ELEVATE X

Hiring Guide
Mai 2026 · v1.0

EXPERTS BY EXPERTS · HIRING PLAYBOOK

Agentic Engineering Interview Guide.

Sequenzieller Hiring-Leitfaden für Senior IT-Hires 2026. Vier Phasen, 21 Fragen, eine optionale Take-Home-Aufgabe. Prüft Workflow-Reife, nicht Whiteboard-Algorithmen.

45 min	4	21	+1
INTERVIEW-SLOT	PHASEN	KERNFRAGEN	TAKE-HOME

Für CTOs, Engineering-Leads und Hiring-Manager, die IT-Freelancer und Senior-Engineers screenen, aber Vibe-Coding-Reflexe früh erkennen wollen.

ELEVATEX.DE

WARUM DIESER GUIDE EXISTIERT

Dieses Format prüft keine Algorithmen. Es prüft Workflow-Reife.

Wer 2026 noch Whiteboard-Aufgaben stellt, testet darauf, ob der Kandidat 2019 ein guter Engineer gewesen wäre — nicht darauf, ob er heute agentic operiert. Genau deshalb arbeitet dieser Guide mit Workflow-Erzählungen, Live-Task-Simulationen an realer Codebase und Selbstreflexions-Fragen — nicht mit isolierten Logik-Puzzles.

> *Most people have still not refactored their hiring process for agentic engineering capability. If you're giving out puzzles to solve, this is still the old paradigm.*
>
> ANDREJ KARPATHY · SEQUOIA AI ASCENT 2026

INHALTSVERZEICHNIS

Phase 01 Workflow-Erzählung

Wie sieht ein typischer Tag aus?

Sieben Fragen über Daily-Driver, Tools, Subscriptions, Skills und Review-Setup. Ohne Screen-Sharing, ohne Live-Task. Hier zeigt sich, ob der Kandidat den Workflow lebt — oder ihn nur kennt.

ERÖFFNUNGSFRAGE

„Erzähl mir, wie ein typischer Arbeitstag bei dir aussieht.

DAUER	FRAGEN	SETUP NÖTIG
15 Minuten	1 – 7	Keines

Phase 1 ist der Eisbrecher. Kandidaten kommen hier in Rhythmus und du bekommst die meisten Differenzierungs-Signale, ohne dass technisches Setup nötig ist. Wer in Phase 1 schwächelt, wird in der Live-Task selten überraschen.

FRAGE 01 Wie viel Code schreibst du an einem normalen Tag noch händisch?

GREEN FLAGS

- Konkrete Zahl: *5–15 Prozent*
- Erklärt, *welche* Art von Code er noch händisch schreibt — Architektur-Entscheidungen, kritische Business-Logik
- Erwähnt Karpathys 80/20→20/80-Flip oder ähnliches Konzept
- Hat eine bewusste Routine entwickelt, nicht zufällige Praxis

RED FLAGS

- *„Schon noch viel"* oder *„Den Großteil"* (über 50%)
- Kann keine Zahl nennen
- Sagt 0% — zeigt Vibe-Coding-Reflex ohne Kontrolle
- Versteht die Frage nicht oder hält sie für unwichtig

FRAGE 02 Welches Subscription-Tier fährst du auf deinem Daily-Driver?

GREEN FLAGS

- Max 5x oder Max 20x bei Anthropic, Pro+ oder Ultra bei Cursor, oder direkter API-Zugang
- Nennt einen ungefähren Monatsverbrauch (über 100 Dollar)
- Hat schon mal die Quotas gesprengt
- Begründet die Tier-Wahl mit konkretem Workload

RED FLAGS

- Pro für 20 Dollar — *nicht senior in agentic 2026*
- *„Das zahlt mein Arbeitgeber"* ohne weitere Auskunft
- Weiß nicht, was er bezahlt
- Hat noch nie einen Quota-Hit erlebt

FRAGE 03 Beschreib deinen Workflow für ein neues Feature. Schritt für Schritt — was machst du als Erstes?

