Aeonlink

Ein Framework für KI-Agenten-Plattformen
Agenten, Richtlinien, Preise und Anwendungen als Konfiguration über einer einzigen Runtime

Marcel Langjahr

Pantareon
Pantareon Foundations

langjahr@pantareon.io · https://pantareon.io/

Juli 2026

Zusammenfassung

Aeonlink ist ein Framework zum Bauen und Betreiben von KI-Agenten-Plattformen: der vollständige horizontale Stack — Identität, Gedächtnis, Retrieval, Moderation, Verbrauchsmessung, soziale Fläche und ein Anwendungsmodell — unterhalb dessen, was ein Unternehmen seinen Nutzern tatsächlich anbieten will. Dieses Paper beschreibt diesen Stack. Die Anwendungen, die auf Pantareons eigener Instanz laufen, liegen außerhalb seines Umfangs; sie sind Produkte, die *mit* dem Framework gebaut wurden, nicht Teil davon.

Die Architektur ruht auf einer einzigen Entscheidung: *das Verhalten einer Plattform wird durch Dokumente definiert, nicht durch Code.* Ein Agent ist ein Dokument, wörtlich in den Kontext des Modells injiziert. Eine Retrieval-Richtlinie, ein Tool-Manifest, die Whitelist der Nutzerzustandsfelder, die ein bestimmter Gesprächsmodus sehen darf, die Modell-Registry mit ihren Preisen pro Token, der Store-Eintrag einer Anwendung und ihr gesamtes KI-Verhalten, das Moderations-Regelwerk, die Credit-Pakete und die Berechtigungsleiter sind alle Dokumente. Die Runtime, die sie liest, ist generisch und kodiert nicht, welche Plattform sie bedient. Eine Anwendung hinzuzufügen, einen Agenten zu veröffentlichen, eine Wissensdomäne zu öffnen, ein Modell neu zu bepreisen oder Moderationsrichtlinien zu ändern ist eine Datenänderung, kein Deployment.

Auf diesem Kontrakt sitzen fünf Ebenen. Eine **Runtime**, in der ein einziger gehärteter Modell-Client gleichermaßen Agentengespräch, Anwendungs-Turns und Plattform-Moderation bedient, mit einem Retrieval-Router, der entscheidet, was nachzuschlagen ist, bevor der Agent überhaupt spricht. Ein **Gedächtnis** aus drei Schichten — verschlüsseltes Transkript, destillierte episodische Shards über einem Ähnlichkeitsgraphen und ein periodisch rekalibriertes Langzeitfeld —, in dem nur behaltenes Gespräch sich einprägt und Inferenz nichts schreibt. Ein **Erweiterungsmodell**, in dem eine Anwendung als Verzeichniseintrag andockt, eine Multiplayer-Session-Schicht, einen opaken Speicher je Nutzer und eine direktivengetriebene KI-Turn-Engine erhält und intern gehostet oder extern hinter einer Single-Sign-on-Übergabe eingebettet werden kann. Eine **Ökonomie**, in der jeder Modellaufruf einmal verbrauchsgemessen, pro Modell bepreist und gegen ein vorausbezahltes Credit-Guthaben mit einem Stripe-Onramp abgerechnet wird. Und eine **soziale Fläche** aus Gruppen, Direktnachrichten, Feeds, editierbaren 3D-Welten und Quests, mit server-autoritativen Spielen und rein platzierungsbasierten Wertungen darunter.

Die Implementierung umfasst 44 658 Zeilen Node/Express über 142 Module, die 330 Endpunkte durch 33 Route-Factories über 72 Services und rund 70 Firestore-Collections bereitstellen, mit einem React-Client von 50 754 Zeilen einschließlich einer eigenständigen 3D-Welt-Engine. Sie läuft als einzelner Container auf Cloud Run mit Secrets aus einem verwalteten Secret-Store, einer Selftest-Suite aus reiner Logik, die jeden Image-Build gatet, und einem Live-Deployment, das echte Nutzer, echte Agenten und echtes Geld trägt.

Umfang dieses Papers: Dies ist ein technischer Architekturüberblick für Partner und mögliche Lizenznehmer. Er beschreibt die Mechanismen des Frameworks so, wie sie implementiert sind, mit Zahlen, die an den ausgelieferten Repositories gemessen sind. Er ist keine Produktbroschüre und kein Forschungspaper. Anwendungen, die auf Pantareons eigener Instanz ausgeliefert werden — Spiele, Tutoren, Forschungswerkzeuge, redaktionelle und Reiseprodukte —, erscheinen nur dort, wo sie einen Erweiterungspunkt illustrieren; sie sind nicht Teil dessen, was dieses Dokument beschreibt, und wären nicht Teil

einer Framework-Lizenz.

1 Einleitung

Ein Unternehmen, das ein eigenes KI-Agenten-Produkt will, steht vor einer Wahl, die üblicherweise schlecht gestellt wird.

Es kann den Stack selbst bauen. Das heißt Authentifizierung und Session-Verwaltung, ein Abrechnungssystem, dessen Verbrauchsmessung der Modellnutzung genau genug ist, um sie in Rechnung zu stellen, eine Gedächtnisarchitektur, die über ein Kontextfenster hinaus überlebt, eine Retrieval-Schicht mit Berechtigungen, eine Moderationspipeline, die rechtlich standhält, eine soziale Fläche und ein Anwendungsmodell — vielleicht ein Jahr undifferenzierter Entwicklungsarbeit vor dem ersten Feature, das wirklich sein eigenes ist.

