HALVIC LABS · NICOLAS SCHMID

AI-Code ohne *Tech-Schuld* — die Checkliste

> Vibe-Coding bringt dich auf 80 %. Diese Checkliste sorgt für die letzten 20 % — damit aus KI-Code ein **Produkt** wird und keine Tech-Schuld. Geh sie durch, **bevor** du KI-generierten Code mergst. Die **Prompt-Boxen** kannst du direkt kopieren.

— Nicolas Schmid, KI-System-Architekt · halvic.ch

01 Bevor du promptst

- [ ] Ich habe **präzise spezifiziert**, was gebaut werden soll — nicht nur „bau X".
- [ ] Inputs, Edge Cases & erwartetes Verhalten sind benannt (null, leer, riesig, ungültig).
- [ ] Ich weiss, **wie das Ergebnis ins System passt** — welche Schicht, welches Muster.

Prompt

```
„Bevor du Code schreibst: Liste Spezifikation, Inputs, Edge Cases (null/leer/riesig/ungültig) und erwartetes Verhalten auf. Stell mir dann 3 Rückfragen zu allem, was unklar ist."
```

02 Den Agenten verhören

- [ ] Welche **Annahmen** hat der Agent getroffen? Explizit abgefragt.
- [ ] Was passiert beim **Edge Case**, den niemand geprompted hat?
- [ ] Verstehe ich jede nicht-triviale Zeile — auch noch in 6 Monaten?
- [ ] Keine **erfundenen oder veralteten Dependencies**.

Prompt

```
„Liste alle Annahmen auf, die du getroffen hast, und markiere die riskanteste. Nenne dann 5 Edge Cases, die diesen Code brechen würden, und was jeweils passiert."
```

03 Tests als Vertrag

- [ ] Ein **Test, der das Verhalten beweist** — nicht nur „es läuft".
- [ ] Kritische Tests **selbst oder getrennt** geschrieben — nicht derselbe Agent, derselbe Durchlauf.
- [ ] **Fehlerfälle und Grenzen** getestet, nicht nur der Happy Path.

Prompt

```
„Schreib einen Test, der das geforderte Verhalten beweist – inklusive Fehlerfälle und Grenzen, nicht nur Happy Path. Mock die Kernlogik nicht."
```

04 Architektur & System-Fit

- [ ] Keine **Logik dupliziert**, die schon existiert.
- [ ] Keine **Schicht-Grenze gebrochen**, kein drittes Muster für etwas mit schon zwei.
- [ ] Bleibt **wartbar und erweiterbar** — nicht nur „fertig".

Prompt

```
„Gibt es im Projekt schon Code oder ein Muster für diese Logik? Zeig mir die Fundstelle und nutze sie, statt etwas Neues zu bauen."
```

05 Sicherheit & Daten

- [ ] **Keine Secrets** im Code/Repo; Eingaben werden validiert.
- [ ] Keine **stille Sicherheitslücke**, die in der Demo nie auffällt.
- [ ] **Datenfluss & -ablage** klar — wo liegen die Daten, wer hat Zugriff.

Prompt

```
„Prüfe diesen Code auf hartcodierte Secrets, fehlende Input-Validierung und stille Sicherheitslücken. Nenne konkrete Zeilen und je einen Fix."
```

06 Vor dem Merge / Ship

- [ ] **Code-Review wie für menschlichen Code** — nicht „war ja nur KI".
- [ ] Lauffähig unter **realer Last**, nicht nur im Demo-Datensatz.
- [ ] **Logging/Monitoring** für den Fall, dass es in Produktion bricht.

Prompt

```
„Review diesen Diff wie ein strenger Senior: Nenne die 3 wahrscheinlichsten Bugs und je eine Frage, die jeden davon aufdeckt."
```

★ Hält es — und verkauft es sich?

- [ ] Löst das ein **echtes Kundenproblem** — oder ist es nur technisch beeindruckend?
- [ ] Weiss **jemand ausser mir**, dass es existiert? (Marketing ist eingeplant.)

Die ersten 80 % sind geschenkt. Für die letzten 20 % wirst du bezahlt.

Mehr: halvic.ch/blog · Tickets → fertiger, getesteter Code: anvil-coder.tech · Hörbücher: nicolasschmid.ch