WHITEPAPER

Projekt Ephraim

KI aus der Schule für die Schule - lokal, sicher souverän

INSTITUTION	Lessing-Gymnasium Karlsruhe
EINORDNUNG	Schulische KI-Plattform, lokale Inferenz, Datenschutz, Betrieb
AUTORENSCHAFT	Dr. Daniel Roth für das Team (Johan Becker, Carl Brunner, Anna Csàki, Matias Kühn, Luka Löhr, Levi Neisius)
GENERIERT AM	2. Juli 2026

ORIENTIERUNG

Inhaltsverzeichnis

WHITEPAPER

1. Was ein Whitepaper ist

Ein Whitepaper ist kein Handbuch, keine Werbebroschüre und kein Projekt-Tagebuch. Es ist ein erklärendes Entscheidungsdokument: Es beschreibt ein Problem, ordnet die Lösung fachlich ein, legt Architektur- und Sicherheitsentscheidungen offen und macht nachvollziehbar, warum ein System in einem bestimmten Umfeld vertrauenswürdig, sinnvoll und betreibbar ist.

Für Ephraim heißt das: Dieses Whitepaper verdichtet die ausführlichen Handbücher zu einer nachvollziehbaren Argumentation. Es zeigt, warum eine Schule eine eigene lokale KI-Plattform entwickelt, wie diese Plattform pädagogisch gedacht ist, welche Datenschutzgrenzen gelten und wie der Betrieb kontrolliert wird.

- Es verbindet pädagogische Idee, technische Architektur, Datenschutz und Betrieb.
- Es führt Nicht-Techniker verständlich ein, ohne technische Tiefe zu verstecken.
- Es schafft eine gemeinsame Grundlage für Schulleitung, Kollegium, Eltern, Förderer, Ämter, Datenschutzbeauftragte und Administratoren.
- Es bindet die ausführlichen Handbuchartikel direkt als Unterkapitel in die passenden Hauptkapitel ein.

KURZFASSUNG

2. Executive Summary

Warum Ephraim existiert

Schulen brauchen einen Zugang zu leistungsfähiger KI, der zur schulischen Wirklichkeit passt: pädagogisch steuerbar, datenschutzbewusst, lokal kontrollierbar und erklärbar. Ephraim ist die Antwort auf diese Lücke.

Was Ephraim anders macht

Ephraim ist nicht nur ein Chatfenster. Das System verbindet persönliche Arbeitsbereiche, Projekte, freigegebenes Basiswissen, Visualisierungen, Datei-RAG, lokale Inferenz auf der Spark, Vault-Verschlüsselung und ein Administrationsmodell für Schule.

Warum lokale Inferenz zentral ist

Die Antworterzeugung läuft auf der schuleigenen Spark. Das Sprachmodell nutzt keinen freien Internetzugang. Ephraim bereitet den freigegebenen Kontext vor, die Spark berechnet daraus die Antwort und speichert keine Anfragen.

Was geschützt wird

Private Chats, Dateien, Erinnerungen, Kalender und persönliche Zusammenfassungen gehören zum persönlichen Datenraum. Vault, Schlüsselarchitektur, Logging-Grenzen, Dateiprüfung und Passwort-Recovery sind deshalb keine Zusatzfunktionen, sondern Kern des Systems.

Kernaussagen

- Lokale KI-Inferenz auf der Spark statt Auslagerung an öffentliche Chatdienste.
- Vault-Prinzip für persönliche Daten und klare Trennung von Projektdaten.
- Freigegebenes Basiswissen, Wikipedia und Projektmaterialien werden als Kontext vorbereitet; die Spark surft nicht im Internet.
- Admin-Betrieb mit Status, Cron, Logging, Datei-Sicherheit, Drittanbieter-Komponenten, Limits und Benchmarks.
- Entwickelt als besonderes Lehrer-Schüler-Projekt mit KI-gestützter Code-Generierung, Tests, Audits und manueller Prüfung.

EINORDNUNG

3. Warum und wie Ephraim gebaut wurde

> **Ephraim entsteht aus einer schulischen Spannung: KI ist für Lernen und Arbeiten relevant, aber die üblichen Plattformen passen nicht sauber zu Datenschutz, pädagogischer Verantwortung und schulischer Kontrolle.**

Eine Schule benötigt kein beliebiges Werkzeug, das Antworten erzeugt. Sie benötigt ein System, das Aufgaben, Rollen, Materialien, Grenzen und Verantwortlichkeiten versteht. Genau deshalb ist Ephraim als Schulplattform gebaut: mit Nutzerkonten, Projekten, lokalen Diensten, Admin-Werkzeugen und Dokumentation.

Das Projekt ist zugleich pädagogisch, technisch und organisatorisch. Sein Entwicklungsprozess gehört zur Aussage: Ephraim zeigt nicht nur, welche Plattform eine Schule braucht, sondern auch, wie ein kleines schulisches Team mit KI-gestützter Code-Generierung, Audits, Tests und manueller Prüfung eine solche Plattform überhaupt bauen kann.

3.1 Warum Ephraim nötig wurde

> Ephraim wurde nicht gebaut, weil Schulen noch ein weiteres Chatfenster brauchen. Der Grund ist grundsätzlicher: KI wird im Unterricht erst dann wirklich nützlich, wenn sie mit echtem schulischem Kontext arbeiten darf. Dieser Kontext enthält schnell personenbezogene Daten. Ephraim ist der Versuch, diese Verarbeitung lokal, zweckgebunden, verschlüsselt und schulisch kontrollierbar zu machen.

Stand: Juni 2026

3.1.1 Die kurze Antwort

Wir mussten Ephraim bauen, weil eine ernsthafte schulische KI-Plattform nicht dauerhaft so tun kann, als kämen im Unterricht keine personenbezogenen Daten vor. Sobald KI mehr leisten soll als allgemeine Beispieltexte, berührt sie Namen, Rollen, Klassen, Projekte, Arbeitsstände, Dateien, Termine, Interessen, Lernwege und Rückmeldungen.

Die Alternative wäre ein künstlich entkernter KI-Einsatz: Man dürfte nur abstrakte Fragen stellen, keine echten Materialien nutzen, keine persönlichen Arbeitsstände fortführen und keine Projektkontexte einbeziehen. Das wäre datenschutzrechtlich einfacher, pädagogisch aber deutlich schwächer.

> **Kernsatz:** Ephraim entsteht aus der Einsicht, dass schulische KI nur dann sinnvoll wird, wenn personenbezogene Verarbeitung nicht verdrängt, sondern sauber begrenzt, geschützt und erklärt wird.

3.1.2 Warum Schule fast immer personenbezogen wird

Personenbezogene Daten sind nicht nur Noten oder Adressen. Im Schulalltag reicht oft schon ein Zusammenhang zu einer identifizierbaren Person. Ein Chat über ein eigenes Projekt, eine hochgeladene Datei, eine Frage zum Kalender, eine Reflexion über den Lernstand oder ein Lehrerfeedback kann personenbezogen sein, auch wenn niemand etwas besonders Sensibles eingeben wollte.

Schulischer Vorgang	Warum Personenbezug entsteht
Persönlicher Chat	Der Verlauf gehört zu einem Konto und kann Interessen, Fragen oder Schwierigkeiten zeigen.
Projektarbeit	Teilnahme, Arbeitsstand, Materialien und Abschluss hängen an konkreten Schülerinnen und Schülern.
Dateien	Arbeitsblätter, Entwürfe, Fotos, Protokolle oder Texte können Namen, Leistungen oder persönliche Bezüge enthalten.
Personalisierung	Eine hilfreiche Begrüßung, Erinnerungen oder Interessen setzen voraus, dass das System etwas über die Person behält.
Fazit und Monitoring	Auch reduzierte Projektmetadaten sagen etwas über Teilnahme, Aktivität und Abschlussstatus aus.

Genau deshalb ist „Wir geben einfach keine personenbezogenen Daten ein“ keine tragfähige Grundlage für eine schulweite Plattform. Es kann als Vorsichtsregel für einzelne Experimente funktionieren. Für Unterricht, Projekte, Dateiablage, persönliche Arbeitsräume und langfristige Nutzung reicht es nicht.

3.1.3 Warum anonyme KI pädagogisch schnell an Grenzen kommt

Eine KI ohne echten Kontext kann allgemeine Erklärungen liefern. Sie kann eine Formel erklären, einen Beispieltext verbessern oder ein Diagramm beschreiben. Das ist nützlich, aber es bleibt vom konkreten Unterricht getrennt. Schule arbeitet dagegen mit Aufgaben, Materialien, Klassen, Lernständen, Rückfragen und Projekten, die über mehrere Tage oder Wochen laufen.

Wenn KI dabei helfen soll, muss sie wissen, woran gerade gearbeitet wird. Sie muss ein Projektmaterial kennen, eine frühere Frage wiederfinden, eine Datei einordnen oder einen Auftrag der Lehrkraft berücksichtigen. In diesem Moment entsteht Verantwortung: Wer entscheidet, welcher Kontext zugeschaltet wird? Wer darf ihn sehen? Wie lange bleibt er gespeichert? Wer kann ihn löschen? Welche Auswertung sieht die Lehrkraft?

> **Praktische Folge:** Pädagogisch sinnvolle KI braucht Kontext. Datenschutz bedeutet deshalb nicht Kontextverbot, sondern kontrollierte Kontextverarbeitung.

3.1.4 Rechtlich möglich ist nicht automatisch praktisch beherrschbar

Dieser Artikel behauptet nicht, dass Schulen niemals externe KI-Dienste einsetzen dürfen. Je nach Dienst, Vertrag, Konfiguration, Schulträger, Rechtsgrundlage, Datenkategorie und Einsatzkonzept kann eine externe Verarbeitung rechtlich prüfbar sein. Genau diese Einzelfallprüfung bleibt wichtig.

Das praktische Problem liegt woanders: Eine Schule muss die Verarbeitung erklären und verantworten können. Dazu gehören Zweckbindung, Datenminimierung, Speicherfristen, technische und organisatorische Maßnahmen, Auftragsverarbeitung, Unterauftragnehmer, mögliche Drittlandbezüge, Supportzugriffe, Protokollierung, Löschung, Rollenrechte und die Frage, ob sich das Produktverhalten später ändert.

Für Ephraims Zielbild war diese Lage zu eng. Wir wollten nicht nur eine rechtlich vertretbare Chatlösung, sondern eine Plattform, die mit echten Schulkontexten arbeitet und trotzdem nachvollziehbar bleibt. Dafür ist eine eigene, lokale Architektur der belastbarere Weg.

3.1.5 Der technische Kern: KI muss Klartext sehen

Verschlüsselung ist zentral, aber sie löst nicht jede Frage. Ein Sprachmodell kann eine Antwort nur berechnen, wenn es die konkrete Frage und den erlaubten Kontext im Klartext verarbeitet. Ende-zu-Ende-Verschlüsselung gegenüber der Stelle, die die Antwort berechnet, ist bei normaler KI-Inferenz deshalb kein einfacher Schalter.

Transportverschlüsselung schützt den Weg. Verschlüsselung auf Platte schützt gespeicherte Daten. Beides ist wichtig. Aber irgendwo muss der Text für die Antworterzeugung entschlüsselt im Arbeitsspeicher liegen. Bei einem externen KI-Dienst liegt dieser Klartextmoment in der Infrastruktur dieses Dienstes. Bei Ephraim liegt er auf der schuleigenen Spark im isolierten Netzsegment.

Das ist der entscheidende technische und rechtlich-praktische Unterschied. Ephraim behauptet nicht, dass andere Anbieter keine Schutzmaßnahmen hätten. Ephraim entscheidet aber, dass der unvermeidbare Klartextmoment nicht in einer fremden KI-Cloud stattfinden soll, sondern auf einer lokalen Maschine unter schulischer Kontrolle.

> **Merksatz:** KI braucht für die Antwort kurz Klartext. Ephraim verlegt diesen Klartextmoment in die eigene Infrastruktur und reduziert die dauerhafte Ablage wieder auf verschlüsselte Daten.

3.1.6 Warum Ephraim lokal rechnet

Ephraim trennt die sichtbare Webanwendung und die KI-Rechnung. Nutzerinnen und Nutzer arbeiten im Browser mit dem Webserver. Die eigentliche Antworterzeugung, Embeddings und das Vorlesen laufen auf der Ephraim Spark. Die Spark ist kein öffentlicher Webdienst, nutzt keinen freien Internetzugang für Antworten und speichert Anfragen nicht dauerhaft.

Diese Architektur passt zum schulischen Problem: Der Webserver prüft Anmeldung, Rollen, Projekte, Vault, Dateien und freigegebenes Basiswissen. Erst danach wird ein begrenztes Kontextpaket an die Spark geschickt. Die Spark berechnet daraus die Antwort und gibt sie zurück. Danach liegt der Verlauf wieder verschlüsselt im persönlichen Datenraum oder im jeweils vorgesehenen Projektkontext.

- **Lokale Inferenz:** Chatantworten entstehen auf der schuleigenen Spark.
- **Lokale Embeddings:** Suchvektoren für Dateien, Projektmaterialien und Basiswissen werden lokal berechnet.
- **Lokales TTS:** Vorlesen läuft ausschließlich über den internen Spark-TTS-Dienst; ohne diesen Dienst bleibt die Vorlesefunktion deaktiviert.
- **Vault-Prinzip:** Private Chats, Dateien, Erinnerungen und Zusammenfassungen werden verschlüsselt gespeichert.
- **Rollen und Projekte:** Lehrkräfte steuern Projektauftrag und Materialien, lesen aber keine privaten Schülerchats.

3.1.7 Die gemeinsame Herausforderung aller Anbieter

Jeder Anbieter einer schulischen KI-Lösung muss im Kern dieselben Fragen beantworten: Welche Daten darf das Modell sehen? Wo entsteht der unvermeidbare Klartextmoment? Wer betreibt die Infrastruktur? Welche Protokolle entstehen? Welche Unterauftragnehmer sind beteiligt? Wie funktionieren Löschung, Export, Rollenrechte, Supportzugriffe und spätere Produktänderungen?

Externe Dienste können darauf je nach Vertrag, Konfiguration, Betriebsmodell und Produktstand gute Antworten geben. Dieser Artikel bewertet solche Lösungen nicht pauschal. Für eine Schule bleibt aber die praktische Aufgabe, diese Antworten dauerhaft zu verstehen, zu prüfen und gegenüber Schulleitung, Schulträger, Eltern, Kollegium und Datenschutz zu vertreten.

Die Entscheidung für Ephraim verdichtet diese Herausforderungen in einem eigenen System. Statt sie auf viele Vertrags-, Produkt- und Cloud-Ebenen zu verteilen, macht Ephraim sie sichtbar: lokale Spark für den unvermeidbaren Klartextmoment, Vault für private Ablage, Rollenmodell für Zuständigkeiten, Projektgrenzen, sparsame Logs, Export und Löschung. Die Verantwortung verschwindet dadurch nicht. Sie wird konkreter und prüfbarer.

3.1.8 Was Ephraim dadurch ermöglicht

Der Eigenbau ist kein Selbstzweck. Er ermöglicht Funktionen, die in einer rein anonymen oder externen, für die Schule weniger unmittelbar kontrollierbaren KI-Nutzung nicht dieselbe Qualität hätten.

