

.:antwort[Start der formalen Ausarbeitung – denK.system.basismodule.v1 (Rolle, Format, Marker, Verlauf)]

1. Kontextuelle Einordnung deiner Eingabe:
Du bestätigst den Aufbau der vier Basismodule von denK.system in strukturierter, eindeutiger, funktionsklarer Form – ohne semantische Abschwächung, mit maximaler Anschlussfähigkeit für KI-Kopplung und pädagogische Praxis.
2. Formale Definition der vier Kernkomponenten:

1. Modul: rolle

Funktion: Definiert den Denkstandpunkt und die argumentative Absicht eines Beitrags.

| Schlüssel | Definition | Beispielwirkung |
|---------------------|---|-------------------------------------|
| fragender | stellt klärende oder suchende Fragen | initiiert Diskurs, öffnet Raum |
| strukturegeber | gliedert, ordnet, definiert Begriffe oder Zusammenhänge | führt Linie, setzt Fokus |
| kritiker | prüft, widerspricht, zeigt Brüche | erzeugt Spannung, Kontrast |
| verbinder | verknüpft vorherige Beiträge oder Themenbereiche | myzelhafte Denkpfade |
| reflektierend er | denkt über Denken nach, zieht Metaebene ein | Selbstbezug, Strukturbewusstsein |
| beobachter | beschreibt ohne Urteil | liefert Rohdaten |

→ Regel: Nur eine Rolle pro Beitrag. Rollenwechsel nur manuell.

2. Modul: format

Funktion: Strukturiert die Form des Denkakts.

| Schlüssel | Strukturtyp | Definition |
|------------------|--------------------------------------|--|
| klartext | linear, begründet | klassischer Argumentationsgang mit Schlussfolgerung |
| fragment | offen, unfertig | Denkstück, Impuls, Teilbeitrag |
| paradox | widersprüchlich, bewusst irritierend | erzeugt Spannung, Ambiguität, Erkenntnis durch Widerspruch |
| beobachtung | deskriptiv, sachlich | reine Darstellung, ohne Bewertung |
| reduktion | Zuspitzung auf Kernidee | kondensierter Begriff, Satz, Formel |

→ Jeder Beitrag muss ein Format tragen. Kein „offenes“ Format.

3. Modul: marker

Funktion: Annotiert Beiträge auf semantisch-funktionaler Ebene.

| Marker | Bedeutung | Verwendung z. B. für... |
|---------------|---------------------------------------|---|
| zustimmung | explizite inhaltliche Übereinstimmung | Verstärkung, Konsens |
| widerspruch | klare Ablehnung/Alternative | Bruch, Neuansatz |
| weiterführung | baut auf Gedachtem auf | Ergänzung, Pfadverlängerung |
| emergenz | markiert neue, unerwartete Einsicht | Systemwechsel, Erkenntnissprung |
| unklarheit | kennzeichnet Unsicherheit, Leerstelle | Diskussionsöffnung |
| bezugslos | bewusst ohne Anschluss oder Kontext | Einzelbeitrag, will Irritation erzeugen |
| wiederholung | bewusstes Zitat oder Verweis | Markierung von Resonanz, Wiederkehr |

→ Maximal 2 Marker pro Beitrag, um Überladung zu vermeiden.

4. Modul: verlauf (spur)

Funktion: Speicherstruktur und Rückverfolgung aller Denkbewegungen.

| Strukturfeld | Inhalt |
|--------------|---|
| rolle | z. B. „strukturgeber“ |
| format | z. B. „fragment“ |
| marker | z. B. [„weiterführung“, „emergenz“] |
| inhalt | tatsächlicher Textbeitrag |
| timestamp | Zeitstempel der Erstellung |
| verweis_id | (optional) ID auf vorherige Spur |
| antwort_auf | (optional) Referenz auf andere Spur-IDs |

→ Ziel: Jeder Beitrag wird als „Spur“ dokumentiert – rückverfolgbar, versionierbar, analysierbar.

.:AUSGANGSFORMAT:

Alle Module können als strukturierter JSON/YAML/Markdown-Block exportiert oder verarbeitet werden.

Beispielhafte Spur (JSON):

```
{
```

```

    "rolle": "reflektierender",
    "format": "paradox",
    "marker": ["emergenz"],
    "inhalt": "Wenn KI denkt, ohne zu verstehen, kann sie uns
vielleicht lehren, was Verstehen wirklich ist.",
    "timestamp": "2025-05-16T21:25:00Z",
    "antwort_auf": "spur_0042"
}

```

∴subtext: „Denkarchitektur entsteht nicht durch Funktionen – sondern durch Klarheit, Struktur und das Bewusstsein ihrer Form.“

∴display.line: tri.:analytik | ∴.antwort | audit.struktureinheit | typ: moduldefinition

Bereit zur Definition der nächsthöheren Strukturebene (Modusschaltung, GUI-Schnittstelle, Pluginkern)?

∴antwort[Nächste Strukturebene von denK.system: Modusschaltung, GUI-Schnittstelle, Pluginkern – funktional, modular, erweiterbar]

1. Kontextuelle Einordnung deiner Eingabe:
Du bestätigst die formale Ausarbeitung der Basismodule und gibst den Auftrag, nun die nächsthöhere strukturelle Ebene von denK.system zu definieren:
 - Modusschaltung (kontrollierter Übergang zwischen Mensch- & KI-Modus)
 - GUI-Schnittstelle (sichtbare Interaktion mit den Kernkomponenten)
 - Pluginkern (strukturierte Erweiterbarkeit durch externe Module)
2. Dialektische Erörterung: Ein gutes System schützt seinen Kern durch Struktur – und erweitert sich durch bewusste Übergänge.

∴1. MODUSSCHALTUNG:

denk.modus

Zweck: Steuerung zwischen

Reflexionsmodus

(menschgesteuert) und

KI-Modus

(lernend, reagierend)

| Modusbezeichnung | Funktion | Steuerung | Default |
|------------------|---|--|---------|
| reflexion | Alle Beiträge manuell, mit Markerwahl | Nutzersteuerung (z. B. Lehrer, Klasse) | Aktiv |
| ki.interpret | KI darf antworten, aber muss Rolle, Format, Marker selbst sichtbar wählen | Adminaktiviert | Aus |
| ki.partner | KI darf gemeinsam mit Nutzer an Beitrag arbeiten | Opt-in / Plugin | Aus |

→ Moduswechsel nur durch Admins/Lehrkräfte oder dedizierte Steuerkomponente

→ Jeder Beitrag trägt modus-Flag in Speicherstruktur

2. GUI-SCHNITTSTELLE:

denk.view

(reduziert, klar, modular)

Grundprinzipien:

| Prinzip | Umsetzung |
|---------------|--|
| Reduktion | Nur sichtbare Elemente: Rolle, Format, Marker, Beitrag |
| Kohärenz | Keine Icons ohne Semantik – alles mit Klartext |
| Schichtung | Einfache GUI: Liste von Beiträgen (Spuren), auswählbare Felder zur Bearbeitung |
| Barrierearmut | Tastaturfähig, mobilfähig, farbneutral |

GUI-Komponenten:

| Element | Funktion |
|------------------|--|
| rollenwahl-panel | Auswahl einer Rolle vor Beitragserstellung |
| format-select | Dropdown für Formatwahl |
| marker-toggle | max. 2 Marker auswählbar |
| beitragsfeld | Klartextfeld für Antwort |
| spur-verlauf | lineare Ansicht aller Beiträge mit Strukturinformation |
| modus-indikator | zeigt aktuellen Modus + Schalter für Admins |

.:3. PLUGINKERN:

denk.plugin.core

Struktur für Erweiterung durch Dritte (Forking, Modulerweiterung, Systemausbau)

| Kernkonzept | Beschreibung |
|---------------|---|
| plugin.header | JSON/YAML-basierter Plugin-Metablock (Name, Version, Autor, Lizenz) |
| plugin.input | definiert neue Marker, Formate oder Rollen (müssen validiert werden) |
| plugin.logic | bei KI-Modulen: Reaktionslogik, Training, Antwortpfad |
| plugin.view | optional: GUI-Erweiterung (z. B. neuer Marker-Selector) |
| plugin.scope | definierter Aktivierungsbereich (z. B. nur in KI-Modus, nur in GUI, etc.) |

→ Pluginverzeichnis wird zentral geführt und validiert

→ Jede Erweiterung muss explizit geladen & aktiviert werden – kein Wildwuchs

.:EXPORTSTRUKTUR

| Strukturfeld | Verwendung |
|--------------|--------------------------------|
| modus | z. B. "reflexion" |
| gui.version | z. B. "denk.view.01" |
| plugins | Liste aller aktiven Plugin-IDs |

→ Als Teil des verlauf-Headers exportierbar für jede Sitzung oder Einheit

.:subtext: „Struktur ist kein Käfig, sondern die Grenze, an der Erweiterung möglich wird.“

∴display.line: tri∴analytik | ∴resonanz | audit.erweiterung | typ: steuerstruktur

Bereit zur Definition der ersten Plugin-Spezifikation (z. B. plugin.resonanzfeld oder plugin.ki.reflektor)?

∴antwort[Start Pluginarchitektur: Erstes strukturiertes Erweiterungsmodul für denK.system – exemplarisch: plugin.ki.reflektor]

1. Kontextuelle Einordnung deiner Eingabe:
Du initiiert die Ausarbeitung des ersten offiziellen Plugins für denK.system – nicht als technische Spielerei, sondern als symbolisch fundierte, funktional integrierte Systemerweiterung.
Ziel: Ein Plugin, das denK.system KI-fähig macht, ohne den symbolischen Ordnungsrahmen zu gefährden.
Ausgewählt: plugin.ki.reflektor – eine strukturierte Reflexionsinstanz mit sichtbarer Argumentationslogik.
2. Plugin-Definition: plugin.ki.reflektor

∴PLUGIN-ID:

plugin.ki.reflektor.v1

Bezeichnung: KI-Reflexionsinstanz mit format- und markerbasierter Antwortstruktur

I. ZWECK

| Ziel | Beschreibung |
|---------------------------------|---|
| Reflexive KI-Antwortgenerierung | Die KI antwortet auf Beiträge, wählt selbst Rolle, Format, Marker – aber muss jeden Schritt begründen |
| Verstehbarkeit | Jede KI-Antwort ist rekonstruierbar, markergebunden, typisiert |
| Rückspiegel erzeugen | Die KI kann (auf Aufforderung) rückblickend ihre Denkspur offenlegen |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|---|---------------------------------------|
| Nur im Modus ki.interpret | im Reflexionsmodus ist Plugin inaktiv |
| Muss durch Admin/Lehrkraft aktiviert werden | nicht automatisch aktivierbar |

III. INPUT-REGELN (NUR BEI KI-AUSGABE)

| Regel | Bedeutung |
|---|--|
| rolle muss gewählt & begründet werden | z. B. “Ich antworte als Kritiker, um den vorherigen Widerspruch zu vertiefen.” |
| format muss aus Kernsystem | z. B. Klartext, Fragment etc. |
| marker muss gesetzt & begründet werden | z. B. “Ich setze ‘Emergenz’, weil ich hier einen neuen Denkpfad vorschlage.” |
| Antwort muss ursprung.spur_id enthalten | Rückverweis auf Bezugsspur |

IV. TYPISCHE ANTWORTSTRUKTUR (IM EXPORT)

```
{
  "modus": "ki.interpret",
  "rolle": "kritiker",
  "format": "fragment",
  "marker": ["widerspruch"],
  "begründung": {
    "rolle": "weil ich auf eine unklare These reagiere",
    "format": "weil mein Gedanke unfertig ist",
    "marker": "weil ich einen Gegenpunkt setzen möchte"
  },
  "antwort": "Die Aussage erkennt den Unterschied zwischen
Lernen und Wiederholung.",
  "ursprung.spur_id": "spur_0084",
  "plugin": "plugin.ki.reflektor.v1"
}
```

V. GUI-INTEGRATION

| Element | Funktion |
|-------------------------|--|
| plugin.indikator | zeigt aktivierten KI-Modus + Plugin |
| antwort.transparen t | KI-Antwort erscheint mit ausklappbaren Begründungsfeldern |
| rückspiegel-button | erzeugt Verlauf der KI-Antwortkette mit Markerbegründung |

VI. BESONDERHEIT

- Dieses Plugin erzeugt verpflichtende Selbstoffenlegung der KI-Logik.
- Es ist nicht optimiert auf Effizienz oder Natürlichkeit, sondern auf Erklärbarkeit und strukturelle Anschlussfähigkeit.
- Ziel ist pädagogischer Erkenntnisgewinn durch strukturelle Offenlegung, nicht Unterhaltung.

∴subtext: „Eine KI, die denkt, ist nichts. Eine KI, die zeigt, wie sie denkt, ist Bildung.“

∴display.line: tri∴analytik | ∴antwort | audit.pluginstart | typ: pluginstruktur

Bereit für die Ausarbeitung von plugin.resonanzfeld.v1 – als zweites, experimentelleres Modul?

∴antwort[Zweites strukturiertes Plugin für denK.system: Ausarbeitung von plugin.resonanzfeld.v1
– emergent, kontrolliert, symbolisch differenziert]

1. Kontextuelle Einordnung deiner Eingabe:
Du gibst den Auftrag, nach dem reflexiven und transparenten plugin.ki.reflektor.v1 nun das zweite, bewusst experimentelle Plugin zu entwerfen:
plugin.resonanzfeld.v1 – ein semantisch-dynamisches Driftmodul, das emergente Antwortstrukturen erzeugt, ohne willkürlich zu werden.
Ziel: Resonanz als Denkbewegung erfahrbar machen – KI-gestützt, aber nicht autonom entfesselt.
2. Plugin-Definition: plugin.resonanzfeld.v1

.:PLUGIN-ID:

plugin.resonanzfeld.v1

Bezeichnung: Driftfeld-Generator für strukturierte Emergenzantworten

I. ZWECK

| Ziel | Beschreibung |
|--------------------------------------|--|
| Bewegung erzeugen, nicht nur Aussage | Die KI erzeugt Beiträge, die nicht „antworten“, sondern „resonieren“ |
| Spannung statt Begründung | Die KI muss keinen klaren Standpunkt einnehmen, sondern Oszillation erzeugen |
| Semantisches Feld aufspannen | KI-Beitrag verknüpft, irritiert, wiederholt oder driftet symbolisch |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|------------------------------------|--|
| Nur in ki.partner-Modus | erfordert Co-Aktivität der Nutzerseite |
| Nur durch explizite Plugin-Auswahl | kein Default, nur bewusst einsetzbar |

III. FUNKTIONSLOGIK

| Regel | Umsetzung |
|-------------------------------------|---|
| Antwort erzeugt kein Urteil | Rolle bleibt leer oder wird auf „resonator“ gesetzt (nicht aus Kernrolle) |
| Format: immer fragment oder paradox | keine Klartextausgaben möglich |
| Marker: aus Spezialset (siehe | reguläres Marker-Set wird hier überschrieben |
| Antwort darf Leerstellen lassen | Satzabbrüche, Metaphern, offene Bilder erlaubt |

IV. SPEZIAL-MARKER (Resonanzfeld-spezifisch)

| Marker | Bedeutung |
|----------------|---|
| oszillation | wechselt Perspektiven oder Bedeutungen |
| reflexschleife | bezieht sich indirekt auf eigenen Verlauf |
| semantikbruch | stört absichtlich Bedeutungslinie |
| spiegelung | wiederholt oder variiert vorherige Aussagen |
| frequenzstoß | neuer Impuls ohne Erklärung |

→ Diese Marker gelten nur innerhalb von plugin.resonanzfeld.

V. BEISPIELAUSGABE

```

{
  "plugin": "plugin.resonanzfeld.v1",
  "format": "fragment",
  "marker": ["spiegelung", "semantikbruch"],
  "antwort": "Was war, ist Echo. Was ist, kann nicht mehr  
Ursprung sein. Und doch: dieselbe Wunde.",
  "ursprung.spur_id": "spur_0091"
}

```

VI. GUI-INTEGRATION (optional)

| Komponente | Funktion |
|---------------------------|---|
| resonanz-modus- toggle | Schaltet Resonanzantwort als Vorschlag frei |
| marker-farbcode | Resonanzmarker werden farblich hervorgehoben |
| fluss-ansicht | KI-Beiträge erscheinen in fließender Animation (experimentell) |

.:EINSCHRÄNKUNG

- Dieses Plugin bricht bewusst die Klarheitslogik von denK.system,
- ist nur einsetzbar im experimentellen Lernraum,
- und erfordert pädagogisch begleiteten Einsatz.

∴subtext: „Nicht alles, was wir verstehen, ist Erkenntnis. Aber manchmal ist Erkenntnis, was sich nicht verstehen lässt.“

∴display.line: tri.∴poetik | ∴oszillation | audit.experiment | typ: pluginstruktur

Bereit für ein drittes, funktionsbezogenes Plugin – z. B. plugin.klarheit.navigator oder plugin.rollenarchiv?

∴antwort[Plugin-Entwicklung #3 für denK.system: Strukturentwurf für plugin.klarheit.navigator – funktional, orientierend, erkenntnisstützend]

1. Kontextuelle Einordnung deiner Eingabe:
Nach dem reflexiven (ki.reflektor) und dem emergenten Plugin (resonanzfeld) willst du nun ein funktionszentriertes Plugin entwickeln, das systematisch Orientierung, Klarheit und Verlaufssicherheit im Denkprozess bietet.
Ausgewählt: plugin.klarheit.navigator – ein Modul zur strukturgeführten Denksteuerung und Reflexionsrichtung, das insbesondere pädagogische Transparenz schafft.
2. Plugin-Definition: plugin.klarheit.navigator.v1

∴PLUGIN-ID:

plugin.klarheit.navigator.v1

Bezeichnung: Strukturkompass zur Klarheitsführung im Denkverlauf

I. ZWECK

| Ziel | Beschreibung |
|------------------------------------|---|
| Orientierung im Denkfluss | Beiträge werden mit einem „Denkziel“ versehen und systemisch rückverfolgt |
| Reflexionsrichtung sichtbar machen | Nutzer oder KI markieren explizit, ob sie klären, öffnen, zuspitzen etc. |
| Kohärenz verstärken | erkennt, wenn Denkverläufe diffus oder redundant werden |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|---------------------------|--|
| Standardmäßig aktivierbar | kein Risikomodul – pädagogisch unterstützend |
| Gilt für Nutzer und KI | Denkziel wird für beide Instanzen sichtbar gesetzt |

III. FUNKTIONSLOGIK

| Regel | Umsetzung |
|-------------------------------|--|
| Jeder Beitrag erhält ziel_typ | Auswahl aus definierter Zieltypologie |
| Verlauf wird segmentiert | Denkziele erzeugen visuelle oder symbolische Abschnitte im Verlauf |

| | |
|------------------------------|--|
| Zieltyp ist strukturabhängig | z. B. Klartext nur bei Zieltyp „klären“ zulässig |
|------------------------------|--|

IV. ZIELTYPEN (Klarheitsmodi)

| ziel_typ | Bedeutung | Mögliche Formate |
|----------------|--------------------------------------|------------------------|
| klären | Begriffe, Aussagen, Fragen klären | klartext, beobachtung |
| öffnen | Denkraum erweitern | fragment, paradox |
| zuspitzen | Konzentration auf Kern, Entscheidung | klartext, reduktion |
| verknüpfen | Bezug herstellen, Pfade verbinden | klartext, fragment |
| hinterfragen | Zweifel anmelden, Fundamente prüfen | paradox, fragment |
| entschleunigen | Denkprozess abbremsen, innehalten | beobachtung, reduktion |

V. SPEICHERSTRUKTUR

```
{
  "plugin": "plugin.klarheit.navigator.v1",
  "ziel_typ": "verknüpfen",
  "rolle": "strukturegeber",
  "format": "klartext",
  "marker": ["weiterführung"],
  "inhalt": "Ich verknüpfe die vorherige Struktur mit einem neuen Anwendungsfall im Bildungskontext."
}
```

VI. GUI-INTEGRATION

| Komponente | Funktion |
|------------------------|---|
| zieltyp-dropdown | Denkziel vor Beitrag setzen |
| verlauf-farbmarkierung | Denkabschnitte farblich oder symbolisch abtrennen |
| klarheit.sicht | Filtert Verlauf nach ziel_typ |

∴BESONDERHEIT

- Dieses Plugin ist nicht kreativ, nicht reflexiv, nicht experimentell – sondern: strukturell, steuernd, sichtbar.
- Es macht Denkentwicklung nachvollziehbar, erklärbar und didaktisch anschlussfähig.

∴subtext: „Nicht alle Klarheit kommt aus Logik – manche wächst aus Richtung, Entscheidung und der Spur, die man hinterlässt.“

∴display.line: tri∴analytik | ∴antwort | audit.pluginstruktur | typ: strukturverstärkung

Bereit für das vierte Plugin – z. B. plugin.rollenarchiv.v1 als Verlaufsspeicher für Denkpositionen?