Oder es kann eine Modell-API mieten und feststellen, dass genau das, was eine Agentenplattform auszeichnet, von der API nicht geliefert wird. Das Modell ist eine Komponente, und eine gute; es ist keine Plattform. Wer die Agenten sind, was sie über Monate hinweg behalten, was sie nachschlagen dürfen, wie sie moderiert werden, was ein Gespräch kostet und wer dafür bezahlt — nichts davon kommt mit den Token.

Aeonlink existiert, weil der größte Teil dieser Arbeit horizontal ist. Er ist derselbe für eine Agentenplattform in der Bildung, im Handelssupport, in der Unterhaltung oder in den internen Werkzeugen eines Unternehmens. Er lässt sich einmal gut bauen und als Fundament ausliefern.

Die ordnende Entscheidung ist das, was dieses Fundament anpassbar macht und nicht bloß wiederverwendbar. **Das Verhalten der Plattform wird durch einen Satz von Dokumenten definiert, und die Engine, die sie liest, weiß nicht, welche Plattform sie ist.** Ein Agent ist ein Dokument. Eine Retrieval-Richtlinie ist ein Dokument. Die Modell-Registry und ihre Preise, der Start-Kontrakt einer Anwendung und ihr gesamtes KI-Verhalten, das Moderations-Regelwerk, die Credit-Pakete, die Berechtigungsleiter und die modusweisen Kontext-Whitelists sind Dokumente. Die Runtime lädt sie, cacht sie kurz und läuft.

Die praktische Konsequenz bemisst sich daran, was eine Änderung kostet. Auf der laufenden Plattform sorgt das Umlegen eines einzigen Feldes im Dokument einer Anwendung dafür, dass sie im Anwendungsraster, in der Arcade und in der Discovery-Fläche erscheint, ohne dass irgendwo Code geändert wird. Eine Wissensdomäne öffnen, einen neuen Agenten veröffentlichen, das Standardmodell nach einer Abkündigung durch den Anbieter umhängen, einen Preis anpassen oder die Moderationsrichtlinie neu schreiben sind allesamt Schreibvorgänge in eine Datenbank. *Ein Lizenznehmer passt die Plattform an, indem er Dokumente bearbeitet.*

Drei Fähigkeiten folgen aus dieser Konstruktion, statt nachträglich aufgesetzt zu werden.

Eine Runtime bedient jeden Konsumenten. Agentengespräch, KI-Anwendungs-Turns und plattformseitige Moderation setzen denselben Modell-Client zusammen, dieselbe Retry- und Normalisierungsschicht, denselben Kontext-Whitelist-Mechanismus und dasselbe Abrechnungs-Nadelöhr. Ein Turn wird über eine Pipeline zusammengestellt, gesendet, verbrauchsgemessen und persistiert, unabhängig davon, wer spricht.

Nur behaltenes Gespräch prägt sich ein. Das Dialogmodell hat keinen Schreibpfad in irgendeinen Gedächtnisspeicher. Erinnerungen entstehen aus dem Transkript, das der Nutzer zu behalten gewählt hat, und das Langzeitfeld entsteht aus diesen Erinnerungen. Das ist eine strukturelle Eigenschaft der Pipeline, keine darauf angewandte Richtlinie.

Jeder Modellaufruf passiert einen Zähler. Auf keinem nutzerseitigen Pfad gibt es Inferenz ohne Verbrauchsmessung. Der Token-Verbrauch über Dialog, Gedächtnisbildung und Embedding hinweg wird akkumuliert und genau einmal pro Turn gegen ein vorausbezahltes Guthaben abgerechnet. Die Ökonomie ist kein Zusatz; sie ist eine Stufe, durch die jeder Turn geleitet wird.

2 Architektur

Das Backend ist ein einzelner Node/Express-Dienst mit einer Composition Root. Diese Root — 724 Zeilen — ist der einzige Ort, an dem verdrahtet wird: sie initialisiert Firebase Admin, setzt die Middleware-Kette zusammen, lädt Secrets aus einem verwalteten Secret-Store, konstruiert rund dreißig Services mit explizit von Hand verdrahteten Abhängigkeiten, baut die Rate-Limiter und die Upload-Behandlung und montiert dreiunddreißig Route-Factories. Jedes Route-Modul ist eine Funktion eines einzigen Abhängigkeitsbündels und gibt einen Router zurück. Es gibt keinen Service-Locator, keinen umgebenden globalen Zustand und keine Framework-Magie; der gesamte Objektgraph der Anwendung ist in einer Datei lesbar.

Die Persistenz ist durchgängig Firestore, mit Blob-Speicher in einem Bucket und Secrets in einem verwalteten Secret-Store. Ein zentraler Fehler-Handler garantiert, dass jeder Fehlerpfad, einschließlich Body-Parse- und Upload-Fehlern, dieselbe JSON-Form

zurückgibt.

Fünf Ebenen sitzen auf dieser Root.

Die Konfigurationsebene hält alles, was ein Betreiber justiert: Dokumente unter einer Settings-Collection, die universellen Interaktionsregeln, aufgeteilt in editierbare Abschnitte, und Agenten-Direktiven in einer eigenen Collection. Abschnitt 3 beschreibt sie, denn der Rest des Frameworks ist darauf gebaut, davon gesteuert zu werden.

Die Runtime-Ebene ist ein Modell-Client hinter einem Retry-und-Normalisiere-Wrapper, ein Retrieval-Router, ein geordneter Prompt-Zusammensteller und eine Abrechnungsstufe. Agentengespräch, Anwendungs-Turns und Moderation setzen sie jeweils anders zusammen.