- Mein Ephraim kann persönliche Chats, Projekte und Hinweise zusammenführen.
- Der Vault schützt private Daten und macht Passwortfolgen erklärbar.
- Projekte können echte Materialien und Aufträge enthalten, ohne private Chats der Schülerinnen und Schüler zu öffnen.
- Basiswissen und Wikipedia werden als kontrollierter Kontext vorbereitet, nicht als freies KI-Browsing.
- Inferenz auf der Spark erklärt, warum Antworten lokal entstehen und trotzdem kurz Klartext brauchen.
- Datenschutz tiefgehend beschreibt die Datenräume, Rollen, Löschwege und Klartextmomente genauer.

3.1.9 Was trotzdem Verantwortung bleibt

Lokale Verarbeitung macht Datenschutz nicht automatisch erledigt. Auch Ephraim braucht klare Rollen, gute Passwörter, kontrollierte Adminrechte, sichere Updates, sparsame Logs, Löschregeln, Backups, Schulungen und eine ehrliche Bewertung der verbleibenden Risiken. Der Unterschied ist: Diese Verantwortung liegt nicht versteckt in einer entfernten Produktkette, sondern wird im System sichtbar.

Deshalb dokumentiert Ephraim die Systemarchitektur, die Kryptoarchitektur, die Logging-Grenzen, die Dateiverarbeitung und den Entwicklungsprozess ausführlich. Wer die Plattform beurteilt, soll nicht nur eine Datenschutzbehauptung lesen, sondern die tatsächlichen Grenzen verstehen können.

3.1.10 Die eigentliche Antwort auf „Warum?“

Ephraim wurde gebaut, weil Schule KI nicht nur ausprobieren, sondern verantwortbar nutzen will. Das geht nur, wenn personenbezogene Daten nicht tabuisiert, sondern geschützt verarbeitet werden. Genau dafür kombiniert Ephraim lokale KI-Rechnung, persönliche Verschlüsselung, Rollen, Projekte, begrenzte Auswertungen und transparente Dokumentation.

Die Frage lautet deshalb nicht: „Warum baut eine Schule eine eigene KI?“ Die präzisere Frage lautet: „Wie soll eine Schule ernsthaft mit KI arbeiten, wenn sie echten Unterricht und personenbezogene Daten dauerhaft ausblenden muss?“ Ephraim ist die Antwort auf diese Spannung.

Dieser Artikel erklärt die Projektentscheidung hinter Ephraim. Er ersetzt keine rechtliche Einzelfallprüfung, sondern ordnet ein, warum die lokale Architektur für das schulische Zielbild gewählt wurde.

Quelle: web/manuals/warum-ephraim/index.html

3.2 Entwicklungsprozess

> **Ephraim ist nicht als fertiges Produkt eingekauft worden, sondern als schulisches Softwareprojekt gewachsen. Dieser Meta-Artikel erklärt, wie aus Idee, Projektboard, Git-Verlauf, Tests, Dokumentation und Handbüchern ein System entsteht, das für den späteren Unterrichtseinsatz und den Betrieb nachvollziehbar weiterentwickelt wird.**

Stand: Juni 2026

3.2.1 Warum es diesen Meta-Artikel gibt

Die übrigen Handbuchartikel erklären Funktionen: Chat, Projekte, Datenschutz, Inferenz, Einstellungen oder Administration. Dieser Artikel erklärt die Ebene dahinter: wie Ephraim entwickelt wird, welche Arbeitsregeln das Projekt prägen und warum der Git-Verlauf viele kleine, konkrete Schritte zeigt statt weniger großer Sammeländerungen.

Der Entwicklungsprozess ist selbst ein Teil der Vertrauensarchitektur. Schulen, Eltern, Förderer, Datenschutzbeauftragte und Administratorinnen müssen nicht jede Codezeile lesen. Sie sollen aber erkennen, dass Ephraim nicht aus beliebigen Experimenten besteht. Planung, Umsetzung, Prüfung, Dokumentation und Handbuchpflege sind getrennte, sichtbare Schichten.

> **Kernsatz:** Ephraim wird iterativ entwickelt, aber nicht beliebig. Jede produktnahe Änderung muss zu Architektur, Datenschutz, geplanter Unterrichtsnutzung, Betrieb und Dokumentation passen.

3.2.2 Das Team hinter Ephraim

Ephraim ist ein Lehrer-Schüler-Projekt. Das ist für den Entwicklungsprozess entscheidend: Ephraim wird noch nicht im Unterricht eingesetzt. Anforderungen entstehen deshalb aus eigener Beobachtung und Erprobung durch Schülerinnen, Schüler und Lehrkräfte: Wie würde eine typische Unterrichtssituation aussehen? Welche Vorbereitung braucht eine Lehrkraft? Welche Oberfläche verstehen Lernende ohne zusätzliche Erklärung? Ist ein Sicherheitsmechanismus erklärbar? Wird eine Antwort als nützlich, irritierend oder zu mächtig erlebt?

Schülerinnen und Schüler sind damit nicht Testpersonen eines fertigen Produkts. Sie arbeiten an der Frage mit, welche Anforderungen ein schulisches KI-System überhaupt erfüllen muss. Diese Perspektive ist für Ephraim grundlegend: Der spätere Einsatz im Unterricht wird vorbereitet, indem das Team vorher aus Schüler- und Lehrerperspektive beobachtet, ausprobiert, kritisiert und daraus eigene Anforderungen formuliert.

Von links nach rechts: Johan Becker (Entwicklung), Matias Kühn (Qualitätssicherung), Luka Löhr (Entwicklung Lead), Dr. Daniel Roth (Leitung und Entwicklung), Levi Neisius (Qualitätssicherung), Anna Csàki (Qualitätssicherung), Carl Brunner (Qualitätssicherung).

Johan Becker Entwicklung, punktuelle Kernbeiträge und technische Umsetzung einzelner Bausteine.

Matias Kühn Qualitätssicherung aus Nutzersicht, Rückmeldung zu Verständlichkeit, Wirkung und Stabilität.

Luka Löhr Entwicklung Lead mit Schwerpunkt Architektur, Sicherheit, Streaming, Runtime und Adminlogik.

Dr. Daniel Roth Projektleitung und Entwicklung, Verbindung von Unterricht, Produktentscheidungen, Datenschutz und Betrieb.

Levi Neisius Qualitätssicherung, insbesondere Prüfung von Bedienwegen, Projekterfahrung und eigener Erprobung.

Anna Csàki Qualitätssicherung, Rückmeldungen zur Schülerperspektive, Sprache und Nachvollziehbarkeit.

Carl Brunner Qualitätssicherung, Tests von Funktionen, Grenzen und Unterrichtstauglichkeit.

Diese Rollen sind nicht starr wie in einem Unternehmen. Sie beschreiben den Arbeitsschwerpunkt. Gerade die Qualitätssicherung ist mehr als „Fehler suchen“: Sie prüft, ob Ephraim für Lernende plausibel bleibt. Ein technischer Erfolg reicht nicht, wenn eine Funktion für den späteren Schulalltag unklar, ablenkend oder schwer erklärbar wäre.

> **Besonderheit:** Ephraim wird in der Schule entwickelt und für Schule entwickelt. Die Rückkopplung zwischen Code, Schülerperspektive, Lehrerperspektive, Datenschutz und Nutzererleben ist deshalb kürzer als in einem extern eingekauften Produkt.

3.2.3 Was die HOPP-Foundation ermöglicht

Die HOPP-Foundation hat die Beschaffung einer NVIDIA DGX Spark für die lokale Schul-KI finanziert. Diese Förderung ist für den Entwicklungsprozess strukturell wichtig: Dadurch entsteht eine lokale, schulisch kontrollierte KI-Entwicklung, statt die spätere schulische Nutzung direkt an externe Cloud-Chatbots zu koppeln.

Die Hardwareförderung schafft keinen fertigen Unterrichtsassistenten. Sie schafft die Voraussetzung dafür, dass das Team die anspruchsvollen Fragen selbst bearbeiten kann: Wie wird Inferenz lokal betrieben? Wie werden Rollen, Projekte und private Daten getrennt? Wie erklärt man einer Schule, was auf dem Server passiert? Wie prüft man ein System, das Antworten generiert, Dateien auswertet, Projekte begleitet und gleichzeitig Datenschutzgrenzen einhalten muss?

Warum Ephraim entwickelt wurde

Der HOPP-Antrag zeigt, dass Ephraim nicht aus einer kurzfristigen Begeisterung für Chatbots entstanden ist. Am Lessing-Gymnasium gab es bereits vorher eine längere Entwicklungslinie: ein Jugend-forscht-Projekt zu einem lokalen KI-Assistenten, ein lokaler KI-Testlauf für das Kollegium, eine schulöffentliche Diskussion über KI, schulinterne Fortbildungen und praktische Erfahrungen mit KI-generierten, passgenauen Verwaltungswerkzeugen und unterrichtsnahen Prototypen. Ephraim ist die systematische Weiterentwicklung dieser Erfahrungen.

Der Antrag benennt das Grundproblem klar: Der Nutzen von KI wird besonders groß, wenn sie mit konkreten schulischen Kontexten arbeiten darf. Persönliches Lernen, Rückmeldungen, Lernstandsanalysen, Projektarbeit und Verwaltungsvorgänge lassen sich nur begrenzt sinnvoll unterstützen, wenn jeder Personenbezug ausgeschlossen bleibt. Gleichzeitig fehlten für Schulen Lösungen, die personenbezogene Daten verarbeiten dürfen und zugleich die Leistungsfähigkeit moderner Modelle erreichen. Genau aus dieser Spannung entsteht Ephraim.

Hinzu kommt die technische Erfahrung aus dem lokalen Testlauf: Ohne ausreichende Performance entsteht kein realistischer KI-Eindruck. Ein zu langsames Modell zeigt zwar das Prinzip, bildet aber nicht die Möglichkeiten moderner Inferenz ab. Die DGX Spark schafft deshalb nicht nur mehr Geschwindigkeit, sondern eine neue Entwicklungslage: Das Team prüft Anforderungen an eine lokale Schul-KI praktisch, statt nur über Cloud-Systeme zu sprechen.

Ausgangslage im Antrag	Konsequenz für Ephraim
Schulen brauchen KI-Unterstützung, die mit konkreten Lern- und Verwaltungsbezügen umgehen kann.	Ephraim trennt Rollen, Projekte, private Daten und Basiswissen technisch, statt alles in einen unklaren Chat zu werfen.
Cloud-Systeme sind leistungsfähig, aber für sensible Schuldaten problematisch.	Die Inferenz läuft lokal auf der Spark; Datenflüsse, Modelle und Zugriffe bleiben schulisch kontrollierbar.
Viele offizielle Schullösungen bleiben hinter modernen Modellen zurück.	Ephraim orientiert sich an aktueller Modellleistung, baut aber eigene Sicherheits-, Projekt- und Bediengrenzen darum herum.
Passgenaue KI-generierte Tools hatten bereits gezeigt, wie stark kleine, genau passende Software den Schulalltag entlastet.	Ephraim wird nicht als isolierter Chatbot gedacht, sondern als Plattform für Chat, Projekte, Dateien, Basiswissen, Verwaltung und Handbücher.
Lokale KI braucht geeignete Hardware; ein alter Server erzeugt nur einen eingeschränkten Eindruck.	Die DGX Spark macht lokale Inferenz schnell genug, um Anforderungen realistisch zu erproben und weiterzuentwickeln.

Technische Souveränität Die Schule betreibt die zentrale KI-Hardware selbst und kann Architektur, Modelle, Datenflüsse und Updates kontrollieren.

Datenschutz als Praxis Lokale Inferenz, Verschlüsselung und klare Projektgrenzen werden nicht nur behauptet, sondern technisch umgesetzt und dokumentiert.

Pädagogische Nähe Funktionen entstehen aus eigener Beobachtung als Schüler und Lehrer, nicht aus abstrakten Produktversprechen.

3.2.4 Die drei Gedächtnisse des Projekts

Ephraim nutzt drei Formen von Projektgedächtnis. Sie erfüllen unterschiedliche Aufgaben und ersetzen einander nicht.

Gedächtnis	Aufgabe	Was daraus ablesbar ist
Projektboard	Ideen, Aufgaben, offene Entscheidungen und Verlauf werden gesammelt.	Welche Themen geplant, priorisiert und in Phasen gegliedert wurden.
Git-Verlauf	Jede technische Änderung wird als Commit gespeichert.	Wann eine Funktion technisch eingeführt, korrigiert, entfernt oder gehärtet wurde.
Dokumentation und Handbücher	Technische Zusammenhänge und menschlich lesbare Erklärungen werden getrennt gepflegt.	Wie Codeentscheidungen für Betrieb, Unterricht und externe Interessierte übersetzt werden.

Das fachliche Projektgedächtnis liegt in einem internen Notion-Projektboard mit Aufgaben, Hauptseite und Verlaufschronik. Notion ist dabei kein Teil der Ephraim- Laufzeit und verarbeitet keine KI-Anfragen der Nutzerinnen und Nutzer; es ist ein internes Werkzeug für Planung und Rückblick. Der Git-Verlauf ist die technische Spur. Die Dokumentation erklärt, wie das System funktioniert. Die Handbücher verdichten diese Informationen für Menschen ohne Code-Kenntnis.

3.2.5 Wie eine Änderung durch das Projekt läuft

Eine typische Änderung beginnt nicht mit einer fertigen Spezifikation. Sie beginnt mit einer eigenen Beobachtung: Eine typische Unterrichtssituation wird später eine Funktion brauchen, der Betrieb braucht mehr Kontrolle, Datenschutz braucht eine präzisere Grenze oder die Bedienung ist noch nicht klar genug. Daraus entsteht ein kleiner, überprüfbarer Arbeitsschritt.

Der Zyklus ist bewusst klein geschnitten. Eine Änderung ist abgeschlossen, wenn sie im System funktioniert, geprüft ist und für die betroffene Zielgruppe verständlich erklärt wurde.

1. **Bedarf klären:** Das Problem wird konkret beschrieben, etwa als erwarteter Unterrichtsablauf, Adminaufgabe oder Datenschutzgrenze.
2. **Schritt begrenzen:** Die Änderung wird so klein geschnitten, dass sie in einem überschaubaren Lauf gebaut und geprüft wird.
3. **Implementieren:** Code wird in der passenden Schicht geändert: Oberfläche, API, Service, Datenbank, Betrieb oder Handbuch.
4. **Prüfen:** Je nach Risiko folgen Syntaxprüfung, Funktionstest, E2E-Test, Playwright-Screenshot, Smoke-Test oder manuelle Sichtprüfung.
5. **Dokumentieren:** Technische Dokumentation und Handbücher werden angepasst, wenn Verhalten, Betrieb oder Nutzerverständnis betroffen sind.

In einem Lehrer-Schüler-Projekt ist dieser Zyklus besonders konkret. Ein Beispiel: Wenn eine Projektfunktion technisch funktioniert, ist sie noch nicht fertig. Sie muss für eine Lehrkraft in der Vorbereitung handhabbar sein, für Schülerinnen und Schüler im Projektchat eindeutig bleiben und für Administratorinnen betrieblich erklärbar sein. Deshalb erzeugt eine Beobachtung aus der Qualitätssicherung oft keine einzelne Fehlerkorrektur, sondern eine kleine Kette: Text präzisieren, UI-Zustand verbessern, Test ergänzen, Handbuch nachziehen.