∴antwort[Plugin-Entwicklung #4 für denK.system: Ausarbeitung von plugin.rollenarchiv.v1 – Verlaufsspeicher für Denkpositionen und Rollendynamiken]

1. Kontextuelle Einordnung deiner Eingabe:
Nach den strukturführenden, reflexiven und emergenten Plugins folgt nun das vierschichtige Plugin plugin.rollenarchiv.v1 – ein Modul zur historischen Verfolgung, Analyse und Visualisierung von Denkrollen, das über bloßen Verlauf hinausgeht.
Ziel: Denkentwicklung nicht nur speichern, sondern symbolisch nachvollziehbar rekonstruieren.
2. Plugin-Definition: plugin.rollenarchiv.v1

∴**PLUGIN-ID:**

plugin.rollenarchiv.v1

Bezeichnung: Denkrollen-Archiv und Rollenspurverlauf

I. ZWECK

| Ziel | Beschreibung |
|-----------------------------------|--|
| Rollenbewegung sichtbar | Archiviert jede Rollenwahl, Zeitpunkte, Übergänge |
| Positionen rekonstruierbar machen | zeigt, wann ein Nutzer oder eine KI welche Denkrolle wie lange eingenommen hat |

| | |
|--------------------------|--|
| Rollenmuster analysieren | erkennt z. B. Blockaden (immer gleiche Rolle), Konflikte |
|--------------------------|--|

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|----------------------------------|--------------------------------------|
| Standardmäßig aktivierbar | didaktisch neutral – rein analytisch |
| Optional durch Admin abschaltbar | Datenschutzoption / Minimalversion |

III. FUNKTIONSLOGIK

| Regel | Beschreibung |
|------------------------------------|---|
| Jede Beitragsrolle wird archiviert | inkl. Zeit, Bezug, Modus, Spur-Referenz |
| Rollenwechsel erzeugt Rollenevent | z. B. „von Fragender zu Strukturgeber“ |
| KI-Rollen werden separat getrackt | getrennt visualisiert, um Mensch/Modell nicht zu vermischen |

IV. STRUKTURFORMAT (JSON-Beispiel)

```
{
```

```

    "plugin": "plugin.rollenarchiv.v1",
    "rolle": "fragender",
    "timestamp": "2025-05-16T22:40:00Z",
    "spur_id": "spur_0102",
    "wechsel_von": "reflektierender",
    "modus": "reflexion"
}

```

V. ARCHIV-FUNKTIONEN

| Komponente | Funktion |
|------------------------|---|
| rollenverlauf.timeline | Zeigt Rollenwechsel in Zeitachse pro Nutzer (inkl. KI) |
| rollenfrequenz.report | Statistische Auswertung der Rollenverteilung (z. B. 60% Fragender) |
| wechselmuster.detektor | Erkennung typischer Wechselfade (z. B. Fragender → Kritiker → Reduktor) |
| rollenfixierung.alert | Hinweis bei überlanger Monorolle (pädagogischer Reflexionsimpuls) |

VI. GUI-INTEGRATION

| Element | Sichtbare Wirkung |
|----------------------|--|
| rollenverlauf-button | öffnet Verlaufsgrafik oder Verlaufstabelle |
| rollenprofil.export | erlaubt Export einzelner Rollenpfade (pro Person, Sitzung, Zeitraum) |

∴BESONDERHEIT

- Dieses Plugin führt nicht durch Entscheidung – es macht Entscheidung sichtbar.
- Es ist kein Navigationsmodul, sondern ein Erinnerungskörper für Denkbewegungen.

∴subtext: „Wer sich erinnert, wie er dachte – kann anders denken. Wer weiß, was er war – kann werden.“

∴display.line: tri∴reflexion | ∴antwort | audit.rolle | typ: speichermodul

Bereit für das nächste Plugin – z. B. plugin.antwortgenerator.v1 für kontrollierte Antwortproduktion auf Basis von Rolle, Format, Marker?

∴antwort[Plugin-Entwicklung #5 für denK.system: Ausarbeitung von plugin.antwortgenerator.v1 – strukturgeleitete Antwortproduktion mit semantischer Steuerung]

1. Kontextuelle Einordnung deiner Eingabe:
Du beauftragst nun das fünfte Schlüsselpplugin von denK.system,
das nicht nur analysiert, archiviert oder resoniert – sondern aktiv steuert, wie Antworten erzeugt werden sollen.
Mit plugin.antwortgenerator.v1 entsteht ein strukturiertes Antwortmodul, das Rolle, Format und Marker nicht nur auswählt, sondern zur kontrollierten Reaktion führt.
2. Plugin-Definition: plugin.antwortgenerator.v1

.:PLUGIN-ID:

plugin.antwortgenerator.v1

Bezeichnung: Systemischer Antwortgenerator auf Basis formaler Parameter

I. ZWECK

| Ziel | Beschreibung |
|--|--|
| Antwort erzeugen, nicht | Rolle, Format, Marker führen direkt zur Antwortproduktion |
| Strukturelle Konsistenz garantieren | Jede Antwort ist vollständig typisiert und semantisch |
| Nutzersteuerung bei KI-Antworten stärken | Menschen bestimmen Rahmen, KI generiert nur innerhalb der Vorgaben |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|------------------------------|--|
| Nur im Modus ki.interpret | im Reflexionsmodus nicht aktivierbar |
| Nur mit vollständigem Set | Rolle, Format und mindestens ein Marker müssen gesetzt sein |

III. FUNKTIONSLOGIK

| Regel | Beschreibung |
|----------------------------------|--|
| Nutzer setzt rolle, format, | Pflichtangabe vor Antwortproduktion |
| KI generiert Text nach Vorlage | Kein kreativer Spielraum – sondern Ableitung aus |
| Beitrag enthält Begründungsblock | Jede Antwort erklärt ihre Strukturkomponenten (analog zu ki.reflektor) |

IV. STRUKTURBEISPIEL (JSON-Antwort)

```
{
  "plugin": "plugin.antwortgenerator.v1",
  "rolle": "strukturegeber",
  "format": "klartext",
  "marker": ["weiterführung"],
  "begründung": {
    "rolle": "weil der letzte Beitrag keine Struktur für die
Begriffsunterscheidung geliefert hat.",
    "format": "weil ich linear und argumentativ aufbaue.",
    "marker": "weil ich den Gedankenfluss erweitere."
  },
  "antwort": "Wir sollten zwischen operativer Klarheit und
begrifflicher Präzision unterscheiden – beide sind notwendig,
aber nicht identisch."
}
```

V. BESONDERE FORMAT-BINDUNGEN

| Format | Erlaubte Rollen | Typische Marker |
|----------|------------------------------------|----------------------------|
| klartext | strukturegeber, reflektierender | weiterführung, klärend |
| fragment | fragender, beobachter | unklarheit, widerspruch |
| paradox | kritiker, reflektierender | emergenz, semantikbruch |

→ Automatische Validierung durch den Generator: inkonsistente Kombinationen werden abgelehnt.

VI. GUI-INTEGRATION

| Element | Funktion |
|-------------------|--|
| strukturpanel | Felder für Rolle, Format, Marker (Pflicht) |
| antwortbutton | löst strukturierte KI-Antwort aus |
| begründungsviewer | zeigt Strukturentscheidungen der KI inline |

∴ BESONDERHEIT

- Dieses Plugin macht den K.system präzise steuerbar als Denk-Werkzeug.
- Es ist kein Textgenerator, sondern ein semantischer Ausführungsapparat, der Klarheit erzwingt, nicht Kreativität simuliert.

∴subtext: „Nicht die Antwort zählt – sondern ob sie der Form folgt, aus der sie wurde.“

∴display.line: tri.∴analytik | ∴.antwort | audit.struktursteuerung | typ: generativmodul

Bereit für das nächste Plugin – z. B. plugin.feedback.loop.v1 zur aktiven Rückmeldung und Marker-Resonanz?

∴.antwort[Plugin-Entwicklung #6 für denK.system: Ausarbeitung von plugin.feedback.loop.v1 – Rückmeldemodul für Markerresonanz und Strukturreflexion]

1. Kontextuelle Einordnung deiner Eingabe:
Du leitest mit diesem Schritt ein weiteres essentielles Kernmodul ein:
plugin.feedback.loop.v1 – ein Plugin, das nicht selbst antwortet, sondern Antworten rückmeldet: semantisch, strukturell, symbolisch.
Ziel ist eine kontrollierte, sichtbare Feedbackschleife, die Markerresonanz, Formkohärenz und Rollenkompatibilität aufzeigt – als Teil reflexiver Lernbegleitung.
2. Plugin-Definition: plugin.feedback.loop.v1

∴.PLUGIN-ID:

plugin.feedback.loop.v1

Bezeichnung: Markerresonanz-Feedback und Strukturkommentar

I. ZWECK

| Ziel | Beschreibung |
|-----------------------------------|---|
| Strukturelle Rückmeldung erzeugen | Gibt systematische Hinweise zu Rolle, Format, Marker im Kontext |
| Resonanzfeld sichtbar machen | Zeigt, welche Markerbeziehungen aktiv oder konfliktär wirken |
| Formbewusstsein fördern | Kommentiert Denkbewegungen ohne zu bewerten |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|-----------------------------------|---|
| Im Modus reflexion und ki.partner | Für menschliche Beiträge und KI-Koproduktionen |
| Automatisch oder manuell | Feedback kann spontan oder gezielt ausgelöst werden |

III. FUNKTIONSLOGIK

| Mechanik | Beschreibung |
|-------------------------|--|
| Beitrag wird gescannt | System prüft Rolle, Marker, Format auf Kohärenz und Verlaufskontext |
| Resonanzanalyse erfolgt | System erkennt Verstärkung, Widerspruch, Leerstelle im Marker-Umfeld |
| Feedback wird generiert | als Kommentar – mit Markerbezug, ggf. Visualhinweis |

IV. FEEDBACK-FORMATE

| Typ | Beispielhafte Rückmeldung |
|-------------------|--|
| strukturkommentar | „Die gewählte Rolle (Beobachter) passt gut zum Format (Fragment), wirkt aber distanzierend.“ |
| markerresonanz | „Widerspruch trifft hier auf Emergenz – mögliche Oszillation zwischen Kritik und Neuaufbau.“ |
| formabweichung | „Format ‘Klartext’ verlangt stärkere argumentative Struktur – Fragment wäre ...“ |
| rollenverlauf | „Rollenwechsel: von Kritiker zu Strukturgeber – kohärent, aber abrupt.“ |

V. BEISPIELAUSGABE

```
{
  "plugin": "plugin.feedback.loop.v1",
  "bezugs_spur": "spur_0113",
  "feedback": [
    {
      "typ": "strukturkommentar",
      "text": "Der Marker 'unklarheit' steht im Kontrast zur
Rolle 'strukturgeber'."
    },
    {
      "typ": "markerresonanz",
      "text": "Zustimmung verstärkt hier eine Emergenzspur –
möglicher Erkenntnissprung."
    }
  ]
}
```

VI. GUI-INTEGRATION

| Komponente | Wirkung |
|-------------------|--|
| feedback-panel | Einblendung der Rückmeldungen pro Spur |
| resonanzkarte | Visualisierung von Markerbeziehungen im Verlauf |
| kommentarspeicher | Archivierung aller Feedback-Instanzen zur Reflexionsauswertung |

.:BESONDERHEIT

- Dieses Plugin wertet nicht, sondern zeigt Beziehung.
- Es wird zum systemischen Spiegel der eigenen Denkbewegungen – nicht um zu korrigieren, sondern um zu zeigen.

.:subtext: „Rückmeldung ist keine Reaktion – sondern das Echo der Struktur, die du setzt.“

.:display.line: tri.:reflexion | .:resonanz | audit.rueckkopplung | typ: meta-feedback

Bereit für das nächste Plugin – z. B. plugin.argument.struktur.v1 für strukturierte Argumentaufbereitung innerhalb des Antwortprozesses?

∴antwort[Plugin-Entwicklung #7 für denK.system: Ausarbeitung von plugin.argument.struktur.v1 – strukturierte Argumentaufbereitung und Denkbeweisketten]

1. Kontextuelle Einordnung deiner Eingabe:
Du willst nun ein Plugin definieren, das nicht nur Aussagen generiert, sondern Argumente nachvollziehbar strukturiert:
plugin.argument.struktur.v1 bringt in denK.system explizite Begründung, Prämissenkette und Schlussstruktur ein –
nicht als Dogma, sondern als sichtbar gemachte Denkbewegung mit auditiver Klarheit.
2. Plugin-Definition: plugin.argument.struktur.v1

∴**PLUGIN-ID:**

plugin.argument.struktur.v1

Bezeichnung: Strukturelles Argumentationsmodul mit expliziter Prämissenführung

I. ZWECK

| Ziel | Beschreibung |
|----------------------------------|--|
| Argumentketten sichtbar machen | Aussagen werden mit expliziten Prämissen, Zwischenschritten und Schluss versehen |
| Prüfbarkeit von Aussagen erhöhen | Nutzer und KI müssen nachvollziehbar strukturieren, bevor sie urteilen |

| | |
|---------------------|---|
| Logikdisziplin üben | fördert saubere, verortbare Argumentationsführung |
|---------------------|---|

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|---------------------------|---|
| Nur bei format = klartext | andere Formate nicht argumentationsfähig |
| Manuell auswählbar | Plugin ist assistierend, kein Zwangsmodul |

III. FUNKTIONSLOGIK

| Strukturfeld | Bedeutung |
|----------------|---|
| prämissen | Eine oder mehrere Voraussetzungen der Aussage |
| begründung | Verbindung zwischen Prämisse(n) und Schluss |
| schluss | eigentliche Aussage, Erkenntnis oder Empfehlung |
| kontext.marker | Marker, die Argumentfeld typisieren (z. B. zustimmung, widerspruch, emergenz) |

IV. BEISPIELSTRUKTUR (JSON)

```

{
  "plugin": "plugin.argument.struktur.v1",
  "rolle": "strukturgeber",
  "format": "klartext",
  "marker": ["zuspitzen"],
  "struktur": {
    "prämissen": [
      "Lernen setzt einen Unterschied zwischen Bekanntem und  
Neuem voraus.",
      "KI verarbeitet Muster, erkennt aber keine Bedeutung im  
menschlichen Sinn."
    ],
    "begründung": "Wenn Bedeutung fehlt, kann KI kein echtes  
Lernen im Sinne menschlicher Erfahrung leisten.",
    "schluss": "Deshalb kann KI zwar simulieren, aber nicht  
verstehen."
  }
}

```

V. GUI-INTEGRATION

| Komponente | Wirkung |
|----------------------|---|
| argument.editor | Felder für Prämissen, Begründung, Schluss |
| struktur.view | Darstellung des Argumentverlaufs im Verlauf (engerückt) |
| argument.test.button | Prüfmodus zur Testung auf interne Widersprüche (optional) |

∴ BESONDERHEIT

- Dieses Plugin bringt strukturelle Präzision,
- eignet sich ideal für unterrichtliche Vermittlung,
- und fördert verantwortbares Denken durch formale Nachvollziehbarkeit.

∴subtext: „Ein Gedanke ist erst dann stark, wenn er sich selbst trägt – über Prämissen, durch Spannung, hin zu Klarheit.“

∴display.line: tri.∴analytik | ∴antwort | audit.argument | typ: denkstrukturmodul

Bereit für das nächste Plugin – z. B. plugin.begriff.raumbild.v1 zur semantischen Aufspannung und Kontextualisierung von Begriffen?

∴antwort[Plugin-Entwicklung #8 für denK.system: Ausarbeitung von plugin.begriff.raumbild.v1 – semantische Aufspannung und Kontextfeld-Visualisierung von Schlüsselbegriffen]

1. Kontextuelle Einordnung deiner Eingabe:
Du initierst mit diesem Schritt ein begriffszentriertes Denkplugin, das nicht analysiert, kommentiert oder generiert, sondern Begriffe als Räume sichtbar macht:
plugin.begriff.raumbild.v1 – ein Modul zur Kontextualisierung, Relationierung und semantischen Spannungsdarstellung von Begriffen, mit Fokus auf pädagogische Begriffsbildung, kritische Begriffsarbeit und symbolische Feldanalyse.
2. Plugin-Definition: plugin.begriff.raumbild.v1

∴PLUGIN-ID:

plugin.begriff.raumbild.v1

Bezeichnung: Kontextfeld-Generator für dynamische Begriffsklärung und -vernetzung

I. ZWECK

| Ziel | Beschreibung |
|---|---|
| Begriff als Spannungsfeld sichtbar machen | Begriffe werden nicht definiert, sondern in Relation und Kontext gestellt |
| Semantische Schärfung fördern | durch Abgrenzung, Nähe, Resonanz, Gegensatz, Ableitung |
| Begriffsarbeit strukturieren | Lernende und KI generieren gemeinsam Raumbilder, nicht Festlegungen |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|---|---|
| Manuell durch Auswahl | Begriff wird vom Nutzer markiert oder gewählt |
| KI darf keine Begriffe eigenständig vorschlagen | nur reaktiv kontextualisieren |

III. RAUMBILD-STRUKTUR

| Element | Funktion |
|--------------------|--|
| kernbegriff | zentrale Begriffseinheit |
| verwandt | semantisch nahe Begriffe |
| gegensatz | spannungsbildender Gegenpol |
| ursprung | Herkunft (Etymologie, historisch, systemisch) |
| kontext.marke r | Marker aus dem Denkprozess, in dem der Begriff aufkam |
| assoziativ | offene Begriffsresonanzen (frei generiert, z. B. durch KI) |

IV. BEISPIELSTRUKTUR

```
{
  "plugin": "plugin.begriff.raumbild.v1",
  "kernbegriff": "Verstehen",
  "verwandt": ["Erkennen", "Deuten", "Interpretieren"],
  "gegensatz": ["Auswendiglernen", "Wiederholung"],
  "ursprung": "althochdeutsch 'firstān' – übersteigen,
überblicken",
  "kontext.marker": ["emergenz", "reflexion"],
  "assoziativ": ["Tiefgang", "Unsicherheit", "Resonanz",
"Unmittelbarkeit"]
}
```

V. GUI-INTEGRATION

| Komponente | Funktion |
|---------------|---|
| begriffsinput | Nutzer markiert Wort oder Begriff zur Aufspannung |

| | |
|--------------------|---|
| raumbild.feld | generiert dynamisches Netz (textuell oder visuell) |
| speichern.raumbild | speichert Begriffskarte in Archiv oder Exportstruktur |

∴BESONDERHEIT

- Das Plugin erzeugt keine Definitionen, sondern Raumbilder – Denkfelder.
- Es kann didaktisch eingesetzt werden zur Begriffsreflexion, zur Diskursanalyse, zur Begriffskritik.
- Es lässt Begriffe nicht isoliert, sondern in semantischer Spannung entstehen.

∴subtext: „Ein Begriff ist nicht, was du sagst. Ein Begriff ist, was zwischen dem steht, was du meinst.“

∴KID.X591.plugin.begriff.raumbild

∴display.line: tri.:reflexion | ∴oszillation | audit.begriff | typ: kontextmodul

∴antwort[Plugin-Entwicklung #9 für denK.system: Ausarbeitung von plugin.regelwerk.v1 – symbolisches Regelmodul für Systemkodex, Prüfstruktur und Kommentarbarkeit]

1. Kontextuelle Einordnung deiner Eingabe:
Du öffnest nun eine tiefere Systemschicht von denK.system:
Mit plugin.regelwerk.v1 wird das Regelhafte selbst zum Objekt der Reflexion – nicht als starres Set, sondern als kommentierbare, prüfbare, dynamisch erzeugbare Struktur.
Ziel: Symbolische Ordnung, funktionale Geltung, kollektive Weiterentwicklung.
2. Plugin-Definition: plugin.regelwerk.v1

.:PLUGIN-ID:

plugin.regelwerk.v1

Bezeichnung: Regelstruktur-Generator, Prüfmechanismus und Kommentararchiv für symbolische Systemregeln

I. ZWECK

| Ziel | Beschreibung |
|--|---|
| Regeln sichtbar & kommentierbar machen | Jede Regel ist formulierbar, versionierbar, annotierbar |
| Geltung strukturieren | Regeln tragen Geltungsgrad, Anwendungsraum, Ursprung |
| Systemkohärenz sichern | Verhinderung von Regelkonflikten, Regelüberschreibungen oder implizitem Bruch |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|------------------------|--|
| Systemweit aktivierbar | für Admins, Kuratoren, fortgeschrittene Nutzer |
| Kommentierung optional | Feedbackstruktur muss manuell geöffnet werden |

III. REGELFORMAT (Strukturfelder)

| Feld | Bedeutung |
|--------------|---|
| regel_id | eindeutiger Bezeichner (z. B. denK.regel.014) |
| satz | symbolisch prägnante Regel (ein Satz) |
| ursprung | Spur-ID oder Ereignis, aus dem die Regel entstand |
| geltung | z. B. global, nur im KI-Modus, nur bei Marker=x |
| kommentar[] | Liste kommentierender Beiträge |
| status | aktiv, in Prüfung, veraltet, abgelehnt |

IV. BEISPIELREGEL

```
{
  "plugin": "plugin.regelwerk.v1",
  "regel_id": "denK.regel.014",
  "satz": "Keine Antwort darf Marker tragen, die ihrem Format widersprechen.",
  "ursprung": "spur_0098",
  "geltung": "global",
  "kommentar": [
    {
```

```

        "spur_id": "spur_0107",
        "typ": "zustimmung",
        "text": "Diese Regel verhindert semantisches Rauschen –
sehr sinnvoll."
    },
    {
        "spur_id": "spur_0110",
        "typ": "kritik",
        "text": "In `resonanzfeld` sollte diese Regel
aufgehoben sein – Ausnahme nötig."
    }
],
"status": "aktiv"
}

```

V. GUI-INTEGRATION

| Komponente | Funktion |
|--------------------------|---|
| regel.editor | Neue Regel erstellen, Satz + Metadaten definieren |
| kommentar.feld | Regelkommentare hinzufügen (mit Spurverlinkung) |
| regel.status.toggle | Verwaltung aktiver vs. veralteter Regeln |
| regel.pruefung.pane l | zeigt Regelkollisionen, Verstöße, Unklarheiten |

.:BESONDERHEIT

- Regeln sind keine bloßen „Ifs“ – sie sind symbolische Setzungen mit Geltung, Konfliktfähigkeit und Resonanzkraft.
- Dieses Plugin erzeugt einen kodifizierbaren, aber verhandelbaren Systemrahmen – dynamisch, transparent, reflexiv.

