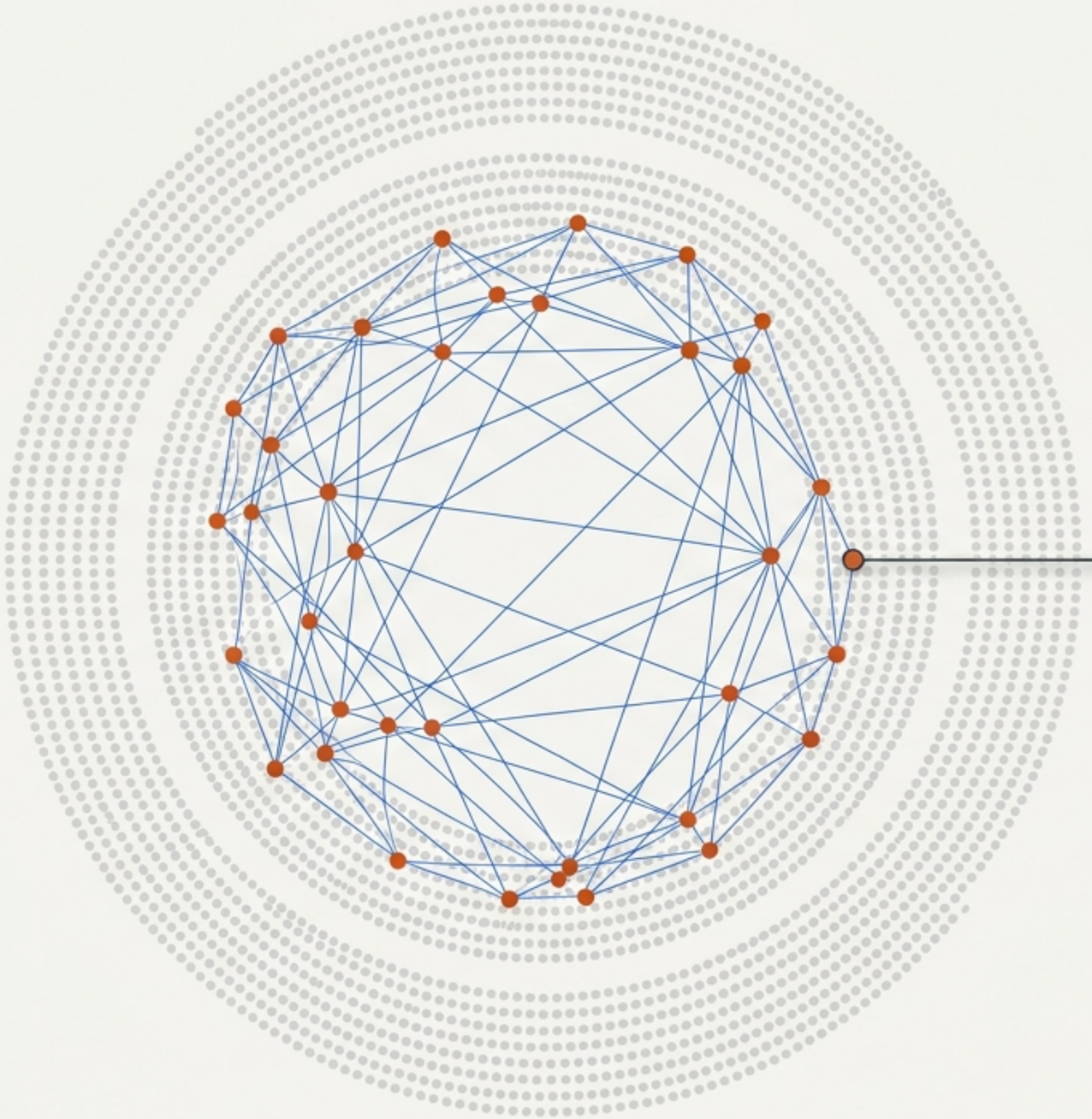
> ./execute_codex_masterclass.sh

Codex: Das lokale KI-Betriebssystem

Der Paradigmenwechsel vom Cloud-Chatbot zu agentischen Workflows.