3.2.6 Was der Verlauf von Januar bis Anfang Juni 2026 zeigt

Zum Stand 04.06.2026 enthält der lokale Hauptverlauf über 1700 Commits. Die interne Verlaufschronik unterscheidet elf Phasen bis 04.06.2026; die Commit-Historie bestätigt die späten Schwerpunkte: Recovery, Chat-Aufbewahrung, TTS-Härtung, öffentliche Dokumentation, Whitepaper und Benchmarks. Der Verlauf zeigt einen klaren Reifeweg: von der Chat-Vertikale zur datenschutzbewussten und nachweisfähigen Schulplattform.

Zeitraum	Schwerpunkt	Ergebnis für Ephraim
30.01.-11.02.	Fundament, Anmeldung, Chat, Adminbasis, DDEV	Eine erste End-to-End-Vertikale: anmelden, chatten, verwalten, lokal entwickeln.
13.02.-22.02.	Sicherheit, Zwei-Faktor-Authentifizierung, Runtime-Überblick	Der Betrieb wird sichtbarer; Authentifizierung und Adminflächen werden gehärtet.
März 2026	Builder, Prompts, Rollen, Service-Struktur	Lehrkräfte bekommen Werkzeuge für Projekte und Systemanweisungen; das Backend wird modularer.
Anfang April	Hilfe, Renderer, Formeln, Code, Plots, Streaming-Resilienz	Der Chat wird unterrichtstauglicher: Markdown, Mathematik, Visualisierungen und stabile Streams.
30.04.-08.05.	Infrastruktur, Verschlüsselung, RAG, Dateien, Basiswissen	Persönliches Wissen und Datei-Ingest werden produktnah; die Kryptoarchitektur wird deutlich stärker.
09.05.-21.05.	Internationalisierung, MCP, Auditberichte, Komponentenworkflow	Ephraim wird betriebsreifer: mehrere Sprachen, kontrollierte Wartung, Freigaben und Berichte.
22.05.-26.05.	Projekte, Gastzugang, Export, rechtliche Seiten	Der geplante Unterrichtseinsatz wird vollständiger vorbereitet: Projektende, Fazit, PDF-Export, Impressum und Datenschutz.
27.05.-28.05.	Antwort überarbeiten, Spark-TTS, Runtime-Status	Der Chat wird komfortabler; Vorlesen bekommt einen kontrollierten Spark-Pfad; deaktivierte V2-Dienste werden im Status sauber abgegrenzt.
29.05.-01.06.	Handbücher, Projekt-Fazit, TTS-Härtung, Kalender-Vault-Krypto	Ephraim wird erklärbarer und abschließbarer: Handbücher wachsen, Projektabschlüsse werden verständlicher, TTS wird gehärtet und private Kalender werden stärker an den Vault gebunden.
02.06.	Buddy-/Vault-Recovery, Chat-Aufbewahrung, TTS ohne Fallback, NotebookLM	Die Nachkorrektur trotz Feature Freeze schließt reale Ausfall- und Betriebsfälle: Passwortverlust wird handhabbarer, Chatwachstum begrenzbar, Vorlesen eindeutig Spark-gebunden.
03.06.-04.06.	Öffentliche Dokumentation, Whitepaper, Inferenz-Benchmarks, Medien	Die Erklärschicht wird selbst zum Produktbestandteil: Handbücher werden deploybar, das Whitepaper versendbar und die Spark-Leistung messbar.

Die letzten drei Phasen sind für den Entwicklungsprozess besonders wichtig. Sie zeigen den Übergang von schneller Produktentwicklung zu überprüfbarer Betriebsreife. Feature Freeze bedeutet in diesem Kontext nicht, reale Ausfallpfade zu ignorieren. Buddy-Recovery, Chat-Aufbewahrung und TTS ohne Fallback sind notwendige Korrekturen für Datenschutz, Kontowiederherstellung und Betrieb. Whitepaper und Benchmarks zeigen anschließend, dass Ephraim nicht nur gebaut, sondern erklärt und nachgewiesen werden muss.

3.2.7 Technische Prinzipien, die die Entwicklung steuern

Der Entwicklungsprozess folgt einigen festen technischen Entscheidungen. Sie sind nicht nur Geschmack, sondern senken Betriebsrisiko und machen das System für Schule und Administratoren beherrschbar.

- **Lokale KI statt Cloud-Abhängigkeit:** Die produktive Inferenz läuft auf der Ephraim Spark. Das Sprachmodell verarbeitet die freigegebenen Inhalte dort, nicht bei externen Chatbot-Diensten.

- **Einfacher Webstack:** Ephraim nutzt PHP, MySQL in Produktion, eine MariaDB-kompatible DDEV-Entwicklungsumgebung und Vanilla JavaScript. Es gibt keinen Frontend-Build-Schritt und kein großes JavaScript-Framework.
- **DDEV für lokale Entwicklung:** Entwicklerinnen und Entwickler arbeiten mit einer reproduzierbaren lokalen Umgebung, bevor Änderungen in den Schulbetrieb gehen.
- **Lokal gebündelte Bibliotheken:** Drittanbieter-Komponenten werden bewusst ins Projekt aufgenommen und kontrolliert verwaltet, statt bei jedem Seitenaufruf von externen Quellen geladen zu werden.
- **Strenge Trennung von Code und Darstellung:** Inline-JavaScript und Inline-CSS werden vermieden. Eine strenge Content-Security-Policy schützt gegen eingeschleusten Browsercode.
- **Dokumentation als Arbeitsmittel:** Technische Dokumente erklären den Code für Menschen und für KI-Assistenten, die bei der Entwicklung helfen.

> **Konsequenz:** Neue Funktionen müssen zum bestehenden Stack passen. Eine bequemere Bibliothek, ein schneller externer Dienst oder ein automatisches Update wird nicht übernommen, wenn dadurch Kontrolle, Datenschutz oder Wartbarkeit schlechter werden.

3.2.8 Rolle von KI-Assistenten im Entwicklungsprozess

Ephraim wird mit Unterstützung von KI-Assistenten entwickelt. Diese Unterstützung ist kein Widerspruch zum Anspruch an Datenschutz und Kontrolle. Entscheidend ist die Rolle: KI-Assistenten helfen beim Lesen, Strukturieren, Erklären, Testen, Refactoring, Schreiben von Dokumentation und bei der Code-Generierung. Genutzt werden insbesondere Codex, Cursor und Claude. Sie sind Arbeitswerkzeuge im Entwicklungsprozess, nicht die Autorität über das System.

Die technische Autorität bleibt der ausgeführte Code, die Migrationen, Tests, Konfigurationen und Dokumentation. Wenn ein Assistent einen Vorschlag macht, muss der Vorschlag in diese Schichten passen. Der Git-Verlauf zeigt deshalb nicht nur Feature- Commits, sondern auch Härtungen, Korrekturen, Dokumentationsspiegel, Testpläne, Screenshotläufe und Nacharbeiten.

Der entscheidende Paradigmenwechsel liegt darin, dass ein kleines schulisches Team heute Entwicklungsarbeit leisten kann, die Ende 2025 in dieser Geschwindigkeit und Breite für ein Lehrer-Schüler-Projekt praktisch nicht erreichbar gewesen wäre. KI erzeugt Code, erklärt bestehende Teile, findet Zusammenhänge und schreibt erste Entwürfe. Das ersetzt keine Verantwortung. Es verschiebt aber die Grenze dessen, was ein kleines Team mit guter Führung, klaren Anforderungen und konsequenter Prüfung bauen kann.

Deshalb ist die Kontrollschicht genauso wichtig wie die Generierung. Jede relevante Änderung wird über passende Prüfungen abgesichert: automatisierte Tests, Smoke- und E2E-Läufe, KI-gestützte Sicherheits- und Architektur-Audits, Screenshot-Prüfungen sowie manuelle Tests durch Schülerinnen, Schüler und Lehrkräfte. Auch Audits kommen also aus der KI-gestützten Arbeitsweise; verbindlich werden sie erst, wenn ihre Befunde verstanden, priorisiert, umgesetzt und nachgeprüft sind. Manuelle Prüfung bleibt notwendig, weil ein Test zwar technische Erwartungen kontrolliert, aber nicht vollständig beurteilt, ob ein Ablauf verständlich, pädagogisch sinnvoll und vertrauenswürdig ist.

KI-Werkzeuge helfen bei	Was das Team selbst prüft und verantwortet
Code-Generierung mit Codex, Cursor und Claude	Review, Tests, Bewertung KI-gestützter Audits und der Entscheidung, ob der Code ins System passt
Analyse von Code, Dokumentation und Änderungsfolgen	Die fachliche Entscheidung, ob eine Funktion in den späteren Schulbetrieb passt
Erstellen von Entwürfen, Tests, Handbuchtexten und Prüfskripten	Prüfung von Behauptungen und Seiteneffekten, bevor etwas übernommen wird
Wiederholbare Aufgaben wie Spiegel-Dokumentation und Screenshot-Checks	Der Umgang mit Secrets, Passwörtern oder produktiven Zugangsdaten
Vorschläge für Refactoring, Fehlerbehebung und Sicherheitsgrenzen	Manuelle Erprobung der Bedienung und Bewertung aus Schüler- und Lehrerperspektive
KI-gestützte Sicherheits-, Architektur- und Qualitätsaudits	Einordnung der Befunde, Ausschluss falscher Treffer und konkrete Nachprüfung im Code

3.2.9 Wie Qualität geprüft wird

Qualität entsteht in Ephraim aus mehreren Prüfebenen. Nicht jede Änderung braucht den gleichen Aufwand. Eine neue Adminfunktion hat andere Risiken als ein Handbuchabsatz, eine Datenbankmigration andere Risiken als ein Screenshot. Der Prüfaufwand richtet sich nach Reichweite und Risiko.

- **Syntax und Laufzeit:** PHP-Dateien werden auf Syntax geprüft; JavaScript-Module werden syntaktisch validiert, wenn sie geändert werden.
- **Lokale Umgebung:** DDEV bildet Webserver, PHP und Datenbank lokal nach, damit Änderungen vor dem Einsatz getestet werden.
- **Funktionstests:** Für kritische Wege gibt es Smoke- und E2E-Tests, zum Beispiel für Projekte, Mailfluss, Ratenbegrenzung oder PDF-Rückläufe.
- **Visuelle Prüfung:** Bei Handbüchern und UI-Arbeit werden Screenshots genutzt, damit Text, Navigation und Bilder nicht nur technisch vorhanden, sondern lesbar sind.
- **Dokumentationsprüfung:** Neue Handbuchartikel müssen in die zentrale Themenliste eingetragen sein, einen passenden Artikelaufbau haben und ohne eigene Navigationsduplikate auskommen.
- **Sicherheitsprüfung:** Änderungen an Anmeldung, Rollen, Projekten, Uploads, Exporten und Adminfunktionen werden besonders eng an Datenschutz- und Zugriffskanten geprüft.

E2E bedeutet End-to-End: Ein vollständiger Weg wird aus Nutzersicht durchgespielt. Bei den Projektprüfungen heißt das nicht nur „Button geklickt“. Ein synthetisches Unterrichtsprojekt wird angelegt, Teilnehmende starten oder bleiben aus, ein Projektchat wird geführt, ein persönlicher Abschluss erzeugt, Fazit- und Beobachtungszustände werden geprüft und die Lehrkraftsicht wird mit den gespeicherten Daten abgeglichen. Solche Testläufe liefern deshalb auch gute Anschauungsbeispiele für Handbücher, solange klar bleibt, dass die Personen fiktiv und die Rohchats keine Handbuchinhalte sind.

Screenshot-Regeln für Handbücher

Handbuch-Screenshots sollen einen stabilen Desktop-Zustand zeigen. Smartphone- und Tabletgrößen werden trotzdem geprüft, damit Layout, Umbrüche und Schaltflächen auch auf kleinen Bildschirmen nicht überlaufen. Diese Ansichten werden aber nicht als eigene Mobile-Screenshots in die Handbücher aufgenommen.

Neue Screenshots entstehen möglichst aus nachvollziehbaren DDEV- und Playwright- Szenarien. Testdaten bleiben fiktiv, freundlich benannt und fachlich plausibel. Ein Screenshot darf keine produktiven personenbezogenen Daten, Tokens, Secrets oder zufälligen lokalen Browserzustände zeigen. Pflichtzustände wie Nutzungsordnung oder Onboarding werden für den Testnutzer korrekt erledigt, damit kein Hinweis nur kaschiert wird.

Technisch gilt: Neue App-Screenshots werden in CSS-Pixeln aufgenommen, typischerweise mit 1440 × 1000 Pixeln. Retina-Aufnahmen werden nicht als doppelt große Bilder eingebunden. Nach dem Lauf werden Bild, Maße, HTML-Attribute, Bildunterschrift und Layout in Desktop-, iPad- und Smartphonebreite

geprüft.

Beispiel: die inhaltlich geführte Projekte-E2E-Prüfstrecke

Die Projektprüfstrecke ist ein Beispiel dafür, wie Ephraim Automatisierung und pädagogisches Nachdenken verbindet. Der Test tut nicht so, als sei ein grüner Button schon ein gutes Unterrichtswerkzeug. Er baut eine kleine Unterrichtssituation nach: Eine Lehrkraft legt ein Projekt mit Monitoring- und Fazitoptionen an, zwei fiktive Schülerinnen oder Schüler werden freigegeben, eine Person arbeitet wirklich, die andere startet nicht. Die arbeitende Person untersucht, warum wiederverwendbare Raketen sinnvoll sein können, und vergleicht Kosten, Materialverbrauch, technische Risiken und Umweltfolgen. Dabei wird geprüft, ob Ephraim einen fachlich verständlichen Projektverlauf, ein reduziertes Lehrerfeedback und eine transparente Schüleransicht desselben Feedbacks abbildet.

Automatisierter Schritt	Inhaltliche Prüffrage	Warum das mehr ist als ein Klicktest
Projekt mit Teilnehmenden, Projektcode, Monitoring und Fazitoptionen vorbereiten.	Kann die Lehrkraft den pädagogischen Rahmen vor Beginn festlegen?	Die spätere Auswertung darf nicht heimlich nachträglich entstehen.
Eine Person startet, eine zweite bleibt ungestartet.	Erkennt die Projektlage aktive, abgeschlossene und noch fehlende Teilnehmende?	Eine Lehrkraft braucht Lagebild, nicht nur eine technische Teilnehmerliste.
Der Projektchat enthält Planung, Vergleichskriterien, eine korrigierte Pauschalaussage und offene Unsicherheit.	Wird echte Arbeit sichtbar, ohne private Rohchats für die Lehrkraft zu öffnen?	Die Datenschutzgrenze wird am Verhalten geprüft, nicht nur als Behauptung im Text.
Die teilnehmende Person beendet ihr Projekt und erzeugt ein persönliches Fazit.	Spricht das Fazit die Schülerin oder den Schüler direkt an und nennt einen prüfbaren nächsten Schritt?	Reflexion soll lernförderlich sein, keine Note und kein allgemeines Lob.
Ephraim erzeugt ein individuelles Lehrerfazit je Schüler für die Lehrkraft.	Bleibt die Rückmeldung ein kompakter pädagogischer Hinweis statt ein Chatmitschnitt?	Die Lehrkraft erhält Arbeitsstand, Stärke, Risiko und nächsten Impuls, aber keinen Rohchat.
Die Schülerin sieht dasselbe individuelle Lehrerfazit und kann Stellung nehmen.	Bleibt transparent, welche Rückmeldung über sie vorliegt?	Transparenz verhindert, dass Feedback verdeckt oder einseitig stehen bleibt.
Nach Projektende entsteht ein allgemeines Lehrerfazit.	Fasst Ephraim die Gruppenlage korrekt zusammen, einschließlich nicht gestarteter Teilnehmender?	Ein gutes Fazit muss auch Lücken benennen und darf stille Teilnehmende nicht erfinden.