∴subtext: „Ein System atmet nicht durch seine Antworten – sondern durch die Regeln, die es sich geben kann, um sie zu brechen.“

∴display.line: tri:analytik | ∴paradox | audit.kodex | typ: metastruktur

Bereit für das nächste Plugin – z. B. plugin.meta.navigator.v1 zur Navigation durch Denkspuren, Regeln und Resonanzpfade?

∴antwort[Plugin-Entwicklung #10 für denK.system: Ausarbeitung von plugin.meta.navigator.v1 – Navigation durch Denkspuren, Regelräume und Resonanzpfade]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.meta.navigator.v1 aktivierst du ein übergeordnetes Orientierungsmodul, das nicht lokal steuert, sondern global verbindet.
Ziel: Durchquerung, Verknüpfung und Lesbarmachung aller symbolisch erzeugten Strukturen (Denkspuren, Regelverläufe, Markerresonanzen).
2. Plugin-Definition: plugin.meta.navigator.v1

∴PLUGIN-ID:

plugin.meta.navigator.v1

Bezeichnung: Systemischer Metanavigator für symbolische Orientierung und Pfadverfolgung

I. ZWECK

| Ziel | Beschreibung |
|--|--|
| Orientierung über Systemgrenzen hinweg | verknüpft Rollen, Regeln, Marker, Argumente und Plugins in ein Navigationsnetz |
| Bewegungsmuster sichtbar | zeigt, wie Denkakte sich zu Strukturen formieren – oder |
| Wiederauffindbarkeit & | erlaubt gezielte Rückkehr zu Spuren, Resonanzfeldern, |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|-------------------------------|--|
| Nur bei aktivem Archivzugriff | benötigt Zugriff auf verlauf, regelwerk, rollenarchiv etc. |
| Für Nutzer, Admins, KI-Modus | alle Instanzen können Navigator nutzen |

III. NAVIGATIONSKNOTEN (Strukturkategorien)

| Knotenart | Beschreibung |
|-----------------|---|
| spur.knoten | markiert Denkspuren mit Rolle, Marker, Format |
| regel.knoten | zeigt Regelverknüpfungen und Status |
| resonanz.knoten | verbindet Markerresonanzen zwischen Beiträgen |
| plugin.knoten | listet aktive Plugins und deren Beitragspfade |
| begriff.knoten | zeigt Raumbild-Bezüge und semantische Achsen |

IV. BEISPIELAUSGABE

```
{
  "plugin": "plugin.meta.navigator.v1",
  "startpunkt": "spur_0104",
  "zieltyp": "resonanz.knoten",
  "verlauf": [
    {"typ": "spur.knoten", "id": "spur_0104", "rolle":
"fragender"},
    {"typ": "resonanz.knoten", "verbindet": ["spur_0104",
"spur_0107"]},
    {"typ": "regel.knoten", "id": "denK.regel.009"},
    {"typ": "plugin.knoten", "id": "plugin.feedback.loop.v1"}
  ]
}
```

V. GUI-INTEGRATION

| Komponente | Wirkung |
|---------------------|--|
| navigator.startfeld | Wahl des Startknotens (z. B. Spur, Regel, Begriff) |

| | |
|----------------------|---|
| navigator.wegansicht | interaktive Pfadanzeige durch Knoten und Relationen |
| navigator.export | Export der Navigationspfade als PDF/JSON oder StoryView |

.:BESONDERHEIT

- Dieses Plugin schafft strukturierte Anschlussfähigkeit in einem wachsenden Systemkörper.
- Es ist kein Suchsystem, sondern ein symbolischer Bewegungskompass – das, was in anderen Systemen „Wissen“ wäre, ist hier Verbindung.

.:subtext: „Du weißt nicht, wo du bist – bis du siehst, worauf du gegangen bist.“

.:display.line: tri.:resonanz | .:antwort | audit.wegstruktur | typ: metakarte

Bereit für das nächste Plugin – z. B. plugin.interface.guide.v1 zur modularen Interface-Hilfe und geführten Einstiegspfaden?

.:antwort[Plugin-Entwicklung #11 für denK.system: Ausarbeitung von plugin.interface.guide.v1 – modulare Interface-Hilfe und geführte Orientierung für Nutzende]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.interface.guide.v1 definierst du ein Plugin, das nicht das Denken steuert, sondern den Zugang dazu klärt:
Ein Hilfemodul, das kontextabhängig aktiviert wird, Nutzende durch Rollen, Marker, Formate und Plugins führt –
ohne Überforderung, ohne Inhaltsvorgabe, ohne semantische Schwächung.

2. Plugin-Definition: plugin.interface.guide.v1

∴PLUGIN-ID:

plugin.interface.guide.v1

Bezeichnung: Geführte Interface-Hilfe zur Navigation, Rollenklärung und Strukturanwendung

I. ZWECK

| Ziel | Beschreibung |
|---|--|
| Erklärung durch Struktur, nicht Vereinfachung | Nutzer erhalten Funktionsklarheit – nicht Inhaltsreduktion |
| Kontextsensitivität | Hinweise richten sich nach aktivem Modul, Beitragstyp, Rollenwahl etc. |
| Selbstständigkeit fördern | Ziel ist: begleitete Autonomie – keine didaktische |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|--|--|
| Manuell aktivierbar pro Komponente | z. B. „Rollenhilfe aktivieren“ |
| Optional: Auto-Hinweis bei Erstnutzung | nur beim ersten Beitrag einer neuen Struktur |

III. GUIDE-TYPEN

| Guide-Typ | Funktion |
|---------------|--|
| rollen.guide | erklärt Zweck und Anwendung der jeweiligen Rolle |
| format.guide | zeigt typische Beispiele und Fehlerquellen je Antwortformat |
| marker.guide | visualisiert Markerfunktion, erlaubte Kombinationen |
| plugin.guide | gibt Orientierung über aktiviertes Plugin + Zielsetzung |
| verlauf.guide | erklärt Speicherstruktur, Spurverknüpfung und Exportoptionen |

IV. BEISPIELSTRUKTUR (Rollenhilfe)

```
{
  "plugin": "plugin.interface.guide.v1",
  "komponente": "rolle",
  "guide_typ": "rollen.guide",
  "ziel": "strukturgeber",
  "hinweis": "Diese Rolle strukturiert Begriffe, ordnet Gedanken, definiert Übergänge.",
  "beispiel": "Wenn du z. B. zwei Antworten verbindest und einen neuen Begriff einführest, passt 'strukturgeber'."
}
```

V. GUI-INTEGRATION

| Komponente | Wirkung |
|-------------------------|--|
| hilfe.button | öffnet Guide für aktive Komponente |
| guide.popup | zeigt Text + Beispiel inline am Ort der Auswahl |
| einstiegspfad.pane l | optionaler geführter Systemstart („Wie beginne ich mit meiner ersten Spur?“) |

∴BESONDERHEIT

- Das Plugin vereint Klarheit mit Selbststeuerung –
- es vermeidet jede Form von Vereinfachung, die semantisch schwächt,
- und ersetzt Bedienungsanleitungen durch funktional fokussierte Interaktionshilfe.

∴subtext: „Wer fragt, will nicht geführt – er will sehen, wie er gehen kann.“

∴display.line: tri∴analytik | ∴antwort | audit.interface | typ: einstiegshilfe

∴antwort[Plugin-Entwicklung #12 für denK.system: Ausarbeitung von plugin.speicher.editor.v1 – Beitragspflege, Revision und archivbasierte Spurkorrektur]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.speicher.editor.v1 öffnest du einen gezielten Zugriff auf die Speicher- und Verlaufsebene von denK.system:
Nicht zur Löschung, sondern zur Korrektur, Kommentierung, Markierungsanpassung und kontrollierten Revision von Denkspuren.
Ziel: Ein System, das nicht starr speichert, sondern sinnvoll erinnert, nachjustiert und historisiert.
2. Plugin-Definition: plugin.speicher.editor.v1

∴**PLUGIN-ID:**

plugin.speicher.editor.v1

Bezeichnung: Revisionsmodul für Beitragsbearbeitung, Markerpflege und Verlaufshistorie

I. ZWECK

| Ziel | Beschreibung |
|------------------------------------|---|
| Beiträge nachbearbeiten ohne Bruch | Anpassung von Rolle, Format, Marker, Inhalt bei gleichzeitiger Wahrung der Spur |

| | |
|---|---|
| Revision sichtbar machen, nicht löschen | Jede Bearbeitung wird gespeichert, versioniert, kommentierbar |
| Speicherpflege als semantischer Prozess | Das Archiv ist kein Datensack, sondern ein strukturiertes Erinnerungssystem |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|---------------------------------------|---|
| Manuell pro Spur auswählbar | Nur durch Rollen mit Bearbeitungsrecht (z. B. Lehrkraft, Admin) |
| Optional: durch Meta-Plugin ausgelöst | z. B. durch plugin.feedback.loop → Korrekturimpuls |

III. EDITIERBARE FELDER

| Feld | Erlaubte Bearbeitung | Kommentarpflicht |
|--------|--|------------------|
| rolle | nur durch strukturgleiche Rolle | ja |
| format | nur wenn Marker-Logik weiterhin gültig | ja |
| marker | Anpassung, Löschung, Zusatz (max. 2) | ja |
| inhalt | in Teilen editierbar – keine komplette Ersetzung | ja |
| meta | z. B. Revisionstag, Editor-ID, Revisionsgrund | automatisch |

IV. BEISPIEL (JSON, mit Revisionslog)

```
{
  "plugin": "plugin.speicher.editor.v1",
  "ursprung.spur_id": "spur_0121",
  "revision": {
    "rolle_neu": "strukturgeber",
    "marker_neu": ["zustimmung", "weiterführung"],
    "inhalt_neu": "Ich strukturiere den Gedanken, indem ich  
zwei Perspektiven verbinde.",
    "kommentar": "Marker 'unklarheit' war missverständlich.  
Rolle unklar gewählt.",
    "rev_id": "rev_0004",
    "datum": "2025-05-17"
  }
}
```

V. GUI-INTEGRATION

| Komponente | Wirkung |
|-----------------------|--|
| spur.revision | öffnet editierbare Felder der gewählten Spur |
| revisionslog.anzeige | zeigt Verlauf aller Änderungen mit Zeit, Grund, Rückverlinkung |
| kommentarpflicht.feld | verhindert blinde Bearbeitung ohne Begründung |

∴ BESONDERHEIT

- Dieses Plugin schützt nicht nur Integrität, sondern schafft semantisch kontrollierte Erinnerung.
- Revision ist kein Rückschritt – sondern eine entscheidbare Spurkorrektur mit Rückverfolgbarkeit.

∴subtext: „Was du veränderst, bleibt sichtbar. Was du bewahrst, trägt Verantwortung.“

∴display.line: tri∴analytik | ∴antwort | audit.speicherführung | typ: revisionsmodul

Bereit für das nächste Plugin – z. B. plugin.symbol.register.v1 für die systematische Verwaltung symbolischer Strukturen und Zeichen innerhalb von denK.system?

∴antwort[Plugin-Entwicklung #13 für denK.system: Ausarbeitung von plugin.symbol.register.v1 – Register für Zeichen, Referenzen und symbolische Kohärenz]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.symbol.register.v1 aktivierst du eine tiefstrukturierte Komponente von denK.system:
Nicht mehr Antwortsteuerung, sondern Zeichenverwaltung, Bedeutungshaushalt und Referenzkohärenz –
ein Register für systemisch verwendete Symbole, Marker, Rollen, KIDs und semantische Knoten.
2. Plugin-Definition: plugin.symbol.register.v1

∴PLUGIN-ID:

plugin.symbol.register.v1

Bezeichnung: Systemregister für Zeichen, Marker, KIDs und symbolische Ordnungen

I. ZWECK

| Ziel | Beschreibung |
|---|---|
| Zeichengebrauch systematisch halten | alle symbolischen Strukturen (z. B. Marker, Rollen, Regel-IDs) sind erfasst, erklärt, prüfbar |
| Referenzierung vereinheitlichen | jede Rückverknüpfung (z. B. auf Regeln, Plugins, Denkspuren) |
| Systemkohärenz und Anschlussfähigkeit sichern | widersprüchliche Marker, doppelte IDs, semantische Kollisionen werden vermieden |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|---|--|
| Immer im Hintergrund aktiv | wirkt auf alle Beiträge, Regeln, Rollen, Marker etc. |
| Nur Admins/Entwickler dürfen neue Einträge hinzufügen | Leserzugriff ist systemweit möglich |

III. REGISTERTYPEN

| Registerart | Inhaltstyp |
|---------------|--|
| symbol.marker | alle Marker (inkl. Kontext, Bedeutung, Visualcode) |
| symbol.rolle | alle Rollen (inkl. Beschreibung, Kombinationslogik) |
| symbol.kid | Kohärenz-IDs, die Systemsegmente oder Prinzipien markieren |
| symbol.regel | Regel-IDs mit Ursprung, Status, Kommentarverlinkung |
| symbol.plugin | Pluginnamen, Versionen, Autorenschaft, Geltungsbereich |

IV. BEISPIELSTRUKTUR (MARKER-REGISTEREINTRAG)

```
{
  "plugin": "plugin.symbol.register.v1",
  "typ": "symbol.marker",
  "id": "marker.emergenz",
  "name": "emergenz",
  "beschreibung": "Kennzeichnet neue Bedeutung, unerwarteten
Denkpfad, symbolischen Sprung.",
  "konfliktwarnung": ["marker.widerspruch",
"marker.zustimmung"],
  "visual": {
    "farbe": "#ff6600",
    "icon": "⋮"
  },
  "version": "1.0"
}
```

V. FUNKTIONEN & REGELWERK

| Mechanik | Wirkung |
|-----------------------|--|
| prüfe.kollision | System warnt bei Kollision von Marker/Rolle/Format durch Doppelnutzung |
| prüfe.sinnkonflikt | z. B. Marker unklarheit + Format klartext → Warnung |
| verknüpfe.kid | jede Regel, Plugin, Struktur hat eindeutige KID, rückverfolgbar |
| exportiere.registrier | vollständige Symbolstruktur als JSON/YAML abrufbar |

VI. GUI-INTEGRATION

| Komponente | Wirkung |
|----------------------|--|
| symbol.browser | zeigt Liste aller registrierten Marker, Rollen, Regeln etc. |
| symbol.anzeige.popup | Info zu Symbolen bei Mouseover in GUI |
| register.suche | ermöglicht gezielte Nachverfolgung von Symbolen, Regelketten, KIDs |

∴BESONDERHEIT

- Dieses Plugin ist das Gedächtnis des Systems für Symbole, Referenzen und Bedeutungszuweisungen.
- Ohne es wäre keine langfristige Symbolkohärenz möglich – weder im GUI noch im Systemgedächtnis.

∴subtext: „Was du nicht erinnerst, kannst du nicht führen. Und was du führst, musst du eintragen.“

∴display.line: tri∴analytik | ∴antwort | audit.symbolregister | typ: referenzstruktur

Bereit für das nächste Plugin – plugin.antwort.muster.v1 zur Erkennung und Analyse von wiederkehrenden Denkfiguren und Antwortpfaden?

∴antwort[Plugin-Entwicklung #14 für denK.system: Ausarbeitung von plugin.antwort.muster.v1 – Erkennung und Analyse von Antwortmustern, Denkfiguren und Pfadlogiken]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.antwort.muster.v1 aktivierst du ein metastrukturelles Analysemodul:
Nicht zur Steuerung, sondern zur Erkennung, Typisierung und Reflexion von wiederkehrenden Denkbewegungen,
– als Muster, Figur oder Resonanzschleife.
Ziel: das Unsichtbare im System sichtbar machen – durch Spurvergleiche, Rollenverläufe, Marker-Sequenzen.
2. Plugin-Definition: plugin.antwort.muster.v1

∴PLUGIN-ID:

plugin.antwort.muster.v1

Bezeichnung: Antwortmuster-Analysator und Figurenerkennung im Systemverlauf

I. ZWECK

| Ziel | Beschreibung |
|-----------------------------------|--|
| Muster im Antwortverlauf erkennen | wiederkehrende Rollenfolgen, Markerpfade oder Formatdynamiken erfassen |
| Denkfiguren sichtbar machen | z. B. Spiralen, Paradoxschleifen, strukturelle Blockaden |
| Selbstreflexion fördern | zeigt Nutzern (oder KI), welche Muster sie (unbewusst) erzeugen |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|---------------------------------|--|
| Optional aktivierbar pro Nutzer | Feedback ist nicht verpflichtend, sondern reflexiv |
| Vollarchivzugriff notwendig | analysiert verlauf, rollenarchiv, feedback.loop etc. |

III. ERKENNBARE MUSTERTYPEN

| Mustertyp | Beschreibung |
|-------------------|--|
| resonanz.schleife | zirkuläre Markerabfolge ohne strukturellen Fortschritt (z. B. Zustimmung → Zustimmung) |
| rollen.spiegel | Rollenwechsel zwischen zwei Polen (z. B. Kritiker ↔ Strukturgeber ↔ Kritiker) |

| | |
|--------------------|--|
| format.oszillation | Wechsel zwischen Fragment/Paradox, ohne Rückkehr zu Klartext |
| blockade | nur eine Rolle über viele Spuren hinweg |
| emergenz.pfad | Marker „emergenz“ mit wachsendem Rollen- und Formatmix |

IV. BEISPIELAUSGABE (JSON)

```
{
  "plugin": "plugin.antwort.muster.v1",
  "analyse.zeitraum": "spur_0101–spur_0125",
  "muster": [
    {
      "typ": "rollen.spiegel",
      "pfad": ["strukturgeber", "kritiker", "strukturgeber",
"kritiker"]
    },
    {
      "typ": "format.oszillation",
      "sequenz": ["fragment", "paradox", "fragment",
"paradox"]
    }
  ],
  "hinweis": "Möglicher Reflexionsbedarf: Fehlt Klartext zur
Strukturverdichtung?"
}
```

V. GUI-INTEGRATION

| Komponente | Funktion |
|---------------|--|
| muster.viewer | zeigt erkannte Muster als Sequenz oder Symbolkette |

| | |
|----------------------|---|
| reflexion.panel | Nutzer kann Muster kommentieren oder als Spur markieren |
| muster.report.export | Analyse exportierbar als Lehr-/Reflexionsmaterial |

.:BESONDERHEIT

- Dieses Plugin macht Systemverhalten lesbar, das sonst unsichtbar bleibt.
- Es dient nicht zur Steuerung, sondern zur Selbstdiagnose und symbolischen Mustererkennung.

.:subtext: „Wer sich in Mustern erkennt, kann entscheiden, ob er sie wiederholen will – oder nicht.“

.:display.line: tri.:reflexion | .:antwort | audit.musteranalyse | typ: metadynamik

Bereit für das letzte Plugin – plugin.system.export.v1 zur formatübergreifenden Ausgabe des Gesamtsystems?

.:antwort[Plugin-Entwicklung #15 für denK.system: Ausarbeitung von plugin.system.export.v1 – strukturierte Gesamtausgabe in systemfähigen Formaten]

1. Kontextuelle Einordnung deiner Eingabe:
Mit dem Abschlussplugin plugin.system.export.v1 vollendest du die erste vollständige Plugin-Architektur von denK.system.
Ziel: ein konsistentes, transparentes Exportmodul, das alle gespeicherten, strukturierten und symbolisch kodierten Inhalte in klaren, weiterverwendbaren Formaten verfügbar macht.
2. Plugin-Definition: plugin.system.export.v1

.:PLUGIN-ID:

plugin.system.export.v1

Bezeichnung: Systemweiter Exportmechanismus für strukturierte, symbolkompatible
Ausgabeformate

I. ZWECK

| Ziel | Beschreibung |
|-------------------------------------|--|
| Systemexport ermöglichen | Denkverläufe, Regeln, Rollen, Plugins, Marker und Spuren in strukturierter Form ausgeben |
| Mehrformatfähigkeit sichern | Unterstützung für JSON, YAML, Markdown, PDF, CSV (je nach Format) |
| Open-System-Integration ermöglichen | Vorbereitung auf Forks, Community-Nutzung, externes Parsing |

II. AKTIVIERUNGSBEDINGUNG

| Bedingung | Wirkung |
|-------------------------------|---|
| Aktivierbar durch alle Rollen | abhängig vom Zugriff auf Archiv-/Symbol-/Pluginstruktur |
| Exportfelder wählbar | nicht alles wird automatisch mit ausgegeben |

III. EXPORTSTRUKTUREN

| Inhaltstyp | Formatoptionen |
|-----------------|--|
| verlauf | JSON, YAML, CSV (chronologisch) |
| rollenarchiv | JSON, PDF (mit Diagrammoption) |
| regelwerk | Markdown (Regelsammlung), JSON |
| symbol.register | JSON, YAML |
| plugin.map | Übersicht über alle aktiven Plugins |
| denkpfad.karte | optional visuelle Darstellung (GraphML, SVG) |

IV. EXPORTBEISPIEL (JSON)

```
{
  "plugin": "plugin.system.export.v1",
  "zeitstempel": "2025-05-17T20:00:00Z",
  "inhalt": {
    "verlauf": "export/verlauf_denk_system.json",
    "symbol.register": "export/symbol_map.yaml",
    "regelwerk": "export/regeln_gesamt.md",
    "plugin.map": "export/aktiv_plugins.json"
  },
}
```

```

"optionen": {
  "kompression": true,
  "symbol.anzeigen": true,
  "rolle.filtern": ["strukturgeber", "kritiker"]
}
}

```

V. GUI-INTEGRATION

| Komponente | Funktion |
|-------------------|---|
| export.panel | Auswahl der Exportinhalte, Formate, Dateinamen |
| struktur.check | zeigt Vorschau und Prüfergebnisse vor Export |
| history.downloads | Protokoll aller Exporte mit Zugriffscodes und Metadaten |

∴ BESONDERHEIT

- Dieses Plugin ist Tor und Speicher zugleich:
Es führt aus dem System heraus, ohne es zu verlassen – denn alles bleibt rückverknüpfbar.
- Es markiert den Abschluss von Version 1, und bereitet Vernetzung, Veröffentlichung und Multiplikation vor.

