

Meine Frage an die KI:

Welchen Sinn macht es, einen Geodienst (WMS) nur zur Abfrage der Sachdaten, aber ohne grafische Inhalte und parallel dazu einen nicht abfragbaren WMS mit Grafik zu betreiben? Beide sind öffentlich zugänglich, also dürften Datenschutz, kritischen Infrastruktur u. ä. keine Gründe für die Trennung sein.

Hier die KI-Antwort:

**KI-generiert – bitte kritisch prüfen (Halluzinationsgefahr).**

Wenn **beide WMS öffentlich** sind und damit **Datenschutz/KRITIS/ACLs** als Motivation ausscheiden, bleiben im Kern nur noch ein paar technische/organisatorische Gründe übrig. Viele davon sind legitim – manche sind eher ein Hinweis auf ein „gewachsenes“ Systemdesign.

**Was das Setup dann typischerweise bezweckt**

### **1) Performance-/Stabilitäts-Entkopplung (Caching vs. DB-Queries)**

Ein „Grafik-WMS“ lässt sich sehr gut **cachen** (Tile-Cache/Reverse Proxy), weil GetMap-Antworten bei gleichen Parametern wiederverwendbar sind.

GetFeatureInfo ist dagegen oft **nicht oder schlechter cachebar** und löst häufig **direkte Datenbankabfragen** aus (teils teuer: räumliche Suche + Joins).

**Warum trennen, wenn alles öffentlich ist?**

- Ihr könnt den **Grafik-WMS extrem aggressiv cachen** und damit billig skalieren.
- Den **Auskunfts-WMS** könnt ihr separat dimensionieren, rate-limiten, monitoren, ggf. sogar bei Lastspitzen drosseln, ohne dass die Karte „wegbricht“.

Das ist der häufigste „vernünftige“ Grund ohne Security-Aspekte.

### **2) Unterschiedliche Update-Zyklen / Datenpipeline (Karte „statisch“, Sachdaten „dynamisch“)**

Es kommt vor, dass die Visualisierung aus einer **vorprozessierten/generalisierten** Datenbasis gerendert wird (z. B. nightly build, vereinfachte Geometrien, vorgerenderte Symbologie), während die Sachdaten aus einem **aktuellen** Fachsystem kommen.

Trennung ermöglicht:

- Darstellung bleibt konsistent, schnell, robust
- Attribute können häufiger aktualisiert werden (oder aus einer anderen Quelle stammen)

**Achtung:** Das erzeugt leicht Inkonsistenzen („Ich klicke auf ein Objekt und bekomme Attribute, die nicht zur sichtbaren Darstellung passen“). Dann braucht man stabile IDs/Regeln.

### **3) Client-/Viewer-Workaround (“Identify“ soll gehen, aber der Darstellungs-Layer soll nicht queryable sein)**

In manchen WebGIS-Viewern ist das Handling so:

- Der sichtbare WMS-Layer ist „nicht abfragbar“ (Queryable aus / capabilities aus oder im Viewer deaktiviert)
- Ein zweiter, unsichtbarer WMS liefert *nur* GetFeatureInfo-Antworten in genau dem Format, das der Viewer erwartet

Das ist oft kein „Fachgrund“, sondern ein **Integrationshack**, z. B. weil:

- man FeatureInfo-Templates/Output (HTML/JSON) anders gestalten will,
- man pro Viewer ein anderes Antwortformat braucht,
- man den sichtbaren WMS nicht anfassen will/darf (z. B. externer Provider oder sehr stabiler Legacy-Dienst).

#### 4) Skalierung/DoS-Risiko (ohne „Datenschutz“, aber mit Betriebssicherheit)

Auch öffentlich kann man trennen, um die Plattform gegen „freundliche“ Lastspitzen oder Missbrauch zu härten:

- GetMap kann gut über Cache/CDN abgedeckt werden
- GetFeatureInfo kann (bewusst oder unbeabsichtigt) sehr viele DB-Treffer erzeugen

Das ist nicht „kritische Infrastruktur“ als Inhalt, sondern **Betriebsstabilität** als Ziel.

#### 5) Unterschiedliche Semantik / Layer-Zuschnitte

Manchmal soll die Karte „schön“ sein (zusammengefasste Layer, kartografische Aggregation), während die Auskunft „fachlich“ sein soll (1:1 Objekte, andere Klassifikation, mehr Felder).

Dann ist der Grafik-WMS absichtlich **nicht abfragbar**, weil die Attributlogik auf dem Kartolayer **gar keinen Sinn** ergäbe (z. B. aggregierte Flächen/Heatmaps, verschmolzene Objekte).

#### 6) Technische Grenzen des WMS als Sachdatenkanal (und man „faked“ das)

Ein WMS ist nicht primär ein Datendienst. Wenn man ihn dennoch für Sachdaten nutzt, passiert das meist über:

- GetFeatureInfo (punkt-/pixelbasiert, abhängig von Maßstab/BBOX)
- oder vendor-spezifische Erweiterungen

Ein separater „Sachdaten-WMS“ kann dann so konfiguriert sein, dass FeatureInfo **verlässlicher** funktioniert (z. B. andere Toleranzen, andere Info-Formate, andere Filter), ohne dass man die Darstellung beeinflusst.

#### Wann ist es eher Unsinn / ein Architektur-Geruch?

Wenn...

- der „Sachdaten-WMS“ im Grunde **nur ein Ersatz für WFS/OGC API Features** ist,
- und ihr eigentlich **Suche, Filter, Export, Listen** braucht (nicht nur Klick-Auskunft),
- und es keine klare Begründung für Caching/Last/Update-Zyklen gibt,

...dann wäre es sauberer, die Architektur so zu schneiden:

- **WMTS/XYZ/WMS** für Darstellung (cachebar)
- **OGC API Features** oder **WFS** für echte Datenabfrage
- optional: eine dedizierte **Search-API** (z. B. Volltext, IDs)

Das reduziert Doppelkonfiguration und macht das System verständlicher.

**Mini-Check: Wie kann man schnell herausfinden, \*welcher\* Grund hier zutrifft?**

Wenn du willst, sag mir kurz:

1. Welche Software hinter den Diensten steckt (GeoServer/MapServer/ArcGIS/QGIS Server ...)?
2. Nutzt ihr GetFeatureInfo? In welchem Format (HTML, text/plain, application/json)?
3. Gibt es Caching (GeoWebCache, MapProxy, CDN, Reverse Proxy)?
4. Haben die beiden WMS identische Layernamen/IDs oder unterschiedliche?

Ohne diese Details ist mein „best guess“ bei **öffentlich + getrennt**:

**Caching/Performance-Entkopplung** oder ein **Viewer-Workaround** (Identify/FeatureInfo-Format), ggf. kombiniert mit **unterschiedlichen Datenständen**.