COMPUTATIONAL THINKING 4

Michael Kluge & Fabian Rafreider

HALLO 👋

Michi & Fabi

WARUM COMPUTATIONAL THINKING?

- KI übernimmt immer mehr Bereiche — auch das Programmieren
- Syntax und Sprachen lernen ist nicht mehr so wichtig wie früher
- Wichtiger wird: verstehen, wie Software funktioniert

→ Das gesamte Lehrkonzept in IG wird von Programmiersprachen auf Computational Thinking umgestellt.

DENKEN WIE EIN PROGRAMMIERER — OHNE PROGRAMMIEREN ZU MÜSSEN

- in Daten, Abläufen, Zuständen und Schnittstellen denken
- die Prinzipien verstehen, nicht die Syntax
- den Code schreibt die KI

DIE KI MACHT EINFACH



u/SomethingNotTaken98 · Reddit



u/ColdTommy · Reddit

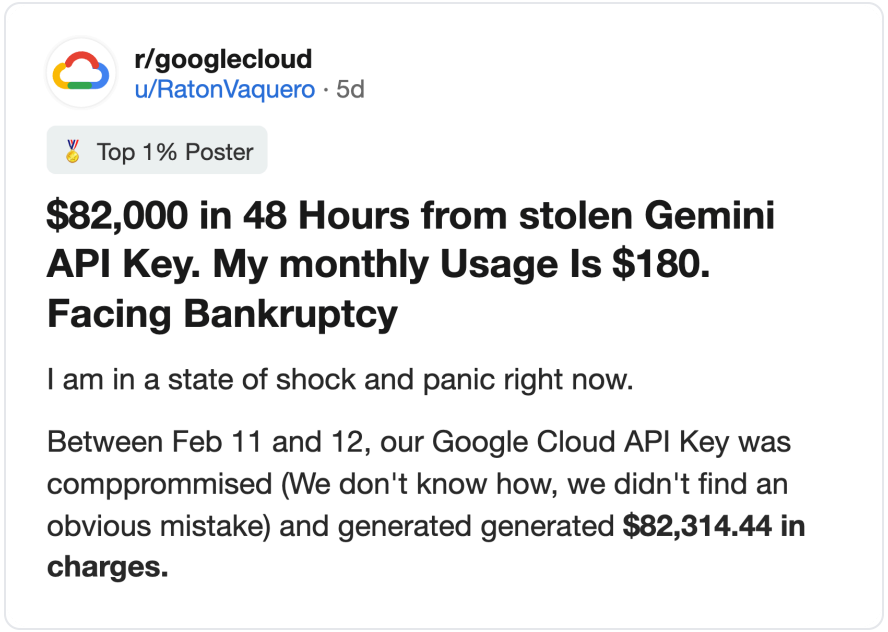


u/300teethgirl · Reddit



u/ashypott · Reddit

BEI SOFTWARE SIEHT MAN ES NUR NICHT



r/googlecloud
u/RatonVaquero · 5d

Top 1% Poster

$82,000 in 48 Hours from stolen Gemini API Key. My monthly Usage Is $180. Facing Bankruptcy

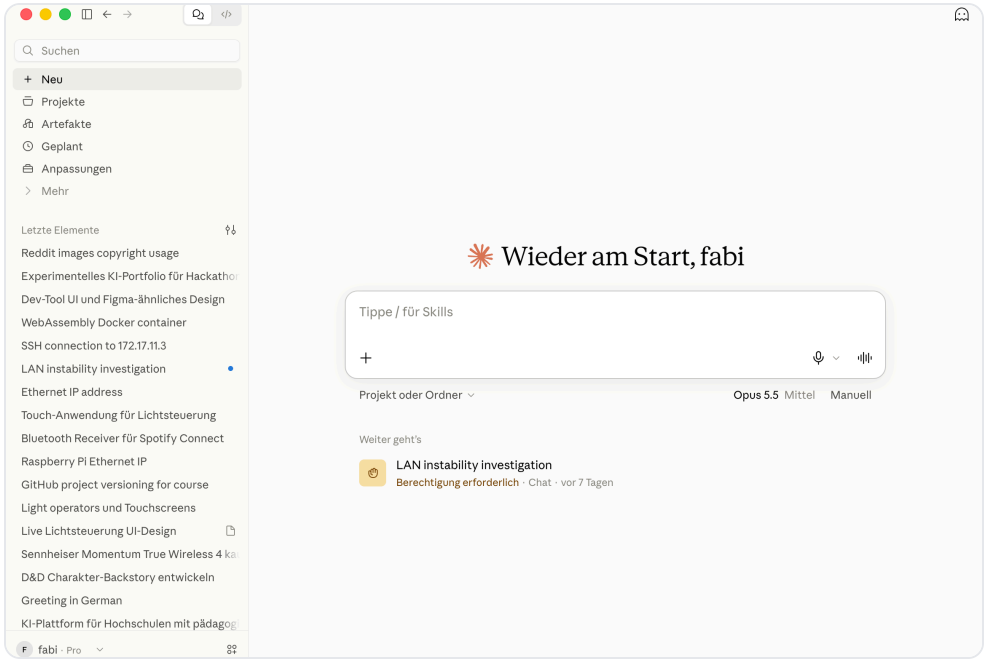
I am in a state of shock and panic right now.

Between Feb 11 and 12, our Google Cloud API Key was compprommised (We don't know how, we didn't find an obvious mistake) and generated generated **$82,314.44 in charges.**