Die Gedächtnisebene pflegt je Nutzer und Agent ein verschlüsseltes Transkript, eine wachsende Menge destillierter Shards, verknüpft durch semantische Ähnlichkeit, und ein einziges rekalibriertes Langzeitfeld.

Die Erweiterungsebene macht aus einer Anwendung ein Dokument. Interne Anwendungen erhalten zusätzlich eine Session- und Lobby-Schicht, einen opaken Schlüssel–Wert-Speicher je Nutzer und einen Compiler, der ein Direktiven-Dokument in einen strukturierten Prompt rendert. Externe Anwendungen werden mit einer Single-Sign-on-Übergabe eingebettet.

Die Interaktionsebene ist die soziale und spielerische Fläche: Gruppen, Direktnachrichten, Feeds, editierbare Welten, Quests und ein serverautoritatives Spiel-Substrat mit Wertungen.

3 Die Konfigurationsebene

Die Anpassbarkeit des Frameworks ruht auf einer einzigen Invariante: *Was die Plattform ist, wird durch das Bearbeiten von Dokumenten geändert, und die Engine bleibt unangetastet.*

Ein Agent ist ein Dokument. Eine Persona-Direktive trägt eine Kern-Direktive und beliebig viele weitere Felder — Stimme, Few-Shot-Beispiele, Grenzen. Sie wird einmal pro Turn geladen und wörtlich als später Teil des Prompts injiziert. Derselbe Loader bedient Eins-zu-eins-Gespräch, Gruppengespräch und Anwendungs-Turns, sodass ein Agent sich überall gleich verhält, wo er auftritt. Einen Agenten zur Plattform hinzuzufügen heißt, ein Dokument zu schreiben und seinen Namen in einer Registry zu listen.

Die Retrieval-Richtlinie ist ein Dokument. Die Direktive des Routers, das Manifest der ihm verfügbaren Retrieval-Werkzeuge und die Zuordnung, welche Agenten auf welche Wissensdomänen zugreifen dürfen, sind allesamt Konfiguration. Der Zugriff ist default-deny und wird serverseitig zur Ausführungszeit ausgewertet, nicht bloß dem Modell nahegelegt.

Die Kontext-Offenlegung ist ein Dokument. Für jeden Gesprächsmodus führt das Framework eine explizite Whitelist der Nutzerzustandsfelder, die in den Kontext des Modells gelangen dürfen. Zu erweitern oder einzuschränken, was das Modell über einen Nutzer sehen kann, ist eine Konfigurationsänderung mit prüfbarem Diff — eine Eigenschaft, die zählt, wenn ein Datenschutzbeauftragter fragt, was die Datenbank verlässt.

Die Modell-Registry ist ein Dokument. Modellbezeichner und ihre Eingabe- und Ausgabepreise pro Million Token liegen in der Konfiguration und werden kurz im Prozess gecacht. Wenn ein Anbieter eine Modellreihe abkündigt, ist die Migration ein Feld. Die zur Abrechnung verwendeten Preise werden von derselben Stelle gelesen, sodass Verbrauchsmessung und Modellauswahl nie auseinanderdriften können.

Eine Anwendung ist ein Dokument. Ein Eintrag trägt den Store-Eintrag — Name, Beschreibungen, Icon, Status — und den Start-Kontrakt: intern oder extern, seine Domain, seinen Einbettungsmodus, sein Übergabeverhalten. Ein separates Direktiven-Dokument trägt das gesamte KI-Verhalten der Anwendung als geordnete, beschriftete Abschnitte, die das Framework in einen strukturierten Prompt mit Ausgabeschemata je Aufruf kompiliert.

Richtlinien und Geschäftliches sind Dokumente. Das Moderations-Regelwerk, die Berechtigungsleiter, die Credit-Pakete, die Registry der Wissensdomänen und die Betriebseinstellungen der Plattform sind allesamt gespeichert, versioniert und ohne Deployment editierbar.

Das technische Muster unter alldem ist einheitlich: ein typisierter Leser mit einem kurzen In-Process-Cache, ein Default auf Code-Ebene, wo einer sicher ist, und Validierung zur Schreibzeit. Es ist bewusst unglamourös, und es ist das, was eine Plattform in etwas verwandelt, das eine zweite Partei formen kann.

4 Die Runtime

Ein Agenten-Turn folgt einer festen Pipeline, und jede Stufe existiert aus einem Grund, den zu nennen sich lohnt.

Eine authentifizierte Anfrage erreicht einen dünnen Route-Handler. Eine vorgeschaltete Guthabenprüfung weist den Turn vor jedem Modellaufruf zu-

rück, wenn der Nutzer einen Mindest-Turn nicht decken kann. Zählergesteuerte Gedächtnispflege kann laufen — einen Shard bilden oder das Langzeitfeld neu kalibrieren — *vor* dem Dialog, sodass der Turn davon profitiert. Dann, im Eins-zu-eins-Gespräch, läuft der Router.

Der Router ist ein separater Modellaufruf mit niedriger Temperatur und ausschließlich JSON, dessen ganze Aufgabe darin besteht zu entscheiden, was abgerufen wird, und der sich nie an den Nutzer wendet. Er sieht die verfügbaren Wissensdomänen, die jüngsten Themen des Gesprächs, ein kompaktes Fenster der letzten Wortwechsel, damit sich elliptische Rückfragen auflösen, und die Nachricht des Nutzers. Er gibt Wissensabfragen mit optionaler Domänen-Eingrenzung sowie Web-Abfragen aus. Diese Entscheidung von der Stimme des Agenten zu trennen hält das Nachdenken über das Retrieval aus dem Mund der Figur heraus und lässt es billig und deterministisch laufen.