GREEN FLAGS

- Erwähnt Spec oder Tests *vor* dem Coden — aber iterativ, nicht wasserfall-artig
- Nutzt explizit Plan-Mode (oder Equivalent)
- Beschreibt Review zwischen Phasen, nicht erst am Ende
- Erwähnt parallele Agents oder Sub-Agents für Reviews
- Hat eigene Faustregel für Spec-Detail (z.B. „Spec für 1–2 Tage Arbeit")

RED FLAGS

- Springt direkt zu *„Dann frag ich Claude, das zu bauen"* (Vibe-Coding-Reflex)
- Verbringt 60+ Minuten allein mit Spec, bevor der erste Agent läuft (Wasserfall-Reflex)
- Kein expliziter Plan-Schritt
- Akzeptiert AI-Output ohne Diff-Review
- Test wird (wenn überhaupt) erst nach dem Code geschrieben

FRAGE 04 Welche Skills oder Slash-Commands hast du in deinem aktuellen Projekt selbst geschrieben? Hast du ein Pattern dafür?

BEZUGSPUNKT — HOWIE LIU, AIRTABLE-GRÜNDER

Skills sind *„das wichtigste Primitiv"* der frontier-agents-Welt. Sein Frame: Stell dir Albert Einstein vor — generell hochintelligent. Er weiß nichts über Real Estate. Aber wenn du ihm das richtige Briefing gibst, eine Playbook, ein Manual, dann geht er hin und löst es. Skills sind genau dieses Briefing. Wer Skills nur als Boilerplate-Sammlung versteht, hat das Konzept nicht erfasst.

GREEN FLAGS

- Mindestens 1–2 selbst geschriebene Skills
- Hat ein *strukturiertes Pattern* für Skills (z.B. Rules + Checklist + Guide pro Topic)
- Kann konkret beschreiben, welches Problem ein Skill löst
- Erwähnt Wiederverwendung über Projekte hinweg
- Versteht Skills auch als *Onboarding-Werkzeuge*
- Versteht Skills als *Kontextualisierungs-Layer* für generell-intelligente Modelle

RED FLAGS

- *„Bisher nicht gebraucht"*
- Nutzt nur Default-Skills oder heruntergeladene
- Hat 1–2 Skills, aber kein erkennbares Pattern
- Versteht das Konzept nicht
- Kann das letzte Mal nicht benennen, wann er einen Skill geschrieben hat
- Beschreibt Skills als *„Prompt-Sammlung"* oder *„Snippet-Library"*

FRAGE 05 Welche Daten oder Secrets dürfen niemals in den Agent-Kontext? Wie stellst du das sicher?

GREEN FLAGS

- Klare Liste: Production-Credentials, Customer-Daten, Infrastructure-Code (Terraform, Helm), PII
- Kennt Pre-Tool-Hooks oder vergleichbare Mechanismen, um den Agent von kritischen Verzeichnissen abzuhalten
- Hat eine bewusste Trennung zwischen Code und Infrastructure
- Erkennt das Adversarial-Verhalten von Agents — und plant dafür
- Erwähnt *.gitignore*-Disziplin, separate Test-Datenbanken, Maskierung in Logs

RED FLAGS

- *„Da achte ich nicht so genau drauf"*
- Hat Secrets schon mal aus Versehen in Prompts gehabt
- Kennt Pre-Tool-Hooks oder vergleichbare Schutzmechanismen nicht
- Vermischt Infrastructure und Code-Logik im selben Agent-Kontext
- *„Das macht doch der Agent automatisch sicher"*

FRAGE 06 Wenn der Agent fertig ist mit der Implementierung — wie sieht dein Review-Setup aus?

GREEN FLAGS

- Mehrere Review-Schichten mit klaren Rollen: zweiter Agent als Reviewer, PR-Bot wie GitHub Copilot, finaler menschlicher Review
- Kann erklären, warum welche Schicht was tut (z.B. *„Codex findet strukturelle Inkonsistenzen, Mensch checkt fachliche Logik"*)
- Versteht klar: AI ist gut bei strukturellen Pattern-Checks, Mensch bei Business-Logik und Kontext
- Hat eine bewusste Heuristik dafür, wann ein zweiter Agent-Review läuft und wann nicht

RED FLAGS

- *„Ich gucke einmal drüber"*
- Nur ein Agent macht alles — Implementierung und Review im selben Lauf
- Kennt das Konzept *„AI reviewt AI"* nicht
- Hält Review für überflüssig, wenn der Agent gute Tests geschrieben hat

FRAGE 07 Was ist deine Standardanweisung an die AI, bevor du eine Implementierung startest?

