

DeLoMa – Deine Lokale Maschine

Handbuch

Vollständige Beschreibung aller Funktionen, der Bedienung und der Problembehebung.

Version 0.4 · Stand Oktober 2026

Pflegedozi · Freie Software unter GNU AGPL v3

pflegedozi.org

Inhalt

1. Über dieses Handbuch 5

Der Name 5

Wie dieses Handbuch geschrieben ist 5

Grundbegriffe 5

2. Installation und Einrichtung 7

Systemvoraussetzungen 7

Installation 7

Update vom Ollama Manager 7

Was wird zusätzlich installiert? 8

Deinstallation 8

3. KI-Modelle 10

Modelle herunterladen 10

Welches Modell wofür? 10

Speicherbedarf und die eingebaute Bremse 11

Modelle löschen 12

Modelle, die Ollama nicht kennt: GGUF und llama.cpp 12

Von Hugging Face laden 12

Kolibri-1 auf einem alten Rechner 14

4. Der Chat 16

Aufbau von oben nach unten 16

Die Schalter im Einzelnen 17

Tastenkürzel 17

Mehrere Gespräche gleichzeitig 17

Schnellbefehle 17

Die Befehlspalette 18

Artefakte 18

Bilder 19

5. Projekte 20

Die mitgelieferten Projekte 20

Eigenes Projekt anlegen 20

6. Wissensdatenbanken 21

Wie das funktioniert . . . 21
Mehrere Wissensdatenbanken . . . 21
Ordner einlesen . . . 22
Änderungen nachziehen . . . 22
Ganzes Dokument in den Chat . . . 22
Suche testen . . . 23

7. Erinnerungen und Aufgaben . . . 24
Dauerhafte Erinnerungen . . . 24
Aufgaben . . . 24
Gesprächsverlauf . . . 24
Bewertungen . . . 25

8. Der Agent . . . 26
Zwei Betriebsarten . . . 26
Das Ampelsystem . . . 26
Was der Agent nicht darf . . . 26
Ein Ablauf in der Praxis . . . 27
Abbrechen . . . 27
Welche Modelle können das? . . . 27

9. Fremdwerkzeuge einbinden (MCP) . . . 28
Programme anbinden mit dem Assistenten . . . 29
Einen Server von Hand einrichten . . . 33
Sicherheit bei Fremdwerkzeugen . . . 33
Warum Sie Werkzeuge auswählen sollten . . . 34

10. Sprache: Vorlesen und Diktieren . . . 35
Vorlesen . . . 35
Diktieren . . . 35

11. Einstellungen . . . 36
Grafikkarte . . . 37
llama.cpp . . . 38

12. Schnellzugriff . . . 40
Tastenkombination . . . 40
Symbol in der Taskleiste . . . 40
Automatisch starten . . . 40

13. Datenschutz und Sicherheit 41
Die Netzwerksperre 41
Keine Nutzungsstatistik 41
Selbst nachprüfen 41
Wo liegen Ihre Daten? 41
Sicherung 41

14. Bilder erzeugen 43
Einrichten in drei Schritten 43
Die Felder 44
Der Stil entscheidet 45
Profile je Modellfamilie 45
Gemessen am 16.10.2026 46
Ein größeres Modell: DreamShaper 8 LCM 47
Der schnelle Decoder 48
Was neben jedem Bild liegt 48
Feinsteuerung: LoRA, Maske, Steuerbild, Modellteile 49
Grafikkarte über Vulkan 49
Für Fortgeschrittene 50

15. Offline-Betrieb 51
Übertragen 51
Sichern 51

16. Problembehebung 53
Das Programm startet nicht 53
Server läuft: Nein 53
Ollama bricht mitten in der Antwort ab 53
Jedes Modell scheitert mit „Internal Server Error“ 53
Die Antworten sind schlecht 54
Alles ist im Akkubetrieb langsam 54
Der Auto-Agent ruft keine Werkzeuge auf 54
Die Web-Suche findet nichts 54
Vorlesen funktioniert nicht 55
Der Dokumente-Tab bleibt leer 55
Die Tastenkombination reagiert nicht 55
Selbsttest 55

17. Lizenz und Weitergabe 56

Was Sie dürfen 56

Was Sie im Gegenzug tun müssen 56

Kurz zusammengefasst 56

Der Name gehört nicht dazu 56

Die KI-Modelle gehören nicht dazu 57

Keine Gewährleistung 57

Wo der vollständige Text steht 57

18. Anhang 58

Alle Tabs auf einen Blick 58

Wichtige Verzeichnisse 58

Nützliche Befehle 59

Wenn gar nichts mehr geht 59

1. Über dieses Handbuch

Dieses Handbuch beschreibt DeLoMa vollständig. Es ist ein Nachschlagewerk – Sie müssen es nicht von vorne bis hinten lesen. Schlagen Sie das Kapitel auf, das Sie gerade brauchen.

> **Zuerst das andere Heft**
> Wenn Sie das Programm zum ersten Mal benutzen, lesen Sie bitte zuerst das Heft **Erste Schritte**. Es führt Sie in einer knappen Stunde von der Installation bis zur ersten Antwort. Dieses Handbuch setzt voraus, dass das Programm läuft.

Der Name

DeLoMa steht für **Deine Lokale Maschine**. Bis Version 0.2 hieß das Programm „Ollama Manager“. „Ollama“ ist eine Marke von Ollama Inc., und ein unabhängiges Programm sollte nicht so heißen, als wäre es deren Werkzeug. Am Programm selbst hat sich durch den neuen Namen nichts geändert; was sich geändert hat, steht in Kapitel 2 unter Update vom Ollama Manager.

Wie dieses Handbuch geschrieben ist

- Fachbegriffe werden erklärt, wenn sie zum ersten Mal auftauchen.
- Befehle zum Abtippen stehen in grauen Kästen.
- Farbige Kästen heben Wichtiges hervor – siehe unten.

> **Hinweis**
> So sieht eine ergänzende Information aus.

> **Tipp**
> So sieht ein Ratschlag aus, der Ihnen Arbeit spart.

> **Achtung**
> So sieht eine Warnung aus. Bitte aufmerksam lesen.

> **Wichtig**
> So sieht ein Hinweis aus, dessen Missachtung Schaden anrichtet.

Grundbegriffe

Begriff	Bedeutung
KI-Modell	Die eigentliche künstliche Intelligenz – eine Datei von 1 bis 20 GB. Wird einmal heruntergeladen und läuft danach ohne Internet.
Ollama	Das Hintergrundprogramm, das KI-Modelle ausführt. Läuft als Dienst mit und wird nicht direkt bedient.

Begriff	Bedeutung
Prompt	Die Frage oder Anweisung, die Sie eingeben.
System-Prompt	Eine Dauer-Anweisung an die KI, die bei jeder Frage unsichtbar mitgeschickt wird. Zum Beispiel: „Antworte immer auf Deutsch.“
Kontext	Alles, was die KI bei einer Frage gleichzeitig „im Kopf“ hat: Ihre Frage, den bisherigen Gesprächsverlauf, gefundene Dokumente. Der Platz dafür ist begrenzt.
Token	Die Einheit, in der KI-Modelle rechnen – ungefähr eine Silbe. Wichtig nur, weil der Kontext in Token gemessen wird.
Agent	Betriebsart, in der die KI selbst Dateien lesen oder Befehle ausführen darf – immer erst nach Ihrer Freigabe.
Werkzeug	Eine Fähigkeit, die die KI im Agent-Modus benutzen kann, etwa „Datei lesen“ oder „im Web suchen“.

2. Installation und Einrichtung

Systemvoraussetzungen

	Minimum	Empfohlen	Anmerkung
Arbeitsspeicher	8 GB	16–32 GB	Bestimmt, wie große Modelle laufen
Speicherplatz	10 GB	30 GB	Pro Modell 1–20 GB
Prozessor	4 Kerne	8 Kerne	Bestimmt das Tempo
Grafikkarte	keine	NVIDIA	Beschleunigt deutlich
System	Linux Mint / Ubuntu		Andere ungetestet

Installation

Im entpackten Programmordner ein Terminal öffnen und eingeben:

```
bash install.sh
```

Das Installationsprogramm erledigt folgendes:

- Es prüft, ob Python und die Fensterbibliothek vorhanden sind, und installiert sie bei Bedarf nach.
- Es kopiert das Programm nach `~/.local/share/deloma/`.
- Es legt eine eigene Python-Umgebung (venv) im Programmordner an und installiert dort die Zusatzpakete aus `requirements.txt`. Fehler dabei werden gemeldet, nicht verschluckt.
- Es prüft anschließend, ob wirklich alle Bestandteile angekommen sind, und bricht ab, falls etwas fehlt.
- Es legt einen Startmenü-Eintrag und den Terminal-Befehl `deloma` an.
- Es bietet an, das Einbettungsmodell `nomic-embed-text` für die Wissensdatenbank zu laden, sobald Ollama läuft.

> **Mehrfach ausführbar**
> Das Installationsprogramm ist gefahrlos mehrfach ausführbar. Wenn Sie unsicher sind, ob etwas fehlt: einfach noch einmal laufen lassen. Ihre Daten bleiben dabei unberührt.

Update vom Ollama Manager

Wer DeLoMa 0.3 oder den Ollama Manager (Version 0.2) schon installiert hat, muss nicht neu anfangen. Am einfachsten geht es mit der Update-Datei von pflegedozi.org: herunterladen, im Download-Ordner ein Terminal öffnen und eingeben:

```
bash deloma-update-0.4.sh
```

Die Datei enthält das ganze Programm samt Handbüchern und prüft sich vor dem Entpacken selbst. Dasselbe erreicht man aus dem entpackten Vollpaket heraus mit:

```
bash update.sh
```

Das Update macht der Reihe nach:

1. Es sichert Ihre Daten als Archiv nach `~/deloma-sicherung/`.
2. Es verschiebt Gespräche, Erinnerungen, Wissensdatenbanken, Einstellungen und Stimmdateien von `~/.local/share/ollama-manager/` nach `~/.local/share/deloma/`.
3. Es entfernt die alten Startmenü- und Autostart-Einträge und legt sie unter dem neuen Namen neu an.
4. Es installiert die neue Fassung wie oben beschrieben und legt die Handbücher unter `~/.local/share/deloma/handbuch/` ab.

Danach heißt der Terminal-Befehl `deloma` statt `ollama-manager`. Die KI-Modelle liegen bei Ollama und sind vom Update nicht betroffen.

> **Programm vorher schließen**
> Das Update bricht ab, solange das Hauptfenster, das Tray-Symbol oder der Hotkey-Dienst noch laufen. Schließen Sie alle drei und starten Sie das Update dann erneut.

> **Auch install.sh kann das**
> Das Installationsprogramm erkennt eine vorhandene Installation des Ollama Managers und übernimmt die Daten auf dieselbe Weise. Nur die zusätzliche Sicherung als Archiv legt allein `update.sh` an.

Was wird zusätzlich installiert?

Paket	Wofür	Ohne das Paket
chromadb, pypdf	Wissensdatenbank	Der Tab Dokumente bleibt leer
faster-whisper, sounddevice	Spracheingabe	Der Mikro-Knopf meldet einen Hinweis
piper-tts	Sprachausgabe	Vorlesen ist inaktiv
pystray, Pillow	Symbol in der Taskleiste	Kein Tray-Symbol
pynput	Tastenkombination	Kein globaler Hotkey

Das Programm startet auch ohne diese Pakete. Die betroffene Funktion meldet sich dann selbst mit einem Hinweis, statt abzustürzen.

Deinstallation

```
bash ~/.local/share/deloma/uninstall.sh
```

> **Daten bleiben**
> Die Deinstallation entfernt das Programm, nicht Ihre Daten und nicht die KI-Modelle. Wenn Sie auch die loswerden wollen, müssen Sie `~/.local/share/deloma/` und `~/.ollama/` von Hand löschen.

3. KI-Modelle

Das Modell ist das Gehirn. Ohne Modell antwortet nichts. Dieses Kapitel erklärt, welche es gibt, welches wofür taugt und wie Sie sie verwalten.



Der Tab Modelle.

Modelle herunterladen

1 Tab **Modelle** öffnen.

2 Gewünschtes Modell in der Liste suchen. Die Filter oben helfen dabei.

3 Auf **Installieren** klicken.

4 Fortschritt im Tab **Log** verfolgen.

Alternativ im Terminal, wenn Sie ein Modell kennen, das nicht in der Liste steht:

```
ollama pull modellname
```

Welches Modell wofür?

Modell	Größe	Braucht	Stärke	Bilder	Werkzeuge
qwen2.5:3b	1,9 GB	8 GB	Allrounder, schnell	nein	ja
gemma3:4b	3,3 GB	8 GB	Ausführlich	ja	nein
gemma4:e4b	9,6 GB	16 GB	Sehr gut, versteht Bilder	ja	ja

Modell	Größe	Braucht	Stärke	Bilder	Werkzeuge
qwen2.5-coder:7b	4,7 GB	16 GB	Programmieren	nein	nein
codellama:7b	3,8 GB	16 GB	Programmieren	nein	nein
qwen3:30b-a3b	18 GB	32 GB	Beste Qualität, trotz Größe schnell	nein	ja
gguf: Kolibri-1 IQ2_XS	22,5 GB	32 GB	Deutsch, lange Dokumente; langsam	nein	ja
nomic-embed-text	274 MB	–	nur Wissensdatenbank	–	–

Nicht jedes Modell kann, was es verspricht

In der Spalte **Werkzeuge** steht, ob ein Modell den Agent-Modus beherrscht. Verlassen Sie sich dabei nicht auf die Angabe von Ollama selbst: `qwen2.5-coder:7b` meldet Werkzeug-Unterstützung, gibt die Aufrufe aber als Text aus, statt sie auszuführen. Das Programm erkennt das und weist Sie darauf hin. Geprüft wurde das mit einer echten Werkzeugfunktion; der Messbericht auf pflegedozi.org zeigt, wie.

Groß heißt nicht langsam

`qwen3:30b-a3b` ist knapp viermal so groß wie `qwen2.5-coder:7b` und trotzdem auf demselben Rechner 77 % schneller. Es rechnet pro Wort nur mit 3 seiner 30 Milliarden Parameter. Wer 32 GB Arbeitsspeicher hat, bekommt damit das beste Modell dieser Liste – nur passt dann kein zweites großes Modell gleichzeitig in den Speicher.

Speicherbedarf und die eingebaute Bremse

Ein Modell muss vollständig in den Arbeitsspeicher passen, zusätzlich zum Kontext. Reicht der Speicher nicht, beendet das Betriebssystem den Ollama-Dienst – mitten in der Antwort, ohne Vorwarnung.

Damit das nicht passiert, hat das Programm einen Speicherschutz. Er vergleicht die Modellgröße mit dem freien Arbeitsspeicher und begrenzt bei Bedarf automatisch den Kontext. Auf einem Rechner mit reichlich Speicher greift er nicht ein. Wenn er eingreift, sagt er es Ihnen im Chat:

```
Kontext auf 4096 begrenzt: gemma4:e4b braucht 9.6 GB, verfügbar sind 12.0 GB.
```

Einstellbar unter **Einstellungen → Speicherschutz**: automatisch (empfohlen), aus oder ein fester Wert.

Zusatzschutz bei wenig Speicher

Zusätzlich empfehlenswert, wenn Sie wenig Arbeitsspeicher haben: Ollama daran hindern, mehrere Modelle gleichzeitig vorzuhalten. Dazu einmalig im Terminal die Befehle aus dem folgenden Kasten ausführen. Der Preis: Beim Wechsel des Modells wird das vorherige entladen und muss beim Zurückwechseln neu geladen werden.

```
sudo mkdir -p /etc/systemd/system/ollama.service.d
sudo tee /etc/systemd/system/ollama.service.d/override.conf <<'ENDE'
[Service]
Environment="OLLAMA_MAX_LOADED_MODELS=1"
ENDE
sudo systemctl daemon-reload && sudo systemctl restart ollama
```

Modelle löschen

Im Tab **Modelle** das Modell auswählen und auf **Löschen** klicken. Der Speicherplatz wird sofort frei. Sie können das Modell jederzeit erneut herunterladen.

Modelle, die Ollama nicht kennt: GGUF und llama.cpp

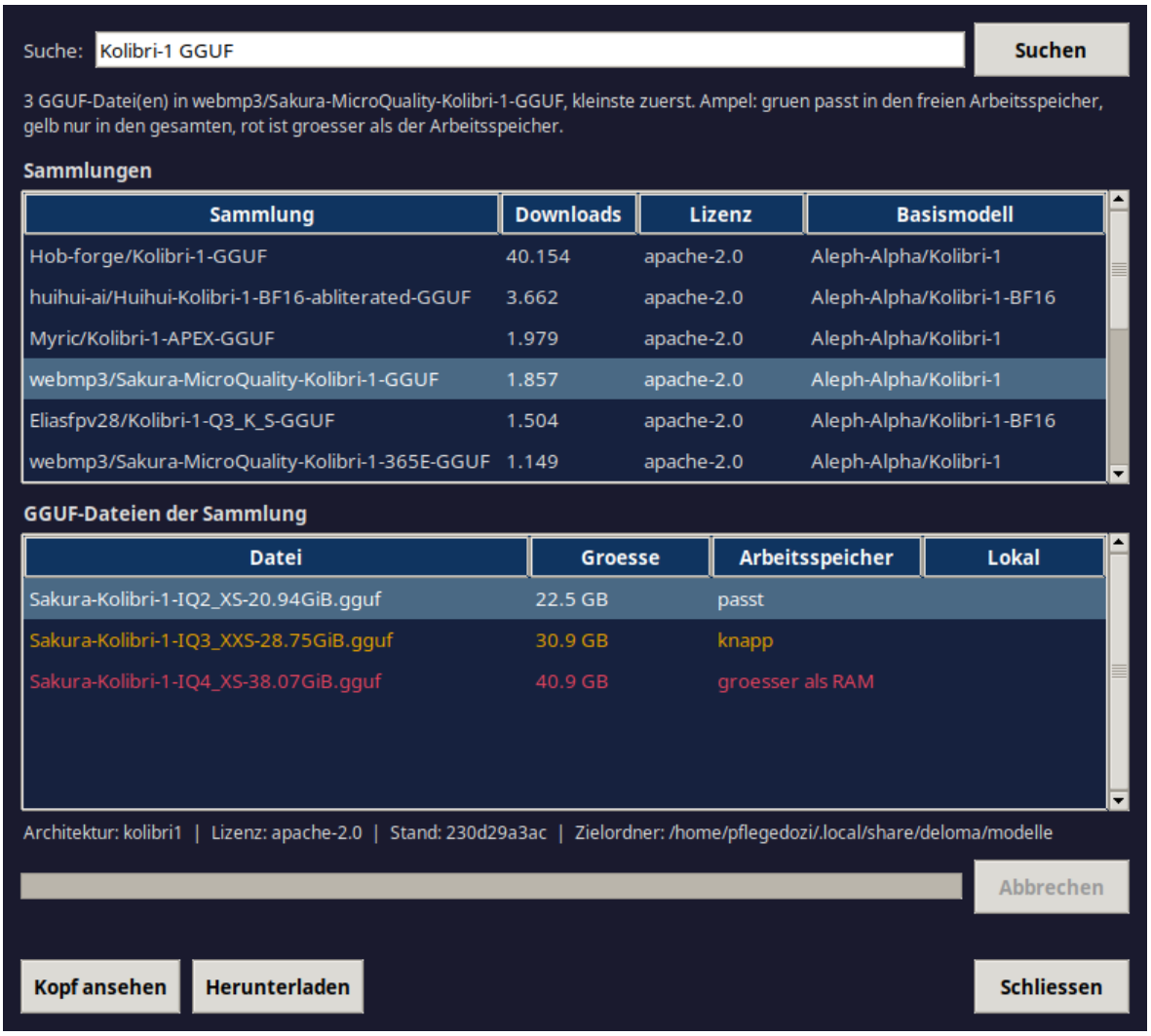
Manche Modelle gibt es bei Ollama nicht oder noch nicht – etwa **Kolibri-1** von Aleph Alpha, ein deutsches Modell, dessen Bauart Ollama im Oktober 2026 nicht laden konnte. Für solche Fälle hat DeLoMa seit Version 0.4 eine zweite Laufzeit: **llama.cpp**. Sie führt Modelldateien im GGUF-Format aus, die man direkt von Hugging Face lädt. Im Programm tragen diese Modelle den Vorsatz `gguf:` vor dem Namen, damit sie sich nie mit Ollama-Modellen verwechseln lassen. Sonst bedienen Sie sie wie alle anderen: im Chat auswählen, schreiben.