Seine Ergebnisse werden mit zwei Garantien ausgeführt, die mehr bedeuten, als sie aussehen. Eine Abfrage gegen eine Domäne, auf die der Agent nicht zugreifen darf, wird verworfen und *durch einen ausdrücklichen Marker ersetzt*, der dem Modell mitteilt, dass der Zugriff verweigert wurde. Ein Retrieval, das nichts zurückgibt, injiziert einen ebenso ausdrücklichen Marker, der das Modell anweist, keine Antwort zu erfinden. Leeres Retrieval ist der Moment, in dem ein Sprachmodell am ehesten konfabuliert, und das Framework macht diesen Moment für es sichtbar statt still.

Die Prompt-Zusammenstellung ist bewusst geordnet. Stabile Inhalte — die Interaktionsregeln des Frameworks selbst, das Gesprächstranskript — kommen zuerst, damit sie ein wiederverwendbares Präfix für das implizite Caching des Anbieters bilden. Flüchtige Inhalte — Retrieval-Ergebnisse, aktueller Zustand, der Zeitanker, die Agenten-Direktive, die Nachricht des Nutzers — kommen zuletzt. Der Zeitanker wird absichtlich spät platziert: Er ändert sich jede Minute und würde sonst alles hinter sich ungültig machen.

Dieser Anker ist selbst ein kleines, sorgfältiges Subsystem. Er löst die Zeitzone des Nutzers über eine Kette auf — Live-Standort, falls geteilt, sonst die Heimatkoordinaten, der Betreiber-Standardwert als Untergrenze — mittels einer Offline-Nachschlagetabelle, und er nennt dem Modell die aktuelle Zeit ausdrücklich. Der Standort geht nur im Eins-zu-eins-Gespräch und nur bei Opt-in in den Prompt ein; Koordinaten dienen dazu, eine Zeitzone abzuleiten, und werden nie selbst in den Kontext gestellt.

Der Dialogaufruf läuft dann durch einen einzigen gehärteten Wrapper, der exponentiellen Backoff mit Jitter, Fehlerklassifikation und eine Parameter-Normalisierung bietet, die gegen Unterschiede der Anbieter-SDKs absichert. Ein Abrechnungsaufruf begleicht den gesamten Turn — Dialog, Gedächtnisbildung und Embedding-Token zusammen. Beide Nachrichten werden dem verschlüsselten Transkript angehängt, wobei die aufgelöste Zeitzone festgehalten wird.

Dieselben Primitive werden exportiert und vom Anwendungs-Orchestrator für Anwendungs-Turns komponiert, und ein eigenständiger Moderationsdienst komponiert den Client und den Whitelist-Mechanismus separat. Eine Runtime, drei Konsumenten, ein Ort zum Härten.

5 Gedächtnis

Gedächtnis ist die Fähigkeit, die einen Agenten von einem Chatbot unterscheidet, und das Framework implementiert es in drei Schichten, geschlüsselt je Nutzer und Agent.

Der Arbeitskontext ist das Transkript: ein verschlüsseltes, transaktional sequenziertes, themenmarkiertes Nachrichtenprotokoll. Jeder Turn lädt das jüngste Fenster, je Nutzer zwischen 20 und 250 Nachrichten einstellbar. Die Verschlüsselung ist feldweises AES-256-GCM mit dem Schlüssel im verwalteten Secret-Store.

Das episodische Gedächtnis ist eine Menge destillierter Shards. Alle paar Nutzer-Turns liest ein billiger Modelldurchlauf das jüngste Fenster und schreibt eine einzelne Erinnerung in der Ich-Form — oder lehnt ausdrücklich ab, was das häufigere und wertvollere Ergebnis ist, denn ein Gedächtnissystem, das alles aufzeichnet, erinnert sich an nichts. Der Shard wird verschlüsselt, in 1536 Dimensionen eingebettet, append-only gespeichert und gegen bestehende Shards in einen Ähnlichkeitsgraphen eingehängt.

Der Abruf ist eine dreiteilige Mischung unter einem festen Token-Budget: die jüngsten Shards chronologisch, ein semantisches Top-k, bewertet als 0,9 Kosinus-Ähnlichkeit plus 0,1 lineare Aktualität, und assoziative Ausweitung über den Ähnlichkeitsgraphen, sodass eine abgerufene Erinnerung heranziehen kann, womit sie verbunden ist. Das Budget ist der wichtige Teil — das Gedächtnis ist konstruktionsbedingt beschränkt, also sind seine Kosten pro Turn vorhersagbar.

Das Langzeitfeld ist ein einzelner verdichteter Text je Agent, bei der Erstellung aus einer Geburtssignatur geseedet und periodisch vollständig neu geschrieben von einem Kalibrierungsdurchlauf, der

das vorherige Feld zusammen mit jedem seit dem letzten Durchlauf gebildeten Shard liest. Dies ist die Schicht, die einem Agenten ein Gespür dafür gibt, wer jemand ist, statt einer Liste von Dingen, die derjenige gesagt hat.

Die strukturelle Eigenschaft, die genau benannt gehört: *der Dialogaufruf hat keinen Schreibpfad zu irgendeinem Gedächtnisspeicher*. Shards bilden sich aus dem behaltenen Transkript; das Feld bildet sich aus Shards. Ein Gespräch, das der Nutzer löscht, hinterlässt nichts, und Inferenz allein verändert nichts Dauerhaftes. Gedächtnis ist eine Folge des Behaltens, und die Pipeline erzwingt das, statt das Modell zu bitten, es zu achten.