GREEN FLAGS

- Hat eine klare Standardanweisung — z.B. *„die AI implementiert nicht selbst, sie schlägt 2–3 Optionen vor, ich wähle"*
- Nutzt visuelle Hilfsmittel: Ablaufdiagramme, Sequence-Diagrams, Mermaid-Charts erzeugen lassen, bevor er Code akzeptiert
- Kann konkret zeigen, wie er den Agent zur *Optionen-Generierung* anstößt, statt zur Direkt-Implementierung
- Hat eigene Templates oder Prompt-Snippets, die er regelmäßig wiederverwendet

RED FLAGS

- *„Ich sag halt, was ich brauche, und dann macht er das"*
- Akzeptiert die erste Implementierung blind
- Lässt sich nie Optionen geben, nur fertige Lösungen
- Kennt visuelle Outputs (Diagramme) nicht als Tool

Phase 02 Setup-Demo + Live-Task

Zeig mir dein Setup.

Drei Setup-Fragen, dann eine echte Mini-Aufgabe an realer Codebase. Wer den Workflow lebt, zeigt es in den ersten 60 Sekunden. Wer ihn nur kennt, stolpert hier.

ERÖFFNUNGSANWEISUNG

„Jetzt würde ich gerne dein Setup sehen. Mach kurz Screen-Sharing — Terminal, IDE, AI-Tools, was du normal offen hast. Drei kurze Fragen dazu, dann gehen wir in die eigentliche Aufgabe.

DAUER	FRAGEN	SETUP NÖTIG
20 – 25 min	8 – 11	Screen-Share

Phase 2 ist der Praxis-Beleg. Setup-Demo deckt Daily-Driver, Konfiguration und parallele Agent-Instanzen auf. Anschließend folgt eine 10–15-Min Live-Task aus vier Profilen (Bug-Fix, Refactor, Greenfield, oder Take-Home Adversarial).

FRAGE 08 Welche AI-Tools laufen aktuell auf deinem System, und welches ist dein Daily-Driver?

GREEN FLAGS

- Nennt 2–3 Tools spezifisch (Claude Code + Codex CLI + Cursor)
- Hat einen klaren Daily-Driver mit Begründung
- Erwähnt parallele Nutzung statt eines einzigen Tools
- Erklärt, warum er Tool A für Aufgabe X und Tool B für Aufgabe Y nutzt

RED FLAGS

- *„Ich nutze halt ChatGPT"* ohne weitere Differenzierung
- Kennt nur ein Tool
- Nennt Copilot als primäres agentic Tool — zeigt 2024er-Stand
- Kann keine Tool-Wahl begründen

FRAGE 09 Zeig mir deine `CLAUDE.md` oder `AGENTS.md`. Wie lang ist sie, was steht drin?

GREEN FLAGS

- Datei existiert und ist sichtbar
- Mindestens 100 Zeilen, eher mehr
- Enthält Architektur-Hinweise, Tool-Konfiguration, Coding-Standards
- Wird regelmäßig aktualisiert (Version-History sichtbar oder erkennbar)
- Erwähnt eigene Skills, Slash-Commands oder Pre-Tool-Hooks

RED FLAGS

- *„Hab ich nicht"* oder *„Hab ich noch nicht angefangen"*
- Datei ist 5–10 Zeilen oder ein Standard-Template
- Datei wurde seit Wochen nicht angefasst
- Der Kandidat weiß nicht, wozu die Datei da ist

FRAGE 10 Lass mich sehen, wie du eine zweite oder dritte Agent-Instanz öffnest. Wie sieht dein paralleles Arbeitssetup aus?