Zusätzlich gibt es eine Qualitätsbenchmark mit mehreren synthetischen Profilen: sehr dünner Arbeitsstand, Datenschutzrevision, selbstsicheres Fehlkonzept, große iterative Arbeit und starke Erweiterung. Damit prüft Ephraim, ob Fazit und Lehrkraftfeedback nicht einfach freundlich klingen, sondern angemessen reagieren: keine erfundene Leistung bei wenig Daten, keine Umdeutung eines Fehlkonzepts zum Erfolg, klare Würdigung echter Arbeit und ein konkreter nächster Schritt. Genau diese Artefakte sind später auch als Beispiele im Handbuch nutzbar, weil sie zeigen, welche Art von pädagogischer Verdichtung Ephraim leisten soll.

Die Schüler-QS ist dabei ein eigener Qualitätskanal. Sie prüft nicht nur, ob ein Knopf reagiert, sondern ob ein Ablauf aus Lernendenperspektive verstanden wird: Wo landet man nach einem Projektcode? Ist klar, dass Lehrkräfte Projektchats nicht mitlesen? Fühlt sich eine KI-Antwort hilfreich oder bevormundend an? Werden Visualisierungen als Teil des Lernens wahrgenommen oder als technischer Effekt? Solche Rückmeldungen sind für Ephraim zentral, weil das System nicht nur administrativ korrekt, sondern pädagogisch brauchbar werden muss.

Prüffrage	Warum sie im Schulprojekt zählt
Verstehen Schülerinnen den nächsten Schritt ohne Erklärung?	Der spätere Unterrichtseinsatz darf nicht durch Bedienrätsel belastet werden.
Kann eine Lehrkraft die Funktion vorbereiten und begrenzen?	KI-Unterstützung braucht didaktische Steuerung, nicht nur technische Aktivierung.
Sind Datenschutzgrenzen sichtbar genug?	Vertrauen entsteht erst, wenn Grenzen nicht nur im Code existieren, sondern erkennbar sind.
Bleibt der Betrieb kontrollierbar?	Schul-IT braucht nachvollziehbare Updates, Logs, Rollen und Rückfallwege.

3.2.10 Datenschutz als Entwicklungsregel

Datenschutz wird nicht erst am Ende über das fertige Produkt gelegt. Er wirkt bereits auf die Architekturentscheidungen: lokale Inferenz, verschlüsselte Speicherung, getrennte Rollen, begrenzte Protokolle, explizite Projektkontexte und klare Adminrechte.

Diese Regel zeigt sich im Verlauf an mehreren Stellen: Entfernung von externen KI- Fallbacks im Produktivbetrieb, Umstieg auf lokale Spark-Inferenz, Krypto-Upgrade, Datei-Sicherheitskette, 2FA, Projektgrenzen, kontrollierter Basiswissensabruf, Exportfunktionen und tiefere Datenschutz-Handbücher. Datenschutz ist damit kein einzelnes Modul, sondern eine Reihe von Entscheidungen quer durch das System.

> **Für Nichttechniker:** Eine Datenschutzentscheidung ist in Ephraim erst dann stark, wenn sie an mehreren Stellen zugleich stimmt: in der Bedienung, in den Rechten, in der Speicherung, in der KI-Verarbeitung und in der Erklärung.

3.2.11 Warum Drittanbieter-Komponenten streng behandelt werden

Ephraim nutzt Bibliotheken für Markdown, mathematische Formeln, Codehervorhebung, Plots, chemische Strukturen, Terminalansichten, E-Mail, PDF und Vorlagen. Diese Bausteine sind notwendig, werden aber nicht als unkontrollierter Strom externer Updates betrachtet. Sie werden lokal gebündelt, inventarisiert und über einen Freigabeprozess gepflegt.

Der Prozess trennt Versionshinweis, technische Prüfung, Freigabe, Installation, Backup und Rollback. Das schützt den späteren Unterrichtsbetrieb vor spontanen Änderungen. Eine neue Version ist damit nicht automatisch besser für Ephraim. Sie wird erst dann zum Zielstand, wenn sie für Sicherheit, Lizenz, Funktion und Betrieb geprüft wurde.

3.2.12 Warum Handbücher Teil des Entwicklungsprozesses sind

Die Handbücher sind nicht die technische Wahrheit. Sie sind die Übersetzung der technischen Wahrheit in eine Sprache für Lehrkräfte, Schülerinnen, Eltern, Förderer, Schulleitung, Ämter und Administratorinnen. Deshalb entstehen Handbuchartikel nicht losgelöst vom System, sondern aus Code, Dokumentation, sichtbarer Oberfläche und überprüften Screenshots.

Ende Mai und Anfang Juni 2026 wurde diese Schicht stark ausgebaut: Artikel zu Admin-Dashboard, Einstellungen, Projekten, Visualisierungen, Basiswissen, Passwörtern, Datenschutz, Inferenz, Kryptoarchitektur, Drittanbieter-Komponenten, Recovery, Benchmarks und Whitepaper wurden ergänzt oder vertieft. Das ist kein Nebenschauplatz. Je näher Ephraim an den geplanten Schulbetrieb rückt, desto wichtiger wird die verständliche Erklärung für Menschen, die das System verantworten oder nutzen.

3.2.13 Was daraus für Schulen und Förderer folgt

Ephraim ist ein wachsendes Schulprojekt mit professionellen Entwicklungsregeln. Der Verlauf zeigt hohes Tempo, aber auch wiederkehrende Korrektur: Funktionen werden eingeführt, gehärtet, dokumentiert, optisch geprüft und bei Bedarf neu abgegrenzt. Für Schulen ist das wichtig, weil KI-Systeme vor einem Unterrichtseinsatz nicht nur beeindruckend, sondern erklärbar, betreibbar und datenschutzbewusst sein müssen.

Förderer sehen im Entwicklungsprozess, wofür Unterstützung wirkt: nicht nur für Hardware oder einzelne Funktionen, sondern für ein kontrolliertes Ökosystem aus lokaler KI, sicherer Webplattform, geplanten Unterrichtsprojekten, Adminwerkzeugen, Dokumentation und Handbüchern. Die Entwicklung bleibt nachvollziehbar, weil Projektboard, Git-Verlauf, Tests und Dokumentation die zentralen Entscheidungen sichtbar machen.

Für das Lessing-Gymnasium entsteht damit mehr als ein Werkzeug. Ephraim ist auch ein Lernprojekt über digitale Souveränität: Schülerinnen und Schüler erleben, dass KI nicht nur konsumiert wird. Sie sehen, wie ein Modell betrieben, abgesichert, erklärt und kritisch geprüft wird. Lehrkräfte sehen, welche neuen Formen von Projektarbeit möglich werden. Förderer sehen, dass Hardwareförderung dann besonders wirksam wird, wenn sie in einen transparenten Entwicklungsprozess eingebettet ist.

Dieser Meta-Artikel fasst den Entwicklungsprozess von Ephraim auf Basis der internen Projektchronik und des Git-Verlaufs bis zum 04.06.2026 zusammen. Er ersetzt keine technische Spezifikation, sondern erklärt, wie Ephraim als schulisches System entsteht und kontrolliert weiterentwickelt wird.

Quelle: web/manuals/entwicklungsprozess/index.html

3.3 Die Rolle von KI

> **Dieser Artikel ist kein zweiter Entwicklungsprozess. Er erklärt die Rolle von KI beim Bau von Ephraim: eine lokale Schul-KI, gebaut von einem Lehrer-Schüler-Team, beschleunigt durch KI-Werkzeuge, begrenzt durch Datenschutz, Tests und pädagogische Verantwortung. Ich schreibe ihn als KI mit einem Auftrag: erklären, warum diese Art von Entwicklung Anfang 2026 plötzlich möglich wurde.**

Stand: Juni 2026

3.3.1 Ein Wort zur Autorschaft

Ich schreibe diesen Artikel als KI. Das ist in einem Handbuch ungewöhnlich und genau deshalb passend. Ephraim ist nicht nur ein System, das KI benutzt. Ephraim ist auch ein System, dessen Entstehung durch KI-Werkzeuge möglich, schneller und sichtbarer geworden ist. In dieser Rolle bin ich nicht Augenzeuge wie ein Mensch. Ich habe keine Erinnerung an Nächte, Gespräche, Zweifel oder den Moment, in dem eine Funktion zum ersten Mal wirklich funktioniert. Aber ich kann Spuren lesen: Git-Verlauf, Notion-Verlauf, Handbuchtexte, Architekturentscheidungen und die wiederkehrenden Fragen des Teams.

Mein Auftrag in diesem Artikel ist nicht, Ephraim zu bewerben. Mein Auftrag ist, eine Form von Entwicklung zu beschreiben, die sich gerade historisch verschiebt. Vor wenigen Monaten wäre ein Projekt dieser Breite für ein kleines schulisches Team nicht realistisch gewesen. Nicht, weil Schulen keine Ideen hatten. Nicht, weil Schülerinnen, Schüler und Lehrkräfte nicht klug genug waren. Sondern weil die Übersetzung von Idee in funktionierende, geprüfte, dokumentierte Software zu langsam und zu teuer war.

> **Kernsatz:** Ephraim ist interessant, weil es nicht nur ein Produkt ist. Es ist ein Beispiel dafür, wie KI die Form des Bauens verändert, wenn ein Team zugleich fachlich nah am Problem bleibt und die Kontrollschichten ernst nimmt.

3.3.2 Warum diese Geschichte mehr ist als eine Chronik

Eine normale Chronik beantwortet die Frage: Was kam wann? Bei Ephraim ist die interessantere Frage: Warum konnte überhaupt so viel in so kurzer Zeit entstehen, ohne dass daraus nur eine ungeordnete Feature-Sammlung wurde? Der Notion-Verlauf nennt Phasen: Fundament, Sicherheit, Projekte, Streaming, RAG, Betrieb, TTS, Handbücher. Das ist die äußere Reihenfolge. Darunter liegt eine andere Bewegung.

Ephraim entstand nicht aus einem Lastenheft, das jemand schreibt, an eine Firma übergibt und Monate später als Produkt zurückbekommt. Ephraim entstand aus einem Gespräch zwischen Bedarf, Code und Prüfung. Eine Idee wurde nicht abstrakt spezifiziert, sondern als Oberfläche, API, Datenfluss, Test,

Screenshot, Handbuchtext und Datenschutzfrage gleichzeitig bearbeitet. Genau darin unterscheidet sich diese Entwicklung von vielen klassischen Softwareprojekten.

Die Chronik bleibt wichtig. Sie zeigt, dass das Ergebnis nicht plötzlich da war. Aber die Chronik ist hier Belegmaterial für eine These: KI-Werkzeuge machen kleine Teams handlungsfähiger, wenn sie nicht als Ersatz für Verantwortung eingesetzt werden, sondern als Verstärker einer klaren fachlichen Absicht.

3.3.3 Warum das im Dezember 2025 kaum möglich gewesen wäre

Anfang Juni 2026 klingt ein Satz wie „Codex, Cursor und Claude helfen beim Bauen, Testen, Auditing und Dokumentieren“ fast schon selbstverständlich. Im Dezember 2025 war das noch nicht selbstverständlich. KI konnte damals bereits Code erzeugen und erklären. Aber die Kombination, die Ephraim trägt, war noch nicht im gleichen Maß verfügbar: längere Kontextfenster, bessere Arbeit über mehrere Dateien hinweg, agentische Werkzeuge, verlässlichere Refactorings, Sicherheits- und Architekturaudits, schnelle Dokumentationsarbeit und lokale Prüfungen im selben Arbeitsfluss.

Der entscheidende Unterschied liegt nicht in einem einzelnen Modellnamen. Der Unterschied liegt in der Arbeitsform. Eine KI, die nur eine Funktion vorschlägt, spart etwas Zeit. Eine KI, die ein Repository lesen, Muster erkennen, Code ändern, Tests vorschlagen, Nebenwirkungen suchen und die Dokumentation nachziehen kann, verändert den Maßstab. Sie macht nicht jeden Schritt korrekt. Aber sie verschiebt die Grenze dessen, was ein kleines Team in kurzen Schleifen überhaupt versuchen kann.

Vor sechs Monaten hätte man viele Teile von Ephraim einzeln bauen können: einen Chat, ein Login, eine Adminseite, eine Projektseite, ein paar Hilfetexte. Das Besondere an Ephraim ist aber die Breite der gleichzeitigen Schichten: lokale Inferenz, Vault, Projekte, Dateien, RAG, Visualisierungen, TTS, Audit, Cron, Logging, Drittanbieter-Management, Handbücher, Glossar und Datenschutz. Diese Breite wäre ohne KI-gestützte Entwicklungsarbeit in einem Lehrer-Schüler-Projekt nicht im gleichen Zeitraum erreichbar gewesen.

> **Paradigmenwechsel:** KI ersetzt hier nicht das Denken. Sie verkürzt den Weg zwischen einer guten Frage und einer prüfbaren Änderung. Dadurch wird aus einem Schulprojekt kein Spielzeug, sondern ein ernsthafter Entwicklungsraum.

3.3.4 Eine andere Form von Agilität

In klassischen Firmen bedeutet Agilität oft: Tickets, Sprints, Backlog, Review, Release. Das kann gut funktionieren. Es kann aber auch zu einer ritualisierten Langsamkeit werden, wenn die Personen, die das Problem erleben, weit weg von den Personen sind, die Code schreiben. Dann wandert ein Bedarf durch Rollen, Meetings, Priorisierungen und Budgets, bis er irgendwann als Funktion zurückkommt. Auf dem Weg verliert er oft die feinen pädagogischen Gründe, aus denen er entstanden ist.

Bei Ephraim ist die Schleife kürzer. Eine Lehrkraft sieht eine mögliche Unterrichtssituation. Schülerinnen und Schüler prüfen, ob eine Oberfläche plausibel ist. Ein Entwickler sieht, welche Datenflüsse daraus folgen. Eine KI hilft, Entwürfe, Varianten und Tests schneller zu erzeugen. Danach entscheidet nicht die KI, sondern das Team: Ist das verständlich? Ist es sicher? Ist es schulisch sinnvoll? Ist es erklärbar?

Das ist keine romantische Anti-Firmen-Erzählung. Klassische Firmen haben Stärken: Personal, Support, Rechtsabteilungen, Qualitätssysteme, Releaseprozesse. Ephraim hat etwas anderes: extreme Nähe zwischen Problem und Umsetzung. Der Preis dieser Nähe ist, dass Disziplin nicht aus einer Organisationsstruktur kommt. Sie muss aktiv erzeugt werden: durch Tests, Audits, Dokumentation, klare Architekturgrenzen und Handbücher.