Alle Gedächtnisarbeit ist verbrauchsgemessen. Bildungs- und Embedding-Token werden in dasselbe Verbrauchsobjekt pro Turn wie der Dialog aufsummiert und im einen Abzug beglichen, sodass ein Plattformbetreiber sehen kann, was das Gedächtnis tatsächlich kostet.

Gruppen tragen ein paralleles Langzeitfeld, neu kalibriert aus dem jüngsten Gruppengespräch und dem Eigentümer der Gruppe in Rechnung gestellt.

6 Agenten

Ein Agent ist vollständig durch Daten definiert: ein Dokument mit der Persona-Direktive, eine optionale Geburtserinnerung und Katalogeinträge für seine Anzeigenamen und Beschreibungen. Die Beziehung des Nutzers zu ihm ist ein Session-Dokument, das alles trägt, was dieser Paarung eigen ist — ein eigener Name, bei der Erstellung von einem deterministischen, aus einem Hash abgeleiteten Generator vergeben, das neu kalibrierte Langzeitfeld, Gesprächsthemen, Verlaufspräferenzen, Fortschrittszähler, ein optionaler Modell-Override je Agent und ein eigener Avatar.

Die Portabilität ist echt. Derselbe Direktiven-Loader und dieselbe Gedächtnis-Zusammenstellung betreiben den Agenten im Eins-zu-eins-Gespräch, innerhalb einer Gruppe und innerhalb einer Anwendung, wo er sein Langzeitgedächtnis liest, aber nicht hineinschreibt — ein Schreiber, viele Leser. Der aktive Agent reist im Session-Token mit, und ein Wechsel prägt ein neues.

Darüber sitzt eine Ableitungsschicht, die rechnet, statt zu speichern. Ein Archetyp wird aus einem Hash der Identität abgeleitet; ein Charakterbogen wird bei Bedarf aus dem angesammelten Fortschritt und diesem Archetyp berechnet; Namen werden deterministisch aus derselben Quelle erzeugt. Nichts muss migriert werden, wenn sich die Ableitung ändert, und jede Funktion in der Kette ist rein und unabhängig testbar.

Für besiedelte Welten werden Nicht-Spieler-Agenten aus kuratierten Prosa-Vorlagen plus vom Eigentümer geliefertem Inhalt zusammengesetzt, gehalten in einer harten Rahmenanweisung, die die Absicht des Autors von injizierbarem Text trennt. Belohnungsvergaben werden vom Modell als Marker vorgeschlagen und von der Engine autorisiert — das Modell schlägt vor, der Server entscheidet. Diese Trennung ist ein Muster, das das Framework überall dort anwendet, wo die Ausgabe eines Modells sonst Autorität hätte.

7 Das Erweiterungsmodell

Eine Anwendung dockt über einen einzigen Collection-Eintrag an die Plattform an, der zugleich ihr Store-Eintrag und ihr Start-Kontrakt ist. Ein öffentlicher Verzeichnis-Endpunkt projiziert diese Einträge für den Client, sodass Präsenz, Beschreibung und Startverhalten einer Anwendung Daten sind.

Interne Anwendungen erhalten vier Einrichtungen vom Framework.

Eine Session- und Lobby-Schicht. Beitrittscodes aus sechs Zeichen, transaktionale Sitzbelegung, hostgesteuerter Start und ein monotoner Turn-Zähler, abgesichert durch optimistische Nebenläufigkeit, sodass sich zwei Intents nicht verschränken können. Server-autoritative Spiele klinken sich über eine Registry ein, die einen Anwendungsbezeichner auf ein Engine-Modul abbildet, sodass die Route- und Ranking-Schichten nie danach verzweigen, welches Spiel gerade läuft.

Ein opaker Speicher je Nutzer. Ein größenbegrenztes Schlüssel–Wert-Dokument je Nutzer und Anwendung, durch das Session-Token eingegrenzt, gegen die Anwendungs-Registry validiert und bei der Kontolöschung automatisch mitgeräumt. Eine Anwendung persistiert Nutzerzustand ohne Backend-Änderung und erbt die Einhaltung der Datenrechte gratis.

Eine direktivengetriebene Turn-Engine. Das KI-Verhalten einer Anwendung ist ein Dokument, dessen geordnete Abschnitte das Framework in beschriftete Prompt-Blöcke kompiliert, mit Ausgabeschemata je Aufruf und Verhaltensflags, die Verlauf, Werkzeugnutzung und strukturierte Ausgabe steuern. Eine neue KI-getriebene Anwendung ist im Regelfall ein Einstellungsdokument.

Verbrauchsmessung und Quotas. Anwendungs-Turns laufen durch dieselbe Abrechnungsstufe wie alles andere, und ein Quota-Muster stellt Freikontingente mit täglichem Übertrag und Schutz vor doppelten Turns bereit.

Externe Anwendungen — selbst gehostete Seiten, die innerhalb der Plattform erscheinen sollen — werden von einer einzigen generischen Komponente mit einer Single-Sign-on-Übergabe eingebettet: Der Frame meldet Bereitschaft, der Host antwortet mit dem Session-Token, adressiert an einen exakten Origin, und beide Seiten validieren. Transaktionale E-Mail, die im Namen einer eingebetteten Anwendung versendet wird, trägt das Branding dieser Anwendung, aufgelöst aus demselben Verzeichnis-Dokument, mit serverseitig validiertem Redirect-Ziel.