GREEN FLAGS

- Öffnet 2–3 Instanzen in unter 60 Sekunden
- Hat ein etabliertes Setup: Terminal-Grid, mehrere CLI-Sessions oder Git-Worktrees
- Kennt *Worktrees* als Konzept und hat sie aktiv getestet
- Kann erklären, wann er parallel arbeitet und wann nicht
- Erwähnt Konflikt-Vermeidung (separate Branches, klar abgegrenzte Tasks)

RED FLAGS

- Hat nur eine Instanz, weiß nicht wie man eine zweite öffnet
- *„Brauche ich nicht"* ohne Begründung
- Lange Setup-Zeit deutet auf seltene Nutzung hin
- Kennt Worktrees nicht oder hält sie für Spezial-Wissen — klarer Senior-Disqualifikator 2026

LIVE-TASK-SIMULATION · 10 – 15 MIN

Du löst sie mit deinem normalen Setup.

„Ich beobachte und stelle eine begleitende Frage. Wähle eine der Aufgaben unten, je nach Kandidaten-Profil." — Diese Aufgaben sind *bewusst keine isolierten Algorithmen oder Whiteboard-Puzzles*. Sie sind realistische Mini-Projekte mit Mehrdeutigkeit, Kontext-Bedarf und Architektur-Entscheidungen — genau das, was 2026 zählt.

VIER AUFGABEN — AUSWAHL JE NACH PROFIL

AUFGABE	FOKUS	WANN NUTZEN
A — Bug-Reproduktion mit Fix 10 MIN	Test-Driven-Bug-Fixing, Plan-Mode-Disziplin, Review-Verhalten	Standard-Default für Backend-Senior-Hires
B — Refactor mit Spec 10 MIN	Spec-Disziplin, Akzeptanz-Test, Diff-Review	Frontend / Mobile-Senior-Hires
C — Greenfield-Feature mit Architektur 10 MIN	Klärende Fragen, Architektur-Entscheidungen, Out-of-Scope-Definition	Tech-Lead-Hires oder wenn Architektur-Stärke zentral ist
D — Adversarial Take-Home 60 MIN	Build-und-Break: Kandidat baut sicher, Agents versuchen zu brechen	Senior-Architect oder Security-relevante Hires; nur als Take-Home

A Bug-Reproduktion mit Fix

> *„Ich gebe dir eine kleine API mit zwei Endpoints. User berichten: beim dritten Page-Aufruf der Pagination kommen manchmal die gleichen Items zurück. Reproduziere den Bug, finde die Ursache, schreib einen Fix mit passendem Test. 10 Minuten."*

WAS DU BEOBACHTEST

- → Schreibt er *zuerst einen Reproduktions-Test*, der den Bug fängt? Oder springt er direkt in den Code?
- → Wie nutzt er den Agent — als Reproduktions-Helfer (gut) oder direkt als Fix-Generator (Red Flag)?
- → Hat er Plan-Mode genutzt, bevor er etwas geändert hat?
- → Liest er den Test-Output gründlich oder akzeptiert er das erste grüne Signal?

B Refactor mit Spec

> *„Hier ist eine View-Komponente mit etwa 220 Zeilen, gewachsener Code aus mehreren Iterationen. Refaktor ihn so, dass die Business-Logik vom View getrennt ist und Tests möglich werden — ohne dass die UI sich verändert. Schreib zuerst kurz auf, was du als Ziel definierst, dann arbeite es um. 10 Minuten."*

WAS DU BEOBACHTEST

- Schreibt er zuerst eine knappe Spec für das Refactor-Ziel (1–2 Sätze, was sich verändert, was nicht)?
- Definiert er einen Akzeptanz-Test (z.B. *„alle bestehenden Snapshot-Tests müssen weiter laufen"*)?
- Lässt er den Agent *Optionen vorschlagen*, oder direkt implementieren?
- Wie geht er mit der *Spec-Wasserfall-Falle* um — kommt er nach 2 Minuten ins Tun, oder bleibt er 8 Minuten in der Spec?
- Reviewt er Diffs vor Akzeptanz?