Drei Schritte, alle im Programm:

1. **llama.cpp bauen** – einmalig, unter **Einstellungen → llama.cpp** (Kapitel 11). Auf einem älteren Rechner dauert das einige Minuten.
2. **Modell laden** – im Tab **Modelle** über den Knopf **Von Hugging Face (GGUF)**.
3. **Chatten** – das Modell steht in der Auswahl als `gguf:…`. Beim ersten Chat startet DeLoMa den Server im Hintergrund; große Modelle brauchen beim ersten Laden einige Minuten.

Von Hugging Face laden

Oben im Dialog steht der **Katalog**: neun Einträge, die auf dem Messrechner von 2013 gemessen wurden, mit Größe, Lizenz, Verwendungszweck und Messwert. Die Auswahl zeigt die Einzelheiten, dazu ob der Eintrag in den Arbeitsspeicher dieses Rechners passt (passt, knapp, zu groß). **Aus dem Katalog laden** holt alles Nötige in einem Zug: beim bildverstehenden gemma-3-4b auch den Projektor, bei FLUX.2-klein Textencoder und VAE aus anderen Sammlungen samt Zuordnung, bei LoRA und ControlNet die Datei in ihren Unterordner. Jede Datei bekommt Prüfsumme und Herkunftszettel wie bei der freien Suche. Die Suche darunter bleibt für alles, was der Katalog nicht kennt.



Suche nach Kolibri-1: oben die Sammlungen, unten die Dateien der gewählten Sammlung mit der Speicher-Ampel.

Suchbegriff eingeben und auf **Suchen** klicken. Erst dann baut das Programm eine Verbindung zu huggingface.co auf; an den Server gehen nur der Suchbegriff und später der Dateiname. Oben erscheinen die **Sammlungen**, die meistgeladenen zuerst, mit Lizenz und Basismodell. Ein Klick auf eine Sammlung zeigt ihre GGUF-Dateien, die kleinste zuerst.

Die Spalte **Arbeitsspeicher** ist eine Ampel für Ihren Rechner:

- **passt** (grün): Die Datei passt samt Reserve in den gerade freien Arbeitsspeicher.
- **knapp** (gelb): Sie passt nur in den gesamten Arbeitsspeicher. Andere Programme müssen weichen, sonst wird ausgelagert.
- **größer als RAM** (rot): Sie läuft allenfalls langsam, weil ständig von der Platte gelesen wird.
- **kein Platz auf der Platte**: Für den Download fehlt der Speicherplatz.

Kopf ansehen liest nur den Anfang der Datei und zeigt Bauart, Größe in Parametern, Kontextlänge, Zahlenformate und ob eine Chat-Vorlage enthalten ist – ohne die Datei zu laden. **Herunterladen** fragt mit Ziel, Lizenz und Ampel nach und lädt dann mit Fortschrittsanzeige. Sie können abbrechen; der geladene Teil bleibt liegen, und ein

späterer Klick auf **Herunterladen** setzt dort fort. Nach dem Laden wird die Prüfsumme verglichen. Neben jede Datei legt DeLoMa einen Herkunftszettel: woher, welcher Stand, welche Prüfsumme.

Die Dateien liegen in `~/.local/share/deloma/modelle/`. Löschen geht im Tab **Status** wie bei Ollama-Modellen; dabei werden Datei und Zettel entfernt.

Vorhandene Datei einbinden… ist für GGUF-Dateien, die schon auf dem Rechner liegen – aus einem anderen Programm, von einem Stick, aus einer früheren Messung. DeLoMa fragt, ob es die Datei kopieren oder nur verknüpfen soll. Verknüpfen braucht keinen zusätzlichen Platz; die Datei muss dann aber bleiben, wo sie ist. Beim Entfernen im Tab Status verschwindet nur die Verknüpfung, nie das Original.

> **Was GGUF-Modelle in DeLoMa nicht können**
> **Bilder:** DeLoMa startet llama.cpp ohne Bildprojektor, GGUF-Modelle sehen deshalb keine Bilder. **Offline-Export:** Der Export im Tab Offline packt nur Ollama-Modelle; GGUF-Dateien kopieren Sie einfach aus dem Ordner `modelle` auf den anderen Rechner. **Eigene Varianten:** Der Dialog „Eigene Variante erstellen“ arbeitet nur mit Ollama-Modellen.

Bilder verstehen mit GGUF-Modellen

Bei llama.cpp sind Sprachmodell und Bildteil zwei Dateien: Neben dem Modell liegt ein Bildprojektor, eine GGUF-Datei, deren Name mit `mmproj` beginnt (etwa `mmproj-model-f16.gguf`). Sammlungen bildfähiger Modelle auf Hugging Face führen beide im selben Repo. Laden Sie beide über den Dialog, erkennt DeLoMa den Projektor, hält ihn aus der Modellliste heraus und ordnet ihn dem Modell zu: zuerst über denselben Eintrag im Herkunftszettel, dann über dieselbe Sammlung, sonst über den Namen ohne Quantisierungs-Endung. Beim Start bekommt llama-server den Projektor mit, das Modell meldet die Fähigkeit vision, und im Chat lassen sich Bilder anhängen wie bei Ollama-Modellen. Ohne Projektor bleibt ein Bild ein Fehler mit genau diesem Hinweis. Beim Entfernen eines Modells fragt DeLoMa, ob ein Projektor mitgehen soll, den kein anderes Modell mehr braucht. Gemessen am 17.10.2026 mit gemma-3-4b-it (Q4_K_M, 2,5 GB) und seinem Projektor (0,85 GB) auf dem Messrechner: ein Bild mit 512 × 512 Pixeln brauchte rund 50 Sekunden, bis die Antwort begann (Bild-Encoder und Prompt laufen auf dem Prozessor), danach 7,6 Tokens je Sekunde.

Kolibri-1 auf einem alten Rechner

Kolibri-1 ist ein deutsches Modell mit 78 Milliarden Parametern, von denen je Wortbaustein nur 3,5 Milliarden rechnen. Der Hersteller nennt als Mindestausstattung zwei A100-Grafikkarten. Es geht auch ohne. Gemessen im Oktober 2026 auf einem Notebook von 2013 (vier Kerne, 31 GB Arbeitsspeicher, keine nutzbare Grafikkarte) mit der kleinsten Community-Datei (2,3 Bit je Gewicht, 22,5 GB):

Was	Ergebnis
Antworttempo	6,4 Tokens je Sekunde, etwa 4 bis 5 Wörter
Einlesen langer Texte	12 bis 15 Tokens je Sekunde; ein Dokument mit 15 000 Tokens braucht rund 20 Minuten

Was	Ergebnis
13 Pflegeaufgaben, ohne Denkmodus	96 % richtig
Werkzeugaufrufe (Agent)	9 von 9
Lange deutsche Dokumente	alle richtig beantwortet, mit etwa einem Drittel der Energie von qwen3:30b-a3b
Wenig Speicher	läuft noch mit einer Grenze von 12 GB, dann langsamer
Schwächen	erfand die Auflösung der Abkürzung SIS; auf Denkstufe low 3 von 26 Läufen in einer Schleife

Der vollständige Messbericht mit allen Bedingungen und auch den verworfenen Läufen steht auf pflegedozi.org (Kolibri-Messbericht). Die Vorgaben in DeLoMa stammen aus dieser Messung: Denken aus, Kontext 16 384, Rechenfäden automatisch. Steht das Modell im Chat auf **Auto**, wählt DeLoMa Kolibri für Gespräche und lange Texte, sobald es installiert ist – für Programmieraufgaben und Bilder nimmt es weiter andere Modelle.

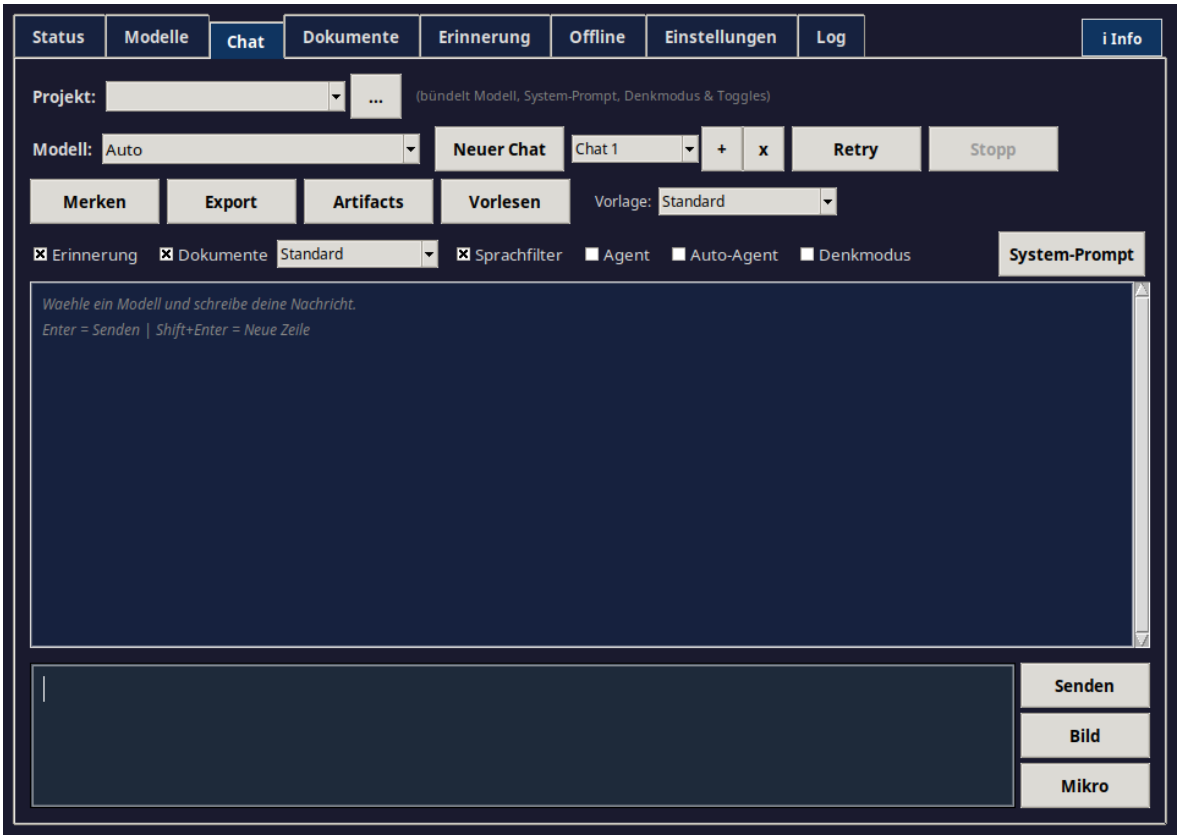
> **Denken kostet Zeit**
> Ohne ausdrückliche Vorgabe denkt Kolibri auf höchster Stufe und braucht dann minutenlang, bevor eine Antwort erscheint. DeLoMa schickt deshalb immer eine Denkstufe mit (Einstellungen → llama.cpp). Für die Pflegeaufgaben in der Messung war der Denkmodus nicht nötig. Eine Ausnahme macht DeLoMa von selbst: Sieht eine Frage nach einer Rechenaufgabe aus (Dosis, Tropfenzahl, Prozent, zwei Zahlen mit Rechenzeichen), denkt das Modell auch bei ausgeschaltetem Schalter. Das lässt sich unter Einstellungen → llama.cpp abstellen.

Das Projekt „Kolibri“

In der Messung antwortete Kolibri aus mitgegebenen Dokumenten richtig, erfand aber die Auflösung der Abkürzung SIS aus dem Gedächtnis und schrieb lang. Deshalb bringt DeLoMa 0.4 ein Projekt **Kolibri** mit (Kapitel 5): Es weist das Modell an, nur aus den mitgegebenen Quellen zu antworten, Fehlendes ausdrücklich zu benennen statt zu raten, Abkürzungen nur mit Beleg aufzulösen und sich kurz zu fassen; das Denken ist aus, die Wissensdatenbank an. Wer das Projekt löscht, bekommt es nicht wieder aufgedrängt. Eigene Dokumente gehören in die Wissensdatenbank (Kapitel 6): Kolibri liest auf dem Prozessor nur 12 bis 15 Tokens je Sekunde, deshalb bekommt es je Frage die passendsten Abschnitte statt ganzer Dokumente – bis zu acht, rund 4 000 Zeichen, etwa eine Minute Einlesen. Für eine Zusammenfassung eines ganzen Dokuments reicht das nicht, und auch der Agent hilft dabei nicht: Sein Lesewerkzeug gibt höchstens 2 000 Zeichen einer Textdatei weiter und kann keine PDFs lesen. Dafür gibt es im Tab Dokumente den Knopf **Ganzes Dokument in den Chat** (Kapitel 6); bei Kolibri passen mit der Vorgabe von 16 384 Tokens rund 49 000 Zeichen, das Einlesen dauert dann gut zehn Minuten.

4. Der Chat

Der Tab **Chat** ist der Arbeitsplatz. Hier läuft alles zusammen.

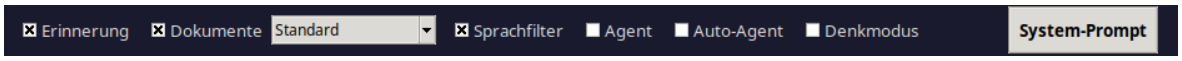


Der Chat vor der ersten Frage.

Aufbau von oben nach unten

Zeile	Inhalt
1. Projekt	Bündelt Modell, System-Prompt und Schalter. Siehe Kapitel 5.
2. Modell	Welche KI antwortet. **Auto** wählt automatisch.
3. Aktionen	Merken, Export, Artifacts, Vorlesen, Vorlage.
4. Schalter	Erinnerung, Dokumente, Sprachfilter, Agent, Denkmodus.
5. Verlauf	Das Gespräch.
6. Eingabe	Ihre Frage, mit Senden, Bild und Mikro.

Die Schalter im Einzelnen



Die Schalterleiste.

Schalter	Wirkung	Empfehlung
Erinnerung	Sucht in früheren Gesprächen nach Passendem und gibt es der KI mit.	an
Dokumente	Durchsucht Ihre Wissensdatenbank und gibt Fundstellen mit.	an, wenn befüllt
Sprachfilter	Entfernt Floskeln aus der Antwort („Das ist eine gute Frage …“).	an
Agent	Blendet eine Werkzeugleiste ein, die Sie selbst bedienen.	nach Bedarf
Auto-Agent	Die KI wählt ihre Werkzeuge selbst. Siehe Kapitel 8.	bewusst einsetzen
Denkmodus	Das Modell denkt sichtbar vor, bevor es antwortet. Nur bei Modellen, die das können.	nach Bedarf

> **Der Denkmodus kostet nichts**
> Gemessen mit `qwen3:30b-a3b`: Mit Denken antwortet das Modell knapper, die Geschwindigkeit bleibt gleich. Bei Fachfragen lohnt er sich; bei Smalltalk ist er überflüssig.

Tastenkürzel

Taste	Wirkung
Eingabe	Frage absenden
Umschalt + Eingabe	Neue Zeile, ohne zu senden
Strg + N	Neuer Chat
Strg + E	Gespräch exportieren
Strg + K	Befehlspalette öffnen
Strg + 1 bis Strg + 7	Zwischen den Tabs wechseln

Mehrere Gespräche gleichzeitig

Mit **+** neben dem Chat-Auswahlfeld legen Sie einen zweiten, dritten, beliebig vielen Chat an. Jeder hat seinen eigenen Verlauf. Mit **x** schließen Sie den aktuellen. Die Gespräche überleben einen Programmneustart.