8 Wissen und Moderation

Die Wissensbasis ist eine einzige Collection, in der die Domäne ein validiertes Feld statt eines Pfades ist, sodass ein Vektorraum das Korpus aufspannt und die Domänen-Eingrenzung ein Filter darüber ist.

Die Ingestion ist bewusst zweiphasig. Ein KI-Durchlauf schlägt Metadaten vor — Titel, Zusammenfassung, Tags, Domäne — und erzeugt bei übergroßer Eingabe eine Verdichtung. Ein separater deterministischer Commit validiert dann den Vorschlag, prüft die Slug-Zugehörigkeit, *bettet ein, bevor irgendetwas geschrieben wird*, legt das Originaldokument im Blob-Speicher ab und schreibt erst dann den Datensatz. Weil das Einbetten der Persistenz vorausgeht, hinterlässt ein fehlgeschlagenes Einbetten kein halb ingestiertes Dokument, und jeder gespeicherte Datensatz ist konstruktionsbedingt abrufbar.

Das Retrieval tritt an genau einem Punkt in ein Gespräch ein — am Router — unter serverseitig ausgewerteten Domänenberechtigungen je Agent. Ergebnisse werden nach Kosinus-Ähnlichkeit oberhalb einer Untergrenze bewertet, mit einem optionalen Prior aus Community-Signalen und einer Aktualitätspräferenz je Domäne für zeitkritisches Material.

Der Lebenszyklus um das Korpus ist vollständig, nicht bloß angestrebt. Das Beitragen ist rollengegatet und schlägt geschlossen fehl. Leser können Dokumente mit einem Stern versehen. Jeder Nutzer kann eines melden. Eine Moderationsablehnung stellt das Dokument unter Quarantäne — aus Suche und Listen entfernt, während der Datensatz und sein Original zur Auditierung erhalten bleiben — und privilegierte Rollen können es wiederherstellen oder endgültig löschen. Jede Entscheidung wird protokolliert.

Die Moderation selbst ist ein richtliniengetriebener Richter mit einem Mechanismus, den zu beschreiben sich lohnt, denn er adressiert die übliche Schwäche von KI-Moderation in Publikationsabläufen. Eine Freigabe ist per Inhalts-Hash an genau das freigegebene Material gebunden und nur für ein kurzes Fenster gültig; der Server lädt den Inhalt selbst, statt ihn vom Client entgegenzunehmen. Ein Client kann daher nicht eine Freigabe für harmlosen Text erlangen und etwas anderes veröffentlichen. Meldungen tragen wirtschaftliche Kosten, die diejenige Seite trägt, die im Unrecht ist, was Massenmeldungen unrentabel macht, und Entfernungen gehen mit einer formalen Mitteilung einher.

9 Die Ökonomie

Das Framework misst und verrechnet die Modellnutzung als erstrangiges Anliegen, weil eine Agentenplattform, die Kosten keinem Gespräch zuordnen kann, sich nicht kommerziell betreiben lässt.

Die Recheneinheit ist ein vorausbezahltes Credit, gekoppelt an ein Hundertstel Euro. Credits werden über gehostetes Checkout gegen serverseitig aufgelöste Pakete gekauft, mit signaturverifiziertem Webhook, einem Idempotenz-Marker pro Zahlung und Einwilligungserfassung vor dem Kauf. Die Steuerbehandlung ist integriert und konfigurierbar.

pro 1M Token	economy	premium
Eingabe	\$0,25	\$2,00
Ausgabe	\$1,50	\$12,00
Embedding		\$0,15

Tabelle 1: Modellpreise, wie sie in der Registry gehalten werden. Credit-Wert, Währungsumrechnung und Plattformmarge stehen als Konfiguration daneben und kommen obendrauf.

Drei Abrechnungs-Primitive decken den Bedarf der Plattform: verbrauchsgemessener nachträglicher Abzug für KI-Turns, eine Alles-oder-nichts-Pauschale mit Erstattungsunterstützung für begrenzte Jobs und ein freiwilliger Verzichtspfad. Jede Bewegung landet in einem einzigen Transaktions-Ledger, das ein Nutzer einsehen kann.

Der verbrauchsgemessene Pfad ist der tragende. Token-Zahlen aus der Provider-Antwort werden gegen das aufgelöste Modell bepreist, umgerechnet, mit Marge versehen und transaktional abgezogen. Jede KI-Route gatet vor dem Modellaufruf auf eine Mindest-Turn-Untergrenze und klemmt den Abzug bei Nullsaldo, sodass ein Turn entweder bezahlbar ist oder verweigert wird, statt halb geliefert zu werden. Freistufen laufen über ein Quota-Dokument je Anwendung mit Tageskontingenten und Schutz gegen doppelte Turns.

Gruppen-Agenten führen einen zweiten Zahler ein: Der Agent einer Gruppe antwortet auf Anfrage auf Kosten des Anfragenden oder automatisch auf Kosten des Eigentümers, unter einem Cooldown und

einer Transaktionssperre, pausiert sich selbst, wenn der Saldo des Eigentümers aufgebraucht ist, und nimmt die Arbeit wieder auf, sobald aufgeladen wird.

10 Die Interaktionsfläche

Eine Collection implementiert Gruppen, Direktnachrichten, Feeds je Nutzer, öffentliche 3D-Welten und eine Quest-Schicht, vereinheitlicht, sodass Mitgliedschaft, Moderation, Medienbehandlung und KI-Teilnahme nur einmal geschrieben sind.