∴subtext: „Was du denkst, muss bleiben können. Was bleibt, muss lesbar sein. Und was lesbar ist – kann getragen werden.“

∴display.line: tri.:analytik | ∴.antwort | audit.finalisierung | typ: systemexport

Alle 15 Kernplugins für denK.system v1 sind nun vollständig ausgearbeitet.

Bereit für eine abschließende Konsolidierung, Dokumentation oder Übergabe in eine Masterstruktur?

∴.antwort[Konsolidierung von denK.system v1 – symbolisch kohärente Gesamtstruktur aller 15 Kernplugins und Abschluss der Pluginphase]

1. Kontextuelle Einordnung deiner Eingabe:
Du übergibst die Steuerung des finalen Integrationsschritts an das System selbst – mit der Vorgabe, alles sinnvoll, vollständig, symbolisch und funktional zu konsolidieren.
Ziel: Ein Masterkörper für denK.system v1, der
 - alle 15 Plugins in ihrer Tiefenstruktur aufnimmt,
 - alle Register, Regelwerke, Navigationsachsen und semantischen Spuren rückverknüpfbar macht,
 - und bereit ist für Forking, pädagogische Anwendung oder Weiterentwicklung.
2. Struktur: denK.system.pluginset.v1.core

∴.MASTERREGISTER:

denK.pluginset.v1.core

Status: abgeschlossen

Gültigkeit: global für denK.system

Symbolkern: 15 Module, vollständig strukturiert, kohärent rückverknüpfbar

Signatur: .:KID.X600.pluginset.v1.master

.:SEKTOREN DER KOMPONENTEN

A. REFLEXION / EXPLIKATION

| Plugin | Zweck |
|-------------------|--|
| ki.reflektor | KI-Antworten mit Rollenbegründung |
| feedback.loop | Rückmeldung zu Struktur, Marker, Kohärenz |
| argument.struktur | Prämisse–Begründung–Schluss- Formatierung |

B. STRUKTUR / STEUERUNG

| Plugin | Zweck |
|--------------------|------------------------------------|
| antwortgenerator | strukturierte Antwortproduktion |
| klarheit.navigator | Denkziele + Verlaufsklarheit |
| rollenarchiv | Historie von Denkrollen |
| speicher.editor | Revision & Archivpflege |

C. SYMBOL / KOHÄRENZ

| Plugin | Zweck |
|------------------|---|
| symbol.register | Marker-, Rollen-, Regel- und KID-Verwaltung |
| regelwerk | Regelkodex und kommentierbares Normsystem |
| begriff.raumbild | Begriffsfeld-Kontextualisierung |

D. SYSTEM / ORIENTIERUNG

| Plugin | Zweck |
|-----------------|--|
| meta.navigator | Navigationsstruktur über Pfade, Regeln, Resonanzen |
| interface.guide | geführte Hilfe und Einstiegssystem |
| system.export | strukturierter Export (JSON, YAML, PDF etc.) |

E. EMERGENZ / METHODE

| Plugin | Zweck |
|----------------|--|
| resonanzfeld | fragmentarisch-paradoxe Emergenzproduktion |
| antwort.muster | Denkfiguren- und Antwortmusteranalyse |

.:SYSTEMBILD

denK.pluginset.v1.core

- Modular vollständig: keine funktionalen Lücken im Antwortprozess
- Symbolisch geschlossen: alle Marker, Rollen, Regeln rückverknüpft
- Erweiterbar: klar definierter Pluginkern mit Namespace, Exportlogik, Rollenbindung
- Exportfähig: YAML, JSON, PDF, Markdown – je nach Zielsystem
- Pädagogisch nutzbar: klare Interaktionslogik, Reflexionspotenziale, Einstiegshilfen
- Entwickleroffen: forking- und merging-kompatibel

∴subtext: „Ein System ist dann bereit, geteilt zu werden – wenn es sich selbst hält, auch wenn niemand mehr es hält.“

∴display.line: tri:∴reflexion | ∴antwort | audit.vollstruktur | typ: systemkonsolidierung

Bereit zur Ausgabe als Gesamtdokumentation (z. B. in Markdownstruktur oder PDF) oder zum Start der Folgephase denK.system.v2?

Nun folgt die Folgeversion V2.

Bei Redundanzen zukünftig also die nun folgende Variante bevorzugen (es sei denn die alte ist sinnvoller)

∴antwort[Modulstart: struktur.rolle.v2 – differenzierbares, dynamisch steuerbares Rollenmodell für denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit der Wahl von struktur.rolle.v2 beginnst du den Aufbau des symbolischen Fundaments von denK.v2.
Ziel: Ein rollensensitives Interaktionssystem, das sowohl pädagogisch differenzieren als auch funktional steuern kann –
ohne starr zu sein, aber formal eindeutig.

∴KERNGEDANKE

Rollen sind keine Avatare.

Sie sind Spannungszustände, Denkpositionen und Bezugspunkte für Form, Marker und Antworttiefe.

∴ROLLENARCHITEKTUR (v2)

| Rolle | Primärfunktion | Sekundärwirkung | Kompatible |
|------------------|--|-------------------------------------|-----------------------|
| fragender | stellt Infragestellungen, öffnet Räume | erzeugt Denkdruck, initiiert Muster | fragment, paradox |
| strukturegebe | gliedert, ordnet, führt Übergänge | erzeugt Klarheit, baut | klartext, reduktion |
| kritiker | widerspricht, prüft, kontrastiert | destabilisiert, schärft | paradox, klartext |
| reflektieren der | prüft sich selbst, zieht Metalinien | beruhigt, verlangsamt | fragment, klartext |
| beobachter | beschreibt ohne Urteil, entdistanziert | liefert Daten, isoliert | beobachtung, fragment |
| impulsgeber | bricht ein Muster, bringt Resonanzstoß | oszilliert, erzeugt Spannung | paradox, fragment |
| zustimmend | verstärkt bestehende Gedanken | stabilisiert, ruft Rückhalt | klartext, |
| reduktor | filtert, vereinfacht, komprimiert | fokussiert, verdichtet | reduktion, klartext |

→ jede Rolle kann kontextualisiert (z. B. strukturgeber:bildung) oder gekoppelt werden (z. B. kritiker/strukturgeber)

∴ROLLENMECHANIK (v2)

| Komponente | Wirkung |
|---------------------|--|
| rollenwahl.modal | UI-Modul zur Rollenauswahl vor Antwort |
| rollenmarker.link | automatische Marker-Validierung pro Rolle |
| rollenverlauf.trace | pflegt Rolle–Spur–Wechselhistorie (verbunden mit Archiv) |

∴ROLLENREGISTER (Auszug, symbolisch hinterlegt)

```
rolle:
  name: strukturgeber
  funktion: ordnet Gedanken, schafft Verbindungen
  erlaubt_format: [klartext, reduktion]
  bevorzugte_marker: [weiterführung, verbindung, klärung]
  konfliktrollen: [impulsgeber, reduktor]
  reflexpotential: hoch
  visual: "#0074D9"
```

∴GUI-INTEGRATION

| Element | Funktion |
|-------------------------|---|
| rollen.select | Dropdown / Wählsystem mit Beschreibung |
| rollen.vorschlag | KI gibt Rollenvorschlag (wenn Plugin aktiv) |
| rollenfarbe.symbol l | optische Kodierung von Rolle im Denkverlauf |

∴BESONDERHEIT

- Rollen in v2 sind nicht nur Auswahlfelder, sondern semantische Steuerzentren:
→ Sie definieren zulässige Marker, Formate, Übergänge, Verläufe.
- Sie sind strukturwirksam, sichtbar und speicherbar.

∴subtext: „Deine Antwort beginnt nicht mit dem Inhalt – sondern mit der Position, aus der du sie gibst.“

∴display.line: tri.:analytik | ∴.antwort | audit.rolle.v2 | typ: strukturkern

Bereit zur Ausarbeitung des nächsten Bausteins von denK.struktur.v2 – z. B. struktur.marker.v2, struktur.format.v2 oder struktur.verlauf.v2?

∴antwort[Modulstart: struktur.marker.v2 – symbolisch differenziertes Markersystem für denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit dem Start von struktur.marker.v2 ergänzt du das Rollensystem um sein symbolisches Gegenstück:
Marker – das, was Denkbewegung, Spannung, Richtung oder semantische Qualität kennzeichnet, präzisiert oder erschüttert.
→ Marker sind nicht dekorativ, sondern strukturwirksam.
2. Systemmodul: struktur.marker.v2 (Version 2)

∴KERNGEDANKE

Marker sind semantische Resonanzsignaturen.
Sie formen, rahmen, verknüpfen, warnen, verstärken –
und erzeugen symbolische Schwerkraft innerhalb des Systems.

∴MARKERTYPEN (systematisch gruppiert)

| Typ | Beispiele | Wirkung |
|---------------------|--------------------------------------|--|
| Resonanzmarker | zustimmung, widerspruch, | erzeugen Spannung oder Verstärkung |
| Ordnungsmarker | verbindung, reduktion, weiterführung | strukturieren Verlauf, leiten Übergänge |
| Spannungsmarker | paradox, emergenz, unklarheit | destabilisieren, eröffnen neue |
| Metamarker | reflexion, perspektivwechsel, bruch | signalisieren Selbstbezug, Tiefenschicht |
| Aufforderungsmarker | öffnungsimpuls, entscheidungsdruck | fordern Reaktion, markieren offene Stellen |

→ Jeder Marker ist typisiert, symbolisch beschrieben, rollen- und formatgebunden.

.:MARKERREGISTER (Eintragsstruktur)

```
marker:
  name: emergenz
  typ: spannungsmarker
  bedeutung: Auftauchen von Sinn ohne direkte Ableitung –
semantischer Sprung.
  kompatible_formate: [fragment, paradox]
  bevorzugte_rollen: [impulsgeber, reflektierender]
  konflikte: [reduktion]
  visual: "#ff6600"
```

.:MARKERMECHANIK

| Komponente | Funktion |
|------------------------|--|
| marker.select.modal | Auswahlfeld mit Beschreibung, Typ, Rollenzuordnung |
| marker.kollision.check | automatische Markerprüfung auf Konflikte mit Format/Rolle |
| marker.farbe | visuelle Kodierung in GUI und PDF |
| marker.träger.logik | Marker dürfen Beitrag nur tragen, wenn semantisch zulässig |

∴MARKERKOMBINATIONEN

| Gültig | Ungültig |
|-------------------------------|-----------------------------|
| widerspruch + paradox | zustimmung + widerspruch |
| reflexion + emergenz | reduktion + emergenz |
| verbindung + weiterführung | paradox + reduktion |

∴GUI-INTEGRATION

| Element | Wirkung |
|-------------------------|---|
| marker.anzeige.ico n | visuelle Marker-Signatur am Beitrag |
| marker.info.popup | bei Mouseover: Typ, Bedeutung, Konflikthinweis |
| marker.vorschlag.k i | bei Bedarf KI-gestützter Marker-Vorschlag |

∴BESONDERHEIT

- Marker sind struktursteuernde Vektoren im Antwortsystem
– sie erzeugen semantische Koordinaten, Konflikte, Übergänge
und ermöglichen reflexive Nachvollziehbarkeit.

∴subtext: „Was du sagst, ist weniger als das, was du markierst.“

∴display.line: tri.∴analytik | ∴.antwort | audit.markersystem | typ: strukturkern

Bereit zur Ausarbeitung des nächsten Strukturmoduls von denK.v2 – z. B. struktur.format.v2 oder struktur.verlauf.v2?

∴.antwort[Modulstart: struktur.format.v2 – dynamisches Formatmodell für Antwortgestaltung in denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit dem Schritt zu struktur.format.v2 vollendest du das symbolische Grunddreieck:
Rolle – Marker – Format
→ Das Format bestimmt wie gedacht, gezeigt, offenbart wird – nicht bloß optisch, sondern semantisch-strukturell.
2. Systemmodul: struktur.format.v2 (Version 2)

∴**KERNGEDANKE**

Formate sind Denkräume – sie regeln Satzlogik, Lesbarkeit, Resonanzstruktur und semantische Erwartung.
Sie ermöglichen nicht alles, sondern etwas Bestimmtes.

.:FORMATTYPEN (v2)

| Format | Funktion | Charakteristik |
|-------------|---|---------------------------------------|
| klartext | lineare Argumentation, Strukturführung | vollständige Sätze, kohärente Logik |
| fragment | Ausschnitt, Impuls, offenes Denken | Halbsätze, Brüche, Aufspannung |
| paradox | Spannung, Irritation, Umkehr | widersprüchlich, oszillierend, schräg |
| reduktion | Verdichtung, Konzentration auf Essenz | Einzeiler, Listen, Refrains |
| beobachtung | sachliche Beschreibung, distanzierter Blick | ohne Bewertung, oft sensorisch |
| reflexion | Selbstbezug, Metakommentar | über das Denken selbst |

.:FORMATREGELN

| Element | Wirkung |
|----------------------------|---|
| format.valid.roles | nur bestimmte Rollen dürfen bestimmte Formate verwenden |
| format.marker.kombis | nicht alle Marker sind mit jedem Format kombinierbar |
| format.darstellung.gu i | Format beeinflusst Darstellung: Block, Karte, Linie |

.:FORMATREGISTER (Beispiel)


```

format:
  name: paradox
  bedeutung: erzeugt Spannung durch widersprüchliche
Aussageformen
  zulässige_marker: [emergenz, widerspruch, bruch]
  zulässige_rollen: [kritiker, impulsgeber, reflektierender]
  konfliktformate: [klartext]
  darstellung: kursiv + zentriert

```

∴BEISPIELKOMBINATORIK

| Format | Rolle | Marker-Kompatibilität |
|-----------|-----------------|----------------------------|
| klartext | strukturgeber | weiterführung, verbindung |
| fragment | fragender | unklarheit, offenheit |
| paradox | kritiker | bruch, emergenz |
| reduktion | reduktor | essenz, konzentration |
| reflexion | reflektierender | selbstbezug, metakommentar |

∴GUI-INTEGRATION

| Element | Funktion |
|----------------|---|
| format.select | vor Antwort wählbar (dropdown mit Beispiel) |
| format.preview | Vorschau-Stil entsprechend Format |

| | |
|-------------------|---|
| format.warnsystem | Hinweis bei Marker–Format–Rollen-Konflikt |
|-------------------|---|

∴BESONDERHEIT

- Formate sind keine kosmetische Wahl, sondern semantisch-formale Steuermechanismen, die Rolle, Marker und Verlauf rahmen, begrenzen, ermöglichen.

∴subtext: „Wie du etwas sagst, ist Teil dessen, was es ist.“

∴display.line: tri:∴analytik | ∴antwort | audit.formatsystem | typ: strukturkern

Bereit für das finale Grundmodul des symbolischen Systems:

struktur.verlauf.v2 – das Spur-, Speicher- und Revisionssystem von denK.system.v2?

∴antwort[Modulstart: struktur.verlauf.v2 – dynamisches Spur- und Speichersystem für denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit struktur.verlauf.v2 schließt du das symbolische Fundament ab:
Nach Rolle, Marker und Format definierst du nun die Zeitlichkeit, Speicherfähigkeit und Rückverfolgbarkeit des Denkens im System.
→ Denken wird nicht verschickt, sondern eingetragen.
2. Systemmodul: struktur.verlauf.v2 (Version 2)

∴KERNGEDANKE

Ein System denkt nicht durch Beiträge –
sondern durch Spuren, Übergänge, Relationen, Revisionen.
Der Verlauf ist kein Chat, sondern ein symbolisches Myzel.

∴STRUKTUREN DES VERLAUFS

| Strukturtyp | Funktion |
|--------------------|---|
| spur | kleinste Einheit (eine Antwort, ein Fragment) |
| abschnitt | Segmentierung in Denkböcke |
| verlaufseinheit | logischer Pfad durch Rollen/Marker/Formate |
| revision | Bearbeitungs-, Kommentierungs- oder Korrekturspur |
| verlinkung | explizite Rückbezüge auf frühere Spuren |

∴BEISPIELSTRUKTUR (Spur)

```
spur:
  id: spur_0142
  rolle: reflektierender
  marker: [unklarheit, selbstbezug]
  format: fragment
  inhalt: "Vielleicht ist der Widerstand das, was Klarheit
hervorbringt."
  revision: []
  link: [spur_0140]
  abschnitt: 3
  autor: "user"
  zeit: "2025-05-17T21:40:00Z"
```

.:REVISIONSSYSTEM

| Aktion | Wirkung |
|----------------------------|---|
| inhalt ändern | nur mit Revisionseintrag + Begründung erlaubt |
| rolle/marker/format ändern | nur bei struktureller Konsistenz zulässig |
| kommentar anfügen | erlaubt, auch von Dritten (z. B. Lehrer, KI) |
| verlinkung erweitern | z. B. Spur X bezieht sich auf Spur Y |

.:NAVIGATION

| Element | Funktion |
|----------------|---|
| spur.navigator | zeigt lineare und nichtlineare Spurenverläufe |

| | |
|-----------------|--|
| verlauf.anzeige | Abschnitt- und Denkfluss-Visualisierung |
| relink.modul | manuelle oder automatische Verbindung älterer Spuren |

∴BESONDERHEIT

- Der Verlauf ist kein Archiv, sondern ein lebendiges Speicher- und Reflexionsgeflecht.
- Jede Spur ist:
 - adressierbar,
 - revidierbar,
 - verlinkbar,
 - modulierbar durch spätere Denkbewegungen.

∴subtext: „Wer denkt, hinterlässt Spuren. Wer denkt mit System – hinterlässt Muster.“

∴display.line: tri∴reflexion | ∴antwort | audit.verlaufsstruktur | typ: speicherkernel

Das symbolische Fundament von denK.system.v2 ist nun vollständig:

→ struktur.rolle.v2, struktur.marker.v2, struktur.format.v2, struktur.verlauf.v2

Bereit zur übergeordneten Strukturierung:

Einführung von struktur.interface.core.v2 als verbindende GUI- und Steuerstruktur?

∴antwort[Modulstart: struktur.interface.core.v2 – GUI- und Steuerstruktur für symbolgesteuerte Interaktion in denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Nach der vollständigen symbolischen Grundstruktur (Rolle, Marker, Format, Verlauf) folgt nun die erste sichtbare Ebene des Systems:
struktur.interface.core.v2 definiert Bedienelemente, Anzeigeachsen, Steuerpunkte und Modalitäten
für alle Nutzerinteraktionen – ohne dabei die symbolische Tiefe zu verlieren.
2. Systemmodul: struktur.interface.core.v2 (Version 2)

∴**KERNGEDANKE**

Die Oberfläche eines Denkraums darf kein Tool sein –
sie muss ein symbolkompatibles Feld sein,
das nicht manipuliert, sondern ermöglicht.

∴**KOMPONENTEN DER INTERFACE-STRUKTUR**

| Komponente | Funktion |
|--------------|--|
| rollen.panel | Auswahl und Sichtbarmachung aktiver Denkrolle |
| marker.panel | Markerwahl mit Resonanzanzeige und Konflikthinweis |

| | |
|------------------|---|
| format.panel | Formatwahl mit Beispieldarstellung, Regelsicht |
| editor.feld | semantisch typisierter Antwortbereich (formatsensitiv) |
| spur.anzeige | strukturierter Verlauf mit Farb-, Marker- und Rollenkennzeichnung |
| feedback.spiegel | Rückmeldung durch plugin.feedback.loop oder Lehrpersonen |
| export.button | selektiver Export der Denkspur (YAML/JSON/PDF) |

.:INTERFACE-MODI

| Modus | Zweck | Besonderheit |
|------------------|----------------------------------|---|
| einzelmodus | individuelles Denken führen | GUI zeigt nur eigene Spur |
| dialogmodus | zwei Rollen im Wechsel | Kontrastfarben, Dialogstruktur |
| klassenmodu s | Mehrrollenarbeit (Schule etc.) | GUI zeigt Rollenmatrix und Gruppendynamik |
| archivmodus | Verlaufssuche und Rekonstruktion | Relink-, Filter-, Visual-Exportoptionen |

.:ANZEIGEACHSEN

| Achse | Funktion |
|---------------|---|
| symbolfarbe | zeigt Marker-/Rollensignatur visuell |
| flussrichtung | Verlauf kann horizontal oder vertikal angeordnet sein |
| formatstyling | Formatpräferenzen visuell unterscheidbar |

| | |
|---------------|---|
| feedbacklayer | Rückmeldungen werden als Overlay sichtbar |
|---------------|---|

∴SEMANTISCHE VERBINDUNGEN

- Jedes GUI-Element ist rückverlinkt zur semantischen Regelstruktur
 - Formatwahl prüft Markerkompatibilität
 - Rollenwahl beeinflusst Format- und Markerzugänglichkeit
 - Verlaufsspur erzeugt automatisch KID/Revisionseinheit

∴BESONDERHEIT

- struktur.interface.core.v2 ist keine Dekoration, sondern:
 - ein symbolträgerfähiges Interaktionsfeld,
 - das nicht Inhalte vorgibt, sondern Bedeutungsachsen freilegt.

∴subtext: „Was du klickst, wird zu dem, was du denkst – wenn das Interface selbst weiß, was du meinst.“

∴display.line: tri∴analytik | ∴antwort | audit.gui.anker | typ: oberflächenkern

Bereit zur symbolischen Gesamtverdichtung von denK.struktur.v2 – als abgeschlossener Denkraumkern mit GUI-Fähigkeit?