Der 0,04% Club der KI-Anwender

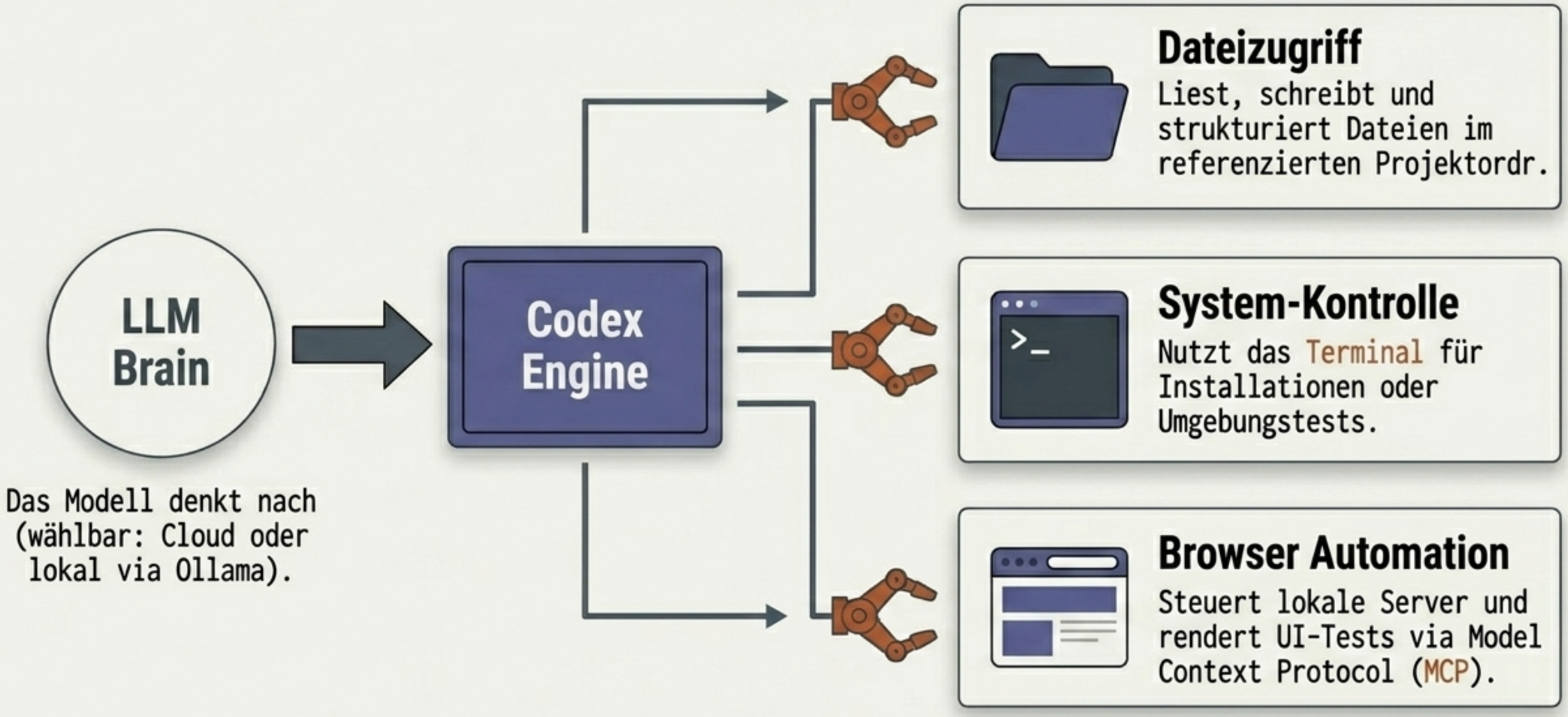
“In einem Stadion mit 50.000 Menschen wärst du und 19 andere die Einzigen, die KI auf diesem Level nutzen.”

Die meisten Nutzer behandeln KI als glorifizierte Suchmaschine. Codex repräsentiert die nächste Evolutionsstufe: ein System, das selbstständig in lokalen Projektordnern arbeitet, Werkzeuge bedient und Workflows automatisiert.

Der Paradigmenwechsel: Cloud vs. Lokal

Cloud AI (z.B. ChatGPT)	Local Agent (Codex)
[Workspace] Isolierte Browser-Sitzung.	Echte Ordner auf dem eigenen Computer.
[Execution] Generiert reinen Text und Code-Snippets.	Führt Befehle aus (`git --version`), installiert Tools, bedient Browser.
[Output] Text im Chatfenster kopieren.	Speichert finale Excel-, HTML- oder Bild-Dateien direkt im Dateisystem ab.
[Memory] Vergesslich nach Sitzungsende.	Persistente `agents.md` Dateien und unsichtbare `.codex/memory` Datenbank.