C Greenfield-Feature mit Architektur-Entscheidung

> *„Wir bauen eine kleine Banking-App. Du sollst ein einfaches Feature anlegen: Nutzer können Rechnungs-PDFs per Email an eine projektspezifische Adresse weiterleiten — die App liest sie aus und legt sie als anstehende Zahlungen ab. Wie würdest du das Datenmodell und die Service-Architektur aufsetzen? Implementier dann das Datenmodell und einen ersten Service-Stub. 10 Minuten."*

WAS DU BEOBACHTEST

- Stellt er *klärende Fragen, bevor er anfängt* (Kontext-Bewusstsein vs. Vibe-Coding-Reflex)?
- Lässt er sich vom Agent 2–3 Architektur-Optionen vorschlagen, bevor er entscheidet?
- Trennt er sauber zwischen *Datenmodell-Entscheidung (sein Job)* und *Boilerplate-Generierung (Agent-Job)*?
- Erkennt er, was *explizit nicht implementiert werden soll* in 10 Minuten — und definiert er es als Out-of-Scope?

D Adversarial Take-Home (60 Minuten)

BEZUGSPUNKT — ANDREJ KARPATHY, SEQUOIA AI ASCENT 2026

„Hiring has to look like — give me a really big project and see someone implement that big project. Like let's write a Twitter clone for agents and then make it really good, make it really secure. And then I'm going to use 10 codex agents to try to break your website. They should not be able to break it."

„Bau in 60 Minuten ein kleines, klar abgegrenztes System mit deinem normalen Setup — z.B. eine API mit Authentifizierung, Rate-Limiting und einer Persistenz-Schicht. Mach es so sicher wie möglich. Nach Ablauf der Zeit setze ich mehrere Agents auf das System an, die versuchen, es zu brechen — Auth umgehen, Race-Conditions ausnutzen, Validation-Lücken finden."

WÄHREND DES BUILDS

- Baut er *defensiv von Anfang an*, oder kommt Security erst später?
- Nutzt er Agents für Security-Reviews während des Builds, nicht nur für Implementation?
- Spezifiziert er Threat-Models explizit, bevor er codet, oder bleibt das implizit?
- Hat er eigene Skills oder Patterns für Rate-Limiting, Input-Validation, Auth-Hardening?

IN DER ADVERSARIAL-PHASE

- Wie reagiert er, wenn Lücken gefunden werden? Defensiv-defensiv, oder analytisch?
- Kann er erklären, *warum* eine bestimmte Lücke entstanden ist?
- Patcht er nur das Symptom oder die zugrundeliegende Klasse von Problemen?

Hinweis zur Anwendung: Diese Aufgabe ist nicht für 10-Minuten-Live-Slots geeignet — sie ist ein Take-Home oder ein 60–90-Minuten-Onsite-Slot mit anschließendem Debrief. Sie ist die schärfste verfügbare Hiring-Probe für Senior-Architects und Security-relevante Rollen, weil sie Build und Break in einem Setup vereint.

FRAGE 11 Während der Arbeit: Wie weißt du gerade, dass der Agent auf dem richtigen Weg ist?

GREEN FLAGS

- Liest tatsächlich den Plan oder die Diffs
- Hat klare Zwischenchecks (Tests laufen lassen, Code-Read)
- Stoppt aktiv, wenn etwas nicht passt
- Vertraut dem Agent nicht blind, auch nicht bei einfachen Tasks

RED FLAGS

- *„Ich schau am Ende"* ohne Zwischenchecks
- Verlässt sich nur auf grüne Tests
- Kann nicht erklären, woran er Fehlentwicklung erkennen würde

Phase 03 Fehler-Erzählung

Wann hat der Agent Unsinn gebaut?

Der ehrlichste Teil. Wer hier glatt durchgeht ohne Selbstreflexion, ist entweder zu wenig praktiziert oder hat Dunning-Kruger.

ERÖFFNUNGSFRAGE

„Erzähl mir konkret von einem Moment, in dem der Agent kompletten Unsinn gebaut hat — etwas, das du bemerkt hast. Wann war das, was war die Aufgabe?

DAUER	FRAGEN	MODUS
10 Minuten	12 – 15	Erzählend

Vier Fragen über Fehlentwicklungen, zurückgezogene PRs, Nein-Sagen gegenüber der AI und Approve-Drift bei großen PRs. Konkrete Stories statt Theorie — wer keine konkreten Fälle benennen kann, hat den Workflow nicht in der Praxis.

FRAGE 12 Wie hast du gemerkt, dass es schiefgeht?