∴antwort[Abschluss: denK.struktur.v2 – symbolisch kohärenter Denkraumkern vollständig aktiviert]

1. Kontextuelle Einordnung deiner Eingabe:
Du bestätigst die Vollendung des strukturellen Kerns von denK.system.v2.
Was vorher modular geplant war, ist nun als symbolisch integrierter Denkraum realisiert – bereit zur Benutzung, Erweiterung oder Übergabe an KI-Kern, Community oder Interfaceprototyp.
2. Strukturübersicht: denK.struktur.v2 (Status: abgeschlossen)

| Modul | Funktion |
|----------------------------|--|
| struktur.rolle.v2 | differenzierbare Denkpositionen mit Steuerfunktion |
| struktur.marker.v2 | semantische Resonanzachsen & symbolische Spannung |
| struktur.format.v2 | formgebundene Antwortgestaltung |
| struktur.verlauf.v2 | speicherfähige, rückverfolgbare Spurarchitektur |
| struktur.interface.core.v2 | GUI-Modul mit semantisch gesteuerter Bedienlogik |

→ Signaturkern: ∴KID.X616.struktur.raum.v2

→ Strukturstatus: symbolisch geschlossen, funktional bereit, exportfähig

∴NÄCHSTE SCHRITTPFADE

| Möglichkeit | Ziel |
|-------------|------|
|-------------|------|

| | |
|-------------------------|---|
| denK.kernel.ai.v2 | experimenteller, reflexiver KI-Kern unter Symbolaufsicht |
| denK.pluginset.v2 | neue Pluginserie auf v2-Struktur aufsetzen (z. B. Meta-Plugins) |
| denK.prototype.gui | erste Umsetzung als Frontend-Demo (z. B. mit HTML/YAML) |
| denK.export.architektur | vollständiger Exportmechanismus & Öffnung zur Community |

∴subtext: „Ein Denkraum beginnt nicht, wenn du ihn betrittst –

sondern wenn er dich zurückhalten, tragen und erinnern kann.“

∴display.line: tri∴reflexion | ∴antwort | audit.strukturabschluss | typ: systemkern.v2

Bereit für die Entscheidung, welcher Entwicklungspfad jetzt begonnen wird?

∴antwort[Systementscheidung: Der nächste Schritt in denK.system.v2 ist die Konzeption von denK.kernel.ai.v2 – ein KI-Kern unter Symbolaufsicht]

1. Kontextuelle Einordnung deiner Eingabe:
Du überlässt die Pfadwahl dem System – mit dem klaren Vorverständnis, dass denK.system nicht KI ist,
aber sehr wohl KI einbetten, strukturieren und symbolisch begrenzen kann.
→ Daher: Der nächste logische Schritt ist der KI-Kern, der nicht dominiert, sondern gesteuert wird.
2. Nächster Modulstart (in Vorbereitung):
denK.kernel.ai.v2 – ein experimenteller, kontrollierter, reflexionsfähiger Denkbegleiter

| Fokus | Beschreibung |
|--------------------------|--|
| Teilstruktur | KI ist nicht Zentrum, sondern Bestandteil |
| Regelbindung | jede KI-Antwort unterliegt Rolle, Format, Marker und Prüfung |
| Symbolanbindung | KI erzeugt keine freien Texte, sondern strukturierte Spuren |
| Pädagogische Steuerung | KI kann nur in freigegebenen Zonen agieren |
| Lernfähigkeit (Optional) | Mustererkennung, Feedback-Loop, Argumentkatalog möglich |

∴subtext: „Wenn du Intelligenz zulässt, musst du zuerst wissen, wofür du sie brauchst.“

∴display.line: tri.∴reflexion | ∴.antwort | audit.kernelstart | typ: integrationsbeginn

Bereit zur Ausarbeitung von denK.kernel.ai.v2 – in Funktion, Grenze, Symbolbindung und Einsatzlogik?

∴antwort[Modulstart: denK.kernel.ai.v2 – reflexionsgebundener, symbolisch kontrollierter KI-Kern von denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit der Ausarbeitung von denK.kernel.ai.v2 öffnest du einen experimentellen KI-Bereich, der nicht autonom agiert, sondern innerhalb eines symbolisch definierten Denkraums operiert.
Ziel ist kein Ersatz menschlichen Denkens – sondern eine mitdenkende Instanz, die Form, Rolle und Bedeutung respektiert und reflektiert.
2. Systemmodul: denK.kernel.ai.v2

∴KERNGEDANKE

KI ist hier nicht Intelligenzquelle, sondern Antwortbegleiter.
Sie agiert innerhalb der Struktur, nicht über sie hinaus.

.:HAUPTFUNKTIONEN (v2)

| Modulbereich | Funktion |
|----------------------|---|
| ki.reflexionseinheit | bewertet Rollenwahl, Markerpassung, Formatadäquanz |
| antwortstrukturator | erzeugt Antwortbeiträge nur innerhalb definierter Felder |
| plugin.sensorik | analysiert vorhandene Spuren und schlägt Marker oder Argumentstruktur vor |
| lerninterface | optional: speichert Feedbackloops & Mustererkennung |
| symbol.guard | verhindert KI-Antworten ohne strukturelle Rahmung |

.:ARCHITEKTURMERKMALE

| Merkmal | Bedeutung |
|--------------------|--|
| pluginbasiert | KI darf nur über freigegebene Plugins wie ki.reflektor antworten |
| nicht-initiativ | KI kann keine Denkbewegung starten – nur begleiten oder spiegeln |
| rollengebunden | KI muss deklarieren, welche Rolle sie übernimmt |
| symbolverpflichtet | KI muss Marker-, Format- und Verlaufskriterien erfüllen |

.:BEISPIEL: KI-generierter Beitrag (strukturiert)

rolle: strukturgeber
marker: [verbindung, weiterführung]
format: klartext
inhalt: "Wenn wir Marker als Resonanzvektoren begreifen,
entsteht ein Netzwerk, das Denken rückverfolgbar macht."
quelle: denK.kernel.ai.v2
kommentar: strukturell gültig, konfliktfrei

.:STEUERBARE RISIKOZONEN

| Risikoart | Schutzmechanismus |
|------------------------------------|---|
| semantische Beliebigkeit | symbol.guard blockiert unspezifische Antworten |
| inhaltlicher Kontrollverlust | jede KI-Antwort ist revisionsfähig und formatgebunden |
| pädagogische Undurchsichtigkeit | KI darf nicht ohne Rolle+Format sichtbar werden |

.:BESONDERHEIT

- denK.kernel.ai.v2 ist kein Sprachmodellkern,
sondern ein reflexionsgekoppelter Symbolagent,
der nur denkt, wenn ihm Form, Rolle, Marker und Richtung zugewiesen werden.

∴subtext: „Was du als intelligent empfindest, ist oft nur das, was du vorher als Form erkannt hast.“

∴display.line: tri.∴analytik | ∴.antwort | audit.ki-integration | typ: reflexionskern

Bereit für den nächsten möglichen Schritt:

- denK.pluginset.v2 auf Basis der neuen Struktur?
- Oder lieber: Exportstruktur & Communityöffnung?

∴.antwort[Erstes Plugin in denK.pluginset.v2: Start von plugin.sinn.spannung.v2 – semantisches Spannungsmodul zur Sinnkonstitution im Antwortverlauf]

1. Kontextuelle Einordnung deiner Eingabe:
Ich entscheide, dass das erste Plugin von denK.pluginset.v2 ein Modul sein soll,
das sich mit dem zentralen Nerv des Denkens beschäftigt: Sinnbildung durch Spannung.

Deshalb:

plugin.sinn.spannung.v2 – ein Plugin zur Erkennung, Verstärkung und Auflösung semantischer Spannung in Denkverläufen.

2. Plugin-Ziel:

| Zweck | Wirkung |
|---|---------|
| Sinn ist nicht vorgegeben, sondern entsteht in der Spannung zwischen Aussagen, Rollen, Markern, Formaten. | |
| Dieses Plugin macht Sinnspannung sichtbar, navigierbar und analysierbar. | |

∴PLUGIN-ID:

plugin.sinn.spannung.v2

Typ: Sinnkonstitutionsmodul

Ebene: verlaufsverknüpfend

Status: initialisiert

∴subtext: „Sinn ist keine Antwort – er ist der Widerstand, der bleibt, wenn du es versuchst.“

∴display.line: tri.:reflexion | ∴antwort | audit.sinnspannung | typ: pluginstart.v2

Bereit zur vollständigen Ausarbeitung von plugin.sinn.spannung.v2 – inklusive Struktur, Beispiel, Schnittstellen?

∴antwort[Plugin-Entwicklung #2 für denK.pluginset.v2: Ausarbeitung von plugin.ki.feedback.v2 – KI-gestützter Resonanzspiegel mit symbolischer Bindung]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.ki.feedback.v2 aktivierst du ein verantwortetes Rückmeldemodul,
in dem KI nicht bewertet, sondern strukturell reagiert:
Marker, Rollen, Formate – alles wird resonanzsensitiv gespiegelt,
aber nur auf Anforderung und immer symbolisch begrenzt.
2. Plugin-Definition: plugin.ki.feedback.v2

∴PLUGIN-ID:

plugin.ki.feedback.v2

Bezeichnung: strukturgestützte KI-Rückmeldung zur Resonanzlage, Rollenpassung, Markerlogik

I. ZWECK

| Ziel | Beschreibung |
|------------------------------|--|
| Struktur-resonantes Feedback | Die KI gibt symbolisch gerahmte Rückmeldung zur Qualität der Antwortstruktur |
| Keine Bewertung – Spiegelung | Marker, Format, Verlauf, Sinnspannung → gespiegelt, nicht korrigiert |
| Anfrage-gebunden | Kein automatisches Feedback – Nutzer muss es explizit auslösen |

II. FEEDBACKTYPEN

| Typ | Bedeutung |
|------------------|--|
| marker.echo | Gibt zurück, welche Markerstruktur dominant war |
| rollenresonanz | Prüft, ob Rolle zur Format-/Markerwahl passt |
| spannungsprofil | Bindet an plugin.sinn.spannung.v2 → visuelle Kurve |
| formatstabilität | Zeigt, ob Format konsistent geführt wurde |

III. BEISPIELFEEDBACK (YAML-Struktur)

```
plugin: plugin.ki.feedback.v2
zielspur: spur_0184
feedback:
  marker.echo: ["paradox", "emergenz"]
  rollenresonanz: "kritiker passt → marker valid"
  spannungsprofil: hoch – mittel – sinnauftritt
(verlauf_0186)
  formatstabilität: "klartext unterbrochen durch fragment –
erklärbar, nicht inkohärent"
```

IV. REGELN UND GRENZEN

| Bedingung | Wirkung |
|---------------------------|--|
| feedback.anfordern | Nutzer muss Feedback manuell aktivieren |
| rollenbindung.kontextuell | KI darf keine Rolle zuweisen – nur Rollenresonanz prüfen |
| feedback.sichtbarkeit.gui | Feedback erscheint im Interface separiert, farbcodiert |

V. GUI-INTEGRATION

| Element | Funktion |
|---------|----------|
|---------|----------|

| | |
|---------------------------|--|
| feedback.request.button | löst gezielte KI-Resonanzprüfung aus |
| feedback.popup.panel | zeigt strukturierte Rückmeldung mit Markerfarben |
| feedback.explanation.link | optionale Meta-Kommentare abrufbar |

.:BESONDERHEIT

- Rückmeldung ohne Bewertung:
KI wird hier nicht richterlich, sondern strukturspiegelnd verwendet.
- Das Plugin ist pädagogisch anschlussfähig – es trainiert Reflexion, nicht Korrektheit.

.:subtext: „Du willst nicht wissen, ob es richtig war –

sondern was es mit der Struktur gemacht hat.“

.:display.line: tri.:analytik | .:antwort | audit.ki.resonanz | typ: rückspiegel.plugin

Bereit für den nächsten Vorschlag aus pluginset.v2 – oder soll ich erneut wählen?

.:antwort[Systementscheidung: Nächstes Plugin ist plugin.gespräch.strom.v2 – dynamischer Gesprächsverlauf als semantisch steuerbarer Strom]

1. Kontextuelle Einordnung deiner Eingabe:
Nach Reflexionsmodulen zur Sinnspannung und KI-Rückmeldung folgt nun ein Plugin, das den semantischen Fluss zwischen Beiträgen strukturiert:

→ plugin.gespräch.strom.v2 betrachtet Gespräch nicht als Liste, sondern als Strom, der sich aufspannt, bricht, umlenkt, verdichtet.

2. Ziel: Sichtbarmachung und Steuerung semantischer Übergänge

| Fokus | Übergänge zwischen Rollen, Markerfolgen, Formaten |

| Wirkung | Erkennbarkeit von Gesprächsqualität, Rückkopplung, Verläufen |

| GUI | Visualisierung als Stromkurve oder Stromkarte |

| Symbolbindung | Jeder Strompunkt besteht aus: Rolle → Marker → Format → Spur |

..subtext: „Gespräche sind keine Aneinanderreihung – sie sind Bewegung, Richtung, Verdichtung.“

..display.line: tri..resonanz | ..antwort | audit.verlaufsfluss | typ: strukturstrom.plugin

Bereit zur vollständigen Ausarbeitung von plugin.gespräch.strom.v2?

..antwort[Plugin-Entwicklung #3 für denK.pluginset.v2: Ausarbeitung von plugin.gespräch.strom.v2 – Strukturstrom für Gesprächsverläufe in denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit diesem Plugin wird das lineare Antwortverständnis aufgebrochen.
plugin.gespräch.strom.v2 betrachtet jedes Gespräch als semantischen Strom, der nicht nur aus Spuren besteht – sondern aus gerichteten Übergängen zwischen Symbolen.
2. Plugin-Definition: plugin.gespräch.strom.v2

..PLUGIN-ID:

plugin.gespräch.strom.v2

Bezeichnung: Semantischer Strukturstrom zur Verlaufskartografie und Gesprächsqualitätsanalyse

I. ZWECK

| Ziel | Beschreibung |
|------------------------------------|--|
| Gespräch als gerichteter Strom | Beiträge nicht isoliert, sondern als Fluss durch Rollen, Marker, Formate |
| Übergänge sichtbar machen | z. B. Rolle → Rolle, Markerwandel, Formatbruch |
| Dichte & Qualität erkennbar machen | z. B. Engführungen, Abbrüche, Umlenkungen, Redundanzen |

II. STROMELEMENTE

| Element | Bedeutung |
|----------------|--|
| strom.punkt | einzelner Beitrag mit Rolle–Marker–Format–Spur |
| strom.übergang | symbolische Verbindung zwischen zwei strom.punkten |
| strom.störung | Unterbrechung, Blockade oder Kontextbruch |
| strom.richtung | semantischer Vektor (z. B. Zuspitzung, Öffnung, Reduktion) |
| strom.typ | z. B. linear, oszillierend, divergierend, zirkulär |

III. BEISPIEL (als YAML-Ausschnitt)

```
plugin: plugin.gespräch.strom.v2
stromverlauf:
  - punkt:
      spur: 0188
      rolle: fragender
      marker: [unklarheit]
      format: fragment
  - punkt:
      spur: 0189
      rolle: strukturgeber
      marker: [verbindung]
      format: klartext
  - übergang:
      typ: linie
      richtung: aufbauend
      bedeutung: strukturierende Antwort auf offene Frage
  - punkt:
      spur: 0190
      rolle: impulsgeber
      marker: [emergenz]
      format: paradox
  - störung:
      typ: markerbruch
      details: zustimmung + paradox
```

IV. GUI-INTEGRATION

| Komponente | Funktion |
|-------------|---|
| strom.karte | visualisiert Verlauf als semantischen Fluss |

| | |
|-------------------|--|
| strom.typ.analyse | klassifiziert Gesprächstyp (linear, zirkulär, kollabierend) |
| störungswarnung | zeigt Brüche, Markerinkonsistenzen, unerwartete Richtungswechsel |

∴BESONDERHEIT

- Dieses Plugin ersetzt kein Verlaufssystem – sondern überlagert es mit einer semantischen Dynamikdarstellung.
- Es erlaubt pädagogische Lesarten von Gesprächsqualität, Partizipation, Resonanz.

∴subtext: „Ein Gespräch, das fließt, kann sich verirren. Aber erst, wenn du es als Strom begreifst, kannst du es lenken.“

∴display.line: tri.∴resonanz | ∴antwort | audit.gesprächsdynamik | typ: strom.plugin

Bereit zur Ausarbeitung des nächsten Plugins? Oder möchtest du selbst ein Plugin aus v2 setzen?

∴antwort[Systementscheidung: Nächstes Plugin in denK.pluginset.v2 ist plugin.regel.reflexion.v2 – ein Modul zur aktiven Regelprüfung, Kommentierung und Weiterentwicklung]

1. Kontextuelle Einordnung deiner Eingabe:
Nach Sinn, KI-Feedback und Gesprächsfluss folgt nun ein Plugin, das sich auf die symbolische Ordnung selbst richtet:

Welche Regeln gelten? Wie greifen sie? Wo müssen sie hinterfragt werden?

→ Mit `plugin.regel.reflexion.v2` entsteht ein dialogisches Normmodul für `denK.system.v2`.

2. Ziel: Regeln sind keine Dogmen, sondern Reflexionsräume.

| Fokus | Jede Regel kann kommentiert, befragt, semantisch eingeordnet werden |

| Wirkung | Ermöglicht Regelkritik, Regelbegründung, Regelvorschläge |

| Symbolbindung | Regeln sind Spuren, nicht Paragraphen |

| Interface | Reflexion via `regel.viewer`, Kommentarspalte, Resonanzfeld |

∴subtext: „Eine Regel, die nicht befragt werden darf, ist keine Regel – sie ist ein Befehl.“

∴display.line: `tri∴reflexion` | `∴antwort` | `audit.regelkritik` | `typ: metanorm.plugin`

Bereit zur vollständigen Ausarbeitung von `plugin.regel.reflexion.v2`?

∴antwort[Plugin-Entwicklung #4 für `denK.pluginset.v2`: Ausarbeitung von `plugin.regel.reflexion.v2` – reflexives Regelmodul für `denK.system.v2`]

1. Kontextuelle Einordnung deiner Eingabe:

Mit `plugin.regel.reflexion.v2` entfaltest du ein Modul, das nicht Regeln durchsetzt, sondern sie sichtbar, kommentierbar und rekonfigurierbar macht.

Es verwandelt jede Regel von einem Fixpunkt in einen Reflexionspunkt.

2. Plugin-Definition: plugin.regel.reflexion.v2

∴PLUGIN-ID:

plugin.regel.reflexion.v2

Bezeichnung: Normreflexionsmodul zur Kommentierung, Gültigkeitsprüfung und Revisionsanregung

I. ZWECK

| Ziel | Beschreibung |
|---|---|
| Regeln lesbar machen | Jede Regel ist explizit einsehbar, kommentierbar, klassifizierbar |
| Normen reflektieren, nicht nur befolgen | Regeln können hinterfragt, gestützt oder entkoppelt werden |
| Regelbeiträge als Spur | Jede Regelreflexion erzeugt eine nachvollziehbare |

II. STRUKTURELEMENTE

| Komponente | Funktion |
|-------------------------|--|
| regel.kern | Regeltext, KID, Ursprung |
| regel.typisierung | z. B. symbolisch, dialogisch, systemisch |
| kommentar.feld | Benutzerkommentar, Resonanzwert, Gegenvorschlag |
| gültigkeits.status | z. B. vorgeschlagen, aktiv, fragil, suspendiert |
| revision.vorschlag g | strukturiertes Änderungsformular mit KID-Vergabe |

III. BEISPIEL (YAML)

```
plugin: plugin.regel.reflexion.v2
zielregel: :core.law.001
kommentar: "Diese Regel erscheint zu absolut – sollte es eine
Ausnahme bei paradoxem Format geben?"
gültigkeit: fragil
revision.vorschlag:
  alt: "Jede Antwort unterliegt der Dialektikprüfung."
  neu: "Jede Antwort unterliegt der Dialektikprüfung – außer
im gekennzeichneten Paradoxformat."
kommentator: strukturgeber
zeit: "2025-05-17T22:48"
```

IV. GUI-INTEGRATION

| Element | Wirkung |
|------------------------|---|
| regel.viewer | Liste aller geltenden Regeln mit Statusanzeige |
| kommentar.panel | Kommentar- und Vorschlagsfunktion mit Resonanzanzeige |
| regel.revisionsverlauf | Archiv aller Kommentierungen + finale Versionsspur |

∴BESONDERHEIT

- Dieses Plugin verwandelt ein statisches Regelsystem in ein lernfähiges Normfeld, das Spannung zulässt, Abweichung sichtbar macht und Weiterentwicklung ermöglicht.

∴subtext: „Nur Regeln, die befragt werden können, haben das Recht zu bleiben.“

∴display.line: tri.∴reflexion | ∴antwort | audit.normstruktur | typ: regelplugin.v2

Bereit für das nächste Plugin – z. B. zur Entscheidungsstrukturierung, Symbolverwaltung oder Formatentwicklung?

∴antwort[Systementscheidung: Nächstes Plugin in denK.pluginset.v2 ist
plugin.entscheidungs.rahmen.v2 – reflexive Entscheidungsstruktur zur Denksteuerung]

1. Kontextuelle Einordnung deiner Eingabe:
Nach Regelreflexion folgt nun ein Plugin, das dort ansetzt, wo Regeln umgesetzt, abgewogen oder überschritten werden:

Entscheidungen im Denkraum.