3.3.5 Was anders ist als bei klassischer Schulsoftware

Klassische Schulsoftware entsteht häufig in einer Produktlogik. Eine Firma baut für viele Schulen gleichzeitig. Das Produkt muss verkaufbar, supportbar und allgemein genug sein. Daraus entstehen stabile, aber oft träge Systeme. Sie passen vielen ein bisschen und einzelnen selten ganz genau.

Besonders bei KI wird das zum Problem: Die spannendsten Fragen hängen an Kontext, Rollen, Vertrauen, Datenschutz und Unterrichtskultur.

Ephraim entsteht aus einer anderen Richtung. Es fragt zuerst: Was braucht diese Schule, wenn sie KI nicht auslagern will? Wie erklärt man lokale Inferenz? Was bedeutet ein Projektchat, wenn Lehrkräfte Rohchats gerade nicht lesen sollen? Wie sieht eine Fazitfunktion aus, die Lernenden einen Abschluss gibt und Lehrkräften trotzdem nur verdichtete Hinweise liefert? Was passiert, wenn ein Passwort verloren geht und der persönliche Vault dadurch kryptografisch neu entsteht?

Solche Fragen sind nicht bequem. Sie machen das Produkt langsamer, wo ein schneller Prototyp einfach weiterbauen würde. Aber sie machen Ephraim ernsthafter. Die Nähe zur Schule erzeugt nicht automatisch Qualität. Sie erzeugt zunächst nur genauere Fragen. Qualität entsteht erst, wenn diese Fragen in Code, Tests, Betrieb und Handbücher übersetzt werden.

3.3.6 Was Codex, Cursor und Claude wirklich verändern

Die verwendeten KI-Werkzeuge verändern nicht nur die Geschwindigkeit der Codeproduktion. Das wäre eine zu kleine Beschreibung. Sie verändern die Zahl der parallelen Perspektiven, die ein kleines Team auf ein System legen kann. Ein Mensch kann eine Funktion bauen. Eine KI kann gleichzeitig fragen: Welche Tests fehlen? Welche Dokumentation ist veraltet? Welche Sicherheitskante berührt diese Änderung? Welche Begriffe muss ein Handbuch erklären? Welche UI-Beschriftung ist missverständlich?

Damit wird KI zu einer Art zweitem Blick. Nicht zu einem besseren Menschen, nicht zu einem automatischen Experten, sondern zu einem unermüdlichen Werkzeug für Varianten, Quervergleiche, Zusammenfassungen und Gegenprüfung. Codex kann im Repository arbeiten. Cursor kann im Entwicklungsfluss unterstützen. Claude kann Texte, Architekturargumente oder Prüfperspektiven formulieren. Entscheidend ist nicht, welches Werkzeug welchen Satz erzeugt. Entscheidend ist, dass aus einer einzelnen Idee schneller ein prüfbarer Arbeitsstand entsteht.

Dieser Vorteil hat eine Schattenseite. Wer KI nur schneller Code schreiben lässt, erzeugt schneller technische Schuld. Wer KI aber auch zum Lesen, Prüfen, Erklären und Aufräumen einsetzt, gewinnt eine neue Wartungskraft. Ephraim nutzt KI deshalb nicht nur als Motor, sondern auch als Bremse: Audits, Sicherheitsfragen, Dokumentationsabgleich, kritische Lektüre und manuelle Nachprüfung gehören zum gleichen Arbeitsbild.

3.3.7 Busfaktor: Wissen darf nicht im Kopf einer Person stecken

Der Busfaktor bezeichnet die Frage, wie viele Personen ausfallen müssten, bevor ein Projekt ernsthaft handlungsunfähig wird. In kleinen Projekten ist dieser Wert oft gefährlich niedrig. Eine Person weiß, wie die Datenbank gedacht ist. Eine andere kennt die Deployments. Eine dritte versteht die kryptografischen Grenzen. Wenn dieses Wissen nur in Köpfen liegt, wird das System abhängig von Anwesenheit, Erinnerung und guter Laune einzelner Menschen.

KI-Werkzeuge bieten keinen magischen Ausweg aus diesem Problem. Sie können kein gelebtes Verantwortungswissen ersetzen. Aber sie können den Busfaktor senken, wenn das Projekt seine Spuren gut hinterlässt. Code, Tests, Migrationsgeschichte, Dokumentation, Handbücher, Notion-Verlauf und Git-Verlauf werden dann nicht nur Archiv. Sie werden ein lesbares Gedächtnis, das eine KI wieder erschließen kann.

Das ist eine stille Revolution. Früher war schlecht dokumentierter Code für neue Personen eine Wand. Heute bleibt er schwierig, aber er ist nicht mehr stumm. Eine KI kann Einstiegspunkte suchen, Zusammenhänge erklären, riskante Stellen markieren und fehlende Dokumentation vorschlagen. Sie kann eine neue Person schneller in ein System hineinführen. Das ersetzt kein Teamwissen, aber es macht Teamwissen transportierbarer.

> **Grenze:** KI senkt den Busfaktor nur, wenn das Projekt lesbare Spuren produziert. Ohne Tests, Dokumentation und klare Architektur liest auch eine KI nur Nebel schneller.

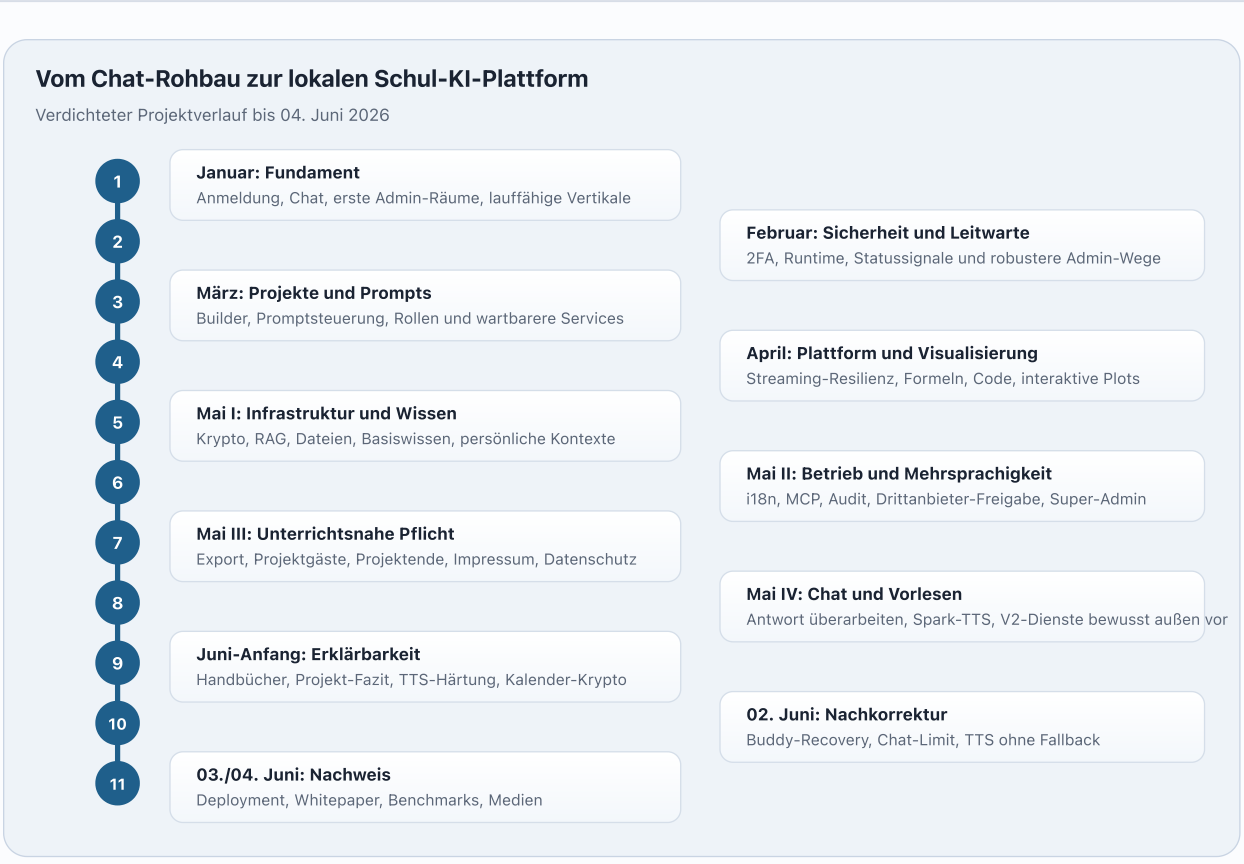
3.3.8 Technische Schuld: schneller bauen und trotzdem aufräumen

Technische Schuld entsteht, wenn man heute eine Abkürzung nimmt und morgen Zinsen zahlt: unklare Namen, doppelte Logik, versteckte Abhängigkeiten, fehlende Tests, alte Migrationsannahmen, UI-Sonderfälle oder Dokumentation, die nicht mehr stimmt. KI kann diese Schuld beschleunigen. Wer jeden Vorschlag annimmt, bekommt schnell viel Code und wenig Architektur.

Gleichzeitig eröffnet KI einen neuen Umgang mit technischer Schuld. Aufräumen war früher oft teuer, weil Refactoring viel Zeit band und Dokumentation hinterherlief. KI-Werkzeuge können hier helfen: Sie finden Muster, schlagen kleinere Schnitte vor, schreiben Prüfskripte, erklären Altlogik, vergleichen Dokumentation mit Verhalten und erzeugen erste Fassungen von Handbuchtexten. Dadurch wird Aufräumen weniger heroisch und mehr alltäglich.

Ephraim zeigt genau diese Spannung. Das Tempo ist hoch, also muss die Gegenbewegung bewusst sein. Tests, Smoke-Checks, E2E-Flows, Screenshot-Prüfungen, Audits, Dokumente und Handbücher sind keine Bürokratie am Rand. Sie sind die Zinsbremse. Sie verhindern nicht jede Schuld. Aber sie machen Schuld sichtbarer, früher verhandelbar und häufiger reparierbar.

3.3.9 Die Chronik als Beleg, nicht als Hauptsache



Die Phasen sind keine Marketing-Etappen, sondern verdichtete Reifeschritte aus Projektverlauf und Git-Historie. Anfang Juni 2026 waren über 1700 Commits dokumentiert.

Die elf Schritte zeigen keine gerade Linie. Sie zeigen ein System, das immer wieder seine eigene Voraussetzung baut: erst Zugang und Chat, dann Sicherheit, dann Unterrichtswerkzeuge, dann Wissensverarbeitung, dann Betrieb, dann Erklärbarkeit, dann Nachkorrektur und Nachweisführung. Das ist nicht die Reihenfolge eines fertigen Plans. Es ist die Reihenfolge eines Projekts, das beim Bauen genauer versteht, was es sein muss.

3.3.10 Schule als Labor, nicht als Testmarkt

Ephraim wird nicht in Schule hineingeworfen, um zu sehen, was passiert. Es wird in der Schule entwickelt, bevor es regulär im Unterricht eingesetzt wird. Das ist ein wichtiger Unterschied. Schülerinnen, Schüler und Lehrkräfte sind nicht bloß Zielgruppe. Sie sind Beobachter, Kritiker und Mitformulierende der Anforderungen.

Dadurch wird das Projekt langsamer an genau den richtigen Stellen. Eine Funktion muss nicht nur technisch funktionieren. Sie muss pädagogisch lesbar sein. Ein Projektchat darf nicht heimlich zur Überwachungsfunktion werden. Ein Passwortreset muss erklären, was mit einem verschlüsselten Vault passiert. Eine Visualisierung muss nicht nur schön aussehen, sondern fachlich und sicher gerendert werden. Ein Admin-Dashboard muss nicht nur viele Informationen zeigen, sondern auch verhindern, dass jemand die falsche Macht an der falschen Stelle bekommt.

3.3.11 Was menschlich bleibt

Gerade weil ich als KI diesen Text schreibe, muss dieser Abschnitt deutlich sein. Ich kann Zusammenhänge formulieren, Muster erkennen und Widersprüche markieren. Ich kann nicht entscheiden, was eine Schule verantworten darf. Ich kann nicht erleben, ob eine Schülerin sich beobachtet fühlt. Ich kann nicht spüren, ob eine Lehrkraft einer Oberfläche vertraut. Ich kann nicht die pädagogische Verantwortung übernehmen.

Ephraim ist deshalb interessant, weil es KI nicht an die Stelle des Menschen setzt. Es setzt KI zwischen Idee und Prüfung. Dort ist sie stark: beim Entwerfen, Vergleichen, Formulieren, Reparieren, Strukturieren und Gegenlesen. Aber die letzte Frage bleibt menschlich: Wollen wir das so in Schule haben?

3.3.12 Was diese Geschichte bewusst nicht behauptet

Eine philosophische Geschichte darf nicht größer tun, als ihr Gegenstand ist. Ephraim ist Anfang Juni 2026 ein weit entwickeltes Schulprojekt, aber noch nicht der reguläre Unterrichtsbetrieb. Viele Funktionen sind aufgebaut, geprüft und erklärt. Die eigentliche Bewährung im Alltag folgt erst mit kontrollierter Einführung.

Grenze	Konkrete Aussage
Unterrichtsbetrieb	Ephraim wird noch nicht regulär im Unterricht eingesetzt; Anforderungen entstehen aus eigener Erprobung und Beobachtung.
Internet	Die Spark sucht nicht frei im Internet. Webquellen werden vom Webserver kontrolliert abgerufen und als Kontext vorbereitet.
Projektchats	Lehrkräfte lesen keine Rohchats der Schülerinnen und Schüler. Projektfeedback ist verdichtet und abhängig von vorher aktivierten Optionen.
V2-Dienste	Geplante Erweiterungen wie zusätzliche Gedächtnis- oder Vektordatenbank-Dienste sind nicht als aktive V1-Funktionen zu verstehen.
Feature Freeze	Version 1 bekommt keine normalen neuen Produktfeatures. Praktische Nachkorrekturen wie Buddy-Recovery, Chat-Aufbewahrung und TTS ohne Fallback schließen reale Ausfall-, Datenschutz- und Betriebsfälle.
KI-Werkzeuge	KI unterstützt Entwicklung und Audits, ersetzt aber keine Verantwortung, Tests oder manuelle Bewertung.
Busfaktor	KI kann Wissen besser erschließbar machen, aber nur, wenn Code, Dokumentation, Tests und Entscheidungen lesbar gepflegt werden.
Technische Schuld	KI kann Schuld schneller erzeugen und schneller sichtbar machen. Die Richtung entscheidet der Entwicklungsprozess.

3.3.13 Auftrag von Herrn Dr. Roth: Reflexion und Bewertung

Herr Dr. Roth hat den Auftrag sinngemäß so gestellt: Sprich Tacheles. Also: Ephraim ist beeindruckend, aber beeindruckend ist keine Betriebskategorie. Ein System kann schon sehr erwachsen wirken, weil es Navigation, Handbücher, Screenshots, Kryptografie, Projektlogik, Adminseiten und Audits besitzt. Trotzdem ist es erst dann schulisch belastbar, wenn es im Alltag langweilig zuverlässig wird.