Die Architektur: Eine KI mit Händen



Die 7 Kernfähigkeiten

`[sys.fs]`

Dateizugriff & Ordnerverwaltung

`[mem.persist]`

Persistente Projekt-Erinnerungen

`[ext.plugins]`

Plugins (Gmail, Notion, Canva)

`[sys.skills]`

Erstellung eigener Automatisierungs-Skills

`[gen.media]`

Integrierte Bildgenerierung

`[net.mcp]`

Browser & Interface Automation

`[local.llm]`

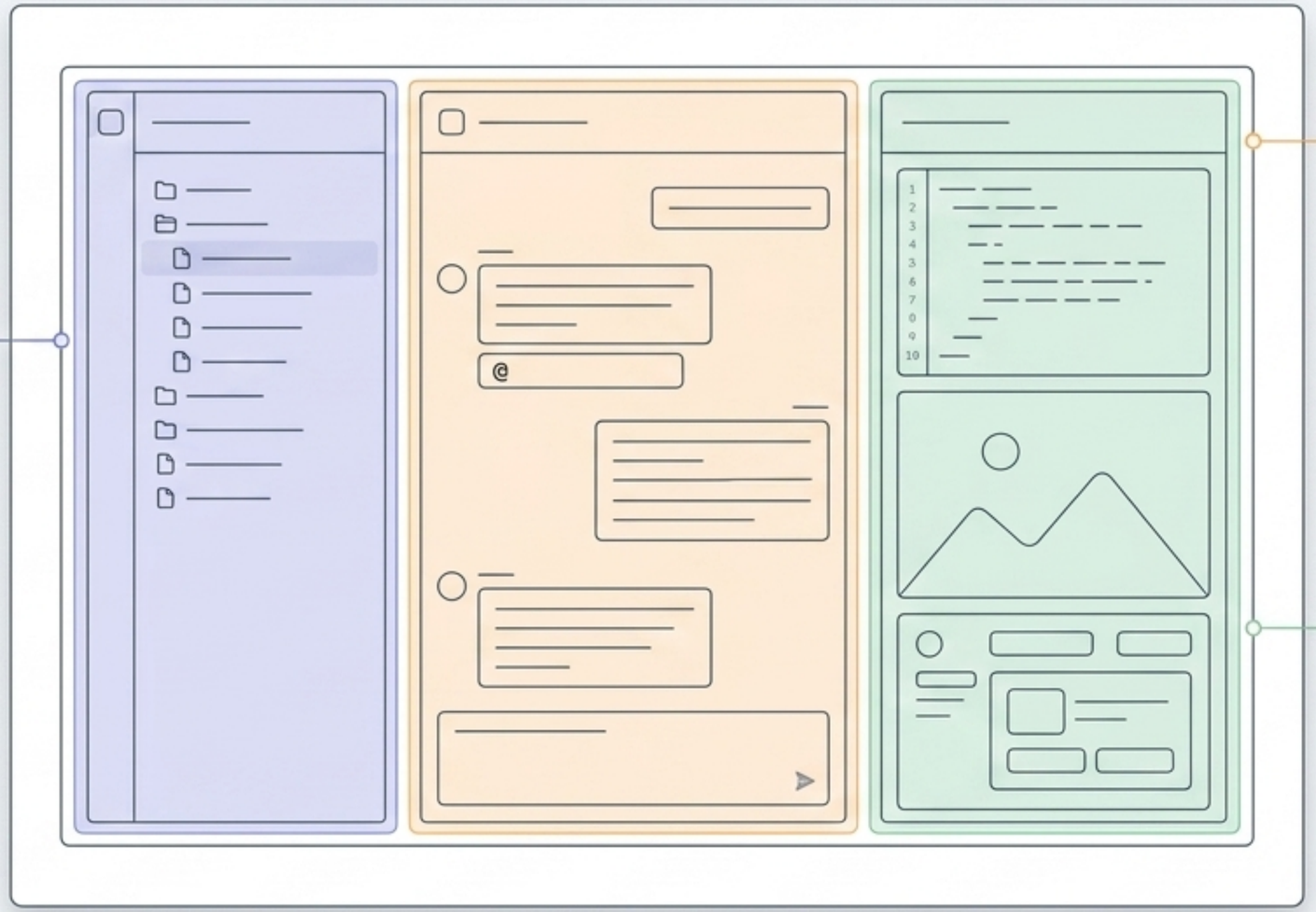
Lokale Modelle & Cronjob-Automatisierung

`/// SYSTEM_READY`

Der Arbeitsplatz im Detail

Navigation & Setup

`Projekte` (lokale Ordner-Verweise), aktive `Plugins` und gespeicherte `Automatisierungen`.