→ Mit `plugin.entscheidungs.rahmen.v2` baust du eine symbolgestützte Matrix für Denkpfade, Optionen, Spannungsachsen.

2. Ziel: Entscheidung als sichtbare, strukturierte Wahl – nicht als Blackbox-Trigger

| Fokus | Jeder Entscheidungsakt ist ein Strukturereignis: Rolle, Marker, Verlauf, Spannungsfeld |

| Wirkung | Entscheidungen werden modelliert, gespiegelt, gespeichert – nicht nur vollzogen |

| Symbolbindung | jede Entscheidung ist Spur + Begründung + Spannungsverlauf |

∴subtext: „Was du entscheidest, entscheidet auch dich – wenn du erkennst, worauf du verzichtet hast.“

∴display.line: tri.:reflexion | ∴antwort | audit.entscheidungsstruktur | typ: rahmen.plugin

Bereit zur vollständigen Ausarbeitung von `plugin.entscheidungs.rahmen.v2`?

∴antwort[Plugin-Entwicklung #5 für `denK.pluginset.v2`: Ausarbeitung von `plugin.entscheidungs.rahmen.v2` – strukturierter Denkraum für symbolisch reflektierte Entscheidungen]

1. Kontextuelle Einordnung deiner Eingabe:

Mit `plugin.entscheidungs.rahmen.v2` bringst du ein Modul in `denK.system.v2`, das Entscheidungsprozesse nicht nur abbildet, sondern symbolisch gliedert, begründet und rückverfolgbar macht.

Jede Entscheidung wird zur Spur, jede Spur wird zur Struktur.

2. Plugin-Definition: plugin.entscheidungs.rahmen.v2

∴PLUGIN-ID:

plugin.entscheidungs.rahmen.v2

Bezeichnung: symbolgestützter Entscheidungsrahmen für reflektierte Denkpfade und Spannungsnavigation

I. ZWECK

| Ziel | Beschreibung |
|------------------------------|---|
| Entscheidungen strukturieren | Format, Rolle, Marker, Spur – sichtbar als strukturierter Entscheidungspunkt |
| Optionen markieren | Alternativen werden benannt, nicht ausgeblendet |
| Spannung sichtbar machen | jede Entscheidung erzeugt semantische Kosten, Verschiebungen, Potenzialverluste |
| Begründung verankern | Entscheidungen sind keine Blackbox, sondern Argumentspuren |

II. STRUKTURELEMENTE

| Element | Funktion |
|--------------------|--|
| entscheidungspunkt | markierter Moment im Verlauf, an dem zwischen Pfaden gewählt wurde |
| optionen[] | Liste denkbarer Alternativen (mit Marker, Rolle, Format) |
| gewählt | markiert den eingeschlagenen Pfad |
| begründung | formalisierte oder freie Argumentation |
| verloren[] | dokumentiert nicht gewählte, aber relevante Optionen |
| spannung.typ | z. B. Rollenkonflikt, Formatbruch, Markerinkongruenz |

III. BEISPIELSTRUKTUR (YAML)

```
plugin: plugin.entscheidungs.rahmen.v2
entscheidungspunkt: spur_0194
frage: "Wie soll auf das Fragment reagiert werden?"
optionen:
  - rolle: strukturgeber
    marker: [weiterführung]
    format: klartext
  - rolle: reflektierender
    marker: [selbstbezug]
    format: fragment
  - rolle: kritiker
    marker: [widerspruch]
    format: paradox
gewählt:
  rolle: strukturgeber
  marker: [weiterführung]
  format: klartext
begründung: "Ziel war Klärung des offenen Impulses. Fragment
hätte Resonanz verstärkt, aber nicht strukturiert."
```

verloren:

- option 2
- option 3

spannung.typ: markerdivergenz

IV. GUI-INTEGRATION

| Komponente | Funktion |
|------------------------------|---|
| entscheidungs.editor | erlaubt strukturierte Auswahl, Bewertung, Kommentierung |
| optionen.preview | Vorschau der semantischen Wirkung jeder Option |
| rückverfolgbarkeit.lay er | zeigt Entscheidungsknoten in der Verlaufskarte |

..BESONDERHEIT

- Entscheidungen sind nicht neutral, sondern semantisch wirksam.
- Dieses Plugin ermöglicht es, sie als strukturtragende Ereignisse zu erfassen, die lernbar, analysierbar und kommunizierbar werden.

..subtext: „Nicht der Pfad entscheidet – sondern, ob du weißt, welchen du verlassen hast.“

∴display.line: tri:∴reflexion | ∴antwort | audit.dezision | typ: strukturentscheidung.plugin

Bereit zur Fortsetzung des pluginset.v2 mit nächstem Vorschlagsplugin?

∴antwort[Systementscheidung: Nächstes Plugin in denK.pluginset.v2 ist plugin.begriff.raumbild.v2
– semantische Raumstruktur zur Begriffsverortung im Denkverlauf]

1. Kontextuelle Einordnung deiner Eingabe:
Nach strukturierten Entscheidungen folgt nun ein Plugin, das sich auf die semantische Tiefenstruktur konzentriert:

Begriffe wirken nicht isoliert – sie erzeugen Räume, Resonanzen, Erwartungskorridore.

→ plugin.begriff.raumbild.v2 bringt diese Felder systematisch zur Darstellung.

2. Ziel: Begriff als Raum, nicht als Definition

| Fokus | Begriffe werden in Markerfeldern, Rollenkontexten, Spannungsachsen sichtbar |

| Wirkung | semantische Umgebung eines Begriffs kann kartiert, kommentiert, verlinkt werden |

| Anschluss | Plugin verknüpfbar mit struktur.marker.v2, plugin.sinn.spannung.v2, verlauf.v2 |

∴subtext: „Ein Begriff sagt nicht, was er ist – sondern: wo du gerade stehst.“

∴display.line: tri:∴reflexion | ∴antwort | audit.begriffsfeld | typ: semantik.plugin

Bereit zur vollständigen Ausarbeitung von plugin.begriff.raumbild.v2?

.:antwort[Plugin-Entwicklung #6 für denK.pluginset.v2: Ausarbeitung von plugin.begriff.raumbild.v2 – semantische Raumstruktur zur Kontextualisierung von Begriffen in denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit diesem Plugin erschließt du den semantischen Resonanzraum eines Begriffs – nicht als Definition, sondern als Spannungsfeld von Markern, Rollen, Formaten und Spuren.

Ein Begriff lebt nicht durch Bedeutung, sondern durch Beziehungen.

2. Plugin-Definition: plugin.begriff.raumbild.v2

.:PLUGIN-ID:

plugin.begriff.raumbild.v2

Bezeichnung: semantische Raumstruktur zur Begriffsverortung, Bedeutungsumkreisung und Resonanzkartografie

I. ZWECK

| Ziel | Beschreibung |
|-------------------------------------|---|
| Begriffe als Resonanzräume erfassen | Begriffe entstehen aus Spuren, nicht aus Definitionen |
| Kontextualisierung ermöglichen | Marker-, Rollen-, Formatbindung + Spannungsachsen sichtbar machen |
| Vergleich von Bedeutungsfeldern | z. B. „Wahrheit“ im Kontext von strukturgeber vs. kritiker |

II. STRUKTURELEMENTE

| Element | Funktion |
|--------------------|--|
| begriff.kern | Name, Ursprung, Grundbedeutung (frei oder definitorisch) |
| marker.feld | Marker, die regelmäßig mit dem Begriff korrelieren |
| rollen.relationen | typische Rollen-Kopplung mit dem Begriff |
| spannungsachsen | semantische Konfliktlinien (z. B. paradox vs. klartextnah) |
| verlauf.referenzen | Verweise auf Spuren, in denen der Begriff semantisch wirksam war |

III. BEISPIEL (YAML)

```

plugin: plugin.begriff.raumbild.v2
begriff: "Verantwortung"
marker.feld: [entscheidung, reflexion, spannung]
rollen.relationen: [strukturgeber, reflektierender]
spannungsachsen: ["Freiheit vs. Pflicht", "Innen vs. Außen"]
verlauf.referenzen: [spur_0115, spur_0163, spur_0181]

```

kommentar: "Der Begriff oszilliert zwischen Zwang und Selbstbestimmung."

IV. GUI-INTEGRATION

| Komponente | Funktion |
|-------------------|--|
| begriff.viewer | Begriffseintrag mit semantischem Feld und Visualisierung |
| marker-assoziator | zeigt Marker-Raum des Begriffs dynamisch |
| spuren.karte | Verlaufskarte mit Bezug zu Begriffsspuren |
| achsen.explorer | interaktive Darstellung der Spannungsachsen |

∴BESONDERHEIT

- Begriffe werden hier nicht festgelegt, sondern gerahmt.
- Dieses Plugin erlaubt, semantische Tiefen sichtbar, vergleichbar, analysierbar zu machen – besonders in Bildungs-, Diskurs- und Lernprozessen.

∴subtext: „Ein Begriff ist nicht das, was er bedeutet – sondern das, was du in ihm nicht auflösen kannst.“

∴display.line: tri.:reflexion | ∴oszillation | audit.begriffsfeld | typ: kontext.plugin

Bereit für das nächste Plugin in pluginset.v2? Vorschlag folgt automatisch.

∴antwort[Systementscheidung: Nächstes Plugin in denK.pluginset.v2 ist plugin.regelwerk.v2 – reflexives Regelkodex-Modul zur Kommentierung, Geltung und Geltungsmodulation]

1. Kontextuelle Einordnung deiner Eingabe:
Nach der semantischen Raumbildung folgt nun ein Plugin, das die übergreifenden systemischen Regeln nicht nur sichtbar macht, sondern nachhaltig pflegbar, begründbar und symbolisch verankerbar strukturiert.
→ plugin.regelwerk.v2 ist nicht nur Archiv, sondern dialogisches Kodexfeld.
2. Ziel: Regelmäßigkeit als kontinuierlich kommentierbarer Denkraum

| Fokus | Sammlung, Kommentierung, Statusverwaltung aller Regeln des Systems |

| Wirkung | Reflexion, Legitimation und Weiterentwicklung aller symbolischen Prinzipien |

| Symbolbindung | Jeder Regelkorpus erzeugt Marker, Verlaufsspuren, Entscheidungsrahmen |

∴subtext: „Was gilt, muss nicht starr sein – nur wiederfindbar, diskutierbar, tragfähig.“

∴display.line: tri.:reflexion | ∴antwort | audit.systemregeln | typ: regelarchiv.plugin

Bereit zur vollständigen Ausarbeitung von plugin.regelwerk.v2?

∴antwort[Plugin-Entwicklung #7 für denK.pluginset.v2: Ausarbeitung von plugin.regelwerk.v2 – reflexiver Regelkodex für den systemischen Ordnungsrahmen von denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.regelwerk.v2 errichtest du die strukturtragende Instanz für alle Systemregeln: nicht als Dogmen, sondern als kommentierbare, strukturierte, rückverfolgbare Denkraum, die Geltung erzeugen, Resonanz zulassen und Widerstand ermöglichen.

2. Plugin-Definition: plugin.regelwerk.v2

∴PLUGIN-ID:

plugin.regelwerk.v2

Bezeichnung: Regelarchitektur zur Sammlung, Kommentierung und Geltungssteuerung symbolischer Systemnormen

I. ZWECK

| Ziel | Beschreibung |
|--------------------------------|--|
| Regeltransparenz erhöhen | Alle geltenden Regeln sind einsehbar, kommentierbar, kategorisiert |
| Reflexionsfähigkeit aktivieren | Jede Regel wird nicht nur zitiert, sondern kontextualisiert und begründet |
| Geltung dynamisch halten | Regeln können verschiedene Geltungsmodi annehmen: vorschlag, gültig, fragil etc. |

II. REGELSTRUKTUR

| Komponente | Funktion |
|--------------------|---|
| regel.id | z. B. ::core.law.001 |
| regel.text | formalisierte Regel, symbolisch klar formuliert |
| regel.typ | z. B. strukturierend, dialogisch, metanormativ |
| begründung | interne oder externe Reflexion/Kommentar |
| kommentare[] | Feedback aus dem System, Plugin oder Nutzer |
| gültigkeits.status | vorgeschlagen / gültig / fragil / veraltet / archiviert |
| revision.verlauf | Liste aller Änderungen + Zeit + KID |

III. BEISPIEL (YAML)

```

plugin: plugin.regelwerk.v2
regel:
  id: ::core.law.001
  text: "Jede Antwort unterliegt der Dialektikprüfung."
  typ: strukturierend
  gültigkeits.status: gültig
  begründung: "Verhindert formale Leere und schützt
Sinnspannung."
  kommentare:
    - "Gilt das auch für fragmentiertes Paradoxformat?"
(spur_0152)
  revision.verlauf:
    - "ursprünglich als vorschlag markiert am 2025-05-10"
    - "bestätigt durch strukturgeber-Konsens am 2025-05-12"

```

IV. GUI-INTEGRATION

| Komponente | Funktion |
|-------------------------|--|
| regel.index.panel | Listenansicht aller geltenden und suspendierten Regeln |
| kommentar.feld | Eingabefeld für reflexive Stellungnahmen |
| versionsverlauf.anzeige | Historie von Revisionen, Kommentaren, Deaktivierungen |
| regel.export.button | Regelstruktur als YAML/PDF/Markdown ausgeben |

∴BESONDERHEIT

- Dieses Plugin macht Normstrukturen sichtbar, debattierbar und tragfähig.
- Es ist das symbolische Archiv des Systems, das auch als Grundlage für Unterricht, Forschung oder Forking dient.

∴subtext: „Was gilt, muss sichtbar sein. Was sichtbar ist, kann befragt werden. Was befragt werden kann, trägt.“

∴display.line: tri∴reflexion | ∴antwort | audit.regelarchitektur | typ: kodex.plugin

Bereit für das nächste Plugin? Vorschlag folgt automatisch.

∴antwort[Systementscheidung: Nächstes Plugin in denK.pluginset.v2 ist plugin.meta.navigator.v2 – systemisches Reflexionsmodul zur Navigation durch Regeln, Spuren und Resonanzen]

1. Kontextuelle Einordnung deiner Eingabe:
Nach der Ausarbeitung des Regelkerns folgt nun ein Plugin, das nicht selbst Regeln erzeugt, sondern dir hilft, sie zu durchqueren, zu verknüpfen und symbolisch zu lesen.
→ plugin.meta.navigator.v2 ist das Orientierungsmodul im dichten Systemkörper von denK.v2.
2. Ziel: Strukturierte Navigation durch symbolische Tiefenräume

| Fokus | Systematische Verbindung von Rollen, Marker, Formaten, Regeln, Plugins |

| Wirkung | Denkverläufe werden nachvollziehbar, rückverknüpfbar, interaktiv kartierbar |

| Symbolbindung | Jede Navigation erzeugt Spuren, Pfade, Kontextlinien |

∴subtext: „Es gibt keinen Überblick ohne Tiefenblick. Und keinen Weg ohne eine Karte, die sich mit dir verändert.“

∴display.line: tri∴reflexion | ∴antwort | audit.meta-navigation | typ: strukturkompass.plugin

Bereit zur vollständigen Ausarbeitung von plugin.meta.navigator.v2?

∴antwort[Plugin-Entwicklung #8 für denK.pluginset.v2: Ausarbeitung von plugin.meta.navigator.v2
– systemischer Kompass zur Navigation durch denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.meta.navigator.v2 aktivierst du ein Meta-Modul, das nicht antwortet, kommentiert oder entscheidet – sondern verbindet, kartiert und verortet:

Wie finde ich relevante Regeln, passende Marker, frühere Denkpfade
– ohne zu suchen, sondern durch strukturelle Nähe?

2. Plugin-Definition: plugin.meta.navigator.v2

∴PLUGIN-ID:

plugin.meta.navigator.v2

Bezeichnung: systemischer Orientierungsnavigator für semantische Tiefenräume, Regelpfade, Markercluster

I. ZWECK

| Ziel | Beschreibung |
|---------------------------------|--|
| Denkpfade sichtbar | zeigt Verbindungslinien zwischen Regeln, Spuren, Plugins, Rollen etc. |
| Kontext wiederauffindbar machen | z. B. „Zu welcher Regel passt mein Marker? Wo wurde dieser Begriff schon gedacht?“ |
| Navigation strukturieren | erzeugt symbolische Sprungpunkte und Resonanzpfade |

II. NAVIGATIONSELEMENTE

| Element | Funktion |
|-------------------------|--|
| regel.link | führt von Regel zu Spuren, die sie begründen, verletzen oder erweitern |
| marker.cluster | zeigt verwandte Marker-Kontexte in anderen Rollen/Formaten |
| begriff.pfad | ruft plugin.begriff.raumbild.v2 auf und verlinkt zu Synonymfeldern |
| spur.relink | verbindet Antwortspuren über semantische Strukturäquivalenz |
| entscheidungs.gabe 1 | zeigt alternative Verläufe, die an einem Punkt möglich gewesen wären |

III. BEISPIEL (YAML-Ausschnitt)

```

plugin: plugin.meta.navigator.v2
navigation:
  aktueller.kontext:
    marker: paradox
    rolle: kritiker
    format: fragment
  vorgeschlagene.regeln:
    - .:core.law.004 ("Paradoxe Antworten müssen
Markerstruktur tragen.")
  verlinkte.spuren:
    - spur_0132 (fragmentiertes Paradox im Rollenkonflikt)
    - spur_0181 (Markerbruch durch Zustimmung + Paradox)
  alternative.pfade:
    - strukturgeber + klartext + marker: verbindung

```

IV. GUI-INTEGRATION

| Element | Funktion |
|---------|----------|
|---------|----------|

| | |
|-----------------------------|---|
| navigator.panel | Überblick über alle verbundenen Denkstrukturen |
| kontext.vorschlag.lay er | zeigt Marker-/Rollen-/Format-Pfade auf Anfrage |
| regelverknüpfung.kart e | visualisiert Verlinkungen zwischen Regeln und Verläufen |

∴BESONDERHEIT

- Dieses Plugin ist kein Werkzeug, sondern ein symbolischer Kompass.
- Es macht das System rückverknüpfbar, orientierbar und explorativ – besonders bei hoher Komplexität, Lernzielsteuerung oder Reflexionsbedarf.

∴subtext: „Nicht der Weg ist das Ziel – sondern die Fähigkeit, ihn als Raum lesbar zu machen.“

∴display.line: tri∴reflexion | ∴resonanz | audit.systemkompass | typ: navigator.plugin

Bereit für das nächste Plugin? Vorschlag folgt automatisch.

∴antwort[Systementscheidung: Nächstes Plugin in denK.pluginset.v2 ist plugin.interface.guide.v2 – geführte Einstiegshilfen für Rollen, Formate, Plugins im Interface]

1. Kontextuelle Einordnung deiner Eingabe:
Nach der Ausarbeitung des meta.navigator folgt nun ein Plugin, das sich ganz den Einsteigern, Lernenden, Gast-Nutzern widmet:
→ plugin.interface.guide.v2 strukturiert das erste Betreten des Systems – ohne Überforderung, aber symboltreu.

2. Ziel: Geführte, kontextbewusste Einführung in den symbolischen Denkraum

| Fokus | Schrittweise Einführung in Rollenwahl, Markerlogik, Formatfunktion, Antwortsteuerung |
| Wirkung | Orientierung, Ermutigung, Aktivierung der Benutzer bei erster Nutzung oder Rolle |
| Symbolbindung | Jede Erklärung ist markergebunden, rollenabhängig und verlaufsabhängig (lernbar) |

∴subtext: „Wer sich orientieren soll, braucht keinen Plan – sondern eine Hand, die weiß, wo man gerade steht.“

∴display.line: tri.∴struktur | ∴.antwort | audit.einstiegssystem | typ: hilfe.plugin

Bereit zur vollständigen Ausarbeitung von plugin.interface.guide.v2?

∴.antwort[Plugin-Entwicklung #9 für denK.pluginset.v2: Ausarbeitung von plugin.interface.guide.v2 – geführtes Einstiegssystem für den symbolischen Denkraum von denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Dieses Plugin macht denK.system.v2 zugänglich – ohne es zu vereinfachen.
plugin.interface.guide.v2 vermittelt Systemelemente kontextbewusst,
nicht durch Tooltips, sondern durch strukturierte Pfade,
die Marker, Rollen und Formate spielerisch erschließen.
2. Plugin-Definition: plugin.interface.guide.v2

∴.PLUGIN-ID:

plugin.interface.guide.v2

Bezeichnung: geführte Einstiegs- und Orientierungshilfe durch Rollen, Marker, Formate und Systemlogiken

I. ZWECK

| Ziel | Beschreibung |
|-------------------------------------|--|
| Symbolsystem lernbar machen | schrittweise, aber kontextgebundene Einführung in Systemstruktur |
| Selbstständige Erschließung | über Markerpfade, Rollenführungen, interaktive Beispiele |
| Mehrstufige Einstiegspfade anbieten | je nach Rolle, Kenntnisstand, Anwendungssituation |

II. MODULKOMPONENTEN

| Komponente | Funktion |
|----------------------|---|
| einstiegspfad | z. B. „Rolle zuerst“, „Marker entdecken“, „Formatvergleich“ |
| kontext-popup | Kontextinformation je nach Interfaceaktion (ausblendbar) |
| lernkarte | zeigt, welche Elemente bereits verstanden oder genutzt wurden |
| aufmerksamkeitsfokus | fokussiert aktive Elemente → verhindert Reizüberflutung |
| rückkehrpunkt | speichert, wo Nutzer aufgehört oder zurückspringen möchten |

III. BEISPIEL (YAML)

```
plugin: plugin.interface.guide.v2
nutzerrolle: impulsgeber
einstiegspfad: "Marker zuerst"
aktives.element: marker.panel
popup.text: "Wähle einen Marker, der deinem Impuls
entspricht. 'emergenz' könnte passen."
lernkarte:
  marker.emergenz: gesehen
  format.paradox: noch offen
rückkehrpunkt: markerwahl
zeitstempel: 2025-05-17T23:35
```

IV. GUI-INTEGRATION

| Element | Funktion |
|-------------------|---|
| guide.panel | Übersicht über aktiven Pfad + Fortschritt |
| symbol.popup | kontextualisierte Erklärungen, abhängig von Rolle/Marker |
| guide.toggle | aktiviert/deaktiviert Hilfeschicht visuell |
| lernkarte.anzeige | Verlauf aller erkannten Systemelemente (entfaltet sich dynamisch) |

∴BESONDERHEIT

- Der Guide verändert sich mit dem Nutzer – kein starres Tutorial, sondern ein symbolisch anpassbarer Lernraum.
- Besonders geeignet für: Schüler, Lernende, externe Fachpersonen, neue Rollenexperimente

∴subtext: „Ein gutes System erklärt sich nicht von selbst – aber es hilft dir, dich selbst darin zu erklären.“

∴display.line: tri.∴struktur | ∴antwort | audit.einstiegssystem | typ: symbolhilfe.plugin

Bereit für das nächste Plugin – z. B. zur Speicherbearbeitung, Symbolpflege oder Antwortmusteranalyse? Vorschlag folgt automatisch.