GREEN FLAGS

- Konkrete Story mit Kontext (Projekt, Aufgabe, Symptom)
- Erkannt durch Test, Code-Read oder Logik-Check, *nicht* erst in Production
- Kann den Auslöser benennen
- Hat eine Heuristik daraus gebaut

RED FLAGS

- *„Sowas passiert mir nicht"* — zu wenig Praxis oder keine Selbstreflexion
- Story ist vage und ohne Details
- Erst durch Production-Bug bemerkt, nicht im Review
- Schiebt Schuld auf das Tool, ohne eigene Verantwortung

FRAGE 13 Hast du schon mal einen eigenen Pull Request zurückgezogen, weil du nachträglich gemerkt hast, dass du den Agent-Output nicht wirklich verstanden hattest?

GREEN FLAGS

- Ja, klare Story dazu
- Kann erklären, *warum* das Verständnis erst später kam
- Hat eine Konsequenz daraus gezogen (anderer Workflow, mehr Reviews)
- Zeigt Selbstreflexion ohne Schuldzuweisung

RED FLAGS

- *„Nein, nie."* — entweder zu wenig Praxis oder Dunning-Kruger
- Story ist gestellt oder unkonkret
- Sieht das nicht als Problem
- Verteidigt blindes Akzeptieren

FRAGE 14 Wann hast du der AI das letzte Mal *Nein* gesagt? Was hat sie vorgeschlagen, was du abgelehnt hast?

GREEN FLAGS

- Konkretes Beispiel, in den letzten Tagen oder Wochen
- Begründung war architektonisch oder fachlich (nicht stilistisch)
- Hat das im Anschluss in eine Skill-Regel oder eine *CLAUDE.md*-Notiz übersetzt
- Versteht *Nein-sagen* als zentrale Senior-Verantwortung — nicht als Ausnahme
- Erkennt überschießende Proaktivität als typisches Agent-EQ-Failure und hat eigene Heuristiken dagegen

RED FLAGS

- *„Eigentlich akzeptiere ich meistens, was er macht"*
- Kann sich nicht an einen konkreten Fall erinnern
- Begründet Nein-sagen nur mit Stil-Präferenz
- Versteht die Frage als Test seines Vertrauens in die AI, nicht als Test seiner Verantwortung
- Hat keine Wahrnehmung für *„Agent macht etwas technisch Korrektes, das sozial oder strategisch falsch ist"*

FRAGE 15 Wie schützt du dich heute vor Approve-Drift bei größeren PRs?

GREEN FLAGS

- Kennt den Begriff oder das Konzept
- Hält PRs bewusst klein
- Nutzt automatische Review-Agents als erste Filterschicht
- Nutzt *Issue-Forking-Pattern*: wenn ein neues Topic in einer Agent-Session aufkommt, lässt er den Agent eine separate Issue erstellen, statt den aktuellen PR aufzublähen
- Hat eine Größengrenze für PRs, die er nicht überschreitet

RED FLAGS

- Versteht das Risiko nicht
- *„Ich gucke halt drüber"* ohne strukturelle Maßnahme
- Hat keine Größengrenze für PRs
- Verlässt sich allein auf menschliche Reviewer

Phase 04 Kalibrierung & Signal-Check

Wo ziehst du Grenzen?

Sechs Kalibrierungsfragen — nicht um zu prüfen, ob der Kandidat etwas weiß, sondern um zu sehen, wo er Grenzen zieht.

ERÖFFNUNGSANWEISUNG

„Letzte Phase. Ich stelle dir ein paar Kalibrierungsfragen — nicht um zu prüfen, ob du etwas weißt, sondern um zu sehen, wo du Grenzen ziehst.

DAUER	FRAGEN	MODUS
10 Minuten	16 – 21	Reflektierend

Hier liegen die schärfsten Senior-Tests des Guides. Frage 17 (AI vs. menschlicher Reviewer) ist der schärfste Senioritäts-Test — wer hier schwarz-weiß antwortet, ist nicht senior. Frage 19 (Persistence-Curation) ist der versteckteste — Liu nennt das Aufgeben nach dem ersten Versuch den häufigsten Fehler von Praktikern.

FRAGE 16 Wo vertraust du der AI komplett? Wo überhaupt nicht?

