Kosten-Report {gestern:%d.%m.%Y} (Schätzung von Hermes)")
for model, quelle, n, tok, usd in rows:
    print(f"  {model[:32]:32} {quelle:9} {n:4} Läufe {tok/1e6:7.2f} Mio. Tokens
{usd:7.3f} $")
print(f"  {'Summe':32} {'':9} {sum(r[2] for r in rows):4} Läufe {sum(r[3] for r in
rows)/1e6:7.2f} Mio. Tokens  {summe:7.3f} $")
```

auto_index.py

```
#!/usr/bin/env python3
"""Auto-Index: neue Seiten aus deiner Sitemap sofort an IndexNow melden (Bing,
Yandex, Seznam und weitere).
Merkt sich bekannte URLs in einer Datei und meldet nur neue. Still, wenn nichts neu
ist.
Vorlage von madebybrain.com/hermes-jobs (MIT). Läuft als Hermes-Job ohne KI (--no-
agent), täglich.
Voraussetzung IndexNow: Schlüsseldatei <KEY>.txt im Wurzelverzeichnis deiner Website
(siehe indexnow.org).
Google nimmt IndexNow nicht an; dort reicht die Sitemap in der Search Console.
Aufruf zum Testen ohne Meldung: auto_index.py --probe
"""
import json
import os
import re
import sys
import urllib.request

SITEMAP = os.environ.get("SITEMAP_URL", "https://example.com/sitemap.xml")
KEY = os.environ.get("INDEXNOW_KEY", "")
STATE = os.path.expanduser(os.environ.get("AUTO_INDEX_STATE", "~/.hermes/
auto_index_bekannt.json"))
PROBE = "--probe" in sys.argv


def lade(url):
    req = urllib.request.Request(url, headers={"User-Agent": "hermes-auto-index"})
    return urllib.request.urlopen(req, timeout=30).read().decode("utf-8", "ignore")


def urls(sitemap):
    xml = lade(sitemap)
    locs = re.findall(r"<loc>\s*([^<]+?)\s*</loc>", xml)
    if "<sitemapindex" in xml:  # Sitemap-Index: Unter-Sitemaps einsammeln
        return sorted({u for s in locs for u in urls(s)})
    return sorted(set(locs))
```

```
aktuell = urls(SITEMAP)
bekannt = set(json.load(open(STATE))) if os.path.exists(STATE) else None
if bekannt is None:  # erster Lauf: nur merken, nichts melden
    neu = []
else:
    neu = [u for u in aktuell if u not in bekannt]

if PROBE:
    print(f"{len(aktuell)} URLs in der Sitemap, davon neu: {len(neu)} (Probelauf,
nichts gemeldet)")
    raise SystemExit
if neu:
    host = re.sub(r"^https?://([^/]+).*", r"\1", SITEMAP)
    body = json.dumps({"host": host, "key": KEY, "keyLocation": f"https://{host}/
{KEY}.txt", "urlList": neu}).encode()
    req = urllib.request.Request("https://api.indexnow.org/indexnow", data=body,
                                 headers={"Content-Type": "application/json;
charset=utf-8"})
    code = urllib.request.urlopen(req, timeout=30).status
    print(f"Auto-Index: {len(neu)} neue URL(s) gemeldet (IndexNow HTTP {code})")
    for u in neu:
        print("  " + u)
os.makedirs(os.path.dirname(STATE), exist_ok=True)
json.dump(aktuell, open(STATE, "w"))
```

release_monitor.sh

```
#!/usr/bin/env bash
# Monitor-Skript für einen Wächter-Job: gibt nur die neueste Release-Version aus.
# Hermes vergleicht die Ausgabe bei jedem Lauf. Gleich = KI schläft. Neu = KI wacht
auf und bekommt den Unterschied.
# Vorlage von madebybrain.com/hermes-jobs (MIT). Eigenes Repo über REPO=owner/name
einstellen.
REPO="${REPO:-NousResearch/hermes-agent}"
curl -s -m 20 "https://api.github.com/repos/${REPO}/releases?per_page=5" \
  | python3 -c 'import json,sys; print("\n".join(r["tag_name"] for r in
json.load(sys.stdin)))'
```

Du willst das lieber fertig? Wir richten Hermes als B.R.A.I.N. für dein Unternehmen ein, mit deinen Postfächern, deinem CRM und deinen Abläufen, auf einem Managed Server bei Mittwald in Deutschland, der dir gehört. madebybrain.com/brain