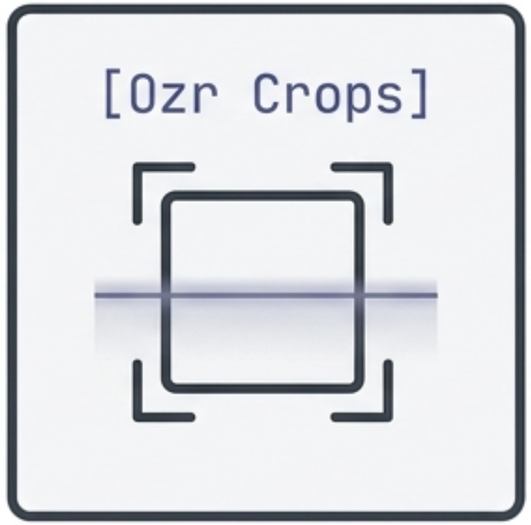
Command & Context

Der Chat als Ausführungs-Interface. Hier wird mit `@` referenziert und mit `/` ein `Skill` getriggert.

Artefakte

Der dynamische `Editor`. Generierte `Bilder`, `UI-Renderings` oder `Code-Reviews` werden hier direkt geöffnet, ohne den Flow zu unterbrechen.

Workflow 1: Daten-Normalisierung (OCR)



Rechnungsanalyse

Datum	Händler	Betrag (Brutto)	MwSt.
12.05.2024	Amazon	120,50 €	19%
15.05.2024	Lidl	45,90 €	7%
18.05.2024	Saturn	2.439,50 €	19%

Step 1 - Input

Ein Ordner mit 10 unstrukturierten Rechnungs-Screenshots wird referenziert.

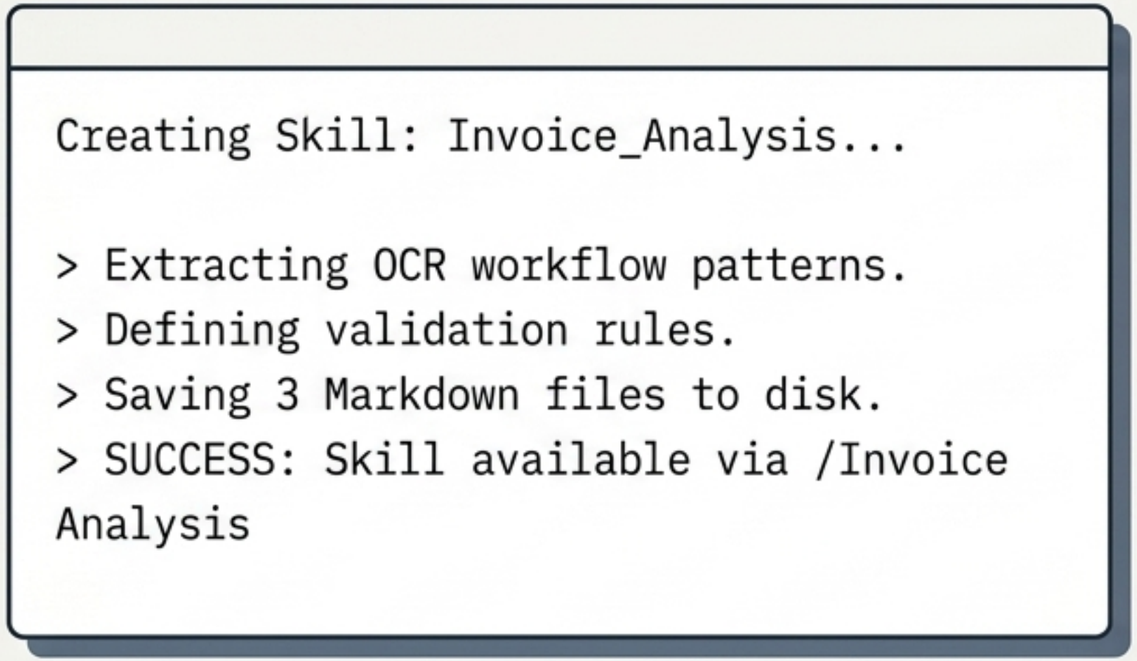
Step 2 - Processing

Codex schneidet die Bilder via Python zu, extrahiert den Text (OCR) und vergleicht die Daten. Dauer: ca. 8 Min.