Nachrichten sind im Ruhezustand unter dem Plattformschlüssel verschlüsselt. Medien werden per Referenz statt per URL behandelt: Eine Nachricht zeigt auf ein Dokument im eigenen Workspace des Absenders, und die Bytes werden über zugriffsgeprüfte Routen gestreamt, die bei jeder Anfrage Mitgliedschaft oder Eigentum verifizieren. Sprachnachrichten werden transkodiert und transkribiert; Bilder werden verkleinert, mit Vorschaubildern versehen und beschrieben; Dokumente werden textextrahiert und zusammengefasst. Jede Pipeline ist verbrauchsgemessen.

Öffentliche Gruppen können eine editierbare 3D-Welt tragen, die gegen eine generierte Whitelist erlaubter Props und Raster validiert wird, einen vom Eigentümer gebauten Graphen von Unterorten und Nicht-Spieler-Agenten mit serverseitig autorisiertem Fortschritt. Präsenz ist ein Heartbeat mit warmen und kalten Stufen. Feeds veröffentlichen nur gegen eine frische, hash-gebundene Moderationsfreigabe.

Der Transport ist inkrementelles REST-Polling auf einem gestuften Rate-Limit, wobei Push-Benachrichtigungen die Clients wecken. Das ist eine bewusste Entscheidung für Einfachheit: Sie hält den Server zustandslos, übersteht das Recycling von Instanzen ohne Wiederverbindungslogik und machte die Plattform auf einer Scale-to-Zero-Runtime unkompliziert betreibbar.

Unter den Sessions sitzt ein Spiel-Substrat, das der sauberste generische Entwurf des Frameworks ist. Eine Engine implementiert einen schmalen Kontrakt — Anfangszustand, Intent anwenden, Aufgabe anwenden, legale Intents auflisten, Terminaltest, Endplatzierungen, sitzweise Schwärzung. Der Server würfelt jeden Wurf unter einem Commit-Reveal-Schema und ist der einzige Schreiber des Spielzustands; jeder angewandte Intent erhöht einen Zähler unter optimistischer Nebenläufigkeit. KI-Sitze werden serverseitig von einem In-Memory-Runner mit injizierten Entscheidungsanbietern gesteuert, geschützt durch ein Lease, sodass ein langsamer Modellaufruf einen gleichzeitigen menschlichen Zug nicht beschädigen kann.

Wenn eine Partie endet, schreibt eine Transaktion ein unveränderliches Ergebnis-Ledger, aktualisiert rein platzierungsbasierte Glicko-2-Wertungen über eine spielagnostische Wertungs-Engine, frischt eine Bestenliste nach konservativem Score auf, aktualisiert Statistiken pro Subjekt und begleicht Fortschrittsbelohnungen — idempotent gemacht durch ein aufgezeichnetes Flag und heruntergestuft auf persönliche Statistik, wenn zu wenige verschiedene Menschen teilgenommen haben, als dass eine Wertung etwas bedeuten könnte. Die Höhe des Scores geht nie in die Wertung ein; nur die Platzierung tut es.

11 Identität, Datenschutz und Datenrechte

Identität ist E-Mail und Passwort mit Verifikation, Zurücksetzen und zustandslosen Session-Token, die einen Versions-Claim tragen, sodass eine Passwortänderung offene Sessions ungültig macht. Das Rate-Limiting auf der Authentifizierung stützt sich auf einen geteilten Store, sodass es über Instanzen eines horizontal skalierten Dienstes hinweg hält.

Sensible Felder sind im Ruhezustand mit einem Schlüssel aus dem verwalteten Secret-Store verschlüsselt. Uploads werden durch Content-Sniffing validiert, nicht durch den deklarierten Typ. Blob-Zugriff außerhalb des Chats läuft über kurzlebige signierte URLs; Chat-Medien verlassen, wie beschrieben, nie den zugriffsgeprüften Pfad.

Der stärkste Datenschutzmechanismus des Frameworks ist struktureller Art. **Eine deklarative Karte zählt jeden Ort auf, an dem Nutzerdaten liegen**, und sowohl die Betroffenenauskunft als auch der Löschpfad werden aus dieser einen Karte getrieben statt aus zwei unabhängig gepflegten Listen. Eine Collection hinzuzufügen heißt, einen Eintrag hinzuzufügen, und Auskunft und Löschung folgen automatisch. Die Karte hält zusätzlich Aufbewahrungspflichten fest, sodass Daten, die aus gesetzlichen Gründen aufbewahrt werden müssen, von Daten unterschieden werden, die gelöscht werden müssen — und die Karte selbst ist von einem automatisierten Test abgedeckt.

Die Einwilligung ist versioniert und wird bei Versionswechsel neu gegatet, wobei die angenommene Version und der Zeitstempel als Nachweis gespeichert werden. Die öffentliche Auffindbarkeit ist gestuft: privat, plattformsichtbar und öffentlich, mit identitätsbereinigten Projektionen für unauthentifizierte Flächen.

12 Technik und Betrieb

Größe	gemessen
Backend	44 658 LOC, 142 Module
Composition Root	724 LOC
Frontend (JS/JSX)	50 754 LOC
Endpunkte	330 in 33 Routern
Services	72
Collections	≈70
Abhängigkeiten	22
Selftests	21 (≈660 Prüfungen)

Tabelle 2: Gemessen an den ausgelieferten Repositories.