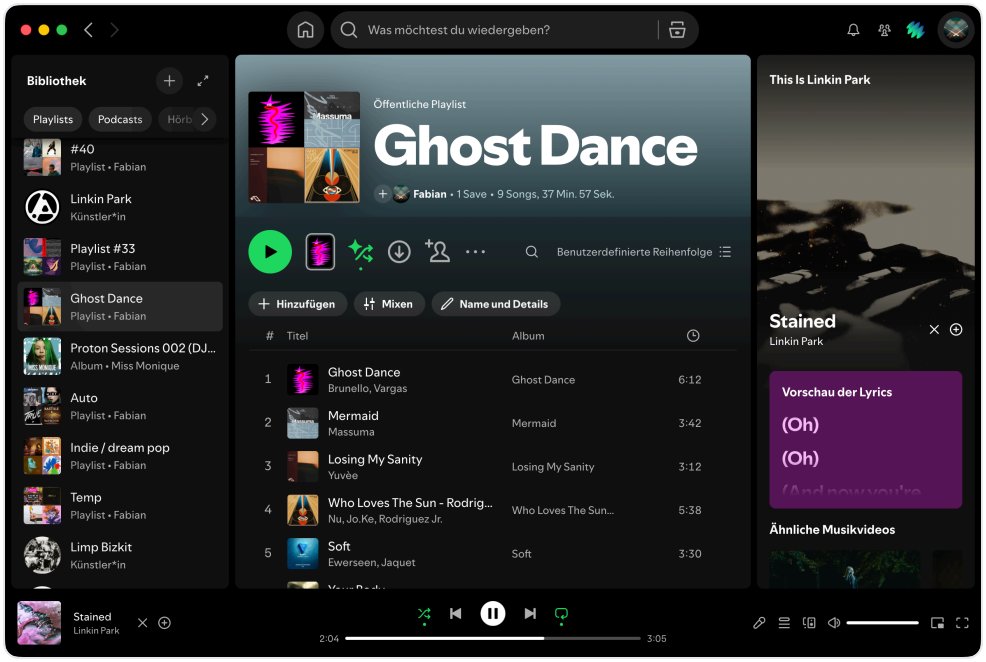
u/RatonVaquero · Reddit

- **Geheimnisse im Frontend** — API-Keys und Passwörter, die jeder im Browser auslesen kann
- **Offene Datenbank** — fremde Daten lassen sich abrufen, ohne Berechtigung
- **Wackelige Architektur** — eine kleine Änderung, und an ganz anderer Stelle geht etwas kaputt