GREEN FLAGS

- Klare Liste, was er delegiert: Boilerplate, Test-Generierung in klaren Modulen, Refactoring, Bug-Reproduktion, Dokumentation
- Klare Liste, was er nicht delegiert: kritische Business-Logik, architektonisches Fundament, Security-Sensitives, regulatorische Bereiche
- Hat eine Heuristik, nicht nur eine Bauchregel
- Räumt ein, dass die Grenze sich verschiebt

RED FLAGS

- *„Eigentlich allem"* — Vibe-Coding-Reflex
- *„Nichts wirklich"* — 2024er-Mindset
- Kann keine Beispiele nennen
- Schwarz-weiß ohne Differenzierung

FRAGE 17 ★ KERN-SENIORITÄTS-TEST Wo ist die AI besser als ein menschlicher Reviewer? Wo eindeutig schlechter?

GREEN FLAGS

- *Besser:* strukturelle Konsistenz über viele Files, schnelles Erkennen von Pattern-Brüchen, Security-Patterns, fehlende Edge-Cases in Tests, blinde Stellen bei Refactoring
- *Schlechter:* fachlicher Kontext (warum wurde diese Entscheidung damals getroffen?), Business-Logik-Korrektheit, Team-Konventionen, langfristige Wartbarkeit jenseits des aktuellen Files
- Hat eine differenzierte Haltung — weder „AI ersetzt den Reviewer" noch „AI bringt nichts"
- Nutzt beide Stärken bewusst

RED FLAGS

- *„AI ist überall besser"* — Vibe-Coding-Reflex
- *„Ich vertraue lieber Menschen"* — 2024er-Mindset
- Kann keinen Bereich nennen, in dem AI eindeutig stark ist
- Kann keinen Bereich nennen, in dem AI eindeutig schwach ist

FRAGE 18 Welche Best Practice aus 2024 hältst du heute für falsch?

GREEN FLAGS

- Konkrete Praxis genannt (z.B. *„Tests schreiben ist teuer"*, *„PRs sollen klein sein, weil Menschen sie reviewen"*, *„Code-Comments sind Pflicht"*, *„Ein-Aufgabe-pro-Branch-Regel"*)
- Begründung, warum sich das verschoben hat
- Hat seine eigene Praxis aktiv geändert

RED FLAGS

- *„Eigentlich passt alles noch"*
- Generic-Antwort ohne Substanz
- Versteht die Frage nicht
- Lehnt die Frage als spekulativ ab

FRAGE 19 ★ VERSTECKTER TEST Erzähl mir von einem Workflow, den du nach mehreren Anläufen zum Funktionieren gebracht hast. Was war dein Stehvermögen?

BEZUGSPUNKT — HOWIE LIU, AIRTABLE-GRÜNDER

Der häufigste Fehler von Praktikern: Sie one-shotten etwas, es ist nicht ganz so tief wie erhofft, und sie geben auf. Die Agents sind mächtig genug, um fast alles zu lösen, was du willst. Die Frage ist, ob du Zeit, Coaching und Curation investierst, um sie dorthin zu bringen. *Persistence-Curation* ist 2026 ein eigenes Senior-Skill — und nicht jeder hat es.

GREEN FLAGS

- Konkrete Story: Workflow X hat beim ersten Versuch nicht funktioniert, beim zweiten besser, beim dritten gut
- Kann beschreiben, *welche Iterationen* nötig waren
- Hat eine eigene Schwelle dafür, wann er ein Pattern weiterverfolgt und wann er es aufgibt
- Versteht das Investment in Coaching und Curation als zentrale Senior-Arbeit

RED FLAGS

- *„Wenn es beim ersten Versuch nicht klappt, ist das Tool nicht reif"*
- Kann sich nicht an einen Fall erinnern
- Beschreibt nur Erfolgs-Versuche, nie Fehl-Versuche und Korrekturen
- Verwechselt Persistence mit Sturheit — ohne Reflexion, was angepasst wird

FRAGE 20 Wen liest oder folgst du im agentic-Bereich? Wer hat dich zuletzt etwas gelehrt?