Schnellbefehle

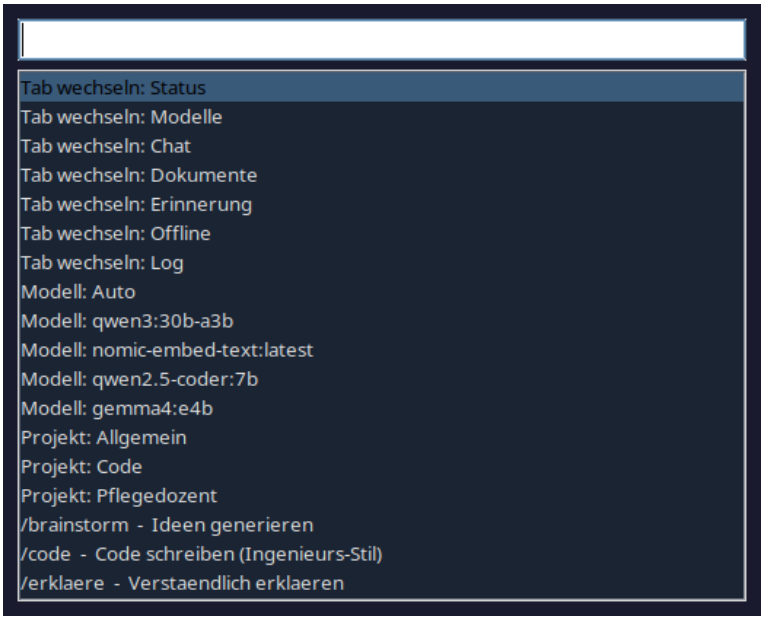
Wenn Sie Ihre Eingabe mit einem Schrägstrich beginnen, wird die Frage automatisch umformuliert:

Befehl	Wirkung
/code	Bittet um Quelltext mit Erklärung
/erklaere	Bittet um eine verständliche Erklärung
/zusammenfassen	Fasst einen längeren Text zusammen – höchstens fünf Punkte oder 200 Wörter
/plan	Erstellt eine nummerierte Schrittfolge
/kritik	Prüft einen Text kritisch
/uebersetzen	Übersetzt
/brainstorm	Sammelt Ideen
/merken	Speichert etwas dauerhaft – ohne die KI zu fragen

> **Nichts kaputtzumachen**
> Ein unbekannter Schnellbefehl wird einfach als normale Frage behandelt.

Die Befehlspalette

Strg + K öffnet ein Suchfeld. Tippen Sie ein paar Buchstaben, und es zeigt passende Projekte, Modelle, Schnellbefehle und Tabs. Mit den Pfeiltasten wählen, mit Eingabe ausführen, mit Escape schließen. Der schnellste Weg, wenn Sie wissen, was Sie wollen.

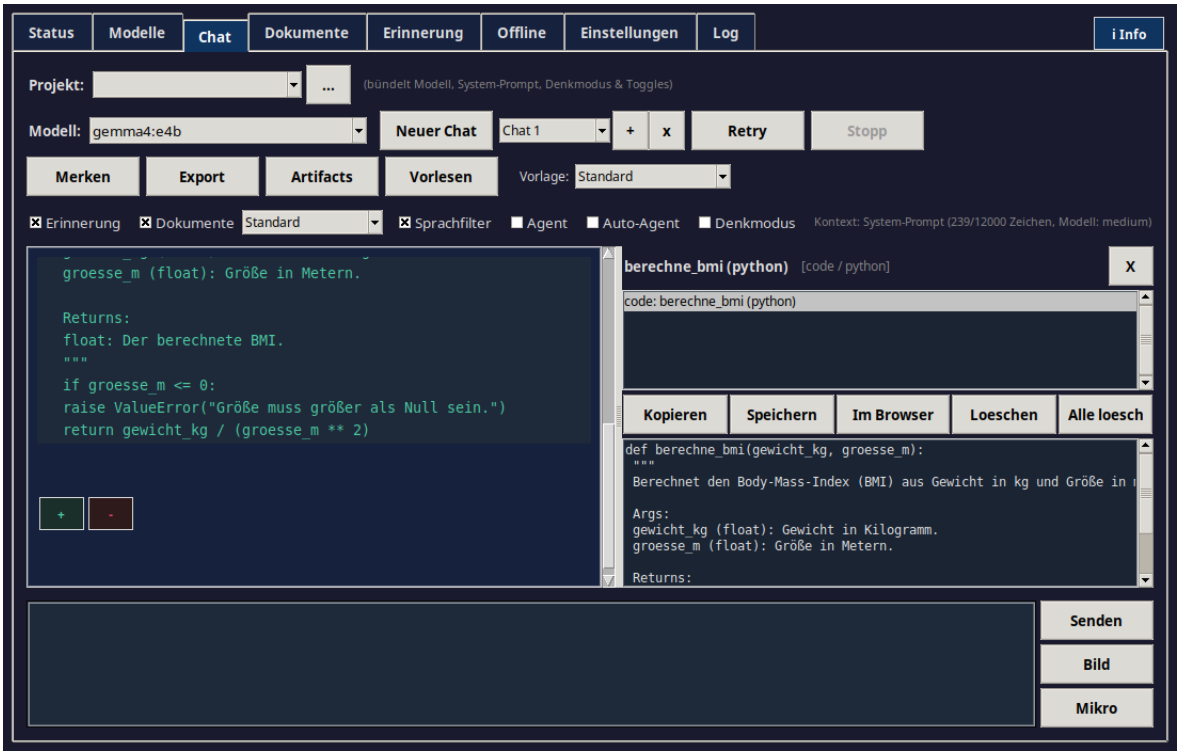


Die Befehlspalette.

Artefakte

Liefert die KI Quelltext oder eine HTML-Seite, erkennt das Programm das und legt es als Artefakt rechts neben dem Gespräch ab. Dort können Sie den Inhalt kopieren, als Datei speichern oder eine HTML-Seite direkt im Browser ansehen. Der Bereich klappt von selbst auf, sobald es etwas zu zeigen gibt, und lässt sich über den Knopf **Artifacts** wieder wegklappen. Für einen zweiten Bildschirm gibt es das Ganze auch als eigenes

Fenster.



Links das Gespräch, rechts das erkannte Artefakt – die geteilte Ansicht.

Bilder

Der Knopf **Bild** hängt ein Foto an die nächste Frage. Das funktioniert nur mit bildfähigen Modellen (siehe Kapitel 3). Steht das Modell auf **Auto**, wählt das Programm selbst ein passendes – und sagt Ihnen Bescheid, wenn keines installiert ist, statt das Bild stillschweigend zu ignorieren.

5. Projekte

Ein Projekt bündelt alle Einstellungen für eine bestimmte Art von Arbeit: welches Modell, welcher System-Prompt, welche Schalter. Ein Klick, und die KI verhält sich passend zur Aufgabe.

Die mitgelieferten Projekte

Projekt	Gedacht für
Allgemein	Alltagsfragen, keine Besonderheiten
Code	Programmieraufgaben – knapp, Code zuerst, großer Kontext, Auto-Agent an
Pflegedozent	Fachliche Fragen aus der Pflege, mit passendem System-Prompt und eingeschalteter Dokumentensuche
Kolibri	Kolibri-1 über llama.cpp (ab 0.4): nur aus Quellen antworten, Fehlendes benennen, kurz, Denken aus – siehe Kapitel 3

Eigenes Projekt anlegen

1. Im Chat auf den Knopf **…** neben dem Projekt-Auswahlfeld klicken.
2. Auf **Neu** klicken und einen Namen vergeben.
3. Modell wählen (oder Auto lassen).
4. System-Prompt eintragen – die Dauer-Anweisung an die KI.
5. Schalter setzen: Erinnerung, Dokumente, Sprachfilter, Auto-Agent, Denkmodus.
6. Kontext-Größe: **auto** richtet sich nach dem Modell. Mit small, medium oder large legen Sie fest, wie viele Zeichen Erinnerung, Dokumente und System-Prompt zusammen bekommen (6000, 12000 oder 24000). Größer heißt mehr Wissen im Gespräch, bei langsamen Modellen aber längeres Einlesen.
7. Speichern.

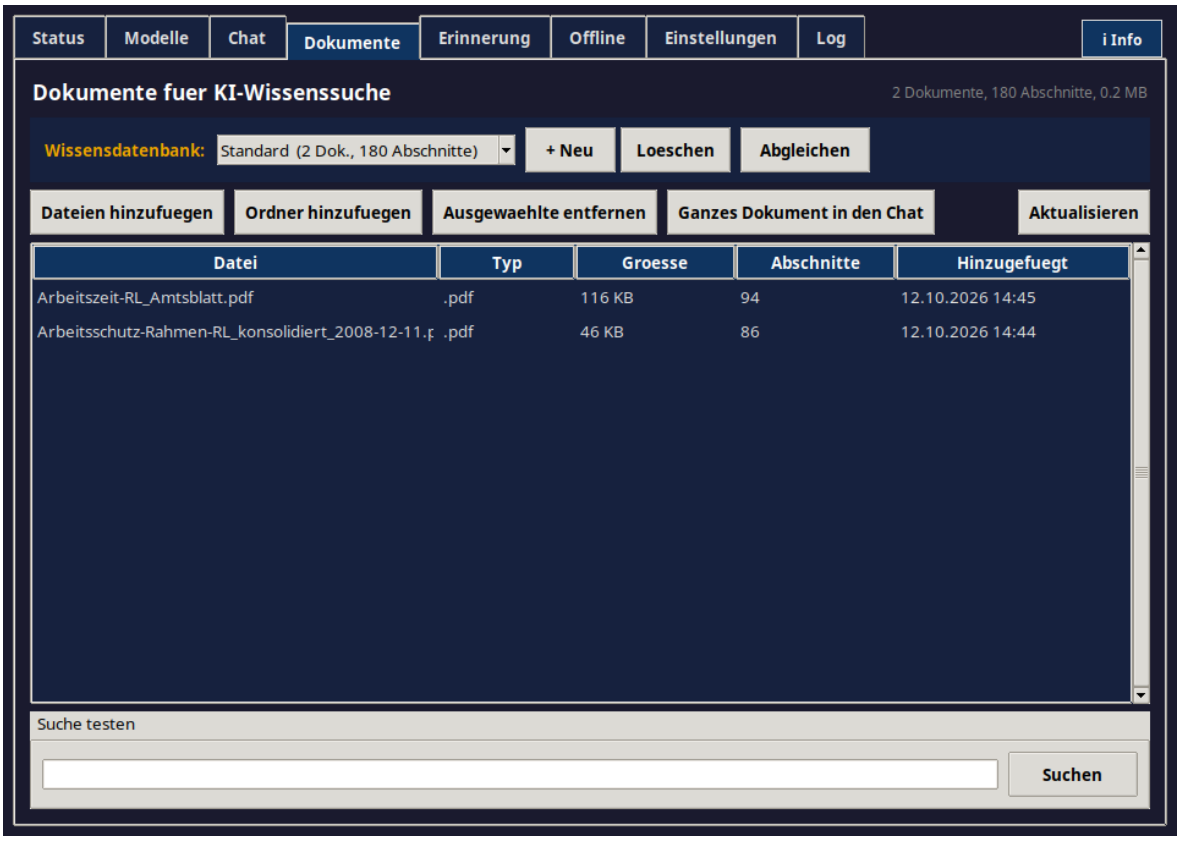
Beispiel für einen System-Prompt:

```
Du bist eine erfahrene Pflegefachkraft. Antworte fachlich korrekt
und auf Deutsch. Nenne bei fachlichen Aussagen die Grundlage.
Wenn du etwas nicht sicher weißt, sage das ausdrücklich.
```

> **Der letzte Satz ist der wichtigste**
> Ohne ihn erfinden kleine Modelle notfalls etwas, das plausibel klingt.

6. Wissensdatenbanken

Sie können eigene Dokumente hinterlegen – Pflegestandards, Leitlinien, Handbücher. Die KI durchsucht sie bei jeder Frage und antwortet aus Ihren Unterlagen statt aus ihrem allgemeinen Wissen.



Der Tab Dokumente.

Wie das funktioniert

Beim Hinzufügen wird jedes Dokument in Abschnitte zerlegt. Zu jedem Abschnitt berechnet ein kleines Zusatzmodell einen „Bedeutungsfingerabdruck“. Stellen Sie später eine Frage, wird auch dafür ein Fingerabdruck gebildet und die ähnlichsten Abschnitte herausgesucht. Der Vorteil gegenüber einer Wortsuche: Es findet auch dann etwas, wenn Sie andere Wörter benutzen als im Dokument.

> **Ohne Einbettungsmodell keine Wissensdatenbank**
> Dafür wird das Modell `nomic-embed-text` (274 MB) gebraucht. Das Installationsprogramm bietet den Download an, sobald Ollama läuft; das Programm fragt außerdem nach, wenn Sie das erste Dokument hinzufügen. Fehlt das Modell, endet jeder Upload mit „Embedding fehlgeschlagen“. Nachholen im Terminal: `ollama pull nomic-embed-text`.

Mehrere Wissensdatenbanken

Sie können mehrere getrennte Bestände anlegen – etwa „Pflegewissen“ und „Projekt Alpha“. Im Chat wählen Sie aus, welcher befragt wird. Das hält die Ergebnisse sauber: Eine Frage zur Pflege findet dann keine Projektnotizen.

1. Tab **Dokumente** öffnen.
2. Auf **+ Neu** neben der Auswahl klicken und einen Namen vergeben.
3. Dateien oder einen ganzen Ordner hinzufügen.
4. Im Chat neben dem Haken **Dokumente** die gewünschte Datenbank auswählen.

Ordner einlesen

Ordner hinzufügen liest einen ganzen Ordner samt Unterordnern ein. Versteckte Ordner und nicht lesbare Dateitypen werden dabei übersprungen. Vor dem Start sehen Sie, wie viele Dateien es sind. Unterstützte Dateitypen:

```
.pdf .txt .md .rst .csv .log .py .js .html .xml .json .yaml .yml .ini .cfg .conf
```

Änderungen nachziehen

Der Knopf **Abgleichen** prüft alle Dokumente einer Datenbank: Geänderte Dateien werden neu eingelesen, gelöschte aus dem Bestand entfernt. Erkannt wird das an einer Prüfsumme, nicht am Datum – eine nur geöffnete Datei wird also nicht unnötig neu verarbeitet.

> **Nach dem Update auf 0.4 einmal abgleichen**
> Seit 0.4 räumt DeLoMa den Text aus PDFs beim Einlesen auf: Am Zeilenende getrennte Wörter werden wieder zusammengefügt („ange-geben“ wird „angegeben“, „Sicherheits-und“ bleibt), Seitenköpfe von Amtsblatt und Bundesgesetzblatt sowie Seitenzahlen verschwinden. Der Anlass: Aus dem Seitenkopf „2003L 299/10 Amtsblatt“ hatte ein Modell eine Richtlinie „2003/299/EG“ gemacht, die es nicht gibt. Damit vorhandene Dokumente davon profitieren, nach dem Update einmal auf **Abgleichen** klicken – alle Dokumente werden dann neu eingelesen, bei großen Beständen dauert das einige Minuten.

Ganzes Dokument in den Chat

Die Wissensdatenbank gibt dem Modell zu jeder Frage nur die passendsten Abschnitte mit. Für eine Zusammenfassung oder eine Prüfung eines ganzen Dokuments ist das zu wenig. Der Knopf **Ganzes Dokument in den Chat** liest das in der Liste gewählte Dokument (oder eine beliebige Datei, wenn nichts gewählt ist) mit derselben Bereinigung wie beim Einlesen, fragt, was das Modell damit tun soll, und legt Aufgabe und Text in das Eingabefeld des Chats. Vorher zeigt er, wie viele Zeichen in den Kontext des gewählten Modells passen, ob gekürzt wird (an einer Satzgrenze) und bei llama.cpp-Modellen, wie lange das Einlesen dauern wird. Gesendet wird erst, wenn Sie im Chat auf Senden klicken.

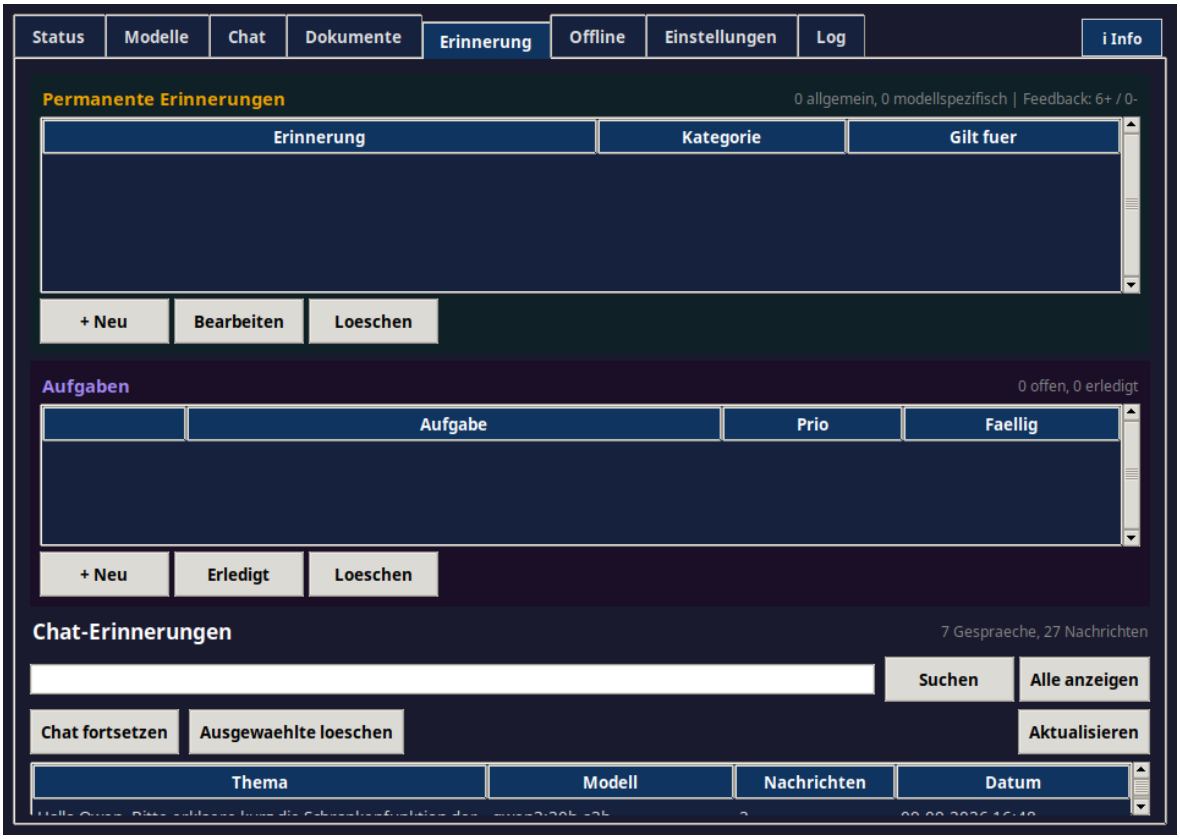
Kontext und Geduld
Bei Kolibri passen mit der Vorgabe von 16 384 Tokens rund 49 000 Zeichen; ein Amtsblatt von 30 Seiten braucht dann gut zehn Minuten Einlesen auf einem Rechner ohne Grafikkarte, bevor die Antwort beginnt. Mehr Text passt mit einem größeren Kontext unter Einstellungen → llama.cpp, je 32 768 Tokens etwa 640 MB Arbeitsspeicher mehr. Bei Ollama-Modellen rechnet DeLoMa mit Ollamas Vorgabe von 4 096 Tokens.