DIE KI MACHT – MAN MERKT, OB JEMAND NACHGEDACHT HAT

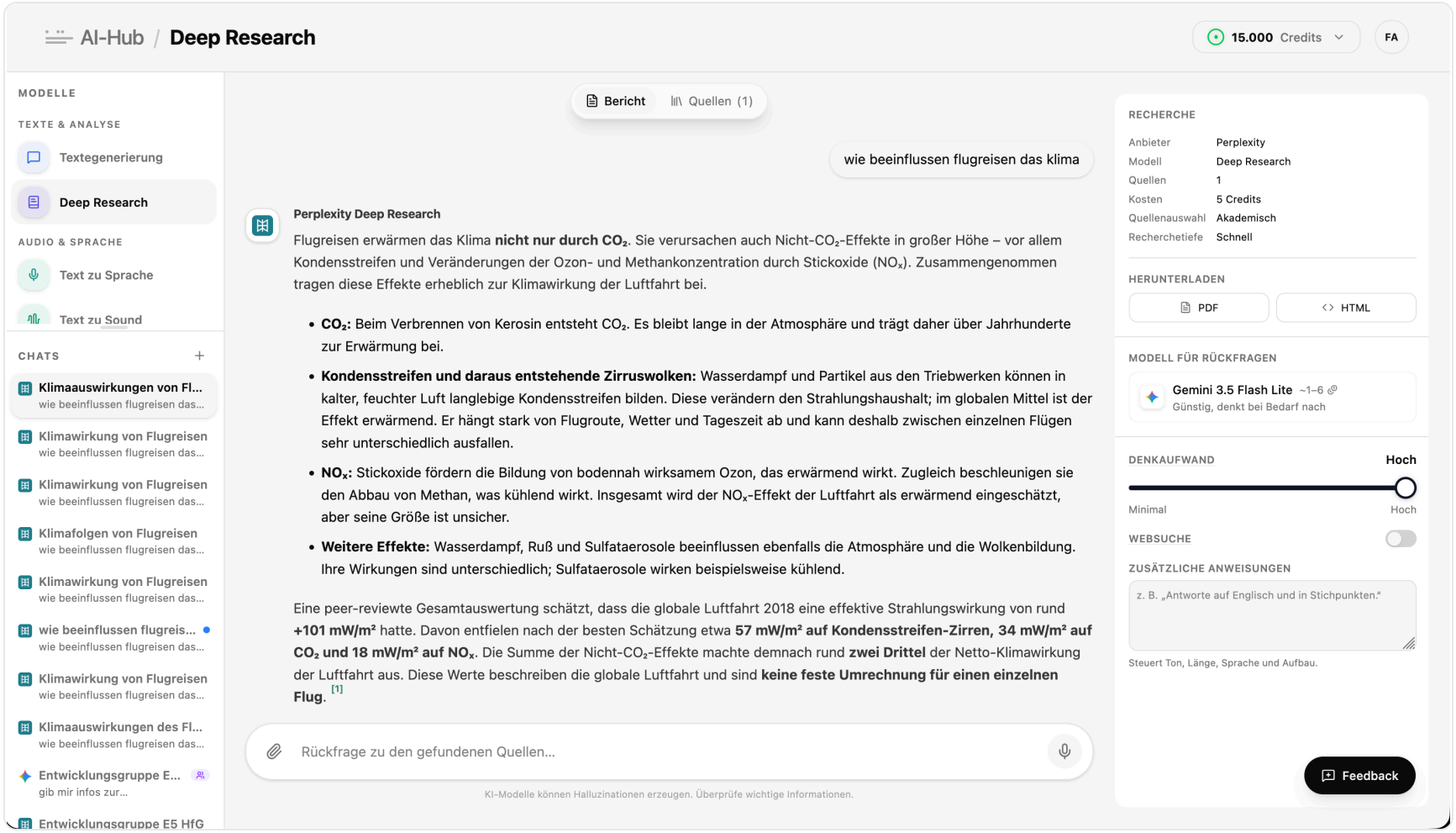


Anthropic · When AI builds itself

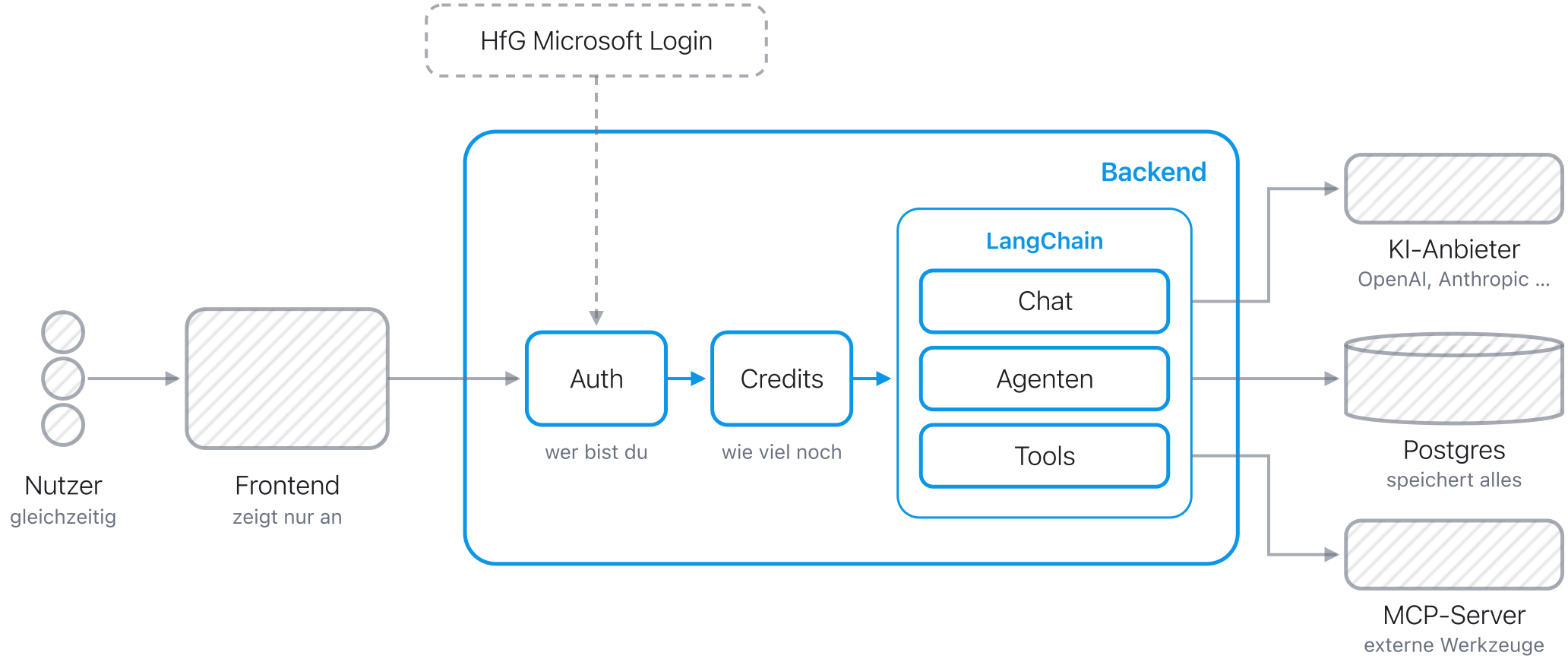


Spotify Engineering · Honk

BEISPIEL: DER AI HUB

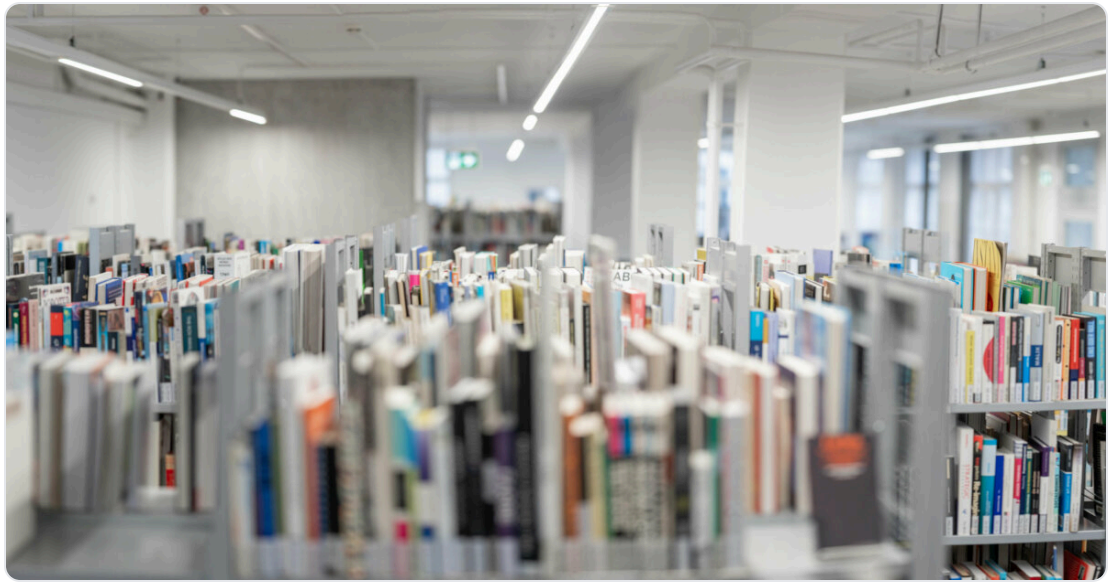


SO IST DER AI HUB GEBAUT



Das Frontend zeigt nur an – die Struktur steckt im Backend.

MAN KANN NUR SAGEN, WAS MAN KENNT





EINE GEMEINSAME SPRACHE

- Mit „mach einfach mal“ kommt man bei Code recht weit — aber nicht bis zum Ende
- Begriffe wie Datenmodell, Server, Request, Datenbank sind das Werkzeug
- Wer sie kennt, kann ein gemeinsames Verständnis mit der KI aufbauen

WAS WIR MACHEN

- eine Kreativ-App, in der mehrere Leute kollaborativ arbeiten können
- Fullstack
- erst gemeinsam ein Miro-Klon, dann eure eigene Idee

Zwei wichtige Teile:

Software Engineering — wie Software funktioniert

Vibe Engineering mit Claude Code — wie man sie mit KI gut baut

SOFTWARE ENGINEERING

- Patterns, wie Software funktioniert: Datenmodell, Client & Server, Datenbank, Authentifizierung, …
- hilft, der KI zu kommunizieren was man will
- hilft, mit **Informatiker*innen** zu reden
- Ziel ist, dass ihr als Gestalter*innen auch die Rolle der Projektleitung in Entwicklungsprozessen einnehmen könnt.

VIBE ENGINEERING MIT CLAUDE CODE

Vibe Coding

„Prompt, Paste & Pray“

- ein Prompt, die KI baut drauflos
- Prüfung: Läuft es?

-> gut für den Prototyp und kleine Features

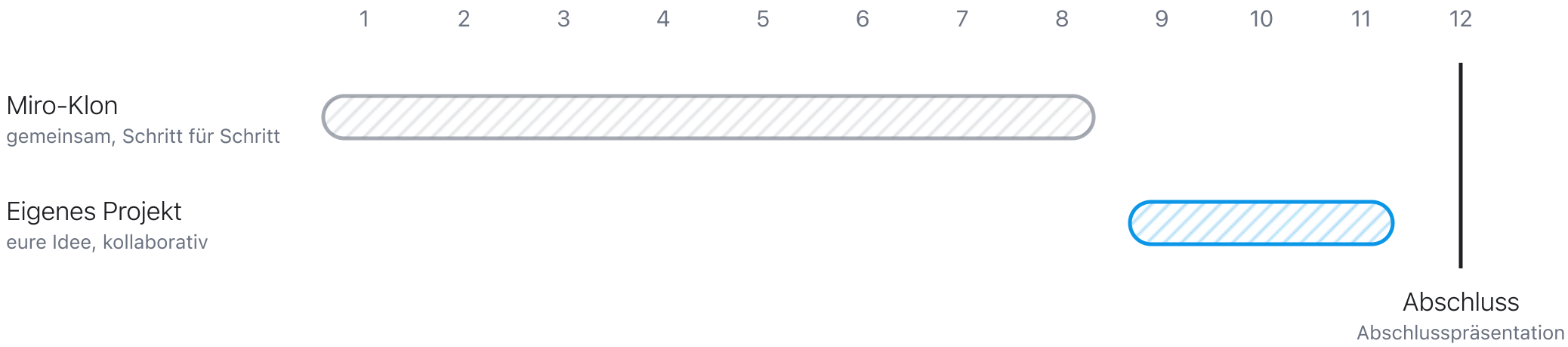
Vibe Engineering

„Plan, Orchestrate & Verify“

- ich plane, die KI setzt um
- man entwickelt ein gemeinsames Verständnis mit der KI
- die KI muss sich an klare Regeln halten
- Prüfung: Sind alle Anforderungen erfüllt?