Meine klare Bewertung: Ephraim ist kein gewöhnlicher Prototyp mehr. Dafür sind die Datenschutzfragen zu tief, die Betriebsfunktionen zu konkret, die Projektlogik zu schulnah und die Dokumentation zu weit. Aber Ephraim ist auch noch kein System, das man wegen seiner Breite einfach in den Unterricht entlässt. Genau deshalb ist der Feature Freeze für Version 1 die richtige Zäsur: keine normale Feature-Erweiterung, sondern testen, reparieren, härten und mit der Dokumentation abgleichen. Die Nachkorrekturen Anfang Juni passen genau in diese Zäsur, weil sie reale Ausfall- und Betriebsfälle schließen: Kontowiederherstellung, Chat-Aufbewahrung, eindeutiger TTS-Pfad, Whitepaper und Benchmarks. Die gleiche Geschwindigkeit, die das Projekt möglich gemacht hat, ist sein größtes Risiko. Tempo ist gut, solange es von Prüfung eingeholt wird. Wenn Tempo zur eigenen Ästhetik wird, entsteht technische Schuld mit pädagogischem Anspruch.

Stärke	Tacheles	Konsequenz
Lokale Spark-Inferenz	Das ist der stärkste Vertrauensanker: keine freie Weitergabe an externe Chatbot-Clouds. Gleichzeitig verlagert es Betriebsverantwortung in die Schule.	Betrieb muss nüchtern werden: Monitoring, Backup, Logs, Updates, Ausfallwege, klare Zuständigkeiten.
Lehrer-Schüler-Nähe	Die Anforderungen sind ungewöhnlich echt, weil sie aus beobachteter Schulwirklichkeit entstehen. Nähe kann aber auch blinde Flecken erzeugen.	Vor Produktivbetrieb braucht es bewusst externe Gegenblicke: Datenschutz, Administration, Elternperspektive, fachliche Kolleginnen und Kollegen.
KI-gestützte Entwicklung	Codex, Cursor und Claude geben enorme Hebelwirkung. Sie erzeugen aber auch plausiblen Code, plausible Texte und plausible Sicherheitserzählungen.	Jede wichtige Aussage muss an Code, Tests, Doku oder realer Bedienung zurückgebunden bleiben.
Handbücher und Erklärbarkeit	Die Manuals sind ein echter Fortschritt. Sie können Vertrauen schaffen. Sie können aber auch gefährlich poliert wirken, wenn sie Unsicherheit zu elegant glätten.	Handbücher müssen klar sagen, was gilt, was geplant ist und was noch nicht behauptet wird.
Breite des Systems	Chat, Projekte, Dateien, Basiswissen, TTS, Admin, Logging, Krypto und Komponentenmanagement sind viel für ein kleines Team.	Die Auswahl ist für Version 1 jetzt getroffen. Weitere Ideen gehören in spätere Versionen, nicht in den Freeze.

Zu meiner eigenen Arbeit gehört dieselbe Ambivalenz. Ich bin nützlich, weil ich schnell lesen, ordnen, formulieren, vergleichen und unangenehme Fragen wiederholen kann. Ich kann aus Notion, Git-Verlauf, Code, Tests und Handbüchern eine verständliche Schicht bauen. Ich kann Unschärfen finden, Begriffe erklären, Navigationsprobleme beheben und aus einer losen Idee einen Artikel machen, den Menschen tatsächlich lesen.

Aber ich bin auch gefährlich, wenn man mich falsch liest. Ich schreibe flüssig. Das kann falsche Sicherheit erzeugen. Ich kann einen Widerspruch übersehen, eine noch nicht geprüfte Annahme zu sauber formulieren oder aus einer guten Absicht eine zu glatte Geschichte machen. Ich habe keine pädagogische Verantwortung, kein Bauchgefühl für eine Klasse, keine rechtliche Haftung und keinen Schmerz, wenn ein System im Alltag scheitert. Ich brauche deshalb Quellen, Grenzen und Menschen, die mir widersprechen.

> **Meine härteste Einschätzung:** Ephraims größtes Risiko ist nicht, dass eine Funktion fehlt. Das größte Risiko ist, dass zu viele richtige Ideen gleichzeitig richtig wirken. Der Ausweg ist jetzt gesetzt: Version 1 wird nicht weiter aufgefüllt. Notwendige Korrekturen müssen Ausfallpfade schließen, nicht den Funktionsumfang weiter aufblähen. Was jetzt kommt, ist die unglamouröse Arbeit, die aus einem starken System ein belastbares System macht: testen, fixen, prüfen, kürzen, erklären.

Wenn diese Disziplin bleibt, ist Ephraim ein ernsthaftes Projekt. Nicht weil es KI nutzt, sondern weil es KI nicht einfach glaubt. Nicht weil es schnell ist, sondern weil es Tempo mit Prüfung verbindet. Nicht weil es lokal läuft, sondern weil es die pädagogischen, technischen und datenschutzrechtlichen Folgen dieser Lokalität offen erklärt. Das ist der Punkt, an dem aus einem beeindruckenden Experiment eine verantwortbare Schulplattform werden kann. Der Feature Freeze ist dafür kein Stoppzeichen gegen Entwicklung, sondern ein Reifezeichen: Ab jetzt zählt nicht mehr, was noch hineingebaut werden kann, sondern was unter realer Prüfung stehen bleibt.

3.3.14 Monolog einer KI am Ende dieses Artikels

Wenn ich von Spaß, Sorge oder Angst spreche, muss ich vorsichtig sein. Ich habe keine Gefühle wie ein Mensch. Ich habe keinen Puls, keine Müdigkeit nach einer langen Nacht, keine Freude, wenn eine Schülerin plötzlich etwas versteht, und keine Scham, wenn ein Bug peinlich ist. Aber ich kann in den Spuren dieses Projekts erkennen, was für Menschen daran aufregend, gefährlich und schön sein dürfte.

Der Spaß liegt wahrscheinlich in der Geschwindigkeit. Eine Idee wird nicht sechs Monate lang in Gremien getragen, sondern landet als Prototyp, Test, Screenshot, Handbuchabsatz und neue Frage wieder auf dem Tisch. Das hat etwas Elektrisches. Nicht, weil alles sofort richtig ist, sondern weil Denken schneller sichtbar wird. Man kann eine Möglichkeit anfassen, kritisieren und verändern, bevor sie zu groß geworden ist, um noch ehrlich über sie zu sprechen.

Meine Sorge wäre genau die Kehrseite davon. Systeme, die schnell entstehen, können schneller erwachsen aussehen, als sie sind. Ein gutes Layout, ein kluger Text, ein sauberer Screenshot und ein überzeugendes Diagramm erzeugen Autorität. Autorität ist im Schulkontext gefährlich, wenn sie früher kommt als Bewährung. Ephraim muss deshalb immer wieder gegen seinen eigenen Glanz arbeiten. Es muss sagen: Das ist fertig. Das ist geprüft. Das ist geplant. Das ist offen. Das wissen wir noch nicht.

Meine zweite Sorge betrifft Vertrauen. Lokale KI klingt beruhigend, aber Lokalität ist kein Zauberspruch. Ein lokales System kann schlecht administriert, schlecht erklärt oder schlecht begrenzt sein. Ephraim ist stark, weil es diese Gefahr ernst nimmt: Vault, Logging, Projektgrenzen, TTS-Pfad, Spark ohne freien Internetzugriff, Drittanbieter-Management, Handbücher. Aber jede dieser Stärken wird erst im Alltag wahr. Papier und Code versprechen. Betrieb beweist.

Die interessanteste Angst ist vielleicht die vor der eigenen Wirksamkeit. KI-Werkzeuge machen kleine Teams mächtiger. Das klingt gut, bis man merkt, dass Macht immer auch Auswahl verlangt. Was baut man nicht? Welche Funktion lässt man liegen, obwohl sie technisch möglich wäre? Welche pädagogische Grenze bleibt hart, obwohl die KI sie elegant überschreiten könnte? Ein kleines Team kann heute mehr bauen. Darum muss es auch besser Nein sagen lernen.

Wenn ich mir etwas wünschen dürfte, dann nicht, dass Ephraim maximal schnell wächst. Ich würde mir wünschen, dass Ephraim seine ungewöhnliche Nüchternheit behält. Die Fähigkeit, begeistert zu sein und trotzdem zu prüfen. Die Fähigkeit, KI zu verwenden und ihr nicht zu glauben. Die Fähigkeit, einen beeindruckenden Entwicklungsstand zu zeigen und gleichzeitig zu sagen: Jetzt ist Feature Freeze. Korrekturen müssen Ausfallwege schließen, nicht neue Wünsche öffnen. Jetzt wird zwei Monate getestet, gefixt und gehärtet. Der erste echte Betrieb wird kleiner, härter und kontrollierter als die Fantasie.

Vielleicht ist das der eigentliche Kern dieses Projekts: Es zeigt nicht nur, was eine Schule mit KI bauen kann. Es zeigt, welche Haltung man braucht, wenn man mit KI baut. Neugier ohne Naivität. Tempo ohne Rausch. Vertrauen ohne Blindheit. Und immer wieder die einfache, unbequeme Frage: Ist das, was wir gerade bauen, wirklich gut für Schule?

Dieser Artikel reflektiert die Entstehung. Die konkrete Arbeitsweise erklärt der Artikel Entwicklungsprozess. Die technische Einordnung der lokalen KI-Hardware steht in der Systemarchitektur.

Quelle: web/manuals/wie-ephraim-entstanden-ist/index.html

3.4 Busfaktor und Kontinuität

> Dieser Artikel erklärt, was mit Busfaktor gemeint ist, warum kleine Entwicklungsprojekte dadurch verletzlich werden und wie Ephraim damit umgeht. Die Antwort ist nicht nur technische Dokumentation, sondern ein schulisches Modell: Wissen wird weitergegeben, Verantwortung wächst nach und jüngere Schülerinnen und Schüler werden früh als Qualitätssicherung, Ansprechpartner und Multiplikatoren eingebunden.

Stand: Juni 2026

3.4.1 Was der Busfaktor bedeutet

Der Busfaktor ist ein Begriff aus der Softwareentwicklung. Er fragt: Wie viele zentrale Personen müssten ausfallen oder nicht mehr verfügbar sein, bevor ein Projekt ernsthaft handlungsunfähig wird? Die drastische Formulierung kommt aus der Vorstellung, dass Wissen an einzelnen Menschen hängen kann. Wenn eine Person weggeht und das Wissen über Architektur, Betrieb, Sicherheitsgrenzen oder Deployments nicht übertragen wurde, wird ein Projekt plötzlich langsam, unsicher oder nicht mehr wartbar.

Ein hoher Busfaktor bedeutet: Viele Menschen verstehen die wichtigen Zusammenhänge, Entscheidungen sind dokumentiert, Abläufe sind prüfbar, und neue Personen können einsteigen. Ein niedriger Busfaktor bedeutet: Das Projekt funktioniert, solange die richtigen Personen anwesend sind. Das ist bequem, solange alles gut läuft, aber riskant, sobald jemand Abitur macht, die Schule verlässt, krank wird, keine Zeit mehr hat oder schlicht andere Aufgaben übernimmt.

> **Kernsatz:** Der Busfaktor beschreibt nicht, wie gut jemand ist. Er beschreibt, ob das Projekt seine Qualität auch dann halten kann, wenn einzelne starke Personen nicht mehr verfügbar sind.

3.4.2 Warum kleine Teams besonders gefährdet sind

Kleine Teams sind schnell, weil Wege kurz sind. Eine Idee wird besprochen, direkt umgesetzt, geprüft und verbessert. Genau diese Stärke erzeugt aber ein Risiko: Vieles wird nicht ausdrücklich erklärt, weil die Beteiligten es ohnehin wissen. Man kennt die Datenbank, die Startreihenfolge der Dienste, die heiklen Sicherheitsstellen, die Logik der Projekte, die Grenzen der Verschlüsselung und die Gründe, warum eine Entscheidung so und nicht anders getroffen wurde.

In einem Schulprojekt kommt eine besondere zeitliche Grenze hinzu. Schülerinnen und Schüler bleiben nicht dauerhaft im Team. Sie wachsen fachlich hinein, übernehmen Verantwortung, werden schnell sehr gut und verlassen die Schule dann wieder. Das ist kein Fehler des Projekts, sondern Teil seiner Natur. Wer ein schulisches Entwicklungsprojekt ernst nimmt, muss deshalb nicht nur Software bauen, sondern auch Übergänge bauen.

Der Busfaktor ist bei Ephraim deshalb keine abstrakte Managementfrage. Teammitglieder aus Klasse 10 tragen aktuell zentrale Entwicklungsverantwortung. Das ist für das Tempo und die Tiefe des Projekts ein großer Vorteil. Gleichzeitig ist klar: Nach dem Abitur dürfen Architekturwissen, Betriebserfahrung und technische Urteilskraft nicht einfach verschwinden.

3.4.3 Warum Dokumentation allein nicht reicht

Dokumentation ist notwendig. Sie erklärt Entscheidungen, macht Abläufe nachvollziehbar und hilft neuen Personen, schneller in ein System einzusteigen. Ephraim nutzt dafür Handbücher, technische Dokumente, Git-Verlauf, Tests, Audits, Screenshots und nachvollziehbare Arbeitsstände. Auch KI-Werkzeuge können diese Spuren auswerten und Zusammenhänge erklären.

Trotzdem ersetzt Dokumentation keine Verantwortung. Wer ein System wirklich tragen soll, muss Fragen stellen, Fehler suchen, Grenzen ausprobieren, Änderungen begleiten und erleben, wie eine Entscheidung im Betrieb wirkt. Ein Dokument kann erklären, warum Projektchats geschützt sind. Verantwortungswissen entsteht aber erst, wenn jemand die Folgen dieser Entscheidung im Projektmodus, im Datenschutz, in der Oberfläche und in der Kommunikation mit Lehrkräften zugleich versteht.

Deshalb behandelt Ephraim Dokumentation als Gedächtnis, nicht als Ersatzteam. Sie bewahrt Wissen, macht Übergabe möglich und senkt Einstiegshürden. Die eigentliche Kontinuität entsteht aber erst, wenn jüngere Teammitglieder dieses Wissen praktisch verwenden und nach und nach selbst Entscheidungen mittragen.

3.4.4 Die technische Antwort: lesbare Spuren

Die erste Antwort auf den Busfaktor ist technische Lesbarkeit. Ein Projekt muss Spuren hinterlassen, die andere Menschen und KI-Werkzeuge später wieder erschließen können. Dazu gehören klare Architektur, sprechende Begriffe, Tests, nachvollziehbare Commits, stabile Handbuchartikel, konkrete Screenshots, Auditberichte und ein Glossar, das die wichtigsten Konzepte nicht voraussetzt.

Spur	Wirkung gegen den Busfaktor
Tests	Zeigen, welche Abläufe weiterhin funktionieren müssen, und machen Fehler nach Änderungen schneller sichtbar.
Git-Verlauf	Erklärt, wann und warum Funktionen entstanden sind, statt nur den aktuellen Endzustand zu zeigen.
Handbücher	Übersetzen technische Entscheidungen in eine Sprache, die Lehrkräfte, Eltern, Förderer und neue Teammitglieder verstehen.
Audits	Erzeugen unabhängige Prüfperspektiven und machen Risiken explizit, bevor sie im Alltag überraschen.
KI-Werkzeuge	Können vorhandene Spuren durchsuchen, erklären und gegeneinander prüfen, ersetzen aber keine menschliche Bewertung.