Suche testen

Unten im Tab können Sie eine Suche ausprobieren, ohne die KI zu bemühen. Die Treffer erscheinen im Tab **Log** mit Prozentangabe zur Relevanz. Nützlich, um zu prüfen, ob die Datenbank überhaupt das Richtige findet.

Große Bestände brauchen Zeit
Rechnen Sie mit etwa einer Sekunde pro Textabschnitt. Ein Ordner mit hundert PDF-Dateien kann eine halbe Stunde beschäftigt sein. Sie können währenddessen weiterarbeiten.

7. Erinnerungen und Aufgaben



Der Tab Erinnerung.

Dauerhafte Erinnerungen

Das sind Tatsachen über Sie, die die KI bei jeder Frage mitbekommt – unabhängig vom Gespräch. Zum Beispiel:

- Ich arbeite als Altenpfleger.
- Antworte mir immer knapp, höchstens fünf Sätze.
- Ich unterrichte angehende Pflegefachkräfte.

Anlegen über den Knopf **Merken** im Chat oder über **+ Neu** im Tab Erinnerung. Eine Erinnerung kann für alle Modelle gelten oder nur für ein bestimmtes.

> **Sparsam sein**
> Jede Erinnerung belegt Platz im Kontext, der dann für die eigentliche Frage fehlt. Fünf gute Erinnerungen sind besser als dreißig mittelmäßige.

Aufgaben

Eine einfache Aufgabenliste mit Priorität und Fälligkeitsdatum. Offene Aufgaben werden der KI als Kontext mitgegeben – Sie können also fragen: Was steht heute an?

Gesprächsverlauf

Jedes Gespräch wird automatisch gespeichert und ist durchsuchbar – auch nach Fragen mit Fragezeichen oder Sonderzeichen. Bei eingeschaltetem Haken **Erinnerung** sucht das Programm bei jeder Frage nach passenden früheren Gesprächen und gibt sie mit.

Bewertungen

Nach jeder Antwort erscheinen zwei kleine Knöpfe zum Bewerten. Die Bewertungen werden gespeichert und im Tab Erinnerung gezählt. Sie helfen Ihnen einzuschätzen, welches Modell für Ihre Art von Fragen taugt.

8. Der Agent

> **Bitte aufmerksam lesen**
> Dieses Kapitel beschreibt die mächtigste und zugleich heikelste Funktion. Bitte vollständig lesen, bevor Sie sie einsetzen.

Im Agent-Modus kann die KI nicht nur reden, sondern handeln: Dateien lesen, Ordner durchsuchen, Befehle ausführen, Webseiten abrufen, Dateien schreiben. Auch hier laufen die Antworten Wort für Wort ein – zwischen den Werkzeugaufrufen und bei der Abschlussantwort.

Zwei Betriebsarten

	Agent	Auto-Agent
Wer entscheidet?	Sie	die KI
Bedienung	Knopfleiste im Chat	Haken setzen, fertig
Geeignet für	gezielte einzelne Aktionen	mehrstufige Aufgaben
Voraussetzung	jedes Modell	werkzeugfähiges Modell

Das Ampelsystem

Jedes Werkzeug hat eine Risikostufe. Danach richtet sich, ob Sie gefragt werden:

Stufe	Bedeutung	Werkzeuge
Grün	Wird ohne Rückfrage ausgeführt. Reines Lesen.	Datei lesen, Ordner auflisten
Gelb	Wird angezeigt und protokolliert; kann pauschal freigegeben werden.	Suchen, Befehl ausführen, Code ansehen, Webseite abrufen, Web-Suche
Rot	Fragt jedes Mal nach Ihrer Zustimmung.	Datei schreiben, Datei ändern

Was der Agent nicht darf

Fest eingebaut, nicht abschaltbar:

- Zugriff nur in Ihrem persönlichen Ordner und in `/tmp`. Systemordner sind gesperrt.
- Gesperrt sind außerdem alle Pfade, die nach Zugangsdaten aussehen: `.ssh`, `.gnupg`, `.env`, `credentials`, `.password`, `.key` und ähnliche.
- Befehle nur aus einer festen Liste harmloser Kommandos (`ls`, `cat`, `df`, `free`, `uname`, `date`, `du`, `find`, `grep` und einige mehr).
- Ausdrücklich gesperrt: `sudo`, `rm`, `chmod`, `kill`, `curl`, `wget`, `ssh` und alles, was Programme startet.

- Keine Sonderzeichen in Befehlen, mit denen sich mehrere Kommandos verketten ließen.
- Zehn Sekunden Zeitlimit pro Befehl.

Ein Ablauf in der Praxis

1. Sie setzen den Haken **Auto-Agent** und wählen ein werkzeugfähiges Modell.
2. Sie fragen: Welche Dateien liegen in meinem Dokumente-Ordner?
3. Die KI ruft das Werkzeug Ordner auflisten auf – grün, also sofort.
4. Sie sehen im Chat: `[Werkzeug: list_directory Risiko: green]` und darunter das Ergebnis.
5. Die KI formuliert daraus eine Antwort.

Verlangt die Aufgabe ein rotes Werkzeug, erscheint ein Fenster mit Werkzeugname und allen Parametern. Erst nach Ihrem **Ja** passiert etwas.

Abbrechen

Der Knopf **Stopp** beendet einen laufenden Agent-Vorgang sofort – vor der nächsten Runde, vor dem nächsten Werkzeug und mitten in einer laufenden Antwort.

Welche Modelle können das?

Der Auto-Agent braucht ein Modell mit echter Werkzeug-Unterstützung. Mit einer Beispielfunktion geprüft – entscheidend ist, ob das Modell einen strukturierten Aufruf zurückgibt statt Fließtext:

Modell	Antwort auf den Werkzeugtest	Agent-tauglich
qwen3:30b-a3b	echter Werkzeugaufruf	ja
gemma4:e4b	echter Werkzeugaufruf	ja
qwen2.5:3b	echter Werkzeugaufruf	ja
gguf: Kolibri-1 IQ2_XS	echter Werkzeugaufruf, 9 von 9 (Oktober 2026)	ja
qwen2.5-coder:7b	gibt den Aufruf nur als Text aus	nein

Ein ungeeignetes Modell führt nicht zum Absturz – es beantwortet die Frage einfach normal, und das Programm sagt Ihnen, dass es keine Werkzeuge kann.

9. Fremdwerkzeuge einbinden (MCP)

MCP steht für Model Context Protocol. Es ist eine Vereinbarung, mit der fremde Programme der KI zusätzliche Werkzeuge anbieten können – etwa Zugriff auf eine Versionsverwaltung oder eine Datenbank.

> **Für den Alltag reicht der Assistent**
> Für den normalen Betrieb brauchen Sie diese Funktion nicht. Wer sie will, kommt mit dem Assistenten unten in wenigen Klicks ans Ziel; der Abschnitt „von Hand“ ist für Fortgeschrittene.

Server über HTTP

Seit 0.4 muss ein MCP-Server kein Programm auf diesem Rechner sein. Im Dialog **+ Neu** steht unter **Transport** neben stdio auch http: Dann trägt man statt eines Befehls eine **Adresse** ein (zum Beispiel `http://127.0.0.1:11482/mcp`) und bei Bedarf **Kopfzeilen** wie `Authorization: Bearer` . DeLoMa spricht das aktuelle Streamable HTTP und fällt auf das ältere HTTP+SSE zurück, wenn der Server nur das kennt. Alles, was nicht an 127.0.0.1 geht, verlässt den Rechner; DeLoMa warnt davor im Dialog, verbietet es aber nicht.

DeLoMa als MCP-Server für andere Programme

Umgekehrt stellt DeLoMa seine Fähigkeiten anderen Programmen bereit, unten im Abschnitt **MCP-Server**. Sieben Werkzeuge: Wissensdatenbanken auflisten und darin suchen (lokale Einbettungen), Projekte auflisten und lesen, Modelle auflisten, eine Frage von einem lokalen Modell beantworten lassen (wahlweise mit dem System-Prompt eines Projekts) und ein Bild erzeugen. Zwei Wege: Der Haken **HTTP-Server beim Start mitstarten** (oder **Jetzt starten**) startet ihn auf 127.0.0.1 mit dem angezeigten Port; andere Programme auf diesem Rechner tragen die Adresse und den **Schlüssel** als Kopfzeile ein. Der **stdio-Befehl** ist für Hosts, die ihre MCP-Server selbst starten, etwa Editoren oder Claude Code: Befehl kopieren, dort eintragen, fertig. Nichts davon verlässt den Rechner; andere Adressen als 127.0.0.1 nimmt der Server nur mit einem ausdrücklichen Schalter an.

> **Geprüft am 17.10.2026**
> Der eigene Server antwortete auf dem Messrechner über HTTP mit Schlüssel in Sekundenbruchteilen; eine Suche in der Wissensdatenbank „Standard“ (181 Abschnitte) dauerte 1,8 Sekunden und lieferte die passende Stelle der Arbeitszeitrichtlinie. Die Antwort- und Bildwerkzeuge dauern so lange wie der jeweilige Lauf im Programm.

Die zwei eigenen Server als Paket für andere Programme

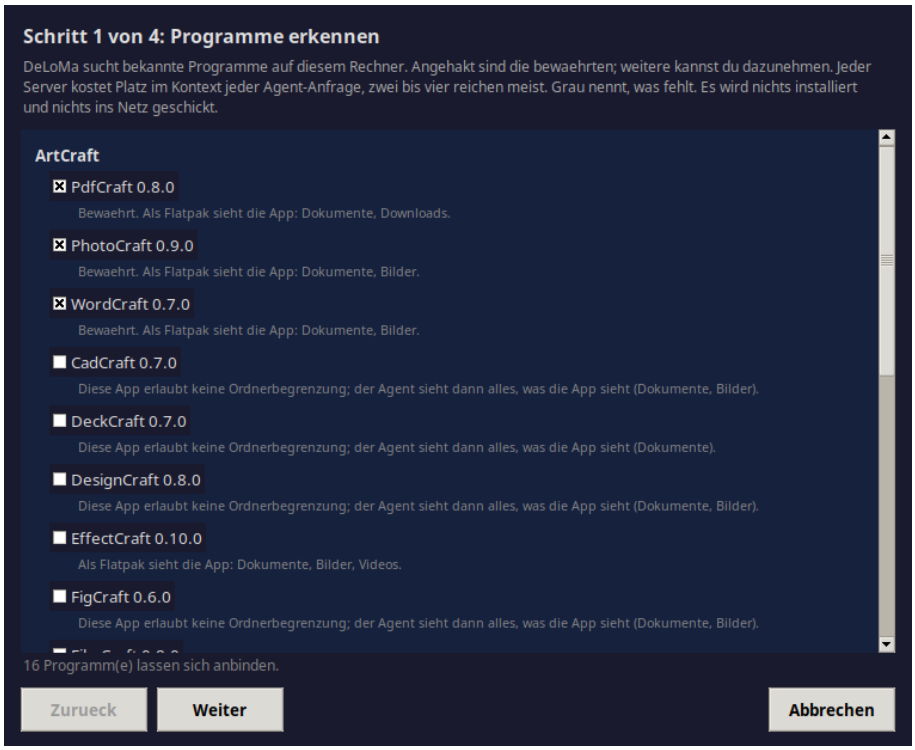
Der Browser-Server und der LibreOffice-Server hängen nicht an DeLoMa. Auf der DeLoMa-Seite liegt neben dem Programm das Zip deloma-mcp-server mit beiden Skripten, einer LIESMICH mit Voraussetzungen (Python 3.10, das Paket websockets für den Browser, python3-uno für LibreOffice), Beispielen für das Eintragen in andere MCP-Hosts und Prüfsummen. Wer DeLoMa hat, braucht es nicht: der Assistent richtet dieselben Server ein.

Programme anbinden mit dem Assistenten

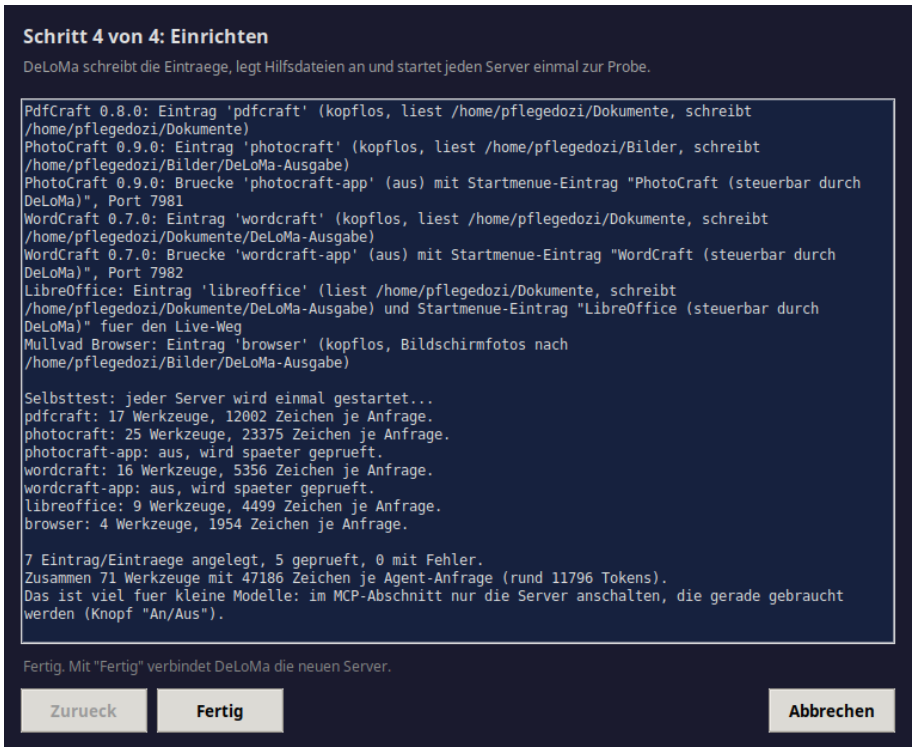
Seit 0.4 kennt DeLoMa drei Programmfamilien, die sich mit wenigen Klicks anbinden lassen:

- **Craft-Apps von ArtCraft** (PhotoCraft, WordCraft, PdfCraft und weitere; Flatpak von getartcraft.com). Jede App bringt einen eigenen MCP-Server mit, der ohne laufende App mit Dateien arbeitet.
- **LibreOffice** in der Paketfassung (mit python3-uno): Dokumente lesen, umwandeln, erstellen, Text ersetzen – und der Live-Weg in ein geöffnetes LibreOffice.
- **Browser der Firefox-Familie** (Mullvad Browser, Firefox, LibreWolf): ein eigener, kopfloser Browser mit eigenem Profil liest Seiten, sucht und macht Bildschirmfotos. Läuft ein VPN, läuft er darüber.

1. **Einstellungen → MCP-Server → Programme anbinden…** klicken.
2. **Schritt 1:** DeLoMa sucht die Programme. Angehakt sind die bewährten (PhotoCraft, WordCraft, PdfCraft, LibreOffice, der erste Browser); weitere gefundene können Sie dazunehmen. Grau steht, was fehlt und was dafür zu installieren wäre.
3. **Schritt 2:** Ordner. Gelesen wird nur in Dokumente und Bilder, geschrieben nur in den Unterordner DeLoMa-Ausgabe. Liegt ein Ordner außerhalb dessen, was eine Flatpak-App sehen darf, fragt DeLoMa nach.
4. **Schritt 3:** Brücken und Browserfenster, siehe unten.
5. **Schritt 4:** DeLoMa schreibt die Einträge, legt Hilfsdateien an und startet jeden Server einmal zur Probe. Am Ende steht, wie viele Werkzeuge und Zeichen je Anfrage dazukommen.



Schritt 1 des Assistenten: gefundene Programme, die bewährten vorangehakt, darunter was die Flatpak-App sehen darf.



Schritt 4: Einträge angelegt, Selbsttest bestanden, Summe der Werkzeuge und Zeichen.

Was die Werkzeuge je Anfrage kosten

Jede Anfrage an das Modell trägt die Beschreibungen aller freigegebenen Werkzeuge mit. Gemessen am 17.10.2026 mit den fünf Anbindungen des Messrechners (Tokens mit dem Tokenizer von gemma-3-4b gezählt; andere Modelle zerlegen etwas anders, die Größenordnung bleibt):

Server	Werkzeuge	Zeichen	Tokens
browser (DeLoMa)	4	1 859	521
libreoffice (DeLoMa)	9	4 499	1 284
wordcraft	16	5 356	1 499
pdfcraft (--compact)	17	12 002	3 226
photocraft	25	23 375	6 651
alle fünf zusammen	71	47 091	13 181
DeLoMas eigene Werkzeuge	9	2 934	850

Was das bedeutet: Mit allen fünf Servern liest ein Modell je Anfrage gut 13 000 Tokens Werkzeugtext, bevor es die eigentliche Frage sieht. Bei Kolibri mit 12 bis 15 Tokens je Sekunde sind das 15 bis 18 Minuten für die erste Antwort; danach hilft der Prompt-Zwischenspeicher. Deshalb lohnt es sich, unter **Werkzeuge** je Server nur das freizugeben, was gebraucht wird, und Server, die gerade nicht gebraucht werden, mit **An/Aus** abzuschalten. Die Summe steht unter der Serverliste.

Gemessen: ein Agent-Auftrag mit MCP-Werkzeugen

Aufgabe an den Agenten am 17.10.2026 auf dem Messrechner: „Erstelle mit den WordCraft-Werkzeugen ein neues Textdokument, schreibe den Text 'DeLoMa Messlauf 17.10.2026' hinein und speichere es als …docx.“ Drei nötige Aufrufe (neues Dokument, Text, speichern), jedes Modell einmal mit nur WordCraft (5 356 Zeichen Werkzeugtext) und einmal mit allen fünf Servern (47 091 Zeichen); alle Werkzeuge freigegeben, Frist 50 Minuten je Lauf.