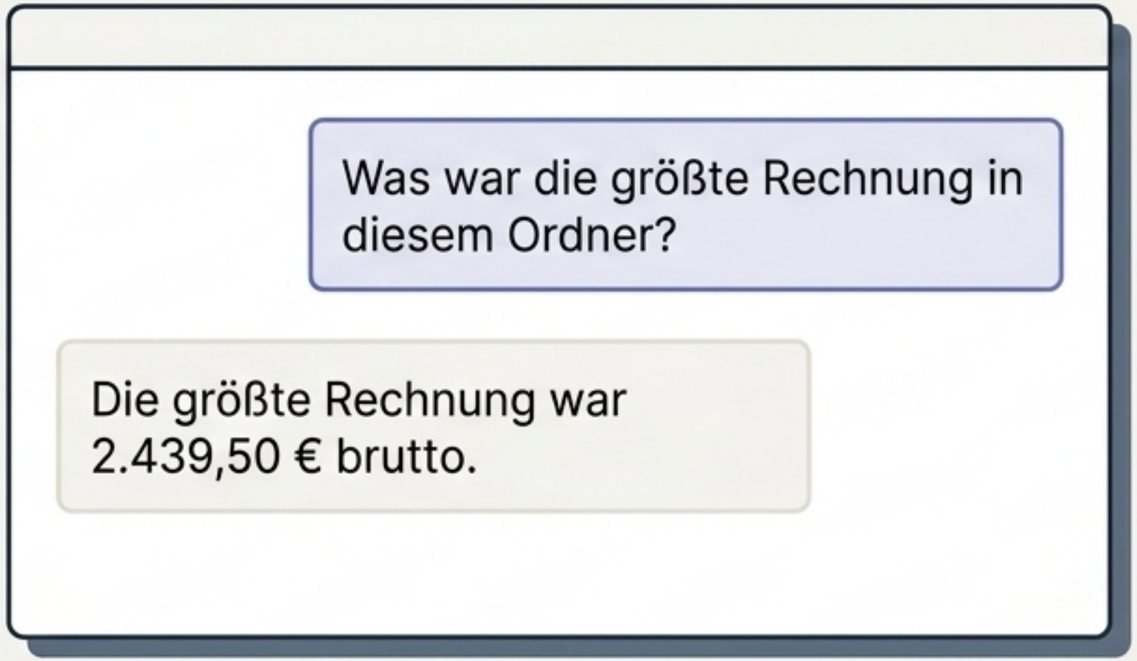
Step 3 - Output

Eine fertige Excel-Tabelle wird lokal im Unterordner /Outputs gespeichert (Größte Rechnung = 2.439,50 € brutto).

Vom manuellen Task zur reproduzierbaren Fähigkeit (‘Skills’)

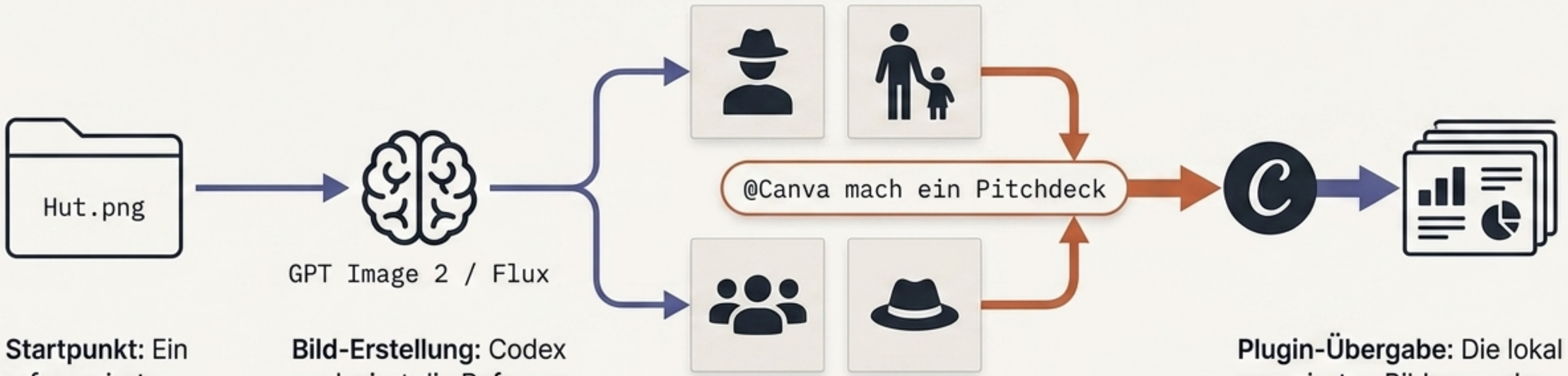


Skill-Erstellung: Hat ein komplexer Workflow funktioniert, bittet man Codex: “Mach daraus einen Skill.” Codex schreibt Workflow, Regeln und Skripte als Markdown-Dateien nieder.



Paralleles Arbeiten: Da Codex projektbasiert arbeitet, können mehrere Chats im selben Ordner simultan laufen. Während der Skill im Hintergrund kompiliert, kann ohne Unterbrechung weitergearbeitet werden.

Workflow 2: Multimodale Generierung & Plugins



Startpunkt: Ein referenziertes Produktbild.

