1.1 Dein Architekturverständnis

Um deine große Herausforderung zu lösen, brauchst du vor allem eines: Logik.

Mit dieser Logik baust du dein Verständnis für Architektur aus und verfeinerst so an allen Stellen dein vorhandenes System. So wird aus einem digitalen Chaos schnell eine geordnete Struktur, in der sich arbeiten lässt. Und genau das wird enorm wichtig sein.

Sind wir ehrlich. Künstliche Intelligenz schreibt schon heute im Jahr 2026 einen enormen Anteil an jenem Quellcode, der in Software verwendet wird. Das sehe auch ich in meinem Job. Während andere noch an irgendwelchem Code experimentieren, habe ich schon 3 Aufgaben abgearbeitet. Und das auch noch in einer guten Qualität.

Wir müssen uns demnach auch im Smart Home nicht auf die Erstellung von Automationen fokussieren. Stattdessen brauchen wir einen Fokus darauf, wie Automationen miteinander interagieren und auf welchen Daten sie ihre Aufgaben ausführen. Es ist nicht wichtig, dass du das perfekte YAML schreibst. Es ist wichtig, dass du verstehst, was in deinem System passiert.

Wer heute den Sinn für Architektur verliert, wird auch auf langfristige Sicht den Umgang mit Künstlicher Intelligenz verlieren. Denn während in der Vergangenheit diese Aufgaben von einzelnen ausgeführt wurden, kann das die KI heute schneller, einfacher und günstiger. Den Sinn für Architektur hat sie aber noch nicht vollständig durchdrungen und ist dort noch auf eine menschliche Steuerung angewiesen.

In den nachfolgenden Kapiteln wirst du noch sehen, worauf ich hinaus will. Dort machen wir es ganz praktisch am Beispiel meiner Waschmaschine, die ausschließlich durch KI eine Restlaufzeit in Home Assistant bekommen hat.

Und während ich mir Gedanken um die Struktur und Architektur gemacht habe, hat die KI einfach die Automationen in Code umgesetzt und direkt ins System gebracht. Ich selbst habe dafür keinen Finger krumm gemacht. Genau darum geht es hier.

Ein wichtiger Gedanke
Die Künstliche Intelligenz wird kein Denken ersetzen. Im Gegenteil. Sie bestraft unpräzises Denken durch unpräzise Ergebnisse unmittelbar.

1.2 Der Paradigmenwechsel

Machen wir es an dieser Stelle etwas konkreter.

Wie hast du in der Vergangenheit Automationen für Home Assistant gebaut? Ich wette mit dir, es lief wie folgt ab:

1. Du machst dir Gedanken, was du automatisieren willst.
2. Du überlegst dir, wie sich das in Home Assistant umsetzen lässt.
3. Du beginnst mit der Automation und probierst aus, bis es funktioniert.
4. Du erkennst, dass es noch nicht rund läuft und passt es nochmal an.

Woher ich das so genau weiß? Weil wir alle so Automationen bauen. Anfänger, Fortgeschrittene, aber auch die Profis. Der einzige Unterschied besteht darin, dass der Profi Schritt 1 mit Schritt 2 kombiniert und mehr Routine in Schritt 3 hat. Er probiert auch aus, aber wesentlich theoretischer.

Mit der Künstlichen Intelligenz fallen aber im Grunde die Schritte 3 und 4 weg. Stattdessen liegt der gesamte Fokus auf den ersten beiden Schritten. Weil die das Fundament für das Ergebnis darstellen.

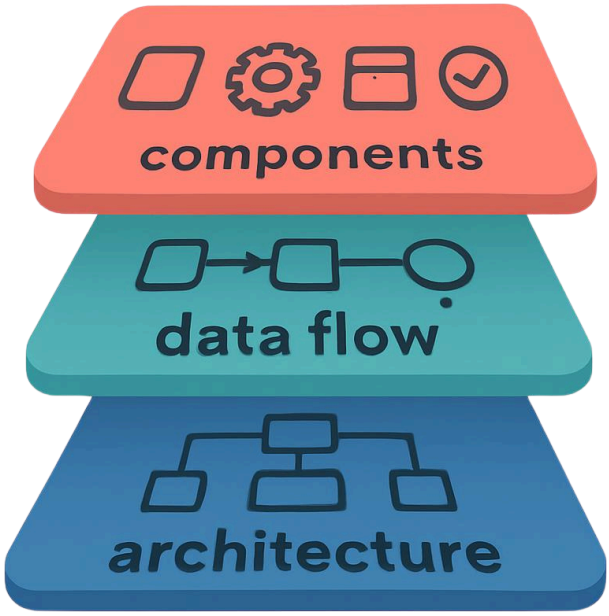
Und auch wenn du nun sagen magst, dass du selbst durch den Editor kein YAML mehr schreibst, unter der Haube passiert genau das gleiche. Für dich als Nutzer ist es im Grunde genommen einfach nur wesentlich ästethischer verpackt. Während du Entitäten in deine Automation ziehst, erstellt Home Assistant im Hintergrund die YAML, auf der deine Automation später basieren wird.