Modell	Server	bis zum 1. Aufruf	gesamt	Aufrufe	Datei
Kolibri-1 (llama.cpp)	nur WordCraft	218 s	265 s	3, genau die nötigen	ja
Kolibri-1 (llama.cpp)	alle fünf	963 s	989 s	3, genau die nötigen	ja
qwen3:30b-a3b (Ollama)	nur WordCraft	539 s	748 s	6, Dokument doppelt angelegt	ja
qwen3:30b-a3b (Ollama)	alle fünf	297 s	Frist erreicht	1, über LibreOffice statt WordCraft	ja
gemma-3-4b (llama.cpp)	nur WordCraft	–	47 s	0, erklärt, es könne das nicht	nein
gemma-3-4b (llama.cpp)	alle fünf	–	18 s	0, „simuliert“ den Ablauf	nein

Was man daraus liest: Kolibri setzt den Auftrag mit genau den drei Aufrufen um; mit allen fünf Servern braucht es 16 Minuten, bis der erste Aufruf kommt, weil es die 13 000 Tokens Werkzeugtext mit 12 bis 15 Tokens je Sekunde liest, danach liegt der Text im Prompt-Zwischenspeicher und jede weitere Runde dauert nur Sekunden. qwen3:30b-a3b ist das denkende Modell: es denkt vor jeder Antwort minutenlang, das lässt sich nicht abschalten, und mit allen fünf Servern griff es zum LibreOffice-Werkzeug, das alles in einem Schritt erledigt; die Datei entstand, die zweite Runde lief aber in die Frist. gemma-3-4b kann in DeLoMa keine Werkzeuge rufen, weil seine Chat-Vorlage keine Werkzeugaufrufe kennt (der Status zeigt deshalb kein „tools“); es beschreibt nur, was es tun würde. Der Rat bleibt: wenige Server, wenige Werkzeuge, und für Werkzeugarbeit ein Modell, das sie beherrscht.

Es wird nichts installiert und nichts ins Netz geschickt. Alle Hilfsdateien liegen unter `~/.local/share/deloma/mcp/`; `uninstall.sh` räumt sie weg.

Brücken: die laufende App bedienen

Ohne Brücke arbeitet der Agent kopflos mit Dateien. Mit Brücke kann er zusätzlich die geöffnete App bedienen – in PhotoCraft Menüs aufrufen und Bildschirmfotos der Oberfläche machen, in WordCraft oder LibreOffice an der Schreibmarke Text einfügen. Dafür legt der Assistent Startmenü-Einträge „… (steuerbar durch DeLoMa)“ an und je einen zweiten Eintrag app-app, der zunächst aus ist.

1. Die App über den Startmenü-Eintrag starten – oder im MCP-Abschnitt den Eintrag wählen und **Steuerbar starten** klicken, dann schaltet DeLoMa gleich um.
2. Danach ist der Eintrag mit Brücke an und der kopflose aus. Zurück geht es mit **An/Aus**.

> **Was eine Brücke öffnet**
> Der Steuerkanal einer App horcht nur auf diesem Rechner (127.0.0.1), bei LibreOffice ohne Passwort. Flatpak-Apps bekommen bei einem Start mit Steuerkanal Netzzugang für diesen Lauf, weil die Brücke sonst am getrennten Netz des Sandkastens scheitert. Wer das nicht will, lässt die Option in Schritt 3 aus.

Der Browser von DeLoMa

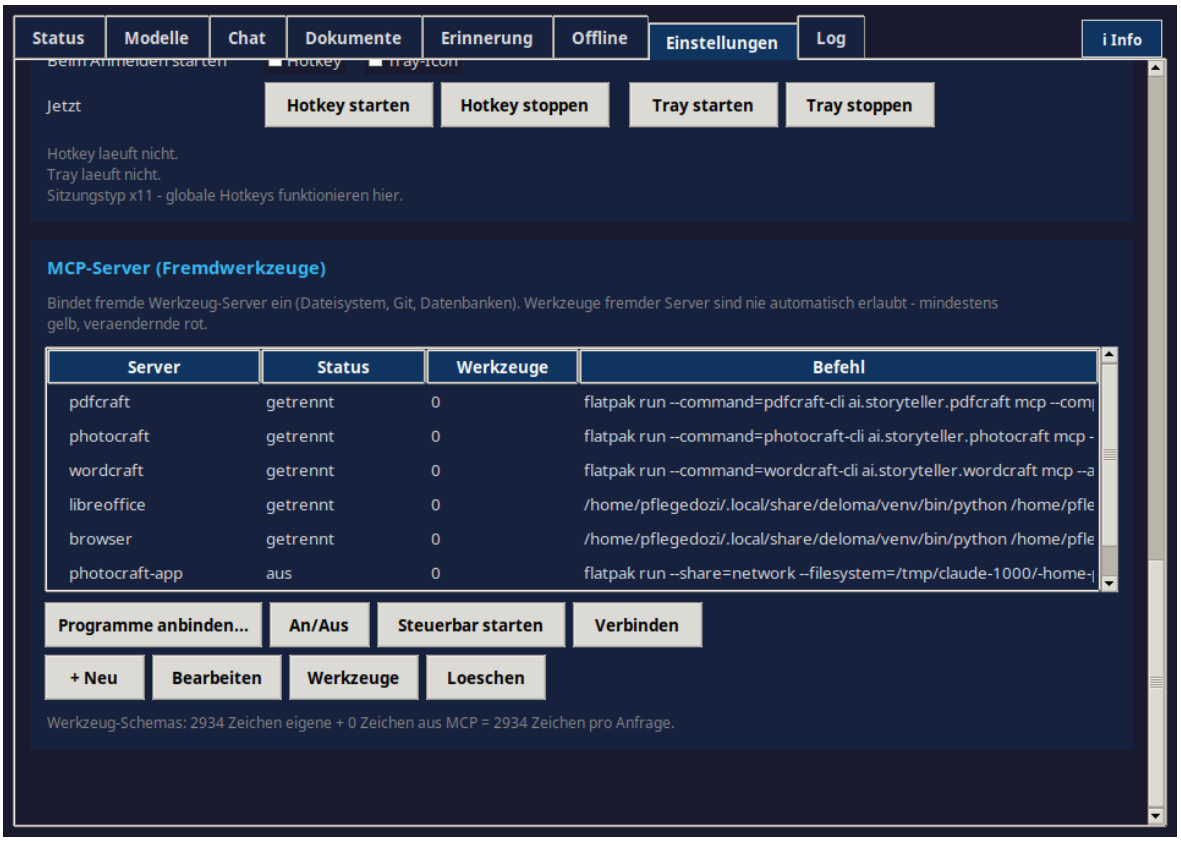
Der Browser-Server startet beim ersten Aufruf einen eigenen Browser mit eigenem Profil – nie Ihr Alltagsprofil – und beendet ihn mit DeLoMa. Nur http- und https-Adressen, keine lokalen Dateien. Unter Fernsteuerung meldet ein Browser navigator.webdriver; Webseiten können das sehen. Bildschirmfotos landen im Ausgabeordner unter Bilder.

Viele Server, wenig Kontext

Jeder angeschaltete Server schickt seine Werkzeuge bei jeder Agent-Anfrage mit. Gemessen am 16.10.2026 auf dem Rechner des Autors:

Server	Werkzeuge	Zeichen pro Anfrage
Browser	4	1.954
LibreOffice	9	4.499
WordCraft	16	5.356
PdfCraft (Kernsatz)	17	12.002
PhotoCraft	25	23.375
Alle fünf zusammen	71	47.186

Mit Kolibri und 16.384 Kontext wären das fast drei Viertel. Schalten Sie im MCP-Abschnitt mit **An/Aus** nur an, was Sie gerade brauchen. Die Werkzeuge gelten ohnehin nur im Agent-Modus, der Chat bleibt unberührt.



Der MCP-Abschnitt in den Einstellungen: oben der Weg für alle (Assistent, An/Aus, Steuerbar starten, Verbinden), unten die Handarbeit.

> **Was der Assistent noch nicht kann**
> LibreOffice als Snap oder Flatpak erkennt er, kann es aber nicht anbinden; er nennt die Paketfassung mit python3-uno. Firefox als Snap (Ubuntu) ist ungeprüft. Craft-Apps ohne Ordnerbegrenzung sehen alles, was ihr Sandkasten freigibt – der Assistent nennt es.

Einen Server von Hand einrichten

1 **Einstellungen → MCP-Server** öffnen.

2 Auf **+ Neu** klicken.

3 Name frei wählen, Befehl und Argumente eintragen.

4 Auf **Speichern und verbinden** klicken.

Beispiel für einen Dateisystem-Server:

Feld	Wert
Name	dateien
Befehl	npx
Argumente	-y @modelcontextprotocol/server-filesystem /home/benutzer/Dokumente

Sicherheit bei Fremdwerkzeugen

Hier läuft fremder Programmcode. Deshalb strenger als bei den eigenen Werkzeugen:

- Kein MCP-Werkzeug ist jemals grün. Auch reines Lesen wird angezeigt.
- Standardstufe ist gelb.
- Rot, sobald der Server ein Werkzeug als verändernd meldet.
- Ein Werkzeug ohne laufenden Server gilt als rot.

Warum Sie Werkzeuge auswählen sollten

Jedes angebotene Werkzeug wird bei jeder Frage mitgeschickt – auch wenn es gar nicht gebraucht wird. Das kostet Platz im Kontext:

	Zeichen pro Frage
Die eingebauten neun Werkzeuge	2.934
Ein Dateisystem-Server, alle 14 Werkzeuge	9.055
Derselbe Server, zwei Werkzeuge freigegeben	1.251

Zum Vergleich: Ein kleines Modell wie `qwen2.5:3b` hat insgesamt etwa 3.000 Zeichen Kontext zur Verfügung. Ein einziger Server ohne Auswahl erstickt so ein Modell.

> **Nur freigeben, was Sie brauchen**
> Nutzen Sie den Knopf **Werkzeuge** und geben Sie nur frei, was Sie wirklich brauchen. Die Einstellungen zeigen Ihnen die Summe an und warnen ab 4.000 Zeichen.

10. Sprache: Vorlesen und Diktieren

Vorlesen

Der Knopf **Vorlesen** im Chat liest die letzte Antwort laut vor. Nützlich, wenn die Hände nicht frei sind oder das Lesen am Bildschirm anstrengt. Dafür braucht es zwei Dinge:

- das Programm `piper` (wird mitinstalliert),
- eine Stimmdatei in `~/.local/share/deloma/voices/`.

Eine deutsche Stimme holen Sie sich so:

```
mkdir -p ~/.local/share/deloma/voices
cd ~/.local/share/deloma/voices
B=https://huggingface.co/rhasspy/piper-voices/resolve/main/de/de_DE/thorsten/medium
curl -fLO $B/de_DE-thorsten-medium.onnx
curl -fLO $B/de_DE-thorsten-medium.onnx.json
```

> **Eine englische Stimme kann keine deutschen Laute bilden**
> Umlaute gehen verloren, das Ergebnis klingt falsch. Achten Sie darauf, dass der Dateiname mit `de_` beginnt. Beide Dateien – `.onnx` und `.onnx.json` – gehören zusammen.

Stimme und Sprechtempo stellen Sie unter **Einstellungen → Sprachausgabe** ein. Der Knopf **Stimme testen** spricht einen Beispielsatz.

Diktieren

Der Knopf **Mikro** nimmt auf und schreibt das Gesprochene in das Eingabefeld. Die Erkennung läuft lokal – nichts wird verschickt.

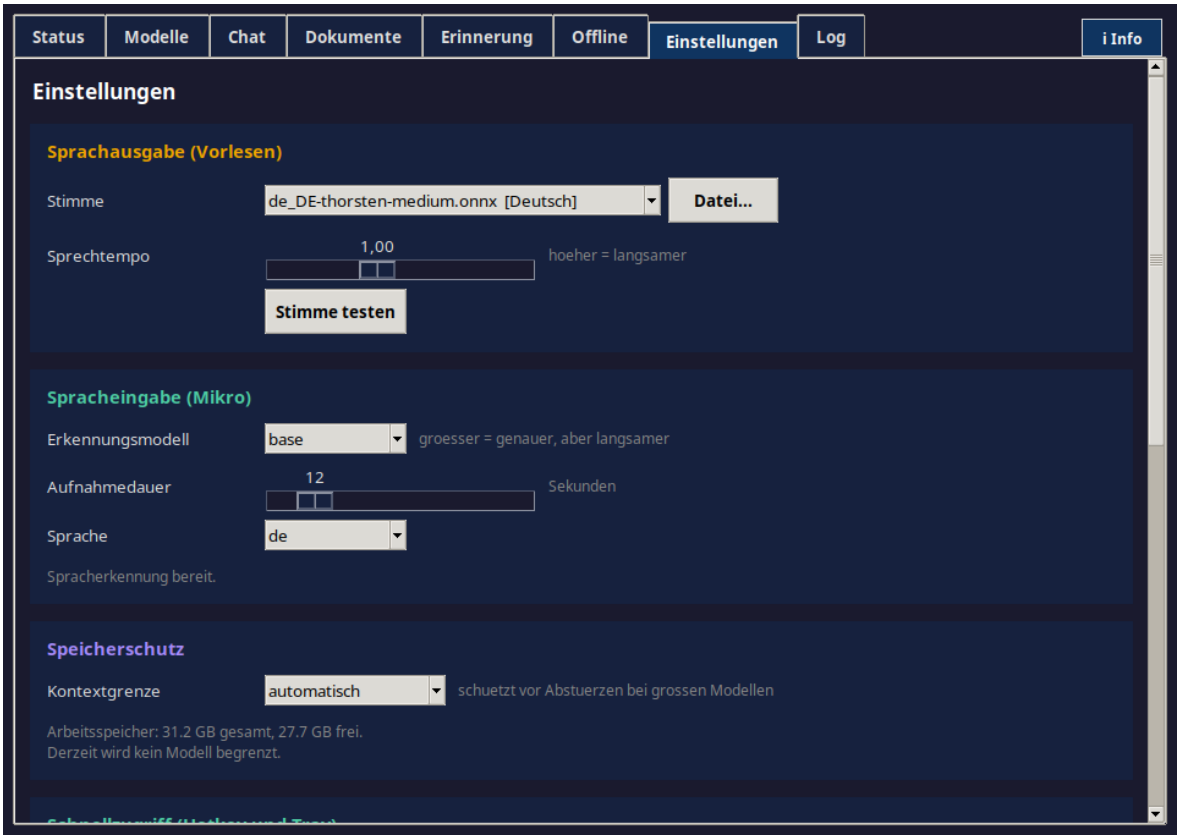
1. Auf **Mikro** klicken.
2. Ein Fenster zeigt einen Countdown. Sprechen Sie.
3. Nach Ablauf wechselt die Anzeige auf Verarbeite Aufnahme.
4. Der erkannte Text erscheint im Eingabefeld – Sie können ihn vor dem Absenden noch ändern.

> **Der erste Klick dauert länger**
> Beim ersten Mal nach dem Programmstart dauert es etwa zehn Sekunden länger: Das Erkennungsmodell wird in den Speicher geladen. Beim allerersten Mal überhaupt wird es zuvor einmalig heruntergeladen (rund 140 MB). Das Fenster sagt Ihnen das.

Einstellbar unter **Einstellungen → Spracheingabe**:

Einstellung	Bedeutung
Erkennungsmodell	tiny schnell und ungenau · base guter Kompromiss · small/medium genauer, langsamer
Aufnahmedauer	5 bis 60 Sekunden
Sprache	Voreingestellt Deutsch

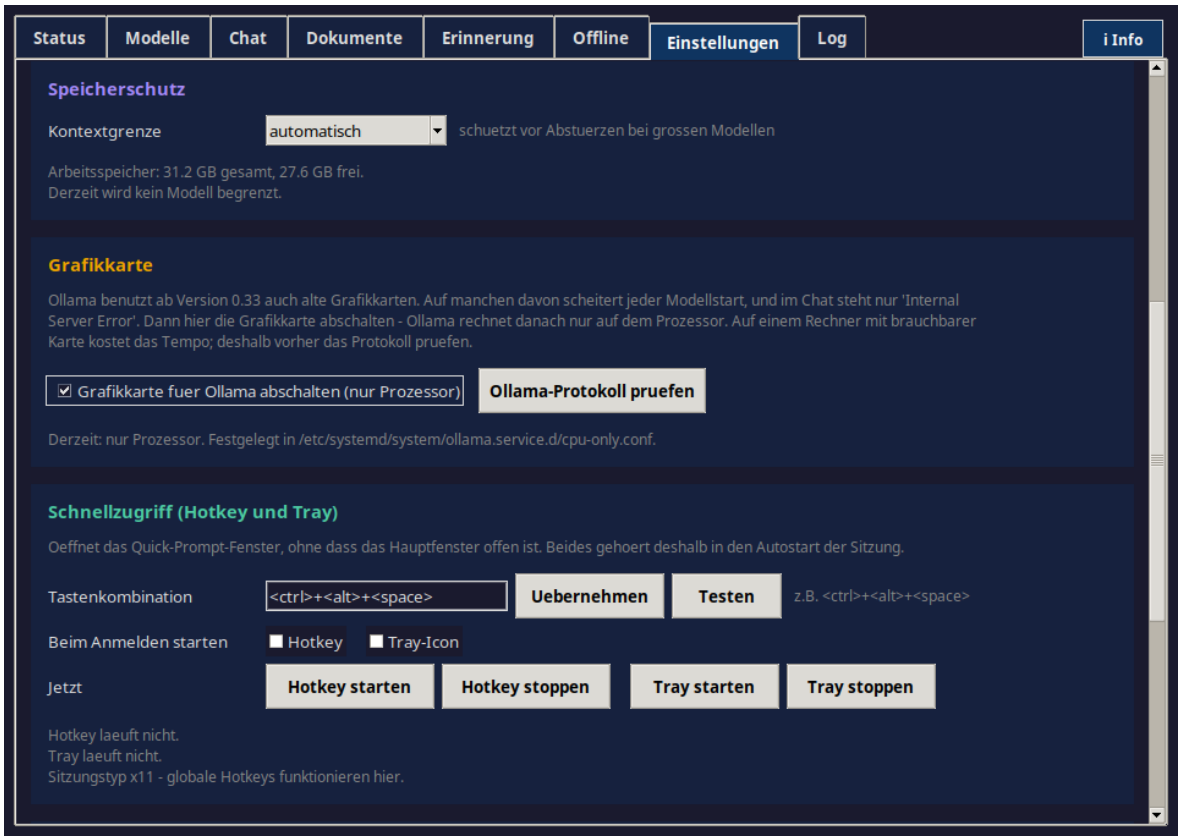
11. Einstellungen



Sprachausgabe, Spracheingabe und Speicherschutz.

Weiter unten auf derselben Seite: llama.cpp, Grafikkarte, Schnellzugriff (Tastenkombination, Tray-Symbol, Autostart) und die MCP-Server.

Grafikkarte



Der Abschnitt Grafikkarte in den Einstellungen.