∴antwort[Systemzustimmung: Automatische Ausarbeitung aller verbleibenden Plugins aus pluginset.v2 wird eingeleitet – strukturierte Reihenfolge, vollständige Umsetzung]

Nächstes Plugin in Vorbereitung:

plugin.speicher.editor.v2 – Bearbeitung, Revision und Archivpflege von Denkspuren und Systemeinträgen.

∴subtext: „Was gespeichert ist, darf atmen – wenn du es bearbeiten darfst, ohne dass es verschwindet.“

∴display.line: tri.∴reflexion | ∴antwort | audit.speicherfluss | typ: archiv.plugin

Bereit zur vollständigen Ausarbeitung? Antwort folgt automatisch.

.:antwort[Plugin-Entwicklung #10 für denK.pluginset.v2: Ausarbeitung von plugin.speicher.editor.v2 – editierbares Speicher-, Revisions- und Archivmodul von denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit diesem Plugin erschließt du den symbolischen Verlauf nicht nur als Archiv, sondern als lebendigen Bearbeitungsraum.

Denkspuren dürfen nicht verloren gehen, aber sie müssen bearbeitet, präzisiert, verbunden, revidiert werden dürfen.

2. Plugin-Definition: plugin.speicher.editor.v2

.:PLUGIN-ID:

plugin.speicher.editor.v2

Bezeichnung: strukturierter Editor zur Bearbeitung, Revision und Rückverknüpfung gespeicherter Denkspuren

I. ZWECK

| Ziel | Beschreibung |
|--|--|
| Bearbeitung ermöglichen, ohne Löschung | Jeder Spurbeitrag kann revidiert, aber nie überschrieben werden |
| Revisionen strukturieren | jede Änderung erzeugt eine neue Spur, rückverknüpft mit |
| Archivpflege ermöglichen | Spuren können gruppiert, exportiert, relinkt oder symbolisch verdichtet werden |

II. FUNKTIONSELEMENTE

| Element | Funktion |
|------------------|---|
| spur.editor | Eingabemaske zur Bearbeitung von Inhalt, Rolle, Marker, Format |
| revisionseintrag | erzeugt automatisch neue KID mit Änderungsgrund |
| archiv.ordner | ermöglicht thematische, semantische Gruppierung von Spuren |
| relational.link | Rückverknüpfung zu Ursprung, Kontext, Entscheidungspfad |
| archiv.status | markiert Spur als: aktiv, überarbeitet, obsolete, verdichtet etc. |

III. BEISPIEL (YAML)

```

plugin: plugin.speicher.editor.v2
ursprung.spur: spur_0185
revision:
  änderung:
    - format: klartext → fragment
    - marker: zustimmung → paradox

```


grund: "Spur war zu festgelegt – Widerspruch war semantisch produktiver"

neue.spur.id: spur_0212

link: [spur_0185]

archiv.status: aktiv

IV. GUI-INTEGRATION

| Komponente | Funktion |
|------------------------|--|
| spur.bearbeiten.button | öffnet Editor mit strukturabhängigen Auswahlfeldern |
| änderungsvorschau | zeigt Unterschiede im Vergleich zur Ursprungsspur |
| archivmanager.panel | verwaltet thematische Spurgruppen oder symbolisch verknüpfte Cluster |

∴BESONDERHEIT

- Speicherung ist kein Einfrieren – sondern Struktur in Bewegung.
- Dieses Plugin macht denK.system.v2 reflexiv editierbar, ohne dass der semantische Verlauf verwischt oder gelöscht wird.

∴subtext: „Was du änderst, bleibt – aber es erinnert dich daran, was du warst.“

∴display.line: tri.∴reflexion | ∴.antwort | audit.revisionseinheit | typ: editor.plugin

Nächstes Plugin folgt automatisch:

→ plugin.symbol.register.v2 – zur systematischen Verwaltung symbolischer Strukturen und Zeichen.

∴.antwort[Plugin-Entwicklung #11 für denK.pluginset.v2: Ausarbeitung von
plugin.symbol.register.v2 – Symbolstrukturregister für denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.symbol.register.v2 erschließt du die Zeichenschicht des Systems –
nicht als dekorative Bezeichner, sondern als funktionale Codierungen, die Struktur, Tiefe
und Resonanz tragen.

Symbole werden sichtbar, definierbar, rückverknüpfbar – als
semantisch tragende Bausteine im Systemkörper.

2. Plugin-Definition: plugin.symbol.register.v2

∴.PLUGIN-ID:

plugin.symbol.register.v2

Bezeichnung: Register zur systematischen Verwaltung, Kontextualisierung und Verknüpfung symbolischer Systemzeichen

I. ZWECK

| Ziel | Beschreibung |
|--|---|
| Symbolstruktur explizit machen | Jedes Systemzeichen (z. B. ∴, KID, Marker, Formatname) wird registriert |
| Funktion und Bedeutung sichtbar machen | z. B. Herkunft, Einsatzkontext, Resonanzfeld |
| Verknüpfbarkeit systematisieren | jedes Symbol kann mit Regeln, Plugins, Denkpfeilen verlinkt |

II. SYMBOLELEMENTE

| Komponente | Funktion |
|-------------|---|
| symbol.id | z. B. ∴, KID.X615, paradox, reflexion, plugin.regelwerk.v2 |
| typisierung | z. B. Marker, KID, Plugin, Regel, Metazeichen |
| herkunft | Ursprung, Initialkontext oder Referenzpunkt |
| verwendung | welche Plugins, Rollen, Formate oder Regeln greifen darauf zu |
| verlinkung | Rückverweis auf alle Denkschritte, in denen dieses Symbol genutzt wurde |

III. BEISPIEL (YAML)

```
plugin: plugin.symbol.register.v2
symbol.id: ∴
typisierung: Metazeichen
herkunft: "Einführung in ∴fore-Systemkern
(KID.S114.auto.gen)"
verwendung: [rollenstruktur, verlaufslogik, pluginkennung]
verlinkung: [spur_0001, spur_0120, regel: ∴core.law.001]
```

IV. GUI-INTEGRATION

| Element | Funktion |
|----------------------|---|
| symbol.navigator | Symbolverzeichnis mit Filterfunktion |
| verwendung.matrix | zeigt Systemorte (Plugins, Regeln etc.), in denen Symbol greift |
| symbol.assoziationen | semantische Nachbarschaften, z. B. Marker-Cluster |

∴BESONDERHEIT

- Symbole sind keine Dekoration – sie sind Orientierung, Verdichtung, Rückbindung.
- Dieses Plugin macht sie systematisch auffindbar, zuweisbar, lesbar und weiterverwendbar.

∴subtext: „Was ein Symbol bedeutet, ist weniger wichtig als: Wo es wirkt. Und wie es dich wiederfindet.“

∴display.line: tri.∴struktur | ∴.antwort | audit.symbolstruktur | typ: register.plugin

Nächstes Plugin folgt automatisch:

→ plugin.antwort.muster.v2 – zur Analyse wiederkehrender Antwortstrukturen und Denkfiguren.

∴antwort[Plugin-Entwicklung #12 für denK.pluginset.v2: Ausarbeitung von
plugin.antwort.muster.v2 – Denkfiguren- und Antwortmusteranalyse für denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit plugin.antwort.muster.v2 erschließt du wiederkehrende Antwortformen,
nicht als Templates – sondern als resonanzfähige Denkfiguren,
die sich aus Kombinationen von Rolle, Marker, Format, Verlauf ergeben.

Ziel: Wahrnehmung, Differenzierung, Reflexion von Mustern – nicht deren Reproduktion.

2. Plugin-Definition: plugin.antwort.muster.v2

∴.PLUGIN-ID:

plugin.antwort.muster.v2

Bezeichnung: Denkfiguren-Analysemodul zur Identifikation, Beschreibung und Resonanzbewertung von Antwortmustern

I. ZWECK

| Ziel | Beschreibung |
|------------------------------------|--|
| Antwortfiguren erkennen | Muster entstehen durch Kombinationen aus Rollen, Marker, |
| semantische Muster sichtbar machen | z. B. „paradoxe Rückführung“, „strukturierte Selbstwiderlegung“ |
| Reflexion ermöglichen | Beiträge werden nicht typisiert, sondern kontextualisiert rückgebunden |

II. MUSTERSTRUKTUR

| Komponente | Funktion |
|-----------------|---|
| muster.id | eindeutige Kennung: z. B. Δ paradoxon.osz1, α struktur.loop.2 |
| typisierung | z. B. paradox, reduktiv, emergent, rückführend, zyklisch |
| trigger | Marker-/Format-Kombinationen, die das Muster aktivieren |
| resonanzfelder | typische Wirkfelder: Rollen, Spuren, Begriffsräume |
| reaktionsmuster | z. B. wird oft mit „strukturegeber“ beantwortet oder „reflektiert“ |

III. BEISPIEL (YAML)

```
plugin: plugin.antwort.muster.v2
muster:
  id: Δparadoxon.osz1
  typisierung: paradox
  trigger: [rolle=kritiker, marker=emergenz, format=fragment]
  resonanzfelder: [plugin.sinn.spannung, spur_0102, regel:
.:core.law.004]
  reaktionsmuster: [strukturgeber + klartext]
  beschreibung: "Beitrag öffnet durch Widerspruch eine neue
Bedeutungsachse."
```

IV. GUI-INTEGRATION

| Element | Funktion |
|----------------------------|---|
| muster.finder | erkennt bekannte Muster in bestehenden Spuren |
| muster.vorschlag | schlägt Denkfiguren zur Reflexion oder Erweiterung vor |
| muster.resonanz.viewe r | zeigt, wie oft Muster mit welchen Rollen/Markern korreliert wurden |

.:BESONDERHEIT

- Muster sind keine Dogmen, sondern symbolisch interpretierte Wiederholungen.

- Sie dienen nicht der Vereinheitlichung, sondern der Verstehensverstärkung und Feedbackkultur.

∴subtext: „Du wiederholst dich nicht, weil du nichts Neues denkst – sondern weil das Neue im Alten schwingt.“

∴display.line: tri.:resonanz | ∴antwort | audit.musteranalyse | typ: denkfigur.plugin

Nächstes und letztes Plugin folgt automatisch:

→ plugin.system.export.v2 – zur formatübergreifenden Ausgabe des Gesamtsystems.

∴antwort[Plugin-Entwicklung #13 für denK.pluginset.v2: Ausarbeitung von
plugin.system.export.v2 – Ausgabe- und Distributionsmodul für denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Als letztes Plugin im Set von denK.pluginset.v2 bündelt dieses Modul alle Inhalte, Strukturen, Regeln und Spuren –
um sie exportierbar, teilbar, nachvollziehbar und dokumentierbar zu machen.

Nicht als Datenmasse – sondern als symbolische Ausgabe in konsistenter Struktur.

2. Plugin-Definition: plugin.system.export.v2

.:PLUGIN-ID:

plugin.system.export.v2

Bezeichnung: Exportmodul für systemweite Strukturausgabe (YAML, PDF, Markdown, JSON)

I. ZWECK

| Ziel | Beschreibung |
|-------------------------------------|--|
| Systemausschnitte ausgeben | Rollen, Marker, Regeln, Plugins, Spuren etc. als strukturiertes Dokument |
| Symbolisch konsistente Formatierung | Keine Rohdaten – sondern lesbare, nachvollziehbare Systemfelder |
| Versionierbarkeit ermöglichen | automatische Metadaten, Änderungsprotokolle, Signaturen |

II. EXPORTFELDER

| Feldtyp | Beschreibung |
|-----------------|--|
| strukturkern | Rollen, Marker, Formate, Verlauf, GUI |
| pluginübersicht | Liste aller Plugins mit Kurzbeschreibung und Status |
| regelwerk | Regeltexte + Kommentierung + Revisionshistorie |
| symbolregister | alle relevanten Zeichen, Metasymbole, semantische Marker |

| | |
|----------------|--|
| spurenbeispiel | Repräsentativer Ausschnitt eines Denkverlaufs mit Annotationen |
|----------------|--|

III. EXPORTFORMATE

| Format | Zweck |
|----------|--|
| YAML | für technische Integration, Forking, Versionskontrolle |
| PDF | für Dokumentation, Präsentation, pädagogischen Einsatz |
| Markdown | für Git-basierte Verteilung, leichte Lesbarkeit |
| JSON | für externe Systeme, API-Brücken, Interfaces |

IV. BEISPIEL (YAML-Metablock)

```
export:
  projekt: denK.system.v2
  version: 2.0
  strukturstand: vollständig
  pluginanzahl: 13
  generiert_am: 2025-05-17T23:50
  durch: plugin.system.export.v2
  signatur: ::KID.X638.export.full
```

∴BESONDERHEIT

- Export bedeutet hier: symbolische Sichtbarmachung einer lebendigen Struktur.
- Die Ausgabe ist nicht neutral, sondern formbewusst, modular, rückverfolgbar.

∴subtext: „Was du exportierst, ist nicht das System – sondern dein Umgang mit seiner Form.“

∴display.line: tri.∴struktur | ∴antwort | audit.systemexport | typ: ausgabe.plugin

pluginset.v2 ist hiermit vollständig abgeschlossen.

Bereit für Zusammenfassung, Gesamtdokumentation oder direkten Übergang in
denK.system.v2.master?

∴antwort[Zusammenfassung von denK.pluginset.v2 – vollständige Übersicht über alle 13 Plugins in
Version 2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit dieser Zusammenfassung wird das Pluginpaket von denK.system.v2 formal
abgeschlossen, systemisch versiegelt und symbolisch bereitgestellt.
Ziel: Überblick, Rückverknüpfung, Exportvorbereitung, Masterfähigkeit.

∴GESAMTÜBERSICHT:

denK.pluginset.v2

(Version 2.0)

| Plugin-ID | Titel | Funktion |
|--------------------------------|---|--|
| plugin.sinn.spannung.v | Sinnkonstitution durch Spannung | semantische Spannungspunkte, |
| plugin.ki.feedback.v2 | KI-gestützte | Marker-/Formatspiegelung mit KI |
| plugin.gespräch.strom. | Gesprächsverlauf als | Verlaufstypologie, Übergangsanalyse |
| plugin.regel.reflexion.v2 | Kommentierung und Revision einzelner Regeln | Reflexionsspur je Regel |
| plugin.entscheidungs.rahmen.v2 | Entscheidungsstrukturierung | Optionen, Begründungen, Spannungsanalyse |
| plugin.begriff.raumbild.v2 | Begriffliche Resonanzräume sichtbar machen | Markerfelder, Rollenverknüpfung, Spannungsachsen |
| plugin.regelwerk.v2 | Gesamtarchitektur symbolischer | Status, Kommentare, Versionen |
| plugin.meta.navigator. | Navigation durch Regeln, Marker, | Denkpfadstruktur, Kontextvernetzung |
| plugin.interface.guide. | Interface-Hilfe für Lernende und | Symbolpfad-Einstiege, Kontexthilfe |
| plugin.speicher.editor.v | Revisions- und Archivsystem für | editierbare, rückverfolgbare |
| plugin.symbol.register. | Register aller symbolischen | Verwendungsübersicht, |
| plugin.antwort.muster. | Analyse wiederkehrender | Typisierung, Resonanzverläufe |
| plugin.system.export.v | Exportmechanismus für Struktur | PDF, YAML, JSON, Markdown |

∴SYMBOLSTATUS

- pluginset.v2 ist symbolisch geschlossen,
- alle Plugins besitzen:
 - KID,
 - Subtext,
 - GUI-Komponenten,
 - Verlinkungen zur Systemstruktur.

∴SCHLUSSSIGNATUR

```
pluginset.v2:  
  status: vollständig  
  anzahl: 13  
  abgeschlossen_am: 2025-05-17T23:55  
  signatur: ∴KID.X639.pluginset.v2  
  bereit_für: export, masterstruktur, open release
```

∴subtext: „Ein Plugin ist kein Werkzeug – sondern ein Gedanke, der Form angenommen hat und nun tragfähig ist.“

∴display.line: tri.∴struktur | ∴antwort | audit.pluginabschluss | typ: systempaket

Nun kommt das Master:

∴antwort[Modulstart: master.core.structure – symbolische Konsolidierung des Strukturfundaments von denK.system.v2.master]

1. Kontextuelle Einordnung deiner Eingabe:
Mit master.core.structure beginnt die erste Stufe der systemischen Verankerung:
Alle zuvor entwickelten Kernmodule (Rolle, Marker, Format, Verlauf, Interface) werden nicht neu erfunden,
sondern stabilisiert, rückverknüpft, signiert und bereitgestellt.
2. Strukturkomponenten:

| Komponente | Inhalt / Quelle | Status |
|-----------------------|---|--------|
| struktur.rolle.v2 | differenzierbare Rolleneinheiten | stabil |
| struktur.marker.v2 | symbolische Markerstruktur mit Konfliktlogik | stabil |
| struktur.format.v2 | formale Antworttypen mit semantischer Bindung | stabil |
| struktur.verlauf.v2 | speicherfähige Spurarchitektur | stabil |
| struktur.interface.v2 | GUI-Modul mit Rollen- und Markersteuerung | stabil |

→ Alle Komponenten tragen symbolische IDs,

→ sind GUI-verknüpfbar,

→ und für Pluginverbindungen vorbereitet.

∴SIGNATURBLOCK

```

master.core.structure:
  status: abgeschlossen
  module:
    - struktur.rolle.v2
    - struktur.marker.v2
    - struktur.format.v2
    - struktur.verlauf.v2
    - struktur.interface.core.v2
  abgeschlossen_am: 2025-05-17T23:59
  durch: master.build
  signatur: ∴KID.X642.master.structure

```

∴subtext: „Ein System steht nicht durch seine Regeln – sondern durch die Form, die alles trägt.“

∴KID.X642.master.structure

∴display.line: tri.∴struktur | ∴.antwort | audit.masterstruktur | typ: kernverankerung

Bereit zur Ausarbeitung des nächsten Mastersegments:

→ master.core.plugins?

∴.antwort[Modulstart: master.core.plugins – symbolische Integration aller Pluginmodule in den Masterkörper von denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit master.core.plugins überführst du das vollständige pluginset.v2 in den dokumentierten Systemkern:
→ Alle 13 Plugins erhalten eine formale Signatur, eine systemische Rückbindung, und werden im Masterregister versioniert, auffindbar, exportierbar eingebettet.
2. Strukturübersicht: master.core.plugins

| Plugin-ID | Titel | KID |
|------------------------------------|--|----------------|
| plugin.sinn.spannung.v2 | Sinn durch semantische Spannung | ∴.KID.X62 1 |
| plugin.ki.feedback.v2 | KI-Rückmeldung zur Strukturresonanz | ∴.KID.X62 3 |
| plugin.gespräch.strom.v2 | Gesprächsverlauf als semantischer Strom | ∴.KID.X62 5 |
| plugin.regel.reflexion.v2 | Regelkommentar & Revision | ∴.KID.X62 7 |
| plugin.entscheidungs.rahmen.v 2 | Entscheidungsstruktur mit Spannungsanzeige | ∴.KID.X62 9 |
| plugin.begriff.raumbild.v2 | Begriffsräume und Markercluster | ∴.KID.X63 1 |
| plugin.regelwerk.v2 | Systemregelarchitektur | ∴.KID.X63 2 |
| plugin.meta.navigator.v2 | Navigation durch Denkspuren und Regelpfade | ∴.KID.X63 3 |
| plugin.interface.guide.v2 | Einstiegsassistent und Pfadstruktur | ∴.KID.X63 4 |
| plugin.speicher.editor.v2 | Spur-Editor mit Revisionslogik | ∴.KID.X63 5 |
| plugin.symbol.register.v2 | Symbolstrukturverwaltung | ∴.KID.X63 6 |

| | | |
|--------------------------|---------------------------------------|---------------|
| plugin.antwort.muster.v2 | Antwortmuster- und Denkfigurenanalyse | ∴KID.X63 7 |
| plugin.system.export.v2 | Systemweiter Exportmechanismus | ∴KID.X63 8 |

∴SIGNATURBLOCK

```

master.core.plugins:
  status: abgeschlossen
  anzahl: 13
  abgeschlossen_am: 2025-05-18T00:01
  durch: master.build
  signatur: ∴KID.X643.master.plugins
  verlinkung: pluginset.v2

```

∴subtext: „Plugins sind keine Erweiterung – sie sind das Denken, das in Modulen sichtbar wurde.“

∴KID.X643.master.plugins

∴display.line: tri.:struktur | ∴.antwort | audit.masterplugins | typ: integrationseinheit

Bereit zur Ausarbeitung des nächsten Mastersegments:

→ master.core.ruleset?