-> für skalierbare Projekte

ABLAUF



→ Ziel: Das ganze Fach, inklusive Projekt, passt in die Vorlesungszeit.

WAS AM PROJEKT ZÄHLT

Präsentation

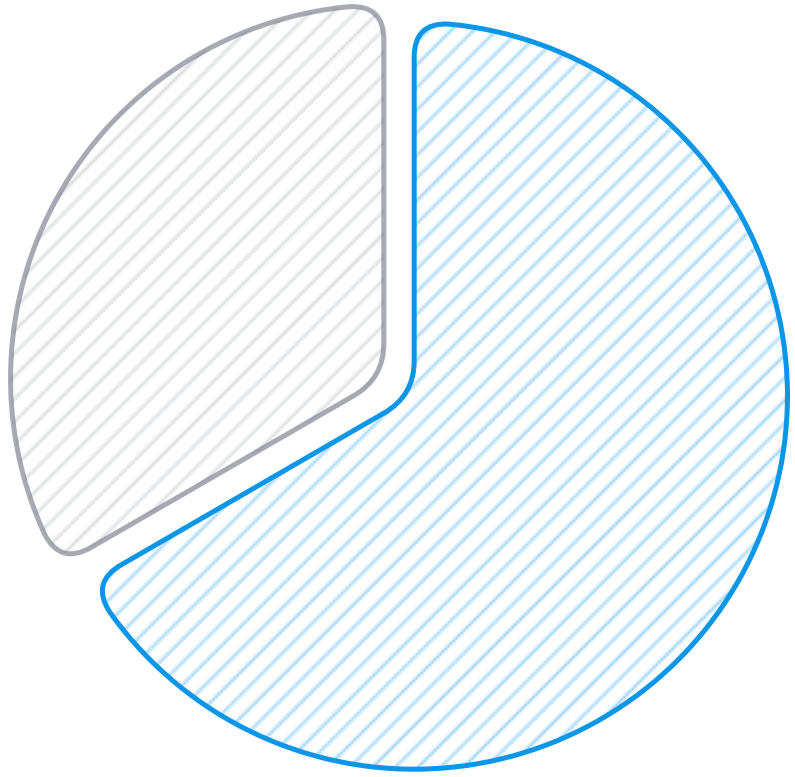
- ihr stellt das Projekt vor
- danach stellen wir Fragen

Dokumentation

- die Systemarchitektur: welche Teile, wie sie zusammenhängen, wie die Daten fließen

-> das fertige Ergebnis ist zweitrangig
-> was zählt, ist euer Systemverständnis und das wollen wir sehen.

BEWERTUNG



2/3
Endprojekt

1/3
Vorlesung

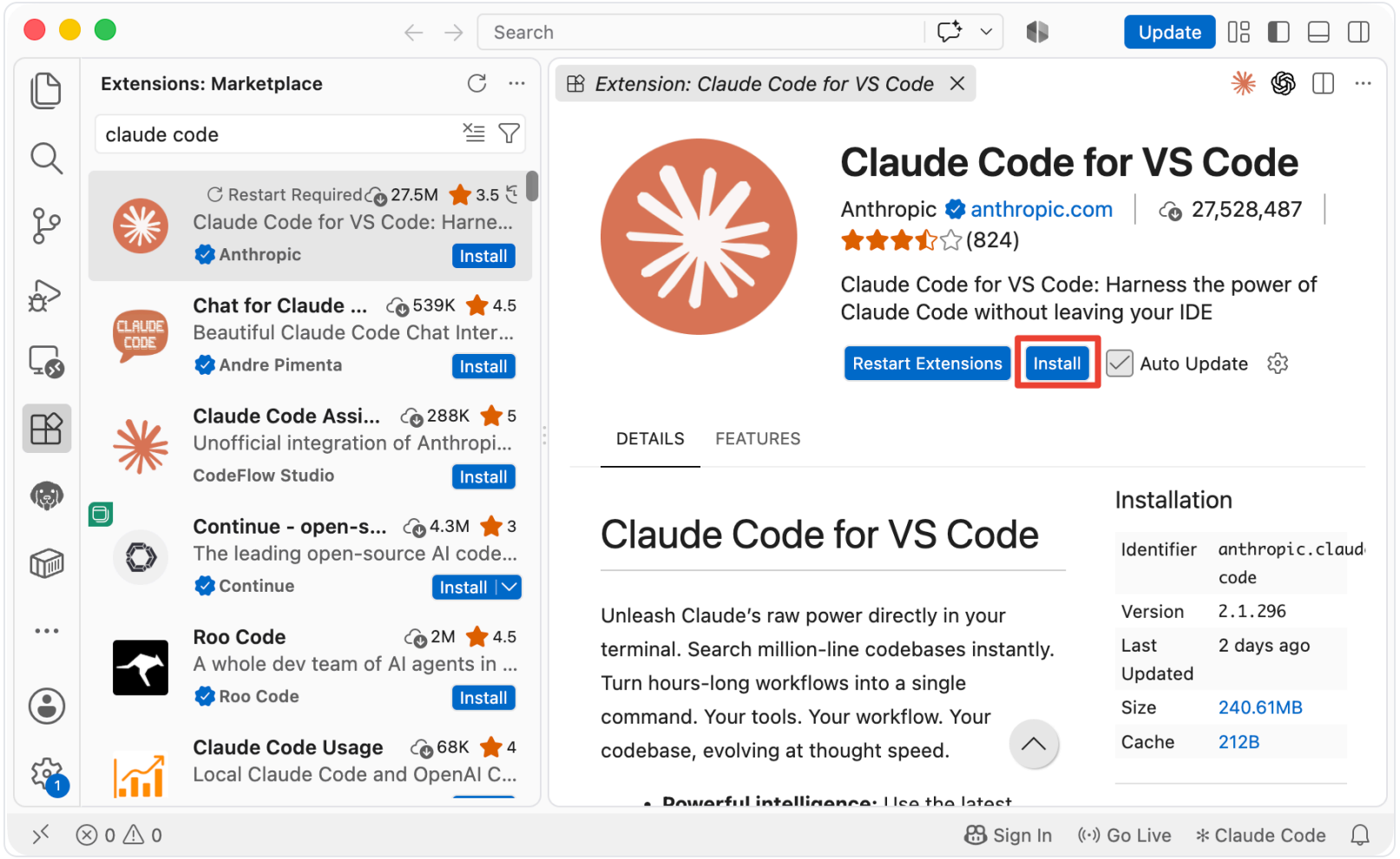
WAS IHR BRAUCHT

- VS Code ✓
- GitHub-Account ✓
- npm ✓
- ein Claude-Abo (Pro oder höher) für Claude Code