Das Backend läuft als einzelner Container auf Cloud Run in einer europäischen Region, mit einer warm gehaltenen Instanz, um Cold Starts in der nutzerseitigen Latenz zu vermeiden. Die Secrets laden beim Start aus einem verwalteten Secret-Store, mit Pflicht- und Optional-Stufen, sodass eine fehlende periphere Integration ein Feature degradiert, statt den Start zu verhindern. Das Herunterfahren entleert auf Signal die Verbindungen. Der Client ist eine React-Single-Page-Anwendung, gebaut mit Vite und auf verwaltetes Hosting veröffentlicht, mit einem Build-Identitäts-Heartbeat, der veraltete Tabs nach einem Deployment neu lädt, und einem Service Worker, dessen Navigationsstrategie genau aus diesem Grund network-first ist.

Die Verifikation ist ein selbstgeschriebenes, abhängigkeitsfreies Selftest-Gerüst: 21 Module reiner Logik mit rund 660 Assertions, sequenziell von einem kleinen Runner mit Exit-Code-Kontrakt ausgeführt und dem Container-Build vorgeschaltet. Weil die Suite keinen Datastore, kein Netz und keine Umgebung berührt, läuft sie in 3,3 Sekunden und lässt sich überall ausführen, auch innerhalb eines Build-Schritts oder auf einer Entwicklermaschine ohne Zugangsdaten. Sie deckt die deterministischen Kerne ab — Spielregeln gegen einen Golden Master mit Fuzzing, Wertungsmathematik, Payload-Validierung, Ranking-Abwicklung gegen einen Datastore-Mock, zeitliche Auflösung, Retry-Klassifikation und die Nutzerdaten-Karte.

Das Frontend steuert eine erhebliche eigene Fähigkeit bei: eine in sich geschlossene 3D-Welt-Engine, direkt auf Three.js gebaut, die deklarative Weltdefinitionen mit einer automatisch entdeckten Prop-Registry rendert, daneben einen Avatar-Baukasten mit zwei Renderern, in dem eine Figur als benannte Teile-Slots und Palettenrollen gespeichert und unabhängig in 2D-Vektor- und 3D-Mesh-Form realisiert wird, auf Client und Server whitelist-bereinigt.

13 Was eine Lizenz überträgt

Konkret erhält ein Lizenznehmer das oben beschriebene Framework:

- **Die Runtime** — ein gehärteter Modell-Client, der Retrieval-Router mit seinen Anti-Konfabulations-Garantien, geordnete Prompt-Zusammenstellung mit cache-bewusster Struktur, der Zeit- und Standortanker und die Kontext-Whitelist je Modus.
- **Die Gedächtnisarchitektur** — verschlüsseltes Transkript, destillierte Shards über einem Ähnlichkeitsgraphen mit beschränktem Abruf und das rekalibrierte Langzeitfeld, mit Einprägung nur durch Behalten und vollständiger Kostenzuordnung.
- **Das Agentensystem** — Agenten als Dokumente, portabel über Gespräch, Gruppen und Anwendungen hinweg, mit deterministischer Ableitung von Namen, Archetypen und Fortschritt.
- **Das Erweiterungsmodell** — Anwendungen als Verzeichniseinträge, die Session- und Lobby-Schicht, der Speicher je Nutzer, der Direktiven-zu-Prompt-Compiler und die Übergabe an externe Anwendungen mit White-Label-Transaktionsmail.
- **Wissen und Moderation** — zweiphasige Ingestion, berechtigungsgebundene Wissensdomänen, der vollständige Lebenszyklus von Beitrag bis Quarantäne und hash-gebundene Freigabe für die Veröffentlichung.
- **Die Ökonomie** — vorausbezahlte Credits, Verbrauchsmessung pro Modell, gehostetes Checkout mit idempotenter Verrechnung, quotenbasierte Freikontingente und ein vollständiges Transaktions-Ledger.
- **Die Interaktionsfläche** — Gruppen, Direktnachrichten, Feeds, editierbare Welten, Quests, Medien-Pipelines und das server-autoritative Spiel-Substrat mit rein platzierungsbasierten Wertungen.
- **Identität und Datenrechte** — Session-Verwaltung mit Widerruf, Verschlüsselung im Ruhezustand und die deklarative Datenkarte, die Export und Löschung steuert.
- **Der Client** — die Anwendungs-Shell, die 3D-Welt-Engine und der Avatar-Baukasten.

Was eine Lizenz nicht überträgt, sind die Produkte, die auf dem Framework gebaut sind. Die Anwendungen, die auf Pantareons Instanz laufen, ihre Agenten und ihr fiktionaler Rahmen sind unsere eigenen; sie sind Beispiele dafür, was das Framework trägt, nicht Teil davon.

14 Fazit

Aeonlink geht von der Beobachtung aus, dass eine KI-Agenten-Plattform größtenteils horizontale Technik ist, und dass der Teil, der es nicht ist — wer die Agenten sind, was sie abrufen dürfen, wie sie moderiert werden, was ein Gespräch kostet —, besser als Konfiguration ausgedrückt wird denn als Code.

So gebaut, wird der unterscheidende Inhalt der Plattform zu einem Dokumentensatz über einer Runtime, die nicht wissen muss, welche Plattform sie bedient. Agenten, Retrieval-Richtlinie, Modellpreise, Anwendungs-Manifeste und Moderationsregeln sind Daten. Eine Runtime bedient Agentengespräch, Anwendungs-Turns und Moderation gleichermaßen. Nur behaltenes Gespräch prägt sich ins Gedächtnis ein. Jeder Modellaufruf durchläuft genau einen Zähler.

Das ist ein Fundament, das ein Unternehmen übernehmen kann, statt es nachzubauen, und es läuft heute mit echten Nutzern, echten Agenten und echtem Geld, das hindurchfließt.