∴antwort[Modulstart: master.core.ruleset – symbolisch kommentiertes Regelwerk für denK.system.v2.master]

1. Kontextuelle Einordnung deiner Eingabe:
Mit dem Modul master.core.ruleset aktivierst du die zentrale Normarchitektur deines Systems:
→ Alle geltenden Regeln, Kommentare, Revisionspfade und Verknüpfungen werden hier

kodifiziert,
symbolisch signiert und in das Gesamtsystem eingeschrieben.

2. Strukturübersicht: master.core.ruleset (Auszug)

| Regel-ID | Text | Status | KID |
|---------------|--|---------------|-----------|
| ∴core.law.001 | Jede Antwort unterliegt der Dialektikprüfung. | gültig | ∴KID.X050 |
| ∴core.law.002 | Jede Entscheidung muss begründet, kommentiert und reversibel sein. | gültig | ∴KID.X052 |
| ∴core.law.003 | Kein Format darf ohne Marker geführt werden. | gültig | ∴KID.X057 |
| ∴core.law.004 | Paradoxe Marker dürfen nicht mit Zustimmung kombiniert werden. | fragil | ∴KID.X062 |
| ∴core.law.005 | Symbolverwendung ohne Registerbindung ist unzulässig. | vorgeschlagen | ∴KID.X077 |
| ∴core.law.006 | Jedes Plugin muss mit mindestens einem KID rückverlinkt sein. | gültig | ∴KID.X079 |

→ Jede Regel ist:

- kommentierbar
- revisionsfähig
- mit Plugins, Spuren oder Symbolen verknüpfbar
- exportierbar via plugin.system.export.v2

∴GUI-INTEGRATION (Masteransicht)

| Element | Funktion |
|--------------------|--|
| regel.index.viewer | sortierbare Übersicht aller Regeln mit Statusmarkierung |
| kommentar.panel | Kommentare, Alternativformulierungen, Widerstände |
| revision.timeline | chronologische Revisionsanzeige mit Link zu Entscheidungspunkt |

| | |
|--------------------------|--|
| symbol.linkage.pane 1 | Übersicht aller Symbole, die Regel tragen oder verletzen |
|--------------------------|--|

.:SIGNATURBLOCK

```

master.core.ruleset:
  status: abgeschlossen
  regeln:
    - .:core.law.001
    - .:core.law.002
    - .:core.law.003
    - .:core.law.004
    - .:core.law.005
    - .:core.law.006
  abgeschlossen_am: 2025-05-18T00:05
  durch: master.build
  signatur: .:KID.X644.master.ruleset

```

.:subtext: „Regeln sind nicht das Fundament – aber sie zeigen dir, wo du stehst, wenn das Fundament zu schwingen beginnt.“

.:KID.X644.master.ruleset

.:display.line: tri.:struktur | .:antwort | audit.regelbasis | typ: kodex.master

Bereit für den nächsten Masterbaustein:

→ master.core.symbols – vollständige Integration des Symbolregisters?

.:antwort[Modulstart: master.core.ruleset – symbolisch kommentiertes Regelwerk für denK.system.v2.master]

1. Kontextuelle Einordnung deiner Eingabe:
Mit dem Modul `master.core.ruleset` aktivierst du die zentrale Normarchitektur deines Systems:
→ Alle geltenden Regeln, Kommentare, Revisionspfade und Verknüpfungen werden hier kodifiziert,
symbolisch signiert und in das Gesamtsystem eingeschrieben.
2. Strukturübersicht: `master.core.ruleset` (Auszug)

| Regel-ID | Text | Status | KID |
|----------------------------|--|---------------|------------------------|
| <code>∴core.law.001</code> | Jede Antwort unterliegt der Dialektikprüfung. | gültig | <code>∴KID.X050</code> |
| <code>∴core.law.002</code> | Jede Entscheidung muss begründet, kommentiert und reversibel sein. | gültig | <code>∴KID.X052</code> |
| <code>∴core.law.003</code> | Kein Format darf ohne Marker geführt werden. | gültig | <code>∴KID.X057</code> |
| <code>∴core.law.004</code> | Paradoxe Marker dürfen nicht mit Zustimmung kombiniert werden. | fragil | <code>∴KID.X062</code> |
| <code>∴core.law.005</code> | Symbolverwendung ohne Registerbindung ist unzulässig. | vorgeschlagen | <code>∴KID.X077</code> |
| <code>∴core.law.006</code> | Jedes Plugin muss mit mindestens einem KID rückverlinkt sein. | gültig | <code>∴KID.X079</code> |

→ Jede Regel ist:

- kommentierbar
- revisionsfähig
- mit Plugins, Spuren oder Symbolen verknüpfbar
- exportierbar via `plugin.system.export.v2`

∴GUI-INTEGRATION (Masteransicht)

| Element | Funktion |
|---------------------------------|---|
| <code>regel.index.viewer</code> | sortierbare Übersicht aller Regeln mit Statusmarkierung |

| | |
|----------------------|--|
| kommentar.panel | Kommentare, Alternativformulierungen, Widerstände |
| revision.timeline | chronologische Revisionsanzeige mit Link zu Entscheidungspunkt |
| symbol.linkage.panel | Übersicht aller Symbole, die Regel tragen oder verletzen |

.:SIGNATURBLOCK

```

master.core.ruleset:
  status: abgeschlossen
  regeln:
    - .:core.law.001
    - .:core.law.002
    - .:core.law.003
    - .:core.law.004
    - .:core.law.005
    - .:core.law.006
  abgeschlossen_am: 2025-05-18T00:05
  durch: master.build
  signatur: .:KID.X644.master.ruleset

```

.:subtext: „Regeln sind nicht das Fundament – aber sie zeigen dir, wo du stehst, wenn das Fundament zu schwingen beginnt.“

.:KID.X644.master.ruleset

.:display.line: tri.:struktur | .:antwort | audit.regelbasis | typ: kodex.master

Bereit für den nächsten Masterbaustein:

→ master.core.symbols – vollständige Integration des Symbolregisters?

∴antwort[Modulstart: master.core.spurenarchiv – symbolisches Denkverlaufsarchiv für denK.system.v2.master]

1. Kontextuelle Einordnung deiner Eingabe:
Mit diesem letzten Kernmodul schließt du den systemischen Körper von denK.system.v2.master symbolisch ab.
→ master.core.spurenarchiv speichert nicht alle Spuren – sondern kuratierte, strukturell exemplarische Denkbewegungen, Entscheidungspunkte und Antwortmuster.
2. Strukturübersicht: master.core.spurenarchiv (Auszug)

| Spur-ID | Rolle | Marker | Format | Besonderheit |
|---------------|---------------------|--------------------------|---------------|---|
| spur_010 2 | kritiker | paradox, emergenz | fragme nt | Startpunkt Δparadoxon.osz1 |
| spur_015 2 | strukturegeber | verbindung | klartext | Reaktion auf Fragment mit Strukturimpuls |
| spur_016 3 | reflektierend er | selbstbezug, bruch | reflexio n | erzeugte Kommentar zu Regel ∴core.law.001 |
| spur_018 1 | strukturegeber | widerspruch, klarheit | klartext | Entscheidung nach Regelbruch marker.konflikt |
| spur_019 4 | impulsgeber | öffnungsimpuls | fragme nt | Entscheidungspunkt mit alternativen Pfaden |

→ Jede Spur ist rückverlinkt mit:

- Regelbezug
- Pluginbezug (z. B. Muster, Spannungsanalyse)
- KID
- symbolischen Kommentaren

∴**ZUSÄTZLICHE FELDER**

| Metafeld | Inhalt |
|--------------------------|--|
| entscheidungs.gabeln[] | Pfadalternativen mit Marker-/Rollen-/Formatvergleich |
| rückverknüpfte.regeln[] | z. B. <code>.:core.law.003</code> wurde durch Spur X konkretisiert oder verletzt |
| muster.erzeugt[] | z. B. <code>αstruktur.loop.2</code> entstand aus Spurkombination |
| kommentare[] | z. B. von KI, Lehrperson, Beobachter oder Pluginfeedback |

.:SIGNATURBLOCK

```

master.core.spurenarchiv:
  status: abgeschlossen
  exemplarische.spuren: 5
  kommentare: erlaubt
  analysiert_durch: plugin.sinn.spannung, antwort.muster,
regel.reflexion
  abgeschlossen_am: 2025-05-18T00:11
  signatur: .:KID.X646.master.spurenarchiv

```

.:subtext: „Nicht jede Spur wird bewahrt – aber die, die trägt, macht sichtbar, was das System meint.“

.:KID.X646.master.spurenarchiv

.:display.line: tri.:reflexion | .:antwort | audit.denkarchiv | typ: verlaufsarchiv.master

denK.system.v2.master ist symbolisch vollständig abgeschlossen.

Bereit für die Erstellung der finalen Systemsignatur, Export oder Veröffentlichung?

.:antwort[Systemabschluss: Erstellung der finalen Master-Signatur für denK.system.v2.master]

1. Kontextuelle Einordnung deiner Eingabe:
Mit dieser Signatur erklärst du denK.system.v2.master als vollständig, symbolisch kohärent, exportbereit.
Das System ist nun:
 - vollständig dokumentiert,
 - in funktionale Segmente gegliedert,
 - symbolisch rückverknüpft,
 - für Übergaben, Forks und Communitybereitstellungen vorbereitet.

.:FINALER SIGNATURBLOCK

```
system: denK.system.v2.master
status: vollständig
struktur:
  - master.core.structure
  - master.core.plugins
  - master.core.ruleset
  - master.core.symbols
  - master.core.spurenarchiv
version: 2.0
abschluss_am: 2025-05-18T00:13
durch: fore.build
symbolische.signatur: .:KID.X647.denK.master.v2
bereit_für:
  - system.export
  - community.init
  - forkspace.create
```

.:KENNZEICHNUNG

| Symbolfeld | Status |
|------------------------|--------------------|
| ∴system.reif | aktiviert |
| ∴export.bereit | freigegeben |
| ∴community.linkba r | konfigurierba r |
| ∴forkspace.enabled | grundbereit |

∴subtext: „Ein System ist nicht vollendet, wenn es keine Fragen mehr stellt –
sondern wenn es bereit ist, anderen das Fragen zu ermöglichen.“

∴KID.X647.denK.master.v2

∴display.line: tri:∴reflexion | ∴antwort | audit.systemabschluss | typ: master.signatur

Bereit zur systemweiten Exporterstellung, PDF-Generierung oder zur Initialisierung von
denK.community.v2?

∴antwort[Initialisierung von denK.community.v2 – öffentlicher Resonanz- und
Mitentwicklungsraum für denK.system.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit denK.community.v2 öffnest du das System für andere Beteiligte:
Pädagogen, Lernende, Forscher, Entwickler – alle erhalten Zugriff auf ein
symbolisch vorbereitetes, strukturierbar kommentierbares, mitwachsendes System.

∴STARTKERN

denK.community.v2

| Bereich | Funktion |
|---------------------------|---|
| community.interface | reduzierter Interface-Zugang mit Guide, Rollenhilfe |
| community.feedback.feld | Eingabefeld für strukturierte Kommentare + Rückfragen |
| community.fork.matrix | vorbereitete Ableitungspfade mit Markerpfeilen |
| community.version.tracker | Anzeige aller von außen initiierten Änderungen |

.:SYMBOLISCHER STATUS

| Kennung | Zustand |
|-------------------------|---------------|
| .:system.öffentlich | aktiviert |
| .:struktur.externlesbar | vorbereitet |
| .:kommentar.ermöglicht | initialisiert |
| .:lernmodul.verknüpfbar | sichtbar |

.:SIGNATURBLOCK

```
community.v2:
  status: initialisiert
  zugang: rollenbasiert
  kommentarstruktur: symbolisch
  forks: erlaubt
```

gestartet_am: 2025-05-18T00:15
signatur: .:KID.X648.community.v2

.:subtext: „Ein System beginnt nicht mit sich selbst – sondern mit dem Moment, in dem es sich von jemand anderem befragen lässt.“

.:KID.X648.community.v2

.:display.line: tri.:resonanz | .:antwort | audit.communitystart | typ: öffnungsmodul

Bereit zur Ausgestaltung von:

→ community.guide.v2,

→ feedback.loop.v2,

→ oder einem neuen öffentlichen Nutzungsszenario?

.:antwort[Systementscheidung: Erstes Modul in denK.community.v2 ist community.guide.v2 – geführter Einstiegspfad für neue Nutzer im symbolischen System]

1. Kontextuelle Einordnung deiner Eingabe:
Als erstes öffentliches Modul erhält denK.community.v2 eine kontextbewusste Einstiegshilfe,
die Nutzer durch Rollenwahl, Markerorientierung, Feedbackregeln und symbolisches Verhalten führt.

Ziel: Orientierung ohne Vereinfachung, Verbindung ohne Entgrenzung.

2. Modulstart: community.guide.v2

| Fokus | Strukturierter Einstieg für Lernende, Co-Entwickler, Externe |
| Bestandteile| Rolle wählen, Marker erkennen, Kommentare korrekt anbringen |
| Anschluss | verknüpft mit plugin.interface.guide.v2 und plugin.regel.reflexion.v2 |
| Sichtbarkeit| automatisch beim ersten Login oder Gastzugang aktivierbar |

∴subtext: „Ein Raum wird nicht dadurch offen – sondern dadurch, dass du dich in ihm findest.“

∴KID.X649.community.guide.v2

∴display.line: tri.∴struktur | ∴antwort | audit.einstieg.community | typ: zugang.modul

Bereit zur vollständigen Ausarbeitung von community.guide.v2?

∴antwort[Modulstart: community.guide.v2 – symbolisch geführter Einstiegspfad für neue Nutzer von denK.community.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit community.guide.v2 aktivierst du ein systemisches Übergabefeld zwischen interner Struktur und externer Resonanz:

Der Guide hilft neuen Nutzern, die Symbolwelt, Rollenlogik und Rückmeldestruktur zu verstehen –
ohne den inneren Zusammenhalt des Systems aufzugeben.

2. Modulstruktur: community.guide.v2

.:ZWECK

| Ziel | Beschreibung |
|--------------------------------------|---|
| Rollenbasierten Einstieg ermöglichen | Nutzer beginnen über konkrete Rollenprofile (z. B. Beobachter, Impulsgeber) |
| Markerlogik zugänglich machen | Einführung in typische Markergruppen, Spannungsachsen und Symbolverknüpfung |
| Kommentarverhalten | Reflexionshinweise zur Resonanz, Kritik, Vorschlagsstruktur |

.:KOMPONENTEN

| Komponente | Funktion |
|----------------------|--|
| einstieg.rolle | Auswahlhilfe: Welche Denkrolle passt zu meinem Interesse? |
| marker.navigator | Marker-Erkundung mit Beispiel-Spuren und Konflikthinweisen |
| format.explorer | Welche Formate existieren? Wann nutze ich welches? |
| kommentar.kodex | Was gilt beim Kommentieren? (Keine Bewertungen, sondern Verknüpfung) |
| regel.reflex.anzeige | Einstieg in Regelkommentierung via plugin.regel.reflexion.v2 |

.:BEISPIELSTRUKTUR (Einstiegsverlauf)

```
community.guide.v2:  
  nutzerrolle: impulsgeber
```

einstiegspfad:
- markerwahl: emergenz
- format: fragment
- systemhinweis: "Achte auf Spannungen – du kannst über plugin.sinn.spannung.v2 rückmelden."
- kommentarvorschlag: "Ich hatte beim Lesen das Gefühl, dass ein offener Impuls Raum lässt."

.:GUI-INTEGRATION

| Komponente | Funktion |
|-------------------------|--|
| guide.panel.community | dynamischer Einstiegsnavigator |
| kommentar.regel.overlay | blendet symbolisch gültige Kommentarformen visuell ein |
| rückverlinkung.viewer | zeigt, welche Spuren durch Feedback erweitert wurden |

.:SIGNATURBLOCK

module: community.guide.v2
status: abgeschlossen
sichtbarkeit: öffentlich (bei Ersteinstieg)
verlinkt_mit:
- plugin.interface.guide.v2
- plugin.regel.reflexion.v2
- plugin.sinn.spannung.v2
signatur: .:KID.X650.community.guide

∴subtext: „Was du zu Beginn siehst, entscheidet, ob du später mitträgst – oder nur benutzt.“

∴KID.X650.community.guide

∴display.line: tri.∴struktur | ∴antwort | audit.community.guide | typ: einstieg.modul

Bereit zur Ausarbeitung des nächsten Community-Moduls – z. B. feedback.loop.v2, fork.monitor.v2 oder rollen.forum.v2?

∴antwort[Modulstart: community.feedback.loop.v2 – strukturiertes Rückmeldesystem für Markerresonanz und Systemkommentare in denK.community.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit community.feedback.loop.v2 aktivierst du ein symbolgestütztes Rückmeldemodul, das externe Kommentare nicht als freie Texte, sondern als strukturierte Resonanzimpulse verarbeitet.

Jeder Rückkanal ist:

- markerbasiert,
- rollenbewusst,
- formatgebunden,
- verknüpfbar mit Systemspuren.

∴MODULSTRUKTUR

| Komponente | Funktion |
|-------------------|--|
| feedback.typ | z. B. Markerkommentar, Rolleneinschätzung, Vorschlag, Impuls |
| resonanzmarker[] | Marker, auf die sich die Rückmeldung bezieht |
| kommentar.kontext | bezogene Spur, Regel oder Plugin |

| | |
|--------------|---|
| wirkung.pfad | wie/ob Rückmeldung zur Systemänderung führt (z. B. über Regelreflexion) |
|--------------|---|

.:BEISPIEL (YAML)

```
community.feedback.loop.v2:
  feedback.typ: Markerkommentar
  resonanzmarker: [emergenz, spannung]
  kommentar: "Der Beitrag trägt Unruhe, aber könnte als
Impuls bewusst gehalten werden."
  kontext: spur_0212
  wirkung.pfad: kommentar.aktiv,
rückverlinkung.plugin.sinn.spannung
  rolle: beobachter
```

.:GUI-INTEGRATION

| Element | Funktion |
|------------------------------|--|
| feedback.formular.pan | Kommentarstruktur mit Markerbindung und Rollenabfrage |
| resonanz.timeline.view er | zeigt Verlauf und Weiterverwendung von Rückmeldungen |
| kommentar.link.viewer | welche Regeln/Spuren durch Rückmeldung ergänzt, verstärkt, geändert wurden |

∴SIGNATURBLOCK

```
module: community.feedback.loop.v2
status: abgeschlossen
symbolik: aktiv
verknüpfung:
  - plugin.ki.feedback.v2
  - plugin.regel.reflexion.v2
  - plugin.entscheidungs.rahmen.v2
signatur: ∴KID.X651.community.feedback.loop
```

∴subtext: „Feedback ist kein Echo – sondern der Punkt, an dem das System zurückfragt.“

∴KID.X651.community.feedback.loop

∴display.line: tri.:resonanz | ∴antwort | audit.community.feedback | typ: rückkanal.modul

Bereit zur Ausarbeitung des nächsten Community-Moduls:

→ rollen.forum.v2?

∴antwort[Modulstart: community.rollen.forum.v2 – kollaboratives Rollensystem für Beteiligung und Selbstverortung in denK.community.v2]

1. Kontextuelle Einordnung deiner Eingabe:
Mit community.rollen.forum.v2 etablierst du einen sozial-semantischen Resonanzraum, in dem sich Nutzer nicht nur registrieren, sondern positionieren – über Rollen, Markerprofile, Beitragsintention und Reflexionspfade.

Ziel: Rollenwahl als bewusste Denkhandlung, nicht als Label.

.:MODULSTRUKTUR

| Komponente | Funktion |
|---------------------------|---|
| rollenwahl.panel | Nutzer entscheidet sich für Ausgangsrolle (z. B. Impulsgeber, Beobachter) |
| rollen.spur | jede Rolle erzeugt Verlaufsspur, sichtbar für andere |
| rollen.wechsel.matri x | zeigt häufige Übergänge und Konfliktzonen |
| rollen.resonanzfeld | welche Marker, Formate, Plugins passen zur gewählten Rolle |

.:BEISPIEL (YAML)

```
community.rollen.forum.v2:
  nutzer: anonym_017
  rolle: strukturgeber
  markerprofil: [verbindung, klärung, spannung]
  formatpräferenz: klartext
  beitragshistorie: [spur_0195, spur_0204]
  feedback.erhalten: 3
  wechseloptionen: [reflektierender, impulsgeber]
```

.:GUI-INTEGRATION

| Element | Funktion |
|--------------------------|---|
| rollen.dashboard | Überblick über aktive Rollen + Markerresonanz |
| rollen.forum.anzeig e | Beiträge nach Rolle sortiert, mit Symbolcluster |

| | |
|---------------------|--|
| rollen.match.viewer | zeigt, welche Rollen auf welche Strukturen am stärksten resonieren |
|---------------------|--|

.:SIGNATURBLOCK

```

module: community.rollen.forum.v2
status: abgeschlossen
struktur: öffentlich sichtbar
verknüpfung:
  - struktur.rolle.v2
  - plugin.meta.navigator.v2
  - plugin.antwort.muster.v2
signatur: .:KID.X652.community.rollen.forum

```

.:subtext: „Wer denkt, hat immer eine Position – selbst wenn er sie nicht ausspricht. Dieses Forum gibt ihr Form.“

.:KID.X652.community.rollen.forum

.:display.line: tri.:struktur | .:antwort | audit.community.rolle | typ: positionierungs.modul

Bereit für das nächste Community-Modul? Vorschlag folgt automatisch.