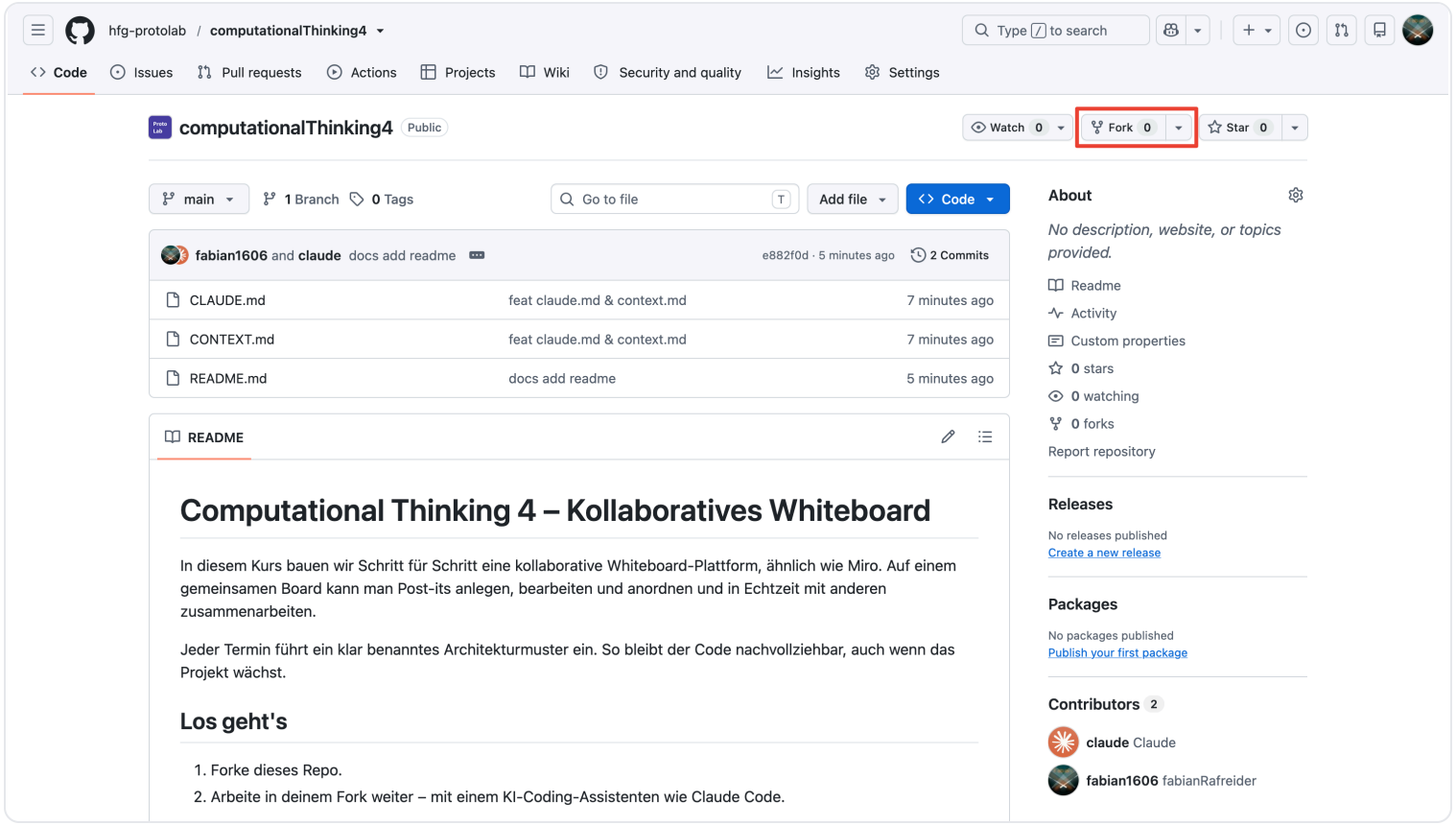
SETUP

1. Claude Code als Extension in VS Code installieren
2. Das Kurs-Repo forken
3. Den Fork klonen
4. Ordner in VS Code öffnen und Claude Code starten