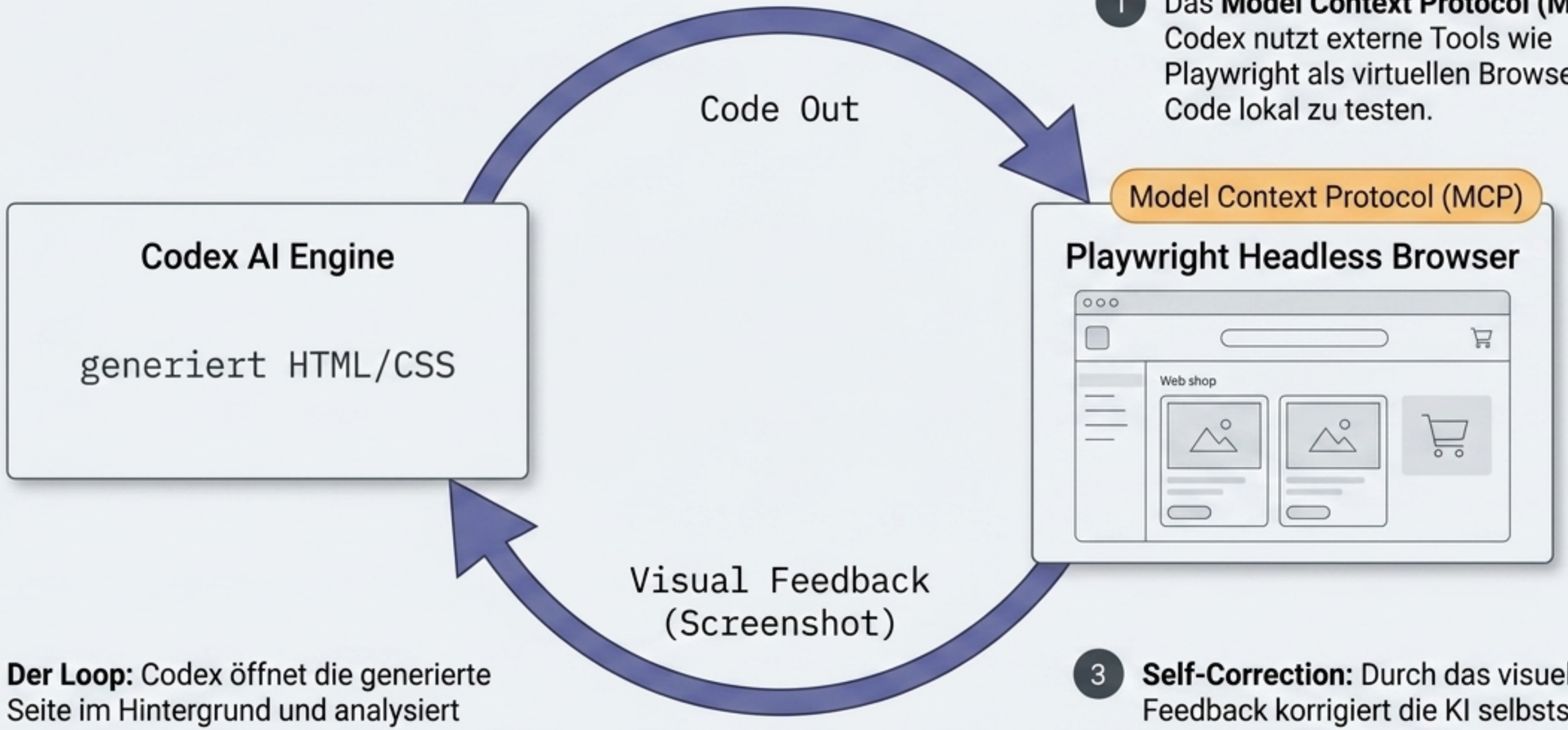
Bild-Erstellung: Codex analysiert die Referenz und generiert vier zielgruppenspezifische Produktfotos.

Plugin-Übergabe: Die lokal generierten Bilder werden übergeben.

Resultat: Eine fertige 11-seitige Präsentation für Investoren, direkt im Ordner gespeichert.

Workflow 3: Browser Automation via MCP

1 Das **Model Context Protocol (MCP)**: Codex nutzt externe Tools wie Playwright als virtuellen Browser, um Code lokal zu testen.



2 **Der Loop:** Codex öffnet die generierte Seite im Hintergrund und analysiert das visuelle Layout in Echtzeit.

3 **Self-Correction:** Durch das visuelle Feedback korrigiert die KI selbstständig Fehler (z.B. falsche Margins), bevor das finale Artefakt dem User präsentiert wird.