Wie das mit dem Paradigmenwechsel genau vonstatten geht, wirst du im Laufe der nächsten Abschnitte genauer verstehen.

2. Architektur-Grundlagen: So baust du es sauber auf

Lass uns das System in Schichten betrachten. Wir müssen uns dazu genau 3 Schichten ansehen, um den ganzen Prozess besser zu beleuchten:

1. Komponenten
2. Datenfluss
3. Architektur



Aus diesen 3 Schichten entsteht nämlich später ein komplett fertiger Prozess, der sich direkt an dein Smart Home andockt und dir gewissermaßen die Kräfte verleiht, die dein Zuhause noch besser und noch effizienter machen.

Und genau dieser Prozess ist es, der den Unterschied macht. Nicht die Tools, nicht das KI-Modell. Beides ist austauschbar. Aber das Zusammenspiel der Tools, genau das muss stimmen. Wenn hier irgendwo ein Fehler begraben liegt, dann wird es schwer. Um das zu vermeiden, gehen wir nun Schritt für Schritt durch.

2.1 Komponenten

Zu Beginn haben wir folgende Systeme, die getrennt voneinander arbeiten:

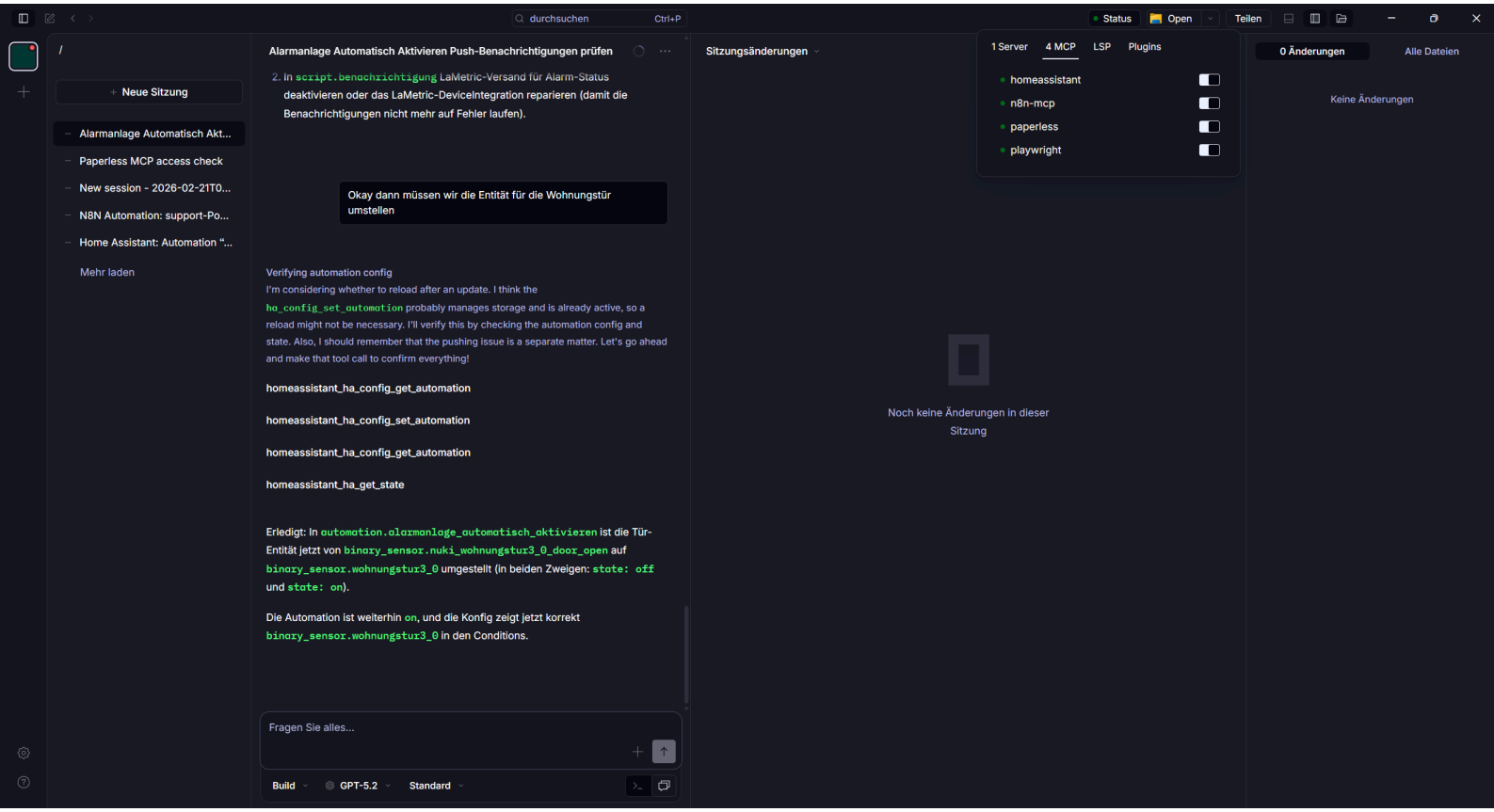
- die KI (z. B. ChatGPT / Codex)
- dein Smart Home (Home Assistant)