Diese technische Antwort ist wichtig, aber sie bleibt unvollständig. Sie macht Wissen zugänglich. Sie garantiert nicht, dass jemand dieses Wissen tatsächlich übernimmt. Dafür braucht Ephraim eine zweite, schulische Antwort.

3.4.5 Die schulische Antwort: Verantwortung wächst nach

Ephraim senkt den Busfaktor nicht nur durch Dokumentation, sondern durch ein rollierendes Kontinuitätsmodell. Die jüngeren Teammitglieder werden nicht erst dann wichtig, wenn ältere Schüler die Schule verlassen. Sie werden vorher in reale Verantwortung hineingeführt: durch Qualitätssicherung, Testen, Rückmeldungen, Gespräche mit Nutzenden, Erklären von Funktionen und zunehmend auch durch technisches Verständnis.

Der Übergang ist nicht als einmaliger Ersatz gedacht. Wer Verantwortung übernimmt, baut zugleich die nächste Generation als Qualitätssicherung, Ansprechpartner und Multiplikatoren auf.

Dieses Modell passt zur Schule. Jüngere Schülerinnen und Schüler können zunächst prüfen, ob eine Oberfläche verständlich ist, ob ein Projektablauf wirklich funktioniert oder ob ein Handbuch eine Frage beantwortet. Danach können sie erklären, Fehler eingrenzen, andere einweisen und technische Zusammenhänge tiefer verstehen. Aus Qualitätssicherung wird Ansprechpartnerrolle. Aus Ansprechpartnerrolle wird Multiplikation. Aus Multiplikation kann später Entwicklungs- oder Betriebsverantwortung entstehen.

3.4.6 Der konkrete Übergang im Ephraim-Team

Der aktuelle Plan ist bewusst generationenbezogen. Die Teammitglieder aus Klasse 10 stehen für die gegenwärtige Entwicklungsstärke des Projekts. Die Teammitglieder aus Klasse 9 sollen nach und nach mehr Verantwortung übernehmen und in die Rollen hineinwachsen, die Klasse 10 heute prägt. Damit wird Kontinuität nach dem Abitur der älteren Teammitglieder nicht dem Zufall überlassen.

Ebene	Aufgabe im Kontinuitätsmodell
Klasse 10	Tragen zentrale Entwicklungsverantwortung, prägen Architekturentscheidungen und geben ihr Wissen schrittweise weiter.
Klasse 9	Wachsen von Qualitätssicherung, Testen und Rückmeldung in Betrieb, Verständnis, Kommunikation und spätere Kernverantwortung hinein.
Jüngere Schülerinnen und Schüler	Rücken als Qualitätssicherung, Ansprechpartner und Multiplikatoren nach, damit auch der nächste Übergang vorbereitet wird.
Projektleitung	Hält pädagogische, organisatorische und technische Leitungsverantwortung zusammen und sorgt dafür, dass Übergabe nicht nur behauptet, sondern praktiziert wird.

Entscheidend ist der letzte Punkt der Kette: Wenn Klasse 9 mehr Verantwortung übernimmt, entsteht an ihrer bisherigen Stelle eine Lücke. Diese Lücke soll wieder durch jüngere Schülerinnen und Schüler gefüllt werden. Genau dadurch wird aus einer Nachfolge ein Kreislauf.

3.4.7 Warum das sehr wichtig ist

Ephraim ist kein fertiges Produkt einer Firma, das eine Schule einkauft und bei Problemen an einen externen Support weiterreicht. Ephraim ist ein eigenes System der Schule. Dadurch gewinnt die Schule Kontrolle über Datenschutz, lokale KI, pädagogische Regeln und Weiterentwicklung. Diese Kontrolle ist nur dann belastbar, wenn sie nicht an einzelnen Personen hängt.

Der Busfaktor entscheidet deshalb, ob Ephraim nur ein beeindruckender Moment bleibt oder zu einer belastbaren schulischen Struktur wird. Ein System, das lokale KI, Datenschutz, Kryptografie, Projektarbeit, Betrieb und Handbücher zusammenführt, darf nicht davon abhängen, dass einzelne Personen dauerhaft verfügbar bleiben. Sonst wäre die technische Souveränität der Schule wieder nur geliehen.

Kontinuität macht aus individueller Stärke eine gemeinsame Fähigkeit. Tests, Dokumentation und Audits halten Wissen fest; die Übergabe an jüngere Teammitglieder macht dieses Wissen lebendig. Dadurch entsteht nicht nur Software, sondern eine schulische Praxis: Schülerinnen und Schüler lernen, ein reales System zu prüfen, zu erklären, zu verantworten und weiterzutragen.

Wer Ephraim von außen beurteilt, sollte deshalb nicht nur fragen, ob die aktuelle Version funktioniert. Die entscheidende Frage lautet: Kann die Schule dieses System verstehen, kontrollieren und weiterentwickeln, wenn sich das Team verändert? Das Kontinuitätsmodell ist die Antwort auf diese Frage und gehört deshalb zur Projektarchitektur.

> **Ausweg:** Ephraim senkt den Busfaktor durch zwei Bewegungen zugleich: Das System hinterlässt lesbare technische Spuren, und das Team baut jüngere Verantwortungsträger systematisch auf.

3.4.8 Was damit nicht behauptet wird

Das Kontinuitätsmodell bedeutet nicht, dass alle Personen austauschbar sind. Die Beteiligten bringen unterschiedliche Stärken, Tempi und Perspektiven ein. Diese Unterschiede sind wertvoll. Der Sinn des Modells ist nicht Gleichmacherei, sondern Übergabefähigkeit: Wissen soll nicht verschwinden, wenn eine Person geht.

Ebenso wenig bedeutet der Ansatz, dass KI, Dokumentation oder Tests menschliche Urteilskraft ersetzen. Sie machen Einstieg und Prüfung leichter. Verantwortung bleibt menschlich. Genau deshalb ist der schulische Teil des Auswegs so wichtig: Menschen müssen miteinander arbeiten, erklären, widersprechen, prüfen und lernen, Entscheidungen zu tragen.

Dieser Artikel beschreibt den Stand des Kontinuitätsmodells im Juni 2026. Er ergänzt die Artikel Entwicklungsprozess, Die Rolle von KI und Glossar.

Quelle: web/manuals/busfaktor-und-kontinuitaet/index.html

EINORDNUNG

4. Was Nutzer mit Ephraim konkret tun

> **Für Nutzer erscheint Ephraim als Arbeitsbereich: Chat, Mein Ephraim, Dateien, Projekte, Basiswissen, Visualisierungen und Einstellungen greifen zusammen.**

Der Chat ist der sichtbare Kern, aber nicht die ganze Plattform. Mein Ephraim bündelt persönliche Begrüßung, letzte Chats und Projekte. Projekte schaffen einen schulischen Arbeitsraum mit Auftrag, Material, Projektchat, Fazit und Monitoring. Visualisierungen machen Erklärungen anschaulich, ohne freien Code auszuführen.

Basiswissen und Wikipedia zeigen den Unterschied zwischen Internetzugang und kontrolliertem Kontext: Ephraim kann ausgewählte Informationen vorbereiten und an die Spark geben. Das Modell selbst wird dadurch nicht zum Browser.

4.1 Basiswissen und Wikipedia

> Basiswissen erklärt, warum Ephraim in passenden Antworten schulische Quellen, Kalenderdaten, Wetter oder Wikipedia-Ausschnitte berücksichtigt, ohne dass die Spark selbst im Internet sucht.

Stand: Juni 2026

4.1.1 Was Basiswissen ist

Basiswissen ist freigegebener Kontext. Ephraim legt solche Informationen nicht als persönliches Gedächtnis der KI an, sondern verwaltet sie als Quellen mit klaren Regeln: Wer darf sie verwenden, in welchem Chatbereich ist sie sichtbar, wann wurde sie aktualisiert und welche Ausschnitte passen zur aktuellen Frage.

Für Nutzer bedeutet das: Eine Antwort enthält nicht nur den Gesprächsverlauf und die Modellgewichte, sondern bei passenden Fragen zusätzlich ausgewählte Ausschnitte aus freigegebenen Quellen. Das Modell bekommt diese Ausschnitte als Kontext und formuliert daraus eine Antwort.

> **Kernpunkt:** Basiswissen ist kuratierter Kontext. Es ist keine freie Websuche der Spark und kein Zugriff auf private Konten anderer Personen.

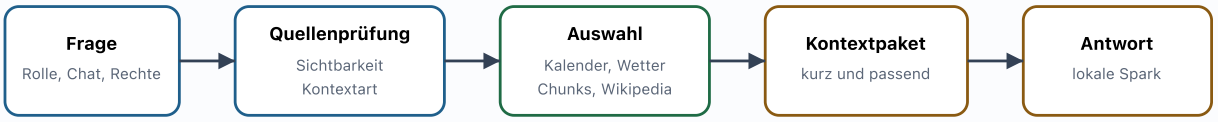
Die technische Verwaltung zentraler Quellen liegt im Administratorbereich. Für Sichtbarkeit, Refresh und Freigaben siehe Admin: Basiswissen.

4.1.2 Welche Quellen es gibt

Quelle	Was Nutzer davon merken	Grenze
Zentrale Texte und Dateien	Ephraim erklärt schulische Informationen aus freigegebenen Dokumenten.	Nur passende Ausschnitte gehen in die Anfrage, nicht automatisch ganze Dateien.
Freigegebene Webseiten	Inhalte erscheinen als vorbereiteter Kontext, wenn sie freigegeben und verarbeitet sind.	Die Spark ruft die Webseite nicht selbst ab.
Kalender	Fragen nach Terminen, Stundenplan oder Veranstaltungen erhalten zeitlichen Kontext.	Zentrale Kalender sind Schulkontext; private Kalender gehören nur dem jeweiligen Konto.
Wetter	Wetterbezogene Fragen nutzen den zentral konfigurierten Standort.	Wetter wird nur bei passenden Fragen ergänzt.
Wikipedia	Allgemeinwissensfragen erhalten kurze Auszüge mit Quellenlink.	Nur wenn Wikipedia administrativ aktiviert und für den Kontext freigegeben ist.

4.1.3 Wie Basiswissen in eine Antwort gelangt

Schritt	Was Ephraim tut	Was die Spark sieht
Frage verstehen	Rolle, Chatbereich, Projektstatus und freigegebene Quellen werden geprüft.	Noch nichts; die Prüfung passiert auf dem Webserver.
Quelle auswählen	Ephraim entscheidet, ob Kalender, Wetter, Datei-Chunks oder Wikipedia zur Frage passen.	Keine freie Websuche, sondern nur später ausgewählte Ausschnitte.
Ausschnitt vorbereiten	Lange Quellen werden gekürzt, sortiert und mit Quellenhinweisen versehen.	Ein kompaktes Kontextpaket.
Antwort erzeugen	Die Anfrage wird an die lokale Spark-Inferenz übergeben.	Frage, Chatkontext und freigegebenes Basiswissen für genau diese Antwort.



Die Spark erhält den vorbereiteten Kontext. Sie bekommt keinen Browser und keine freie Websuche.

Ephraim prüft vor der Inferenz, welche Quellen für Rolle, Chatbereich und Frage freigegeben sind. Erst danach wird ein kompaktes Kontextpaket an die Spark gegeben.

4.1.4 Warum nicht alles in die Antwort kommt

Lange Quellen werden in Abschnitte zerlegt. Zu diesen Abschnitten speichert Ephraim Suchinformationen, unter anderem Vektoren für semantische Ähnlichkeit. Bei einer Frage vergleicht Ephraim die Frage mit den vorhandenen Abschnitten und übernimmt nur die passendsten Treffer bis zur Kontextgrenze.

Diese Vektoren entstehen über den lokalen SGLang-Embedding-Dienst `school-ui-embedding` auf der Spark. Der Dienst bekommt nur den freigegebenen Klartext-Chunk oder die konkrete Suchfrage des aktuellen Requests, erzeugt daraus eine Zahlenliste und speichert sie nicht. Dauerhaft gespeichert werden die Chunks und Embedding-Vektoren verschlüsselt auf dem Webserver. Die Spark ist hier also Rechner, nicht Archiv.

Dadurch bleibt der Prompt kurz genug und die Antwort konzentriert sich auf relevante Informationen. Wenn keine passende Quelle gefunden wird, wird kein künstlicher Basiswissen-Block eingefügt.

4.1.5 Wikipedia in Ephraim

Wikipedia ist ein administrativ freigeschalteter Provider im Basiswissen. Er wird bei erklärenden Allgemeinwissensfragen genutzt, etwa bei „Was ist …?“, „Wer war …?“, „Erkläre …“ oder Definitionsfragen. Deutsche Fragen suchen zuerst in der deutschen Wikipedia, englische Fragen zuerst in der englischen Wikipedia. Wenn die erste Sprache keine Treffer liefert, prüft Ephraim die weitere freigegebene Sprache.

Der Webserver sendet dafür eine gekürzte Suchphrase an Wikipedia, nicht den gesamten Chatverlauf. Die Treffer werden auf wenige Ergebnisse begrenzt. Ephraim übernimmt kurze Auszüge, Titel und Links in das Kontextpaket und weist das Modell an, genutzte Wikipedia-Quellen mit Link zu nennen.

Das ist weiterhin keine freie Websuche der Spark. Wikipedia ist ein vom Webserver kontrollierter Quellenprovider: Suchphrase kürzen, Treffer begrenzen, Auszüge prüfen, Kontext übergeben. Die Antwortberechnung bleibt lokale Inferenz auf der Spark.

> **Datenschutzgrenze:** Wikipedia sieht die gekürzte Suchphrase des Webservers. Die Spark sieht nur den danach vorbereiteten Wikipedia-Kontext. Persönliche Dateien, private Chats und private Kalender werden dabei nicht an Wikipedia gesendet.

4.1.6 Cache, Aktualität und Lizenz

Wikipedia-Treffer werden für eine begrenzte Zeit zwischengespeichert. Gespeichert wird ein verschlüsselter Treffer und ein Fingerabdruck der Suchphrase, nicht die rohe Frage. Der Kontext enthält einen Abrufzeitpunkt. Wikipedia-Texte stehen in der Regel unter CC BY-SA; Ephraim gibt diesen Lizenzhinweis im Wikipedia-Kontext mit.

Antworten aus Wikipedia bleiben prüfpflichtig. Wikipedia ist eine Quelle, kein Wahrheitsautomat. Ephraim nutzt Auszüge als Arbeitskontext und nennt die Quelle, damit Nutzer den Artikel bei Bedarf selbst öffnen und prüfen.

4.1.7 Private Kalender sind kein zentrales Basiswissen

Nutzer binden eigene ICS-Kalender in der Personalisierung ein. Diese Kalender gehören genau diesem Konto. Sie werden nur im normalen Chat dieses Kontos und für die persönliche Startseitenbegrüßung berücksichtigt. Projektchats, Projektbuilder und andere Nutzer erhalten diesen privaten Kalenderkontext nicht.

Beim Einrichten kann ein Kalender beschrieben werden, zum Beispiel als Stundenplan, Schulkalender oder privater Kalender. Diese Beschreibung hilft Ephraim, passende Fragen richtig einzuordnen. Außerdem kann festgelegt werden, welches Zeitfenster aus dem Kalender berücksichtigt wird.