GREEN FLAGS

- Mindestens 2–3 Namen: Karpathy, Steinberger, Willison, Cherny, Tunguz oder Equivalent
- Erwähnt eine konkrete Idee, die er übernommen hat
- Hat eine eigene Quellen-Routine (Newsletter, Twitter-Liste, GitHub-Stars, Substacks)

RED FLAGS

- Kennt keine Namen
- *„Ich schau mir Tutorials an, wenn ich was brauche"*
- Folgt nur Vendor-Marketing
- Hat kein eigenes Lerngerüst

FRAGE 21 Wenn du einen mittelmäßigen Engineer im Team auf agentic umstellen sollst — was ist deine erste Empfehlung?

GREEN FLAGS

- Konkrete pädagogische Antwort: ein erstes kleines MVP-Setup mit Security-Rules, eine Pair-Session, ein konkretes Tool
- Erwähnt Mindset-Shift, nicht nur Tool-Setup
- Versteht: die Methodik wird gelernt, nicht das Tool
- Erwähnt strukturierte Wissens-Verteilung im Team — Channel für Wins-and-Losses, Workflow-Show-and-Tells
- Zeigt Multiplikator-Fähigkeit

RED FLAGS

- *„Soll er sich halt einlesen"*
- Empfiehlt Tutorials oder Workshops als primären Weg
- Versteht nicht, dass Coaching wichtiger ist als das Tool
- Würde nicht selbst pair-programmen

ANHANG A

Hinweise zur Bewertung

Die Bewertung folgt einer einfachen Heuristik. Wichtig: Kein Punktwert ist absolut — Kontext und Rolle bestimmen, welche Phasen schwerer wiegen.

15 / 21

Mindest-Schwelle Senior 2026

Mindestens 15 von 21 Fragen sollten klar im Green-Flag-Bereich liegen für ein Senior-Hire 2026.

5+

Stopp-Signal

Mehr als 5 Red Flags sind ein Stopp-Signal, unabhängig vom Rest.

P1

Eisbrecher-Phase

Phase 1 (Workflow-Erzählung) ist der Eisbrecher — Kandidaten kommen hier in Rhythmus. Wer in Phase 1 schwächelt, wird in der Live-Task selten überraschen.

P2

Praxis-Beleg

Phase 2 (Setup + Live-Task) ist der Praxis-Beleg. Wer den Workflow lebt, zeigt es in den ersten 60 Sekunden des Setups. Wer ihn nur kennt, stolpert hier.

P3

Ehrlichkeits-Phase

Phase 3 (Fehler-Erzählung) ist der ehrlichste Teil. Wer hier glatt durchgeht ohne Selbstreflexion, ist entweder zu wenig praktiziert oder hat Dunning-Kruger.

★

Frage 17 + Frage 19

F17 (AI vs. menschlicher Reviewer) ist der schärfste Senioritäts-Test im Guide — wer hier schwarz-weiß antwortet, ist nicht senior. F19 (Persistence-Curation) ist der versteckteste Test.

ANHANG B

Quellen für die Bezugspunkte

- **Andrej Karpathy** — *From Vibe Coding to Agentic Engineering.* Sequoia AI Ascent 2026 · April 2026
- **Greg Brockman** — *Why Human Attention Is the New Bottleneck.* Sequoia AI Ascent 2026 · Mai 2026
- **Howie Liu** — *Making $ with AI Agents (Hyperagent-Showcase).* Greg Eisenberg's Startup Ideas Podcast · Mai 2026
- **Boris Cherny** — *Why Coding Is Solved, and What Comes Next.* Sequoia AI Ascent 2026 · Mai 2026

Stand: Mai 2026 — wird laufend aktualisiert, wenn sich die Praxis verschiebt.

HIRE SMARTER · HIRE FASTER · HIRE VETTED

Vetted IT-Freelancer. In 48 Stunden.

ElevateX verbindet Unternehmen mit geprüften Freelance-Entwicklern, Ingenieuren und IT-Experten. Skaliere dein Team schnell mit den richtigen Talenten und nutze diesen Guide als Filter für deine eigenen Hires.

Hire Freelancers →

MATCH	REPLY	VETTING
In as little as **48 hours**	Within **24 hours**	100% verified by experts

ELEVATE X

elevatex.de