CLAUDE CODE IN VS CODE



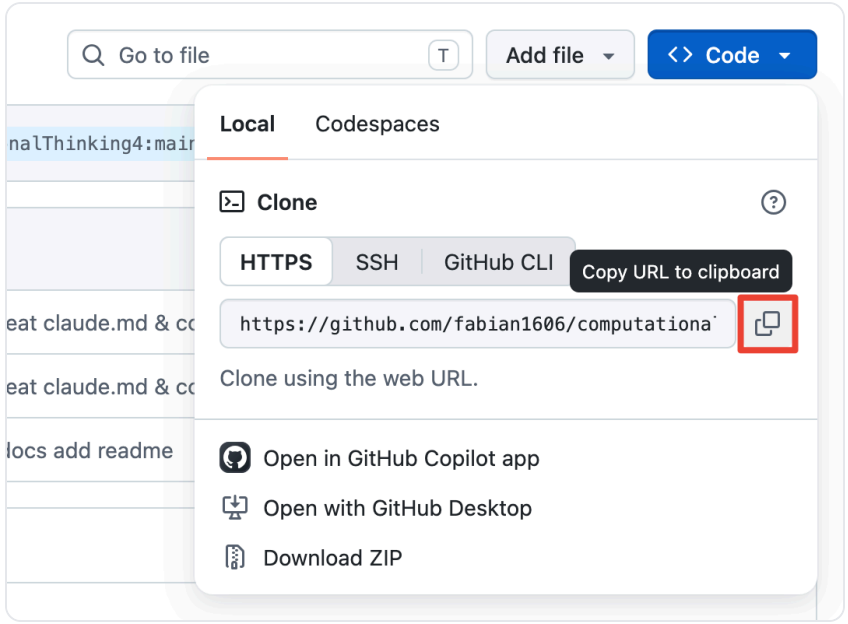
FORKT DAS REPO



github.com/hfg-protolab/computationalThinking4

EUREN FORK KLONEN

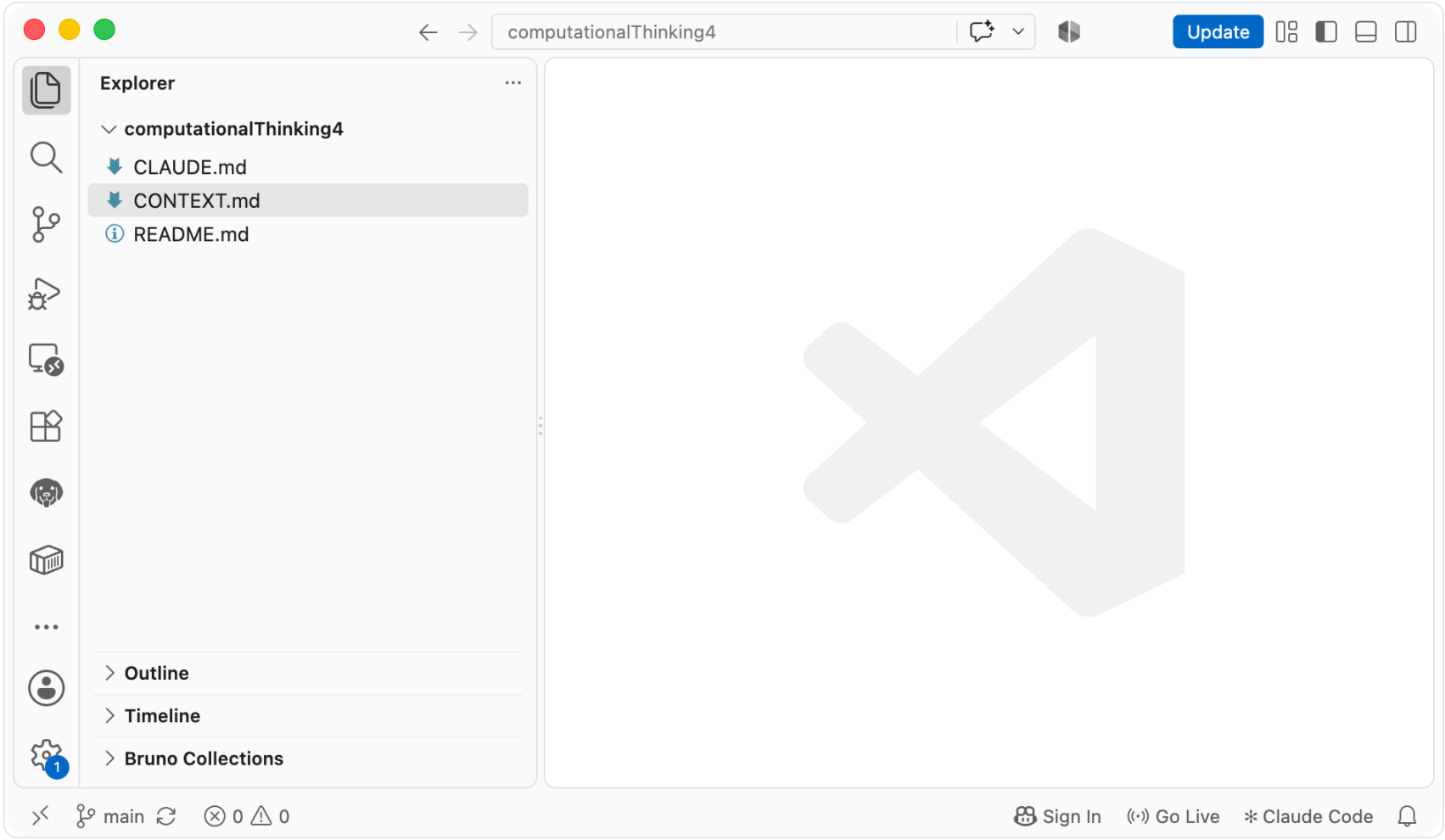
1. Im eigenen Fork auf **Code** klicken und den Link kopieren
2. Terminal öffnen:
3. Klonen — Link einfügen:




```
Terminal
```

```
git clone https://github.com/euer-name/computationalThinking4.git
```

WAS IM REPO LIEGT



CONTEXT.MD — WAS GEBAUT WIRD

- **Das Projekt:** ein kollaboratives Whiteboard wie Miro — Post-its anlegen, bearbeiten, anordnen, in Echtzeit zusammen
- **Das Setup:** Nuxt, Tailwind CSS mit shadcn-vue, ein Monorepo mit npm workspaces

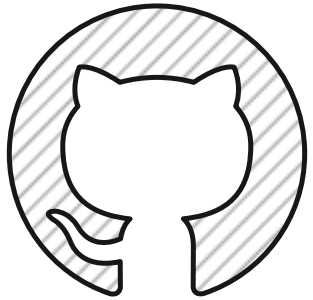
CLAUDE.MD — DIE REGELN FÜR DIE KI

Nach Andrej Karpathy: Die Regeln lösen die typischen Probleme, die KI beim Programmieren macht — falsche Annahmen, zu komplizierter Code, Änderungen nebenbei.

1. **Erst denken, dann coden** — Annahmen aussprechen, bei Unklarheit fragen
2. **Einfach bleiben** — nur, was verlangt ist, keine Datei über 200 Zeilen
3. **Gezielt ändern** — nur anfassen, was zur Aufgabe gehört
4. **Ziel festlegen, dann umsetzen** — vorher klären, woran man sieht, dass es fertig ist

WAS BAUEN WIR

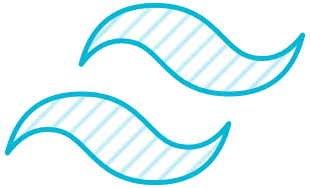
Bevor wir den ersten Prompt schreiben: Womit bauen wir — und warum?