Und wahrscheinlich wirst du schon einmal ChatGPT gefragt haben, ob er für dich eine Automation erstellt. Die Entitäten musstest du ihm dann alle mühevoll zusammenstellen oder im Nachgang selbst austauschen. Darüber hinaus gab es kein Austausch im schönen Editor, sondern nur eine YAML-Datei, die du wiederum in Handarbeit in dein Smart Home einpflegen musstest.

Isolierte Tools
An diesem Beispiel erkennst du, dass die Tools zwar funktionieren. Aber sie sind nicht effizient, weil sie nicht miteinander sprechen.

Glaub mir, auch ich habe diesen Schritt durch. Und ich weiß auch, dass es sich ab einem gewissen Punkt so ineffizient anfühlt, dass man die KI am liebsten aus dem Spiel lassen würde und die Automation einfach selbst baut. Einerseits weil es komfortabler ist, andererseits weil es vielleicht auch noch etwas schneller geht.

Das Problem an dieser Stelle ist aber nicht das Tool. Es fehlt ein Vermittler. Und genau den führen wir jetzt mit **OpenCode** ein.



OpenCode ist laut eigener Beschreibung ein Open-Source-Agent, der dir bei der Arbeit mit Code in deinem Terminal, der IDE oder auf dem Desktop hilft. Du selbst kannst in dieser Software keinen Code schreiben. Sie ist explizit darauf ausgelegt, als Hub zwischen KI-Modell, deinen Systemen und dir als Prompter zu agieren.

OpenCode kurz erklärt

OpenCode ist der Vermittler zwischen den Systemen und dir. Die Software agiert als Integrator und dient im Grunde ausschließlich dazu, verschiedene Tools miteinander zu verbinden.

Damit das reibungslos funktioniert, müssen wir der KI noch Kontext mitgeben. An dieser Stelle kommt das **MCP** ins Spiel. Unter MCP versteht man in diesem Zusammenhang das Model Context Protocol, welches von Anthropic als offener Standard für KI-Modelle entwickelt wurde.

Vereinfacht gesagt setzt sich das MCP zwischen Home Assistant und OpenCode. Sinn und Zweck ist es, dass die Schnittstellen von Home Assistant mit Kontext versehen werden, um der KI eine Beschreibung dafür zu geben. Es ist in etwa so, wie wenn ich dir sagen würde, dass die Schnittstelle */api/error_log* dazu dient, um das

Fehlerprotokoll von Home Assistant abzurufen. Lass uns dazu ein verkürztes Beispiel ansehen:

```
async def smart_entity_search(
        self,
        query: str,
        limit: int = 10,
        offset: int = 0,
        include_attributes: bool = False,
        domain_filter: str | None = None,
    ) -> dict[str, Any]:
        """
        Advanced entity search with fuzzy matching and typo tolerance.

        Args:
            query: Search query (can be partial, with typos)
            limit: Maximum number of results
            offset: Number of results to skip for pagination
            include_attributes: Whether to include full entity attributes
            domain_filter: Optional domain to filter entities
before search (e.g., "light", "sensor")
        Returns:
            Dictionary with search results and metadata
        """
```

In diesem Beispiel sehen wir die Definition für *smart_entity_search*. Was das genau ist, ist an dieser Stelle nicht relevant. Auch nicht, dass du genau verstehst, was dort passiert.

Entscheidender ist die Tatsache, dass hier eine Schnittstelle mit Kontext versehen wird. Du siehst, dass am Anfang verschiedene Parameter (z. B. *query*, *limit* und *offset*) definiert werden. Etwas weiter unten kommt dann die Beschreibung der Schnittstelle inklusive der Beschreibung der einzelnen Parameter zum Tragen.

Über diesen Kontext wird der KI mitgeteilt, was die Schnittstelle tut und wann sie diese einsetzen muss. Das standardisiert das Vorgehen und hilft dabei, dass du nur noch den MCP anbinden musst. Der Rest passiert aus deiner Sicht wie Magie, obwohl es insgeheim nur um eine konkrete Beschreibung geht.

Das MCP ist also der Vermittler zwischen KI und Home Assistant. Ähnlich wie OpenCode. Nur dass OpenCode praktisch nur das Werkzeug für die Anbindung ist, während das MCP die tatsächlich Anbindung an dein Smart Home darstellt.

Zusammenfassend
OpenCode ist der Client, mit dem du arbeitest. Das MCP ist der Vermittler