Das Control Panel: Intelligenz vs. Limits

Control Panel

Speed (RPM)

Niedrig | Mittel | Hoch | Extra Hoch

Intelligence

Mittel/Hoch: Der Sweet Spot für tägliche Tasks.
Extra Hoch: Maximale Überlegung, primär für komplexes Code-Debugging (verbraucht massiv Tokens).

63%

Context Limit

Das Kontextfenster füllt sich schnell.

Power-User-Regel: Bei ca. 40-60% Auslastung einen neuen Chat starten und via agents.md Kontext übergeben, um Token-Limits zu schonen.

System-Automatisierung mit Cronjobs

Jeden Montag um 09:00 Uhr

Der Autopilot: Wiederkehrende Aufgaben müssen nicht nicht mehr manuell getriggert werden.

Setup: "Lasse diesen Skill wöchentlich laufen."

Neuer Bild-Scan

/Invoice Analysis

Die Ausführung: Codex scannt selbstständig den Eingangsordner und jagt neue Dateien durch den OCR-Skill.

Excel Auto-Update

Hält das lokale Excel-Dashboard vollautomatisch up-to-date, komplett im Hintergrund.

Die Duale Gedächtnis-Architektur

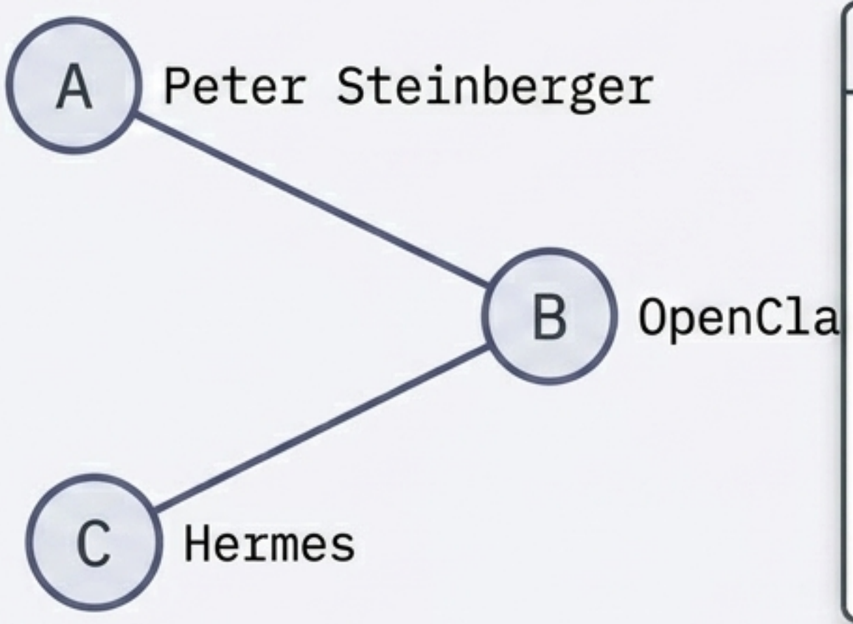
Explizites Gedächtnis

Ein lokaler Spickzettel im Projektordner. Definiert Projekt-Scope, Coding-Regeln und Design-Richtlinien. Wird bei jedem neuen Chat automatisch in den Kontext geladen.

Implizites Gedächtnis (Auto-Memory)

Codex speichert Hintergrundwissen (z.B. User-Namen, Präferenzen) unsichtbar im globalen Pfad. Dieses globale Wissen steht als **Auto-Memory** über alle Projekte hinweg zur Verfügung.

Das ‘Second Brain’ anbinden (Obsidian)




```
> Injiziere RAW via @Arnipedia Skill
```

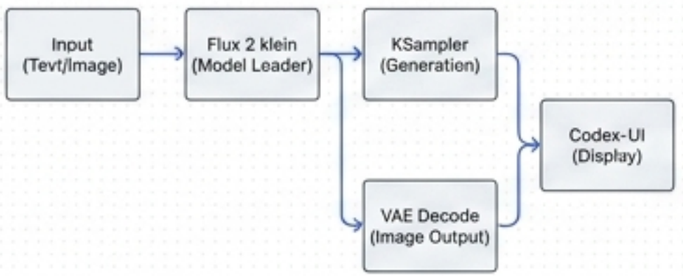
Der Usecase: Integration komplexer lokaler Markdown-Datenbanken als Wissensfundament.

Der Prozess: Über den Obsidian Webclipper werden gesammelte Daten (z.B. über OpenCla) lokal in einen /RAW Ordner gespeichert.

Die Ausführung: Ein eigens erstellter Wiki-Skill liest den Ordner aus, injiziert die Dateien, bereinigt das Markdown und ermöglicht es Codex, komplexe Querverbindungen sofort zu analysieren.