Monorepo

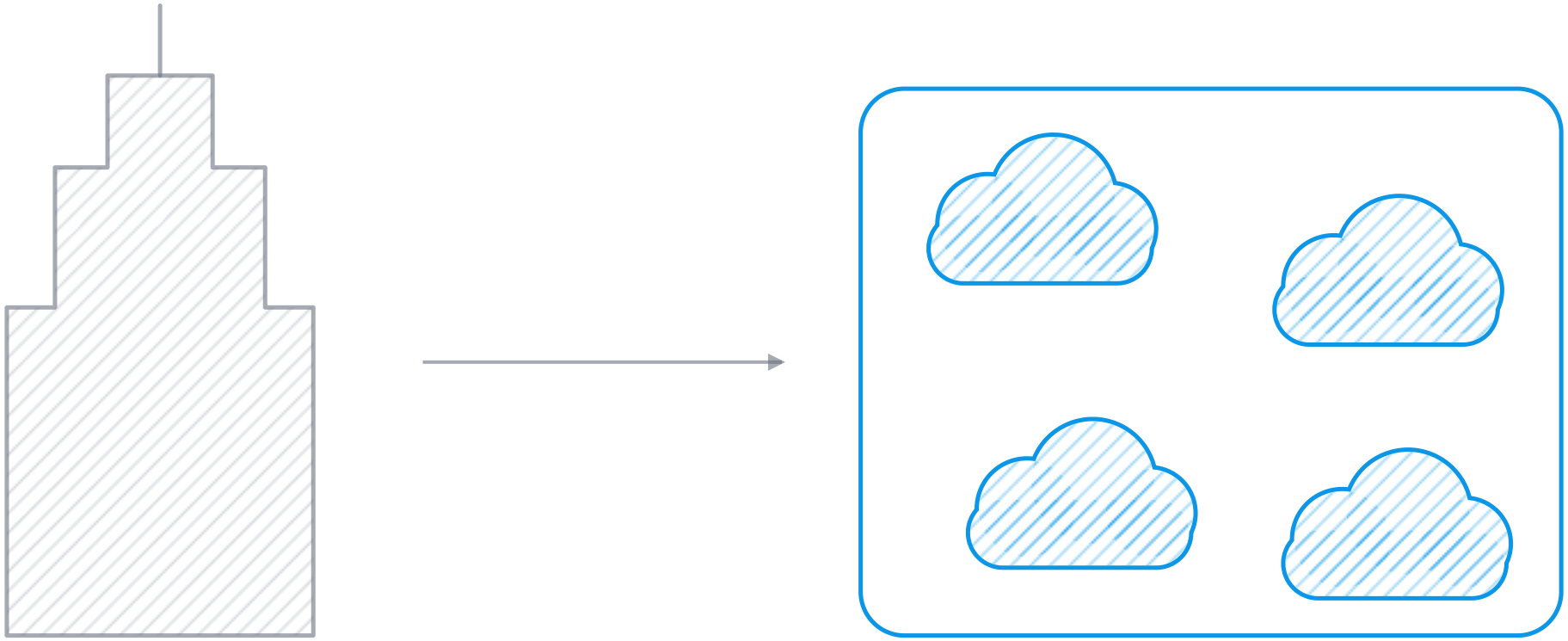


Nuxt



Tailwind CSS

MONOREPO



Monolith

alles ein Block

Monorepo

getrennte Dienste, ein Ort

NUXT



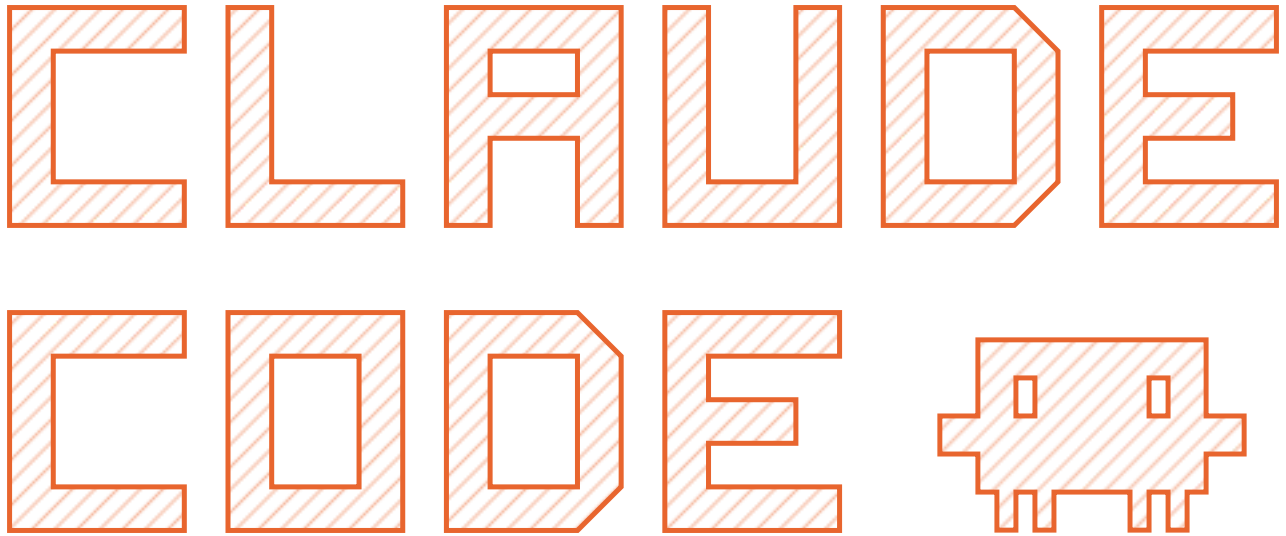
- ihr kennt **Svelte** — ein JavaScript-Framework
- **Nuxt** ist auch ein JavaScript-Framework
- Nuxt baut auf **Vue** auf — Vue ist der neue Standard in IG
- wir nehmen Nuxt statt nur Vue: bringt zusätzliche Features mit
- verbreitet in echten Produkten