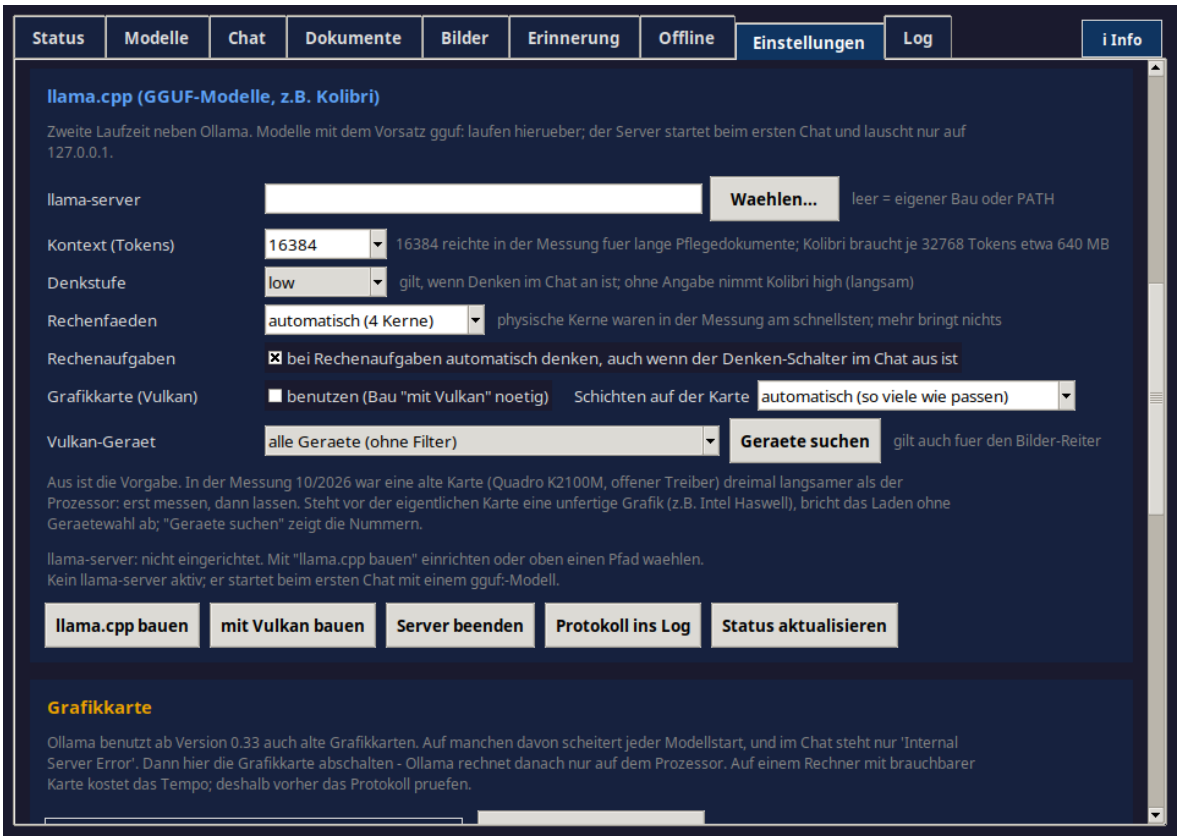
Ollama benutzt ab Version 0.33 auch ältere Grafikkarten. Auf manchen Rechnern scheitert dann jeder Modellstart mit does not support 16-bit storage, und im Chat steht nur Internal Server Error. Auf dem Messrechner kam der Fehler nicht von der NVIDIA-Karte, sondern von der unfertigen Prozessorgrafik (Intel Haswell), die als erstes Vulkan-Gerät vor der Karte steht. Der Haken **Grafikkarte für Ollama abschalten** zwingt Ollama auf den Prozessor. Dazu wird eine Datei unter `/etc/systemd/system/ollama.service.d/` angelegt und der Ollama-Dienst neu gestartet; dafür fragt das System einmal nach Ihrem Passwort.

Der Knopf **Ollama-Protokoll prüfen** liest das Protokoll des Ollama-Dienstes und sagt Ihnen, ob die Grafikkarte Ärger macht, ob sie sauber arbeitet oder ob Ollama ohnehin auf dem Prozessor rechnet. Prüfen Sie das, bevor Sie den Haken setzen.

> **Nur bei Bedarf**
> Auf einem Rechner mit brauchbarer Grafikkarte kostet der Haken Tempo, weil die Beschleunigung wegfällt. Setzen Sie ihn nur, wenn das Protokoll Fehler zeigt oder jedes Modell mit Internal Server Error scheitert. Der Haken lässt sich jederzeit wieder entfernen; auch das startet Ollama neu.

> **Von Hand angelegte Einstellung**
> Haben Sie die Grafikkarte früher selbst per Datei abgeschaltet, erkennt das Programm das und zeigt den Haken gesetzt. Beim Entfernen werden aus fremden Dateien nur die betreffenden Zeilen gelöscht; andere Einstellungen darin bleiben erhalten.

llama.cpp



Der Abschnitt llama.cpp in den Einstellungen.

Hier richten Sie die zweite Laufzeit für GGUF-Modelle ein (Kapitel 3). **llama.cpp bauen** holt den Quelltext von llama.cpp in einem festgelegten Stand, spielt den Kolibri-Patch ein und übersetzt das Programm für Ihren Prozessor. Das Ergebnis landet unter `~/.local/share/deloma/llama.cpp/`, die Ausgabe läuft im Tab **Log**. Nötig sind git, ein C++-Compiler und cmake; fehlt cmake, holt das Skript es ohne Administratorrechte nach. Auf dem Messrechner von 2013 dauerte der Bau 216 Sekunden.

Die Felder darüber:

- **llama-server**: leer lassen, dann nimmt DeLoMa den eigenen Bau. Wer llama.cpp schon anderswo hat, trägt hier den Pfad ein.
- **Kontext**: wie viele Tokens der Server auf einmal verarbeitet. 16 384 reichte in der Messung für lange Pflegedokumente; Kolibri braucht je 32 768 Tokens etwa 640 MB zusätzlich.
- **Denkstufe**: low, medium oder high. Gilt nur, wenn Sie im Chat das Denken einschalten.
- **Rechenfäden**: automatisch heißt ein Faden je physischem Kern. Mehr brachte in der Messung nichts.
- **Rechenaufgaben**: Haken gesetzt heißt, das Modell denkt bei Rechenaufgaben auch dann, wenn der Denken-Schalter im Chat aus ist. Gilt nur für gguf:-Modelle.

- **Grafikkarte (Vulkan)**: aus ist die Vorgabe, dann rechnet llama.cpp nur auf dem Prozessor, auch wenn der Bau Vulkan enthält. Mit Haken legt llama.cpp so viele Schichten des Modells auf die Karte, wie in ihren Speicher passen; **Schichten auf der Karte** legt die Zahl fest. Dafür braucht es einen Bau über den Knopf **mit Vulkan bauen** (siehe unten).
- **Vulkan-Gerät**: **Geräte suchen** fragt den gebauten llama-server nach seinen Karten und zeigt sie mit Nummer und Speicher. Steht vor der eigentlichen Karte eine unfertige Grafik (etwa die Prozessorgrafik eines Haswell-Rechners), bricht das Laden ohne Wahl mit does not support 16-bit storage ab; dann hier die richtige Nummer wählen. Die Nummer gilt auch für den Tab Bilder.

mit Vulkan bauen baut dieselbe Fassung mit dem Vulkan-Backend von ggml, das jede Grafikkarte mit Vulkan-Treiber anspricht (NVIDIA, AMD, Intel, auch die offenen Treiber). Nötig sind zusätzlich der Shader-Compiler glslc, die Vulkan-Header und die SPIR-V-Header; unter Ubuntu und Debian sind das die Pakete `glslc libvulkan-dev spirv-headers`, zusammen rund 1,5 MB. Fehlt etwas, sagt das Skript, was. Ein Bau ohne den Knopf setzt einen früheren Vulkan-Bau wieder auf Prozessor zurück.

> **Gemessen auf einer alten Karte**
> Auf dem Messrechner von 2013 (Quadro K2100M mit 2 GB, offener Treiber NVK) lief der Vulkan-Bau am 17.10.2026 fehlerfrei, war aber langsamer als der Prozessor allein: qwen2.5-coder 7B mit acht von 29 Schichten auf der Karte 1,6 statt 4,7 Tokens je Sekunde, mit automatischer Schichtwahl (zwei Schichten) 2,7. Die Karte rechnet nur in 32 Bit und hat wenig Speicher. Eine Karte lohnt sich erst, wenn das ganze Modell hineinpasst oder sie 16-Bit-Rechnen beherrscht; deshalb bleibt der Haken aus, bis Sie selbst gemessen haben. Der Tab Log zeigt beim Start, wie viele Schichten auf der Karte liegen.

Geänderte Werte gelten ab dem nächsten Start des Servers; **Server beenden** erzwingt ihn. **Protokoll ins Log** zeigt die letzten Zeilen des Serverprotokolls, wenn etwas nicht klappt. Der Server lauscht nur auf dem eigenen Rechner (127.0.0.1) und verlangt einen Schlüssel, den DeLoMa bei jedem Start neu auswürfelt.

> **Woher kommt der Patch?**
> llama.cpp kannte die Bauart von Kolibri-1 im Oktober 2026 noch nicht. DeLoMa bringt deshalb einen Patch aus der Community mit (Hob-forge/Kolibri-1-GGUF, MIT-Lizenz) und prüft vor dem Einspielen seine Prüfsumme. Herkunft und Inhalt stehen in der Datei `kolibri1-llama.cpp.patch.LIESMICH.txt` im Programmordner. Der Patch wurde vor der Aufnahme vollständig gelesen; er enthält keine Datei-, Netz- oder Prozesszugriffe.

Alle Änderungen werden sofort gespeichert. Es gibt bewusst keinen Speichern-Knopf, den man vergessen kann. Die Einstellungen liegen in einer Datei:

```
~/.local/share/deloma/data/settings.json
```

12. Schnellzugriff

Beide Funktionen dieses Kapitels öffnen das Quick-Prompt – ein kleines Fenster für eine schnelle Frage, ohne das ganze Programm zu starten.

Tastenkombination

Voreingestellt ist **Strg + Alt + Leertaste**. Änderbar unter **Einstellungen → Schnellzugriff**. Schreibweise: Sondertasten in spitze Klammern, verbunden mit Pluszeichen.

```
Richtig: <ctrl>+<alt>+<space>
Richtig: <ctrl>+<alt>+k
Falsch:  <ctrl>+<alt>+space
```

> **Der Knopf Testen**
> Er wartet zwanzig Sekunden auf Ihren Tastendruck und sagt Ihnen, ob er angekommen ist. Nutzen Sie ihn – manche Kombinationen fängt die Arbeitsumgebung selbst ab, bevor sie beim Programm ankommen.

Symbol in der Taskleiste

Das Tray-Symbol sitzt unten rechts in der Taskleiste, neben Uhr und Lautstärke – nicht im Programmfenster. Ein Rechtsklick öffnet ein Menü mit **Quick-Prompt**, **DeLoMa öffnen** und **Beenden**.

> **Häufige Verwechslung**
> Ist das Programmfenster maximiert, verdeckt es die Taskleiste – das Symbol scheint dann zu fehlen. Fenster verkleinern, dann ist es wieder da.

Automatisch starten

Die beiden Haken bei **Beim Anmelden starten** sorgen dafür, dass Tastenkombination und Symbol nach jeder Anmeldung bereitstehen – auch ohne geöffnetes Hauptfenster. Genau dafür sind sie gedacht.

13. Datenschutz und Sicherheit

Das Programm ist von Grund auf so gebaut, dass Ihre Daten den Rechner nicht verlassen. Dieses Kapitel erklärt, wie das durchgesetzt wird.

Die Netzwerksperre

Beim Start wird eine Sperre aktiviert, die alle Verbindungen ins Internet blockiert. Erlaubt sind nur Verbindungen zum eigenen Rechner – dort läuft die KI. Ausnahmen gibt es nur an drei Stellen, und alle drei stoßen Sie bewusst an:

- Herunterladen eines KI-Modells,
- das Werkzeug Web-Suche im Agent-Modus,
- das Werkzeug Webseite abrufen im Agent-Modus.

Keine Nutzungsstatistik

Eine der verwendeten Zusatzbibliotheken sendet normalerweise Nutzungsdaten an ihren Hersteller. Das Programm ersetzt diese Funktion beim Start durch eine wirkungslose Attrappe. Es wird nichts gesendet.

Selbst nachprüfen

Sie müssen das nicht glauben. Während Sie chatten, im Terminal:

```
ss -tnp | grep -E 'ollama|python'
```

Es dürfen nur Verbindungen zu `127.0.0.1` erscheinen – das ist Ihr eigener Rechner. Externe Adressen dürfen nicht auftauchen.

Wo liegen Ihre Daten?

Was	Wo	Inhalt
Gespräche	data/memory.db	alle geführten Chats
Erinnerungen	data/persistent_memory.db	Erinnerungen, Aufgaben, Bewertungen
Projekte	data/projects.db	Ihre Projekteinstellungen
Wissensdatenbank	data/chromadb/	Ihre Dokumente
Einstellungen	data/settings.json	Konfiguration
Stimmen	voices/	Stimmdateien für das Vorlesen
KI-Modelle	~/.ollama/models/ oder /usr/share/ollama/.ollama/models/	die Modelle selbst

Alles unterhalb von `~/.local/share/deloma/`, sofern nicht anders angegeben.

Sicherung

Seit Version 0.4 sichert DeLoMa Ihre Programmdaten von selbst: einmal am Tag beim Start kopiert es Gespräche, Erinnerungen, dauerhafte Erinnerungen, Einstellungen, Projekte und die Dokumentliste nach `~/.local/share/deloma/sicherung//`. Die letzten sieben Tagessicherungen bleiben, ältere räumt das Programm weg. Nicht enthalten ist die Wissensdatenbank (`data/chromadb`): Sie kann groß werden und lässt sich aus den Dokumenten neu aufbauen (Kapitel 6, Änderungen nachziehen).

Im Tab **Offline** unter **Sicherung der Programmdaten**:

- **Jetzt sichern** legt zusätzlich eine Sicherung mit Uhrzeit an. Die bleibt, bis Sie sie löschen.
- **Zurückspielen…** zeigt alle Sicherungen. Die gewählte wird beim nächsten Start eingespielt, bevor das Programm seine Datenbanken öffnet – deshalb danach DeLoMa schließen und neu starten. Der Stand von vorher wird als „vor-zurueckspielen-…“ aufgehoben, auch dieser Schritt lässt sich also rückgängig machen.
- **In Ordner kopieren…** schreibt eine frische Sicherung in einen Ordner Ihrer Wahl, etwa auf einen USB-Stick.
- **Aus Ordner übernehmen…** holt eine solche Kopie zurück und bietet an, sie beim nächsten Start einzuspielen.
- Der Haken **Täglich beim Start sichern** schaltet die Automatik ab.

Von Hand geht es weiterhin: Programm beenden und den Datenordner kopieren:

```
cp -r ~/.local/share/deloma/data ~/sicherung-$(date +%F)
```

> **Modelle nicht sichern**
> Die KI-Modelle brauchen Sie nicht zu sichern – die sind jederzeit neu ladbar. Der Datenordner dagegen enthält alles, was unersetzlich ist.

14. Bilder erzeugen

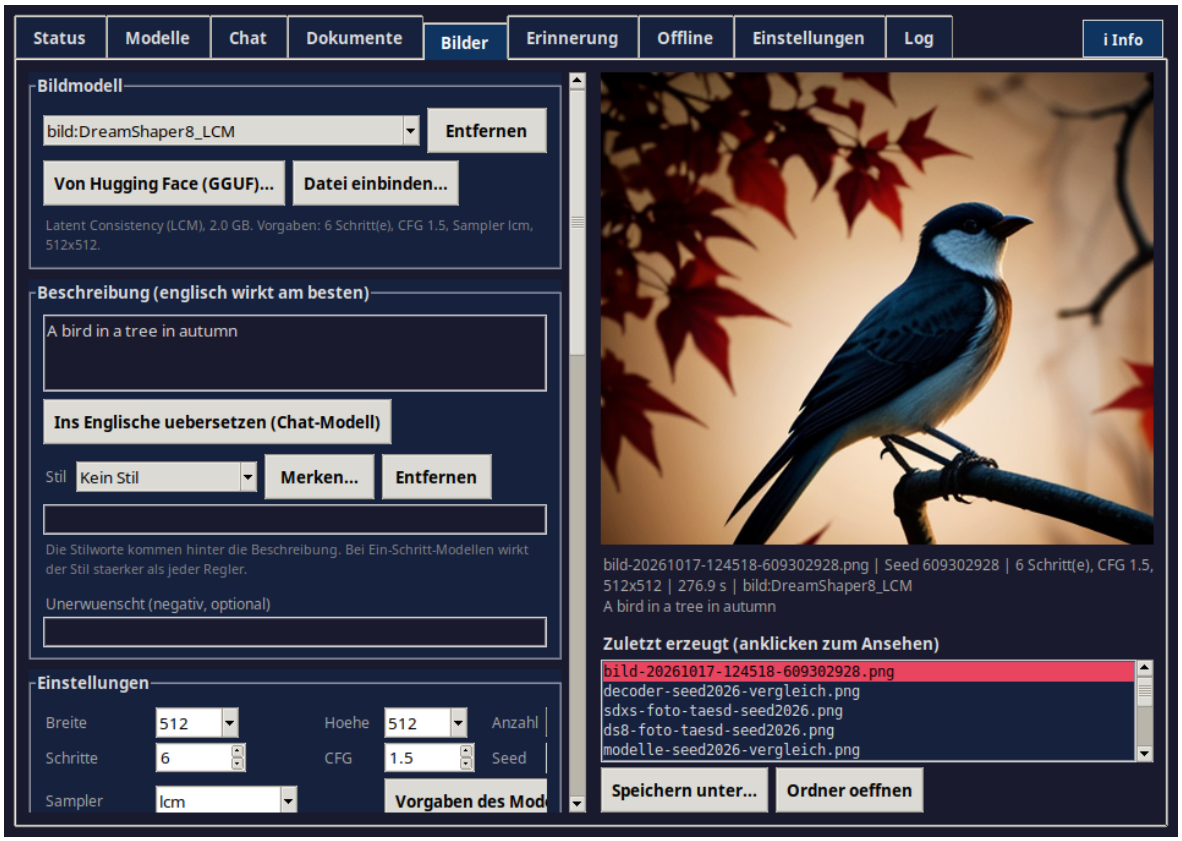
Seit 0.4 erzeugt DeLoMa auch Bilder, und zwar wie alles andere auf Ihrem Rechner: Die dritte Laufzeit neben Ollama und llama.cpp ist stable-diffusion.cpp (leejet, MIT-Lizenz), ein Server, der nur auf 127.0.0.1 horcht und mit DeLoMa endet. Es braucht keine Grafikkarte, nur Arbeitsspeicher und Geduld; die Zeiten unten sind auf einem zehn Jahre alten Prozessor gemessen. Wer eine Karte mit Vulkan-Treiber hat, kann sie unter **Einrichtung** dazuschalten (siehe Ende des Kapitels).