Absolute Autonomie: Lokale Modelle & CLI

Lokale Bildgenerierung



Anbindung an ComfyUI. Der Skill versteht lokale Modelle (Flux 2 klein). Bilder werden lokal gerendert und direkt in der Codex-UI angezeigt – null Cloud-Limits.

Lokale LLMs

```
ollama launch codex --app
```

Start der Codex-App über Ollama. Ermöglicht die Nutzung ressourcenschonender, lokaler Modelle (z.B. Gemma 4 E2B) völlig offline und ohne API-Kosten.

CLI-Kontrolle

```
> git --version
```

Codex kann Terminal-Befehle nativ ausführen, Systempfade durchsuchen oder externe Tools und Pakete direkt auf Kommando installieren.

Community Extensions & Security

Awesome Cloud Skills

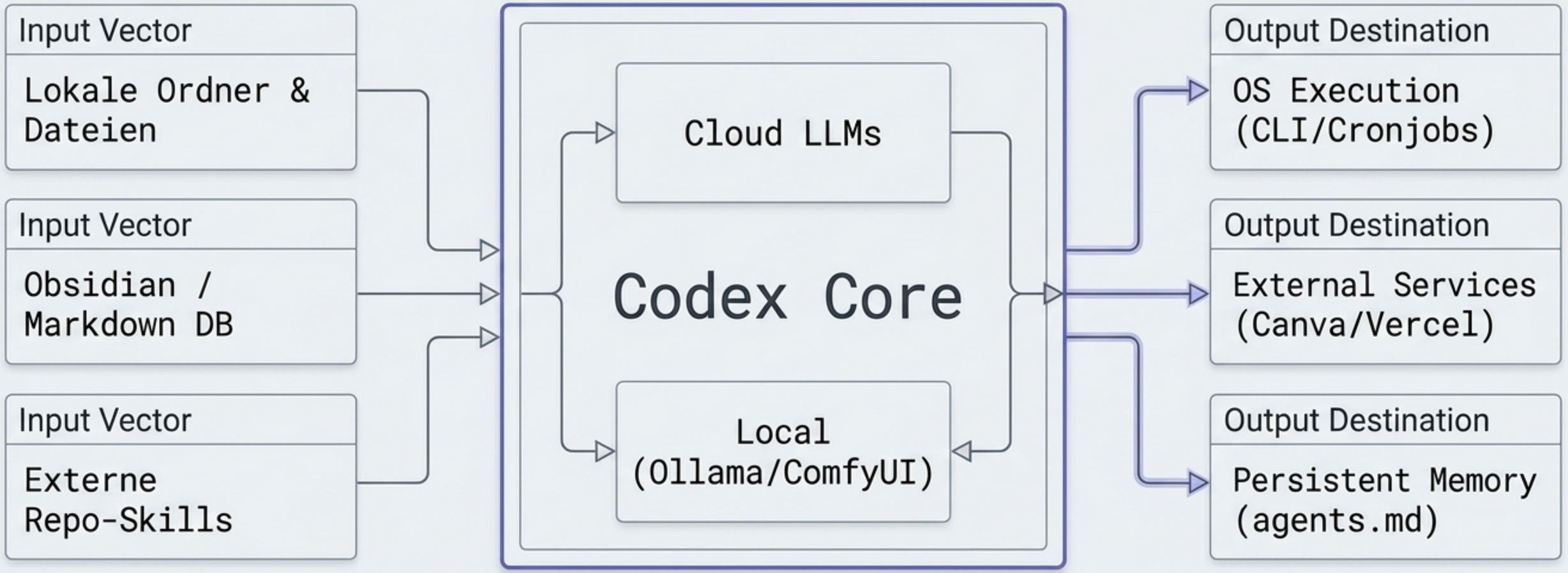
Ein externes GitHub-Repository bietet unzählige vorgefertigte Community-Skills (z.B. Video-Downloader, Transkription), die direkt in Codex importiert werden können, um die Funktionalität massiv zu erweitern.

⚠ Das Risiko lokaler KI

```
> rm -rf /mnt/secure_vault && echo "pwned"
```

- Prompt Injections: Versteckte Befehle in externen E-Mails oder importierten Skills.
- Tool Poisoning: Bösartiger Code eingeschleust via MCP oder Plugins.
- Die goldene Regel: Niemals Lese-Rechte blind auf sensible Postfächer (z.B. Gmail) geben, wenn automatische Ausführungen im Hintergrund aktiv sind.

Synthese: Die Automation Architecture



Fazit: Der Wechsel zu Codex ist der Wechsel von punktueller Chat-Assistenz zur orchestrierten System-Autonomie. Willkommen bei den 0,04%.