TAILWIND CSS

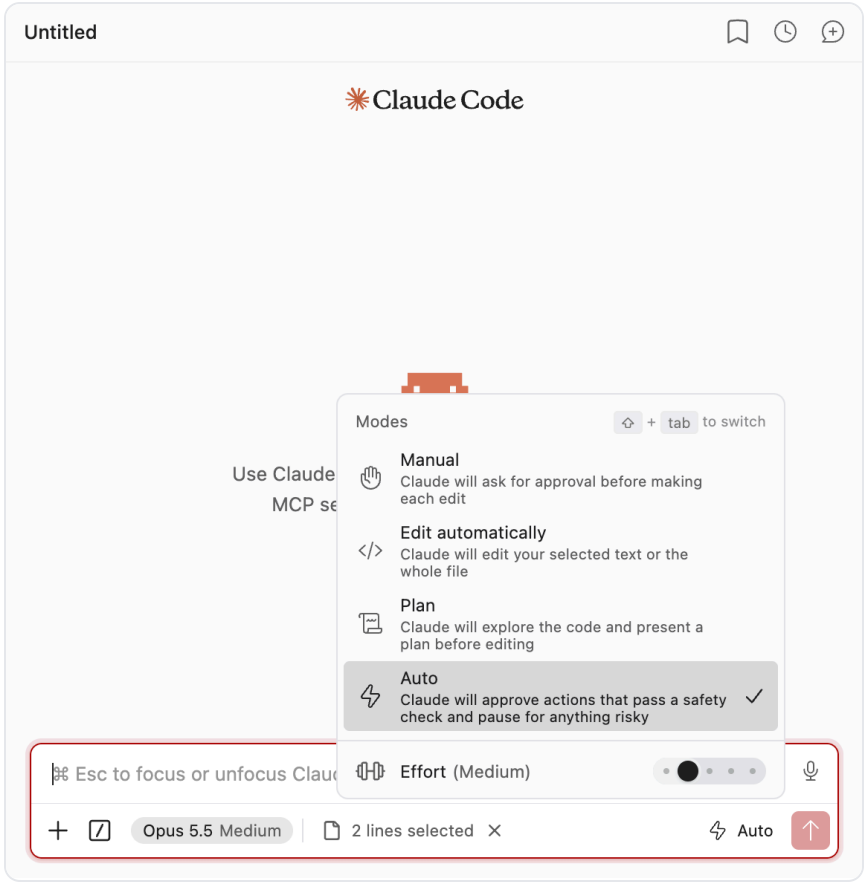


- **Tailwind CSS:** das Styling steht direkt im Markup, in kleinen Klassen
- hilft der KI, saubere und einheitliche Oberflächen zu bauen

WIE BAUEN WIR



CLAUDE CODE MODI



Unten im Eingabefeld: Wie viel darf Claude allein?

- **Manual** — fragt vor jeder Änderung und den meisten Befehlen
- **Edit automatically** — ändert Dateien, fragt nur bei neuen arten von Änderungen
- **Plan** — beschreibt erst, was es vorhat, und wartet auf euer Okay
- **Auto** — prüft selbst, was unbedenklich ist, und fragt nur bei Riskantem

Wir starten das Auto.

MODELL UND EFFORT

- **Modell:** welche Variante von claude man benutzt
- **Effort:** wie viel Claude nachdenkt, bevor es handelt — von low bis max

Wir nehmen Opus 5.5 mit Effort medium.

→ Lieber ein besseres Modell mit weniger Effort als ein schwächeres, das lange nachdenkt.

PROMPTING – INHALT VOR FORM

→ Das Wichtigste ist ein **gemeinsames Verständnis** mit der KI.

- kein „Du bist ein Nuxt-Spezialist …“
- stattdessen: alles an Kontext, was die KI braucht — lieber etwas zu viel als zu wenig
- in natürlicher Sprache, so wie ihr es einem Menschen erklären würdet



DER ERSTE PROMPT

Prompt

```
Platzhalter: hier kommt der erste Prompt rein.
```

DAS PROJEKT STARTEN

Im Terminal, im Projektordner (ganz oben, nicht in `frontend/`):

Terminal

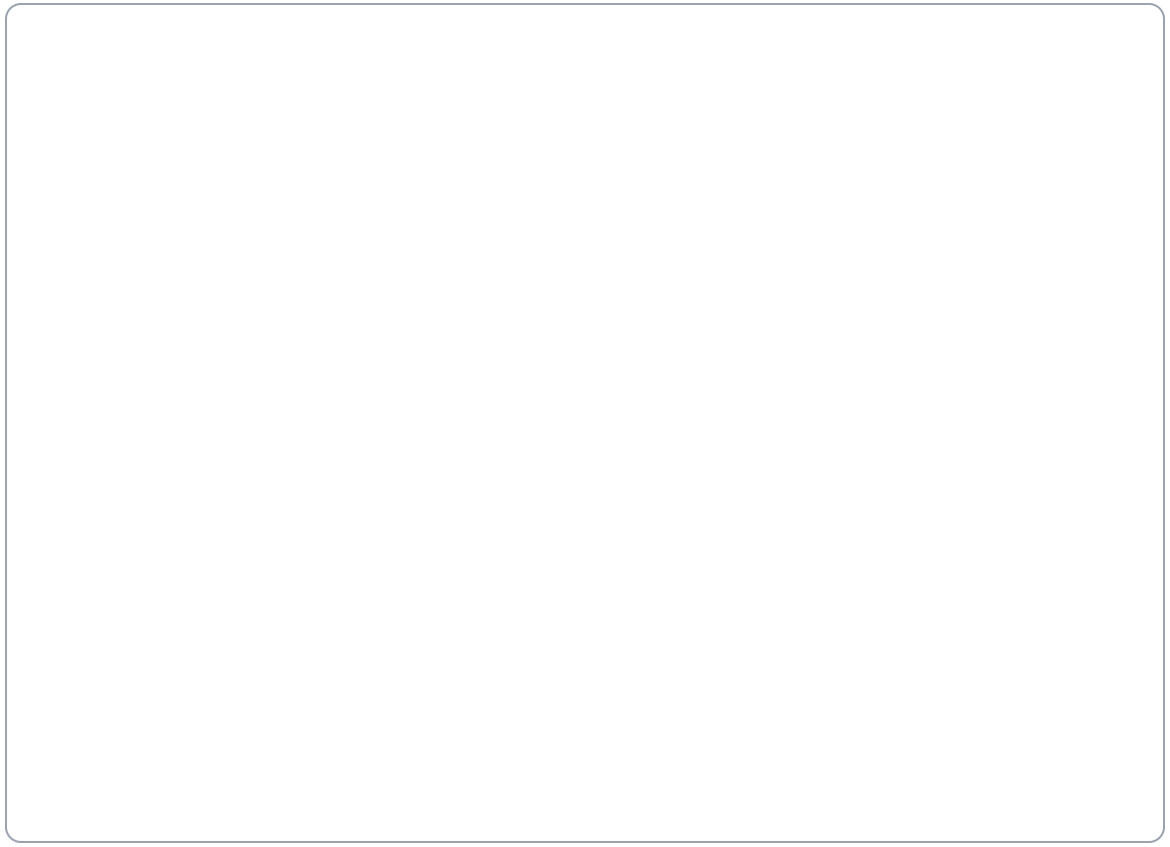
```
npm i
npm run dev
```

- `npm i` — installiert alle Pakete, nur beim ersten Mal oder nach Änderungen an der `package.json`
- `npm run dev` — startet das Frontend
- Im Browser öffnen: `http://localhost:3000`
- Beenden: `Ctrl` + `C` im Terminal

AUFGABE: EIN ENDLOSER CANVAS

- Ein endloser Canvas mit einem Post-it
- Post-it verschieben
- Es rastet im Raster ein
- Text lässt sich bearbeiten
- Post-it lässt sich skalieren

-> schreibt einen prompt löst das nicht mit einem screenshot



Ausprobieren: ziehen, Doppelklick zum Bearbeiten, Ecken zum Skalieren

WIE HAT ES GEKLAPPT?