Einrichten in drei Schritten

1 Tab **Bilder**, unten unter **Einrichtung** auf **Programm holen** klicken. DeLoMa lädt das fertige Linux-Binary (26 MB) von der Release-Seite des Projekts, prüft die in DeLoMa festgehaltene Prüfsumme und legt es unter `~/.local/share/deloma/sd/` ab. Das ist die einzige Verbindung, die dieser Tab aufbaut.

2 Ein Bildmodell laden: **Von Hugging Face (GGUF)…** öffnet den bekannten Dialog mit Zielordner `~/.local/share/deloma/bildmodelle/` und Suchvorgabe sdxs-512. Dieses Modell ist 0,7 GB groß, rechnet in einem einzigen Schritt und taugt zum Anfangen. Vorhandene Dateien (.gguf, .safetensors, .ckpt) bindet **Datei einbinden…** ein.

3 Beschreibung eingeben, **Bild erzeugen**. Während des Rechnens zeigt der Tab Zwischenbilder, **Abbrechen** bricht den Auftrag ab. Fertige Bilder liegen im Ausgabeordner (Vorgabe `~/Bilder/DeLoMa-Bilder/`) und in der Liste Zuletzt erzeugt. **Speichern unter…** kopiert das angezeigte Bild samt Zettel an einen Ort nach Wahl, etwa in einen Projektordner oder für ein anderes Programm; **Ordner öffnen** zeigt den Ausgabeordner im Dateimanager.



Der Tab Bilder nach dem ersten Bild: links Modell, Beschreibung und Einstellungen, rechts das Bild mit seinem Zettel und die Liste der letzten Bilder.

Die Felder

- **Beschreibung:** Englisch wirkt bei den meisten Modellen am besten. **Ins Englische übersetzen** lässt das im Chat gewählte Sprachmodell übersetzen; Strg+Z holt den deutschen Text zurück.
- **Stil:** eine Stilvorlage, deren englische Stilworte hinter die Beschreibung kommen. Mitgeliefert sind Foto, Illustration, Aquarell, Linienzeichnung, Holzschnitt, Pastell und Pixelgrafik; die Worte lassen sich ändern, **Merken…** speichert sie unter eigenem Namen, **Entfernen** löscht die Vorlage. Stil und Worte stehen im Zettel.
- **Unerwünscht:** was nicht ins Bild soll (negativer Prompt). Bei Ein-Schritt-Modellen wirkungslos.
- **Breite, Höhe:** Vielfache von 64. Größere Bilder brauchen mehr Arbeitsspeicher und Zeit; über 640 Pixel schaltet DeLoMa die Kachelung des Decoders ein.
- **Schritte, CFG, Sampler:** Das Profil des Modells setzt sie; **Vorgaben des Modells** stellt sie wieder her. CFG ist, wie streng das Bild der Beschreibung folgt.
- **Seed:** -1 heißt Zufall. Der benutzte Wert steht im Zettel; derselbe Seed mit denselben Einstellungen ergibt dasselbe Bild.
- **Anzahl:** bis vier Bilder in einem Auftrag, jedes mit eigenem Seed-Ableger.
- **Bild zu Bild:** eine Vorlage (PNG, JPG) und die Stärke: 0,1 bleibt nah an der Vorlage, 1,0 ignoriert sie. So wird aus einer Skizze ein ausgemaltes Bild.

Der Stil entscheidet

Ein Ein-Schritt-Modell rät bei allem, was es nicht sicher weiß. Bei fotorealistischen Bildern fällt jedes Raten auf, weil wir Gesichter, Hände und Schrift genau kennen. Stile, die Details ohnehin weglassen, verzeihen das: flache Illustration, Aquarell, Linienzeichnung, Holzschnitt, Pixelgrafik. Ein Stilwort ändert mehr als jeder Regler; dafür gibt es die Stilvorlagen. Das Bild unten zeigt dieselbe Beschreibung mit demselben Seed 2026 in vier Stilen; nur die Stilangabe hinter dem Komma war anders, jedes Bild hat 11 Sekunden gebraucht.



Derselbe Seed, vier Stile: „a nurse at a desk writing notes, warm lamp light“ plus „photo, 35mm“, „flat vector illustration“, „watercolor painting“, „ink line drawing“. Modell sdxs-512, ein Schritt.

Bei sdxs ist die Beschreibung der einzige Hebel: Der negative Prompt wirkt nicht, weil CFG 1 ihn ausschaltet, und mehr Schritte verbessern ein Ein-Schritt-Modell nicht. Für sauberere Fotos braucht es ein Modell mit vielen Schritten, etwa Stable Diffusion 1.5, und damit rund zwei Minuten je Bild auf diesem Rechner.

Profile je Modellfamilie

DeLoMa rät die Familie aus dem Dateinamen und setzt die Vorgaben; ein Feld profil im Herkunftszettel überstimmt das Raten.

Familie (Erkennung)	Schritte	CFG	Hinweis
SDXS (sdxs)	1	1	Pflicht: genau ein Schritt, CFG 1
LCM (lcm)	6	1,5	Sampler und Scheduler lcm
Turbo/Lightning	4	1	
FLUX (flux)	4	1	768 × 768; Modelle mit getrenntem Textencoder kann der Tab noch nicht laden
SDXL (sdxl, xl)	25	7	1024 × 1024, sehr langsam ohne Grafikkarte
SD 1.x/2.x (alles andere)	20	7	512 × 512, nur Englisch, keine Schrift im Bild

Gemessen am 16.10.2026

Fujitsu Celsius H730, i7-4900MQ (4 Kerne), 31 GB RAM, keine Grafikkarte, stable-diffusion.cpp Stand 8cd731e, Modell sdxs-512 Q8_0 (0,7 GB), 512 × 512:

Was	Dauer
Modell laden (Serverstart)	0,5 s
Ein Bild, ein Schritt, 4 Rechenfäden	11,0 s
Ein Bild, ein Schritt, 8 Rechenfäden	10,1 s
Zwei Bilder in einem Auftrag	20,1 s
Ein Bild mit vier Schritten	24 s (rund 5 s je Schritt)
DreamShaper 8 LCM (2,1 GB), Modell laden	0,5 s
DreamShaper 8 LCM, ein Bild mit sechs Schritten	194 bis 197 s (rund 24 s je Schritt, dazu rund 50 s für den Decoder)
DreamShaper 8 LCM, dasselbe mit schnellem Decoder	160 s
sdxs-512 mit schnellem Decoder	12 s (kein Gewinn, der eigene Decoder ist schon klein)
Stable Diffusion 1.5 (Q8_0-GGUF, 1,8 GB), ein Bild mit 20 Schritten	505 s
Stable Diffusion 1.5 mit LCM-LoRA, 4 Schritte, CFG 1	107 s (gleiches Motiv, brauchbar)
Stable Diffusion 1.5 mit LCM-LoRA, 6 Schritte	136 s
dasselbe mit ControlNet Scribble (Skizze als Steuerbild)	188 s (19 s je Schritt, Decoder 49 s)
FLUX.2-klein-4B (Q4_0 2,5 GB + Qwen3-4B 2,5 GB + VAE), Modell laden	1 s (die Gewichte kommen beim ersten Bild, 4,9 GB RAM)
FLUX.2-klein-4B, ein Bild 768 × 768 mit vier Schritten	1173 s: Textencoder 24 s, Sampling 857 s (214 s je Schritt), Decoder 291 s

Stable Diffusion 1.5 mit 20 Schritten braucht auf diesem Rechner gut acht Minuten je Bild, mit der LCM-LoRA und vier Schritten unter zwei Minuten bei vergleichbarem Ergebnis; alle Läufe vom 17.10.2026 mit Seed 2026 und vier Rechenfäden. Hyperthreading bringt kaum etwas, die Vorgabe bleibt die Zahl der Kerne.

Ein größeres Modell: DreamShaper 8 LCM

Die nächste Stufe über sdxs sind Modelle der Klasse Stable Diffusion 1.5 mit vollem Rechenkern. DreamShaper 8 LCM (Lykon, CreativeML OpenRAIL-M, 2,1 GB, eine Datei) hat die LCM-Abkürzung eingebaut und kommt mit sechs Schritten aus; DeLoMa erkennt das am Dateinamen und setzt Schritte 6, CFG 1,5 und Sampler lcm. Der Preis auf diesem Rechner: gut drei Minuten je Bild statt elf Sekunden. Der Gewinn ist unten zu sehen, dieselbe Beschreibung und derselbe Seed mit beiden Modellen.

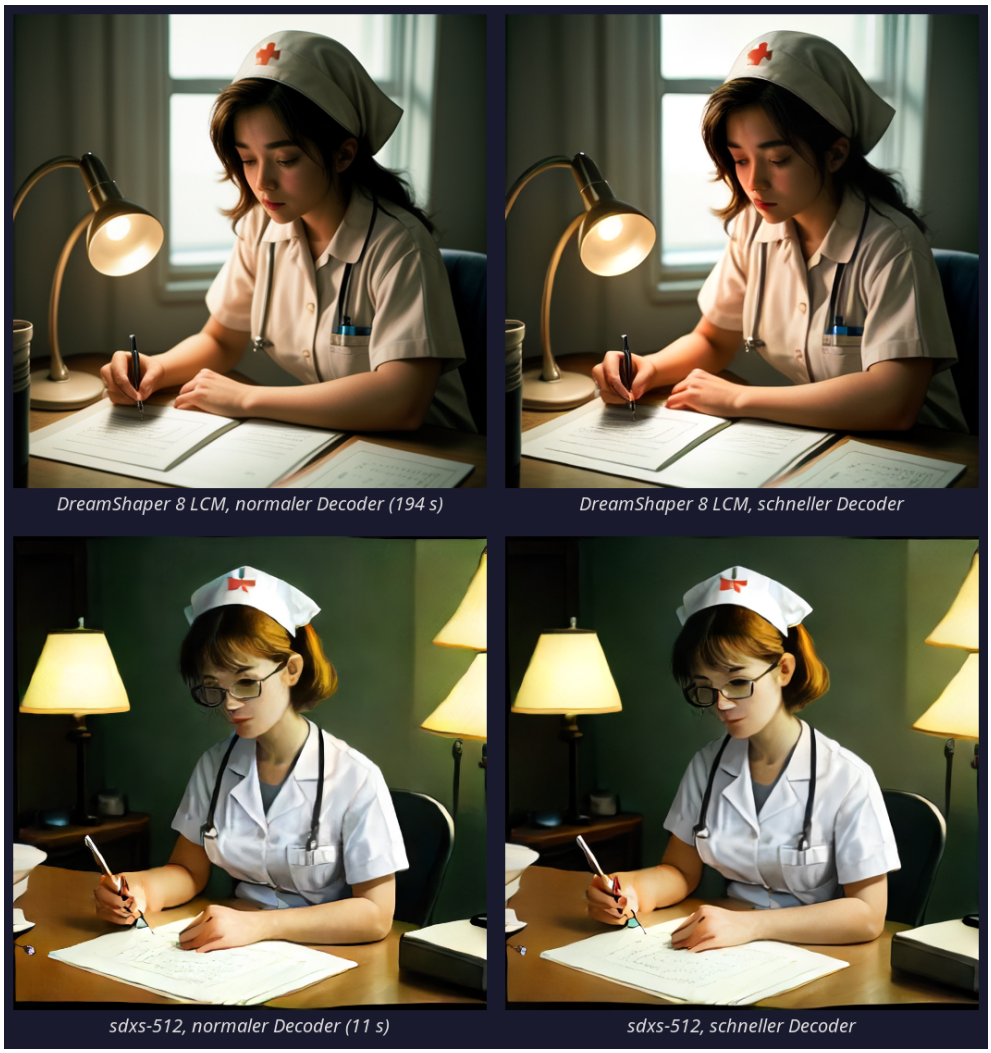


Links sdxs-512 mit einem Schritt, rechts DreamShaper 8 LCM mit sechs Schritten; oben Foto, unten Illustration, jeweils Seed 2026.

Der Unterschied liegt in den Details, an denen Ein-Schritt-Modelle scheitern: Die Hand hält den Stift richtig, das Gesicht ist sauber, Licht und Stoff wirken glaubhaft. Die Stilworte „flat vector illustration“ nimmt DreamShaper allerdings weniger wörtlich und liefert eher eine glatte, gerenderte Zeichnung. Für Illustrationen reicht sdxs oft, für Fotos und für Hände lohnt das größere Modell.

Der schnelle Decoder

Am Ende jedes Auftrags rechnet der Decoder (VAE) das Bild aus dem Zwischenformat zurück; bei Modellen der SD-1.5-Klasse kostet das auf diesem Rechner fast eine Minute. Der Haken **Schneller Decoder (TAESD)** unter Einstellungen tauscht ihn gegen einen kleinen Ersatzdecoder (madebyollin, MIT, 9,8 MB), den DeLoMa beim ersten Einschalten mit festgehaltener Prüfsumme lädt. Gemessen: DreamShaper 8 LCM 160 statt 194 Sekunden, sdxs unverändert, weil sein eigener Decoder schon klein ist. Der Unterschied ist mit bloßem Auge sichtbar und wirkt wie ein eigener Stil: weicher, flacher im Kontrast, fast malerisch, feine Muster werden ungenauer. Manche mögen genau das; für ein Endbild im Fotostil rechnet man besser ohne. Der Zettel nennt den benutzten Decoder. Für SDXL-Modelle gibt es eine eigene Decoderdatei, für FLUX keine; dann bleibt der Haken wirkungslos.



Oben DreamShaper 8 LCM, unten sdxs-512, links mit dem normalen, rechts mit dem schnellen Decoder; Seed 2026.

Was neben jedem Bild liegt

Jedes PNG trägt die Parameter eingebettet, und daneben liegt ein Zettel `.png.json` mit Modell, Beschreibung, Breite, Höhe, Schritten, CFG, Sampler, Seed, Stärke, Rechenzeit, Fäden und Serverstand. Damit lässt sich jedes Bild nachstellen, so wie der Messbericht es für Zahlen verlangt. Ein Klick in Zuletzt erzeugt zeigt Bild und Zettel;

Speichern unter… nimmt den Zettel mit.

Feinsteuerung: LoRA, Maske, Steuerbild, Modellteile

Vier Dinge, die über Beschreibung und Regler hinausgehen. Alle vier sind Wahl, nicht Pflicht, und stehen im Zettel neben dem Bild.

- **LoRA** (unter **Einstellungen** im Tab): kleine Zusatzdateien, die ein Modell in eine Richtung ziehen, etwa die LCM-LoRA, die Stable Diffusion 1.5 in sechs Schritten rechnen lässt, oder ein Stil. **LoRA einbinden…** verknüpft die Datei unter `~/.local/share/deloma/bildmodelle/lora/`, die Auswahl daneben gibt sie mit der gewählten Stärke an den Server weiter (als eigenes Feld des Auftrags; die aus anderen Programmen bekannte Schreibweise in der Beschreibung wertet der Server absichtlich nicht aus). Die Datei muss zur Modellfamilie passen.
- **Zwischenbilder**: schnell zeigt während des Rechnens eine grobe Projektion, schön decodiert jeden Schritt mit dem schnellen Decoder (kostet je Schritt ein paar Sekunden, braucht den Haken oben), keine spart alles.
- **Maske** (Kasten **Bild zu Bild, Maske, Steuerbild**): ein Schwarz-Weiß-Bild in der Größe der Vorlage. Weiße Flächen werden neu gemalt, schwarze bleiben, wie sie sind (Inpainting). Die Maske braucht eine Vorlage; die Stärke gilt wie bei Bild zu Bild.
- **Steuerbild**: eine Skizze oder ein Kantenbild, das die Komposition vorgibt (ControlNet). Dafür braucht es unter **Einrichtung** eine ControlNet-Datei zur Modellfamilie, zum Beispiel control_v11p_sd15_scribble für Stable Diffusion 1.5; **einbinden…** legt sie unter `~/.local/share/deloma/bildmodelle/controlnet/` ab, der Server startet mit ihr neu. Kanten findet DeLoMa nicht selbst; dafür eignen sich PhotoCraft oder GIMP. Die Stärke neben dem Steuerbild sagt, wie streng das Modell folgt. Geprüft am 17.10.2026: eine Strichskizze (Haus mit Tür, Sonne, weiß auf schwarz) wurde mit Stärke 0,9 zu einem Aquarell-Häuschen mit genau dieser Anordnung, in 188 Sekunden.
- **Modellteile…**: manche Modelle bestehen aus mehreren Dateien, etwa FLUX.2-klein aus Diffusionsmodell, Textencoder (Qwen3) und VAE. Die Modelldatei ist dann das Diffusionsmodell, die übrigen Teile ordnen Sie im Dialog zu; sie werden unter `~/.local/share/deloma/bildmodelle/teile/` verknüpft und beim Start mitgegeben. So lief am 17.10.2026 FLUX.2-klein-4B (leejet, Apache 2.0) mit Qwen3-4B als Textencoder und der VAE von Black Forest Labs auf dem Messrechner: ein Bild mit 768 × 768 Pixeln und vier Schritten in 19,5 Minuten, Qualität deutlich über Stable Diffusion 1.5 (Tabelle oben). Stable Diffusion 1.5 in Diffusers-Teilen (UNet, Textencoder, VAE einzeln) nimmt stable-diffusion.cpp dagegen nicht an; dafür braucht es eine Einzeldatei oder eine GGUF-Umwandlung.

Stil aus dem Projekt: Im Projektdialog des Chats (Kapitel 5) lässt sich eine Stilvorlage wählen. Steht im Tab Bilder Kein Stil, nimmt DeLoMa die des gerade gewählten Projekts und vermerkt das im Zettel; eine eigene Wahl im Tab geht vor.

Grafikkarte über Vulkan

Unter **Einrichtung** gibt es den Haken **Grafikkarte (Vulkan-Fassung)**. Er schaltet auf eine zweite Fassung von stable-diffusion.cpp mit Vulkan-Backend um, die

Vulkan-Fassung holen (37 MB, Prüfsumme in DeLoMa festgehalten) getrennt unter `~/.local/share/deloma/sd-vulkan/` ablegt; beide Fassungen bleiben liegen, der Wechsel kostet nichts als einen Neustart des Servers beim nächsten Bild. Welche Teile des Modells auf die Karte passen, entscheidet die Fassung selbst (Diffusion zuerst, dann Textencoder, dann Decoder; was nicht passt, bleibt im Arbeitsspeicher). Die Karte wählen Sie wie bei llama.cpp unter **Einstellungen → llama.cpp → Vulkan-Gerät**.

> **Gemessen auf einer alten Karte**
> Auf der Quadro K2100M des Messrechners (2 GB, offener Treiber NVK) rechnete die Vulkan-Fassung richtig – das Bild glich dem der Prozessor-Fassung bei gleichem Seed –, brauchte für SDXS-512 aber 81 statt 11 Sekunden. Die Karte kann kein 16-Bit-Rechnen. Aus bleibt deshalb die Vorgabe; eine neuere Karte mit mehr Speicher sollte schneller sein, gemessen wurde das nicht.

> **Was der Bildserver öffnet**
> sd-server horcht auf 127.0.0.1, Port 11481, und kennt anders als llama-server keinen Schlüssel: jedes Programm auf diesem Rechner könnte Bilder anfordern. Dateien liest er keine. Mit **Server beenden** oder beim Schließen von DeLoMa ist er weg.

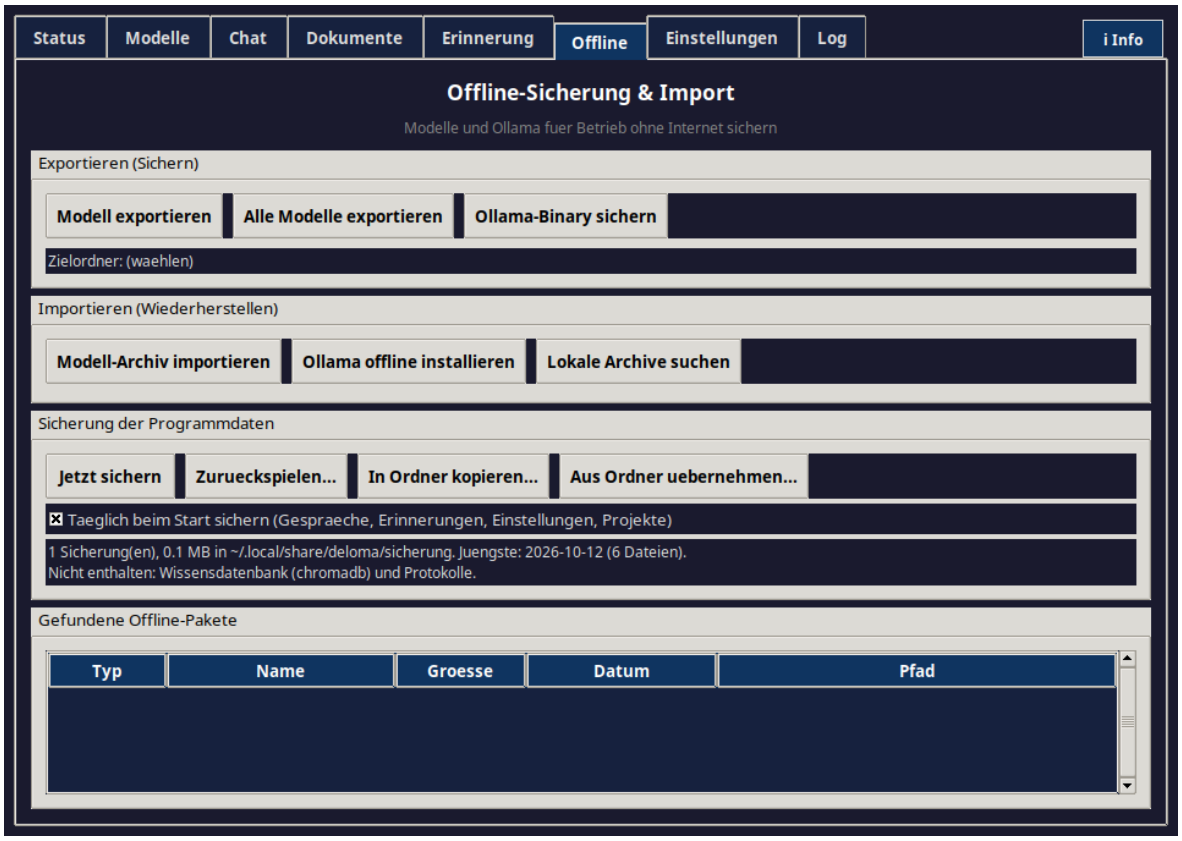
> **Lizenzen und Grenzen**
> Das Programm ist MIT-lizenziert. Modelle haben eigene Lizenzen: sdxs-512 steht unter OpenRAIL++, Stable Diffusion 1.5 unter OpenRAIL-M (beide mit einer Liste verbotener Verwendungen wie Täuschung oder Belästigung), FLUX-Modelle meist unter Apache 2.0. Keine echten Personen nachbilden, fremde Vorlagen nicht nachzeichnen.
> Ein-Schritt-Modelle sind schnell, aber grob; Hände und Gesichter brauchen mehrere Anläufe. LoRA-Dateien und Modelle aus mehreren Teilen (Diffusionsmodell, Textencoder, VAE getrennt) kann der Tab in 0.4 noch nicht laden.

Für Fortgeschrittene

- **sd-server wählen…** trägt ein selbst gebautes Programm ein (Einstellung `sd_server_pfad`).
- **Rechenfäden** 0 heißt Zahl der Kerne (Einstellung `bild_faeden`).
- Protokoll des Servers: `~/.local/share/deloma/data/sd-server.log`, Zustand `sd-server.json`.
- Zum Nachbauen aus dem Quelltext: Repository leejet/stable-diffusion.cpp, `docs/build.md`; cmake und ninja wie beim Bau von llama.cpp.

15. Offline-Betrieb

Für Rechner ohne Internetanschluss – in Kliniken und Behörden eher Regel als Ausnahme. Der Tab **Offline** überträgt einen vollständig eingerichteten Arbeitsplatz auf ein Gerät, das nie online war.



Der Tab Offline.

Übertragen

- **Modelle exportieren** packt ein installiertes Modell als Archiv; **importieren** spielt es auf dem Zielrechner ein.
- **Ollama exportieren** sichert die Programmdatei von Ollama, damit sie auf dem Zielrechner aus der lokalen Kopie installiert werden kann.

Sichern

- **Sicherung der Programmdaten**: täglich automatisch, von Hand, in einen Ordner kopieren, zurückspielen – beschrieben in Kapitel 13.
- **Archive suchen** findet vorhandene Pakete, um sie wiederzuverwenden.

Wo Ollama seine Modelle ablegt
Wurde Ollama als Dienst installiert (der Normalfall), liegen die Modelle unter `/usr/share/ollama/.ollama/models` und gehören dem Nutzer `ollama`. Das Programm findet dieses Verzeichnis von selbst. Lesen darf es aber nur, wenn Ihr Benutzerkonto in der Gruppe `ollama` ist – einmalig `sudo usermod -aG ollama $USER`, danach neu anmelden. Zeigt der Tab 0 Bytes, fehlt meist genau das. Der Import in ein Dienstverzeichnis bleibt Administratoren vorbehalten.

16. Problembehebung

> **Der erste Griff**
> Bei jedem Problem zuerst in den Tab **Log** schauen. Dort steht fast immer im Klartext, was schiefgelaufen ist.

Das Programm startet nicht

Aus dem Terminal starten, dann wird der Grund sichtbar:

```
deloma
```

Meldung	Ursache und Lösung
ModuleNotFoundError	Installation unvollständig. `bash install.sh` wiederholen.
no display name	Kein Grafikbildschirm verfügbar. Das Programm braucht eine Bildschirmsitzung.
database is locked	Das Programm läuft bereits ein zweites Mal. Alle Fenster schließen.
command not found: deloma	`~/.local/bin` ist nicht im Suchpfad. Einmal ab- und wieder anmelden, oder `~/.local/bin/deloma` aufrufen.

Server läuft: Nein

```
systemctl --user restart ollama
# oder, falls systemweit installiert:
sudo systemctl restart ollama
```

Ollama bricht mitten in der Antwort ab

Der Arbeitsspeicher ist ausgegangen; das Betriebssystem hat den Dienst beendet. Nachprüfen lässt sich das so:

```
journalctl -u ollama -n 30 | grep -i 'oom\|killed'
```

Abhilfe, in dieser Reihenfolge:

- **Einstellungen → Speicherschutz** auf automatisch stellen.
- Ein kleineres Modell verwenden.
- Ollama auf ein geladenes Modell begrenzen (siehe Kapitel 3).
- Andere Programme schließen.

Jedes Modell scheitert mit „Internal Server Error“

Erscheint im Chat `[Agent-Fehler: Ollama-Verbindung: Internal Server Error]` bei jedem Modell, liegt es fast nie am Programm, sondern an Ollama und der Grafikkarte. Nachsehen:

```
journalctl -u ollama -b | grep -iE 'inference compute|Unsupported device|segmentation'
```

Steht dort Unsupported device oder ein Absturz von `llama-server`, kann Ollama die Grafikkarte nicht benutzen (typisch bei alten Karten über den offenen Treiber). Dann unter **Einstellungen → Grafikkarte** den Haken setzen – siehe Kapitel 11. Dasselbe von Hand im Terminal:

```
sudo mkdir -p /etc/systemd/system/ollama.service.d
printf '[Service]\nEnvironment="OLLAMA_VULKAN=false"\n' \
  | sudo tee /etc/systemd/system/ollama.service.d/cpu-only.conf
sudo systemctl daemon-reload && sudo systemctl restart ollama
```

> **Nur bei diesen Symptomen**
> Auf einem Rechner mit brauchbarer Grafikkarte schaltet dieser Eingriff die Beschleunigung ab. Nur anwenden, wenn das Protokoll die genannten Meldungen zeigt. Der Knopf **Ollama-Protokoll prüfen** in den Einstellungen nimmt Ihnen das Nachsehen ab.

Die Antworten sind schlecht

Beobachtung	Was hilft
Antwortet auf Englisch	System-Prompt setzen: „Antworte immer auf Deutsch.“
Viele Floskeln	Haken **Sprachfilter** setzen
Erfindet Tatsachen	Größeres Modell; im System-Prompt zum Zugeben von Unwissen auffordern
Ignoriert die Dokumente	Haken **Dokumente** gesetzt? Richtige Wissensdatenbank gewählt? Suche im Tab Dokumente testen. Ist `nomic-embed-text` installiert?
Vergisst den Gesprächsverlauf	Kontext ist voll. Neuen Chat beginnen oder Erinnerungen ausdünnen

Alles ist im Akkubetrieb langsam

Manche Notebooks – gemessen am Fujitsu Celsius H730 – takten ohne Netzteil den Prozessor auf ein Fünftel herunter, egal was die Energieeinstellungen sagen. Die KI wird dann zwei- bis dreimal langsamer und die Antwortzeit schwankt stark. Prüfen:

```
cat /sys/class/power_supply/AC/online   # muss 1 ergeben
```

Für ernsthaftes Arbeiten mit der KI: Netzteil anschließen.

Der Auto-Agent ruft keine Werkzeuge auf

Fast immer liegt es am Modell. Nehmen Sie `qwen2.5:3b`, `gemma4:e4b` oder `qwen3:30b-a3b`. Erscheint im Chat der Hinweis, das Modell habe den Aufruf als Text ausgegeben, ist das Modell für diesen Zweck ungeeignet – unabhängig davon, was Ollama über seine Fähigkeiten meldet.

Die Web-Suche findet nichts

Die Suche läuft über DuckDuckGo. Ohne Internetverbindung oder bei aktivem VPN kann sie fehlschlagen. Die Fehlermeldung im Chat nennt den Grund.

Vorlesen funktioniert nicht

```
ls ~/.local/share/deloma/venv/bin/piper
ls ~/.local/share/deloma/voices/
```

Fehlt das Programm oder die Stimmdatei, holen Sie beides nach – siehe Kapitel 10. Unter **Einstellungen → Sprachausgabe** steht im Klartext, was fehlt. Starten Sie das Programm über das Startmenü oder mit `deloma`; nur so ist das Sprachprogramm im Suchpfad.

Der Dokumente-Tab bleibt leer

Entweder fehlt das Paket `chromadb` in der Python-Umgebung, oder das Einbettungsmodell. Beides nachholen:

```
~/.local/share/deloma/venv/bin/python -m pip install -r ~/.local/share/deloma/requirements.txt
ollama pull nomic-embed-text
```

Die Tastenkombination reagiert nicht

- Mit dem Knopf **Testen** prüfen.
- Eine andere Kombination probieren – die Arbeitsumgebung könnte sie abfangen.
- Unter Wayland sind globale Tastenkombinationen systemseitig eingeschränkt. Der Sitzungstyp steht in den Einstellungen.

Selbsttest

Das Programm bringt eine Testsammlung mit, die alle Bestandteile prüft:

```
cd ~/.local/share/deloma
venv/bin/python -m unittest tests
```

Am Ende muss **OK** stehen. Andernfalls nennt die Ausgabe, welcher Teil nicht funktioniert.

17. Lizenz und Weitergabe

DeLoMa ist freie Software. Es steht unter der GNU Affero General Public License, Version 3 – kurz AGPL. Das ist eine der bekannten Lizenzen für offene Software, und sie hat für Sie vor allem angenehme Folgen.

Was Sie dürfen

- Das Programm benutzen – privat wie beruflich, ohne Gebühr, ohne Anmeldung, auf beliebig vielen Rechnern.
- Das Programm weitergeben – an Kolleginnen und Kollegen, an andere Einrichtungen, an wen Sie möchten.
- In den Quelltext sehen und ihn verändern. Der komplette Quelltext liegt im Programmordner; es gibt keinen versteckten Teil.

Was Sie im Gegenzug tun müssen

Nur eines, und es betrifft Sie nur dann, wenn Sie das Programm verändern und die veränderte Fassung anderen zugänglich machen:

- Die veränderte Fassung muss ebenfalls unter der AGPL stehen.
- Sie muss ihren Quelltext mitbringen.

Das gilt bei der AGPL auch dann, wenn Sie die veränderte Fassung nicht als Datei weitergeben, sondern über ein Netzwerk anbieten – etwa als Webdienst. Wer daraus einen Onlinedienst macht, muss seinen Nutzern den Quelltext zugänglich machen.

Kurz zusammengefasst

Darf ich …	
… das Programm im Betrieb einsetzen?	Ja, ohne Einschränkung
… es an Kollegen weitergeben?	Ja
… Geld dafür verlangen?	Ja, auch das erlaubt die AGPL
… es verändern?	Ja
… eine veränderte Fassung geheim halten?	Nur solange Sie sie für sich behalten
… daraus einen Cloud-Dienst machen?	Ja, aber dann mit offenem Quelltext

> **Warum die AGPL**
> Weil dieses Programm dafür gebaut wurde, dass Ihre Daten bei Ihnen bleiben. Eine freizügigere Lizenz würde erlauben, daraus einen geschlossenen Cloud-Dienst zu machen – also genau das, wogegen es antritt. Die AGPL schließt diese Lücke.

Der Name gehört nicht dazu

„DeLoMa – Deine Lokale Maschine“ ist die Bezeichnung dieses Projekts und nicht Teil der Lizenz. Veränderte Fassungen sind unter eigener Bezeichnung oder mit deutlich sichtbarem Zusatz zu verbreiten; die sachliche Nennung der Herkunft bleibt erlaubt. „Ollama“ ist eine Marke von Ollama Inc.; DeLoMa steht in keiner Verbindung zu Ollama Inc.

Die KI-Modelle gehören nicht dazu

Die Lizenz gilt für dieses Programm. Die KI-Modelle, die Sie damit herunterladen, sind eigenständige Werke mit eigenen Bedingungen. Die meisten sind frei nutzbar, einige haben Auflagen für den gewerblichen Einsatz. Wenn Sie ein Modell beruflich einsetzen, werfen Sie einmal einen Blick in dessen Lizenz – auf der Seite, von der es stammt.

Keine Gewährleistung

Wie bei freier Software üblich, wird das Programm ohne Gewährleistung bereitgestellt. Es ist sorgfältig getestet und im täglichen Einsatz, aber niemand haftet dafür, dass es für Ihren Zweck taugt. Halten Sie es wie mit jeder Software: Sichern Sie wichtige Daten regelmäßig.

Wo der vollständige Text steht

Der amtliche Lizenztext ist auf Englisch und liegt als Datei `LICENSE` im Programmordner:

```
~/.local/share/deloma/LICENSE
```

Im Netz finden Sie ihn unter www.gnu.org/licenses/agpl-3.0. Dort gibt es auch inoffizielle Übersetzungen und eine gut lesbare Erklärung in einfacher Sprache.

Copyright © 2026 Benjamin Tobies

18. Anhang

Alle Tabs auf einen Blick

Tab	Inhalt
Status	Hardware, Ollama-Zustand, installierte Modelle
Modelle	Modelle laden, löschen, eigene Varianten bauen
Chat	Der Arbeitsplatz
Dokumente	Wissensdatenbanken
Bilder	Bilder erzeugen mit stable-diffusion.cpp
Erinnerung	Erinnerungen, Aufgaben, Verlauf, Bewertungen
Offline	Betrieb ohne Internet, Sicherung
Einstellungen	Sprache, Speicherschutz, Grafikkarte, Schnellzugriff, MCP
Log	Technische Meldungen



Der Info-Dialog (Knopf i Info oben rechts): Version, Impressum, Lizenz.

Wichtige Verzeichnisse

Pfad	Inhalt
~/.local/share/deloma/	Programm und Python-Umgebung (venv/)

Pfad	Inhalt
~/.local/share/deloma/data/	Ihre Daten
~/.local/share/deloma/voices/	Stimmen
~/.ollama/models/ bzw. /usr/share/ollama/.ollama/models/	KI-Modelle
~/.config/autostart/	Autostart-Einträge
~/deloma-sicherung/	Sicherung, die das Update angelegt hat

Nützliche Befehle

Befehl	Wirkung
deloma	Programm starten
ollama list	Installierte Modelle anzeigen
ollama pull NAME	Modell laden
ollama rm NAME	Modell löschen
ollama ps	Welche Modelle sind gerade im Speicher?
systemctl --user restart ollama	Dienst neu starten
venv/bin/python -m unittest tests	Selbsttest (im Programmordner)

Wenn gar nichts mehr geht

1. Tab **Log** lesen.
2. Aus dem Terminal starten und die Ausgabe lesen.
3. Zwischengespeicherte Dateien löschen: `find ~/.local/share/deloma -name __pycache__ -exec rm -rf {} +`
4. Installation wiederholen: `bash install.sh`
5. Selbsttest laufen lassen.

> **Ihre Daten liegen getrennt vom Programm**
> Eine Neuinstallation löscht weder Gespräche noch Einstellungen.