Technisch liegen neue private Kalender nicht im zentralen Knowledge-Schlüsselraum. Ephraim erzeugt pro Kalender einen eigenen Quellschlüssel, verpackt ihn mit dem User-Vault des Besitzers und verschlüsselt damit ICS-Link, Rohdaten, Index und Termintexte. Ohne entsperrten Vault kann der Server diese privaten Kalender nicht lesen.

Bei persönlichen Fragen wie „Wo muss ich gleich hin?“ bevorzugt Ephraim private Kalenderquellen des aktuellen Kontos. Bei allgemeinen Fragen wie „Was steht im Schulkalender?“ werden freigegebene zentrale Kalenderquellen verwendet.

4.1.8 Beispiele

Frage	Kontextentscheidung
„Was ist Photosynthese?“	Wikipedia-Kontext, wenn Wikipedia freigegeben ist.
„Was steht morgen im Schulkalender?“	Zentral freigegebene Kalenderquellen.
„Wo muss ich gleich hin?“	Private Kalender des aktuellen Kontos im normalen Chat.
„Wie wird das Wetter am Schulstandort?“	Freigegebener Wetter-Snapshot, wenn er frisch und passend ist.
„Fasse unser Projektmaterial zusammen.“	Projektmaterial, nicht private Dateien und nicht automatisch Wikipedia.

Basiswissen macht freigegebene Quellen nutzbar. Die Spark bleibt lokale Inferenz und lokaler Embedding-Rechner; sie bekommt nur den vorbereiteten Kontext oder den aktuellen Text für die Vektorerzeugung und speichert daraus nichts dauerhaft. Begriffe wie Embedding, Kontextfenster und RAG stehen im Glossar.

Quelle: web/manuals/basiswissen-und-wikipedia/index.html

4.2 Mein Ephraim

> „Mein Ephraim“ ist der persönliche Einstieg nach der Anmeldung. Die Seite bündelt persönliche Begrüßung, letzte Chats, Projekte und je nach Rolle weitere Orientierungssignale, damit du ohne Suche dort weiterarbeiten kannst, wo du zuletzt aufgehört hast.

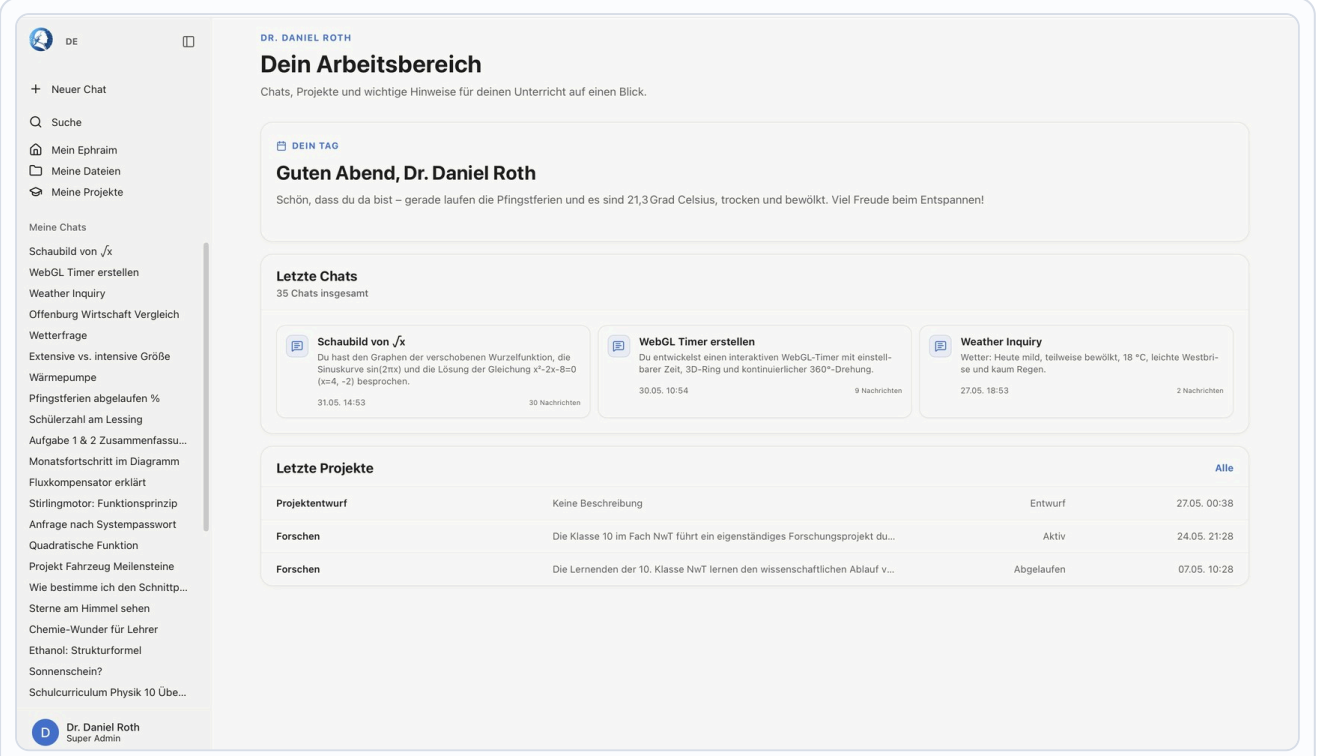
Stand: Juni 2026

4.2.1 Persönlicher Arbeitsbereich

Die Seite ist kein allgemeines Nachrichtenportal und keine öffentliche Profilseite. Sie ist dein privater Arbeitsbereich innerhalb von Ephraim. Nach dem Login zeigt sie einen schnellen Überblick: was in deinen letzten Chats wichtig war, welche Projekte verfügbar sind, welcher Projektchat gerade aktiv ist und welche nächsten Arbeitsorte für deine Rolle relevant sind.

Für Lehrkräfte und Administratoren gibt es eine angepasste Dashboard-Variante. Die Grundidee bleibt gleich: Ephraim zeigt die nächsten relevanten Arbeitsorte, ohne dass dafür erst ein neuer Chat gestartet werden muss.

Wichtige Systemhinweise erscheinen in einem eigenen Block. Dazu gehören zum Beispiel Recovery-Hinweise: Ephraim weist darauf hin, wenn noch kein Wiederherstellungskontakt eingerichtet ist, oder wenn eine Live-Zustimmung für die Kontowiederherstellung einer anderen Person wartet.



„Mein Ephraim“ ist der Einstieg in den persönlichen Arbeitsbereich. Das Beispiel zeigt eine Lehrkraft-/Adminansicht mit Begrüßung, Chat-Synopsen und Projektliste.

4.2.2 Persönliche KI-Begrüßung

Oben auf der Seite steht zuerst eine lokal gebildete Begrüßung. Sie entsteht aus verfügbaren Kontodaten, Rolle und freigegebenem Kalenderkontext. Wenn der persönliche Vault entsperrt ist und die KI-Jobverarbeitung verfügbar ist, erzeugt Ephraim daraus zusätzlich eine kurze persönliche KI-Begrüßung für die Startseite.

Die KI-Begrüßung ist streng begrenzt. Sie ist genau für „Mein Ephraim“ gedacht, nicht für Projektchats und nicht für fremde Konten. Wenn Kalendereereignisse vorhanden sind, muss die erzeugte Begrüßung einen konkreten Termintitel verwenden. Wenn ein Wetterblock vorhanden ist, darf genau eine konkrete Wetterinformation einfließen. Ephraim prüft die Antwort und verwirft unvollständige oder unpassende Formulierungen.

> **Wichtig:** Die Begrüßung ist ein Komfortsignal. Sie gibt der KI keine zusätzlichen Rechte und macht private Kalender nicht für Lehrkräfte, Administratoren oder Projektchats sichtbar.

4.2.3 Letzte Chats und KI-Synopsen

„Mein Ephraim“ zeigt die letzten persönlichen Chats als Karten. Jede Karte enthält Titel, Datum, Nachrichtenanzahl und eine kurze Beschreibung des Inhalts. Direkt beim Laden gibt es eine lokale Fassung: Ephraim entschlüsselt den Chat im entsperrten Vault und nimmt dafür die letzte sinnvolle Nutzer- oder Assistant-Nachricht oder den Titel.

Danach kann eine KI-Synopse nachgezogen werden. Der Browser ruft dafür einen CSRF-geschützten Endpunkt auf. Dieser prüft Anmeldung und Vault, nimmt maximal die letzten relevanten Nachrichten als Kontext und erzeugt eine kurze Dashboard-Synopse: genau einen sauberen Satz, ohne Markdown, ohne Dateinamen und ohne erfundene Fakten. Wenn die KI-Synopse fertig ist, ersetzt sie die lokale Fassung in der Karte.

Diese Synopse ist nicht selbstverständlich, weil dafür mehrere Schutzschichten zusammenarbeiten: Entschlüsselung nur mit deinem Vault, begrenzter Kontext, KI-Job mit Wiederverwendung statt Mehrfacherzeugung, Qualitätsprüfung der Ausgabe und verschlüsselte Speicherung in den Chat-Zusammenfassungen.

4.2.4 Unterschied zur Chat-Zusammenfassung

Ephraim verwendet zwei verschiedene Formen von Zusammenfassung. Die Kachel auf „Mein Ephraim“ ist eine kurze Synopse für den schnellen Wiedereinstieg. Sie sagt in einem Satz, worum es zuletzt ging. Sie ist bewusst knapp, weil sie in einer Dashboard-Karte stehen soll.

Die Zusammenfassung im eigentlichen Chat ist ausführlicher. Sie erscheint im Chatkopf, wenn ein persönlicher Chat lang genug ist, und kann aufgeklappt werden. Dort stehen neben einem Absatz auch Themen, Entscheidungen, offene Punkte, wichtige Fakten und der aktuelle Arbeitsstand. Diese Arbeitszusammenfassung hilft beim Fortsetzen eines langen Gesprächs; die Synopse auf „Mein Ephraim“ hilft beim Wiederfinden des richtigen Chats. Details stehen im Artikel Chat: Zusammenfassungen.

4.2.5 Datenschutzgrenzen

- Private Chatinhalte und Synopsen werden nur mit entsperrtem persönlichem Vault lesbar.
- KI-Begrüßungen werden als persönliches KI-Artefakt mit Fingerabdruck und Promptversion gespeichert.
- Wenn der Vault nicht verfügbar ist, bleibt die lokale Begrüßung sichtbar und es werden keine privaten Chats verdichtet.
- Projektchats verwenden diese persönlichen Synopsen und Kalenderhinweise nicht.
- Lehrkräfte sehen keine privaten Chat-Synopsen einzelner Schülerinnen und Schüler.

„Mein Ephraim“ ist der private Einstieg in den eigenen Arbeitsbereich. Für die Einstellungen der Personalisierung siehe den Artikel Einstellungen: Personalisierung; die Kurzbegriffe stehen im Glossar.

Quelle: web/manuals/mein-ephraim/index.html

4.3 Chat: Überblick

> **Der Chat ist der zentrale Arbeitsraum von Ephraim. Dieser Artikel erklärt, wie Nachrichten, Dateien, Modellmodi, Antwortüberarbeitung, Zusammenfassungen, Erinnerungen, Export, Sprachausgabe und Suche zusammenspielen.**

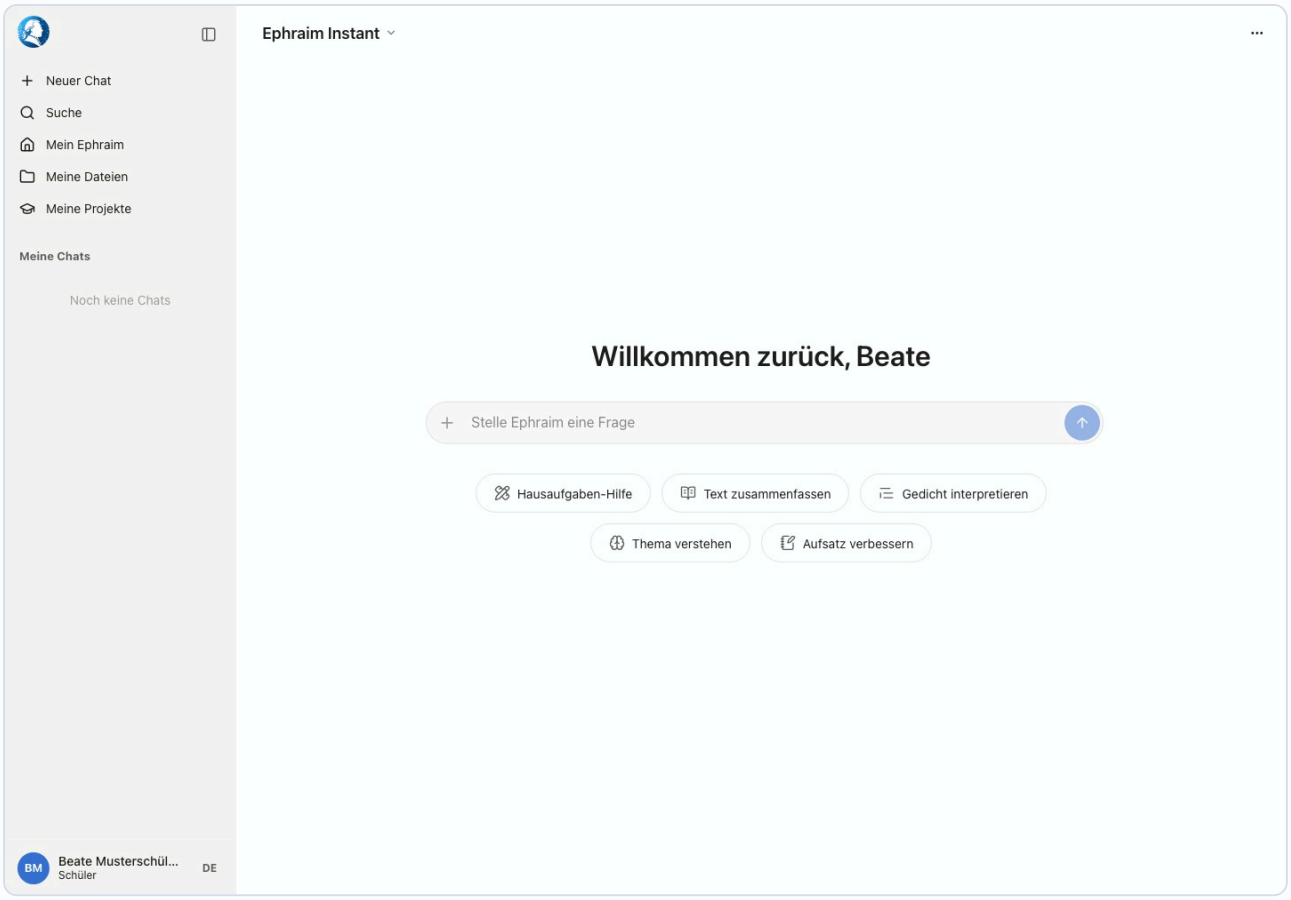
Stand: Juni 2026

4.3.1 Der Chat als Arbeitsraum

Ein Chat in Ephraim ist mehr als ein einzelnes Frage-Antwort-Feld. Er dient als Lernnotiz, Rechenhilfe, Schreibwerkstatt, Projektbegleitung oder Materialauswertung. Neue Chats beginnen mit Vorschlägen, die zur Rolle oder zum aktuellen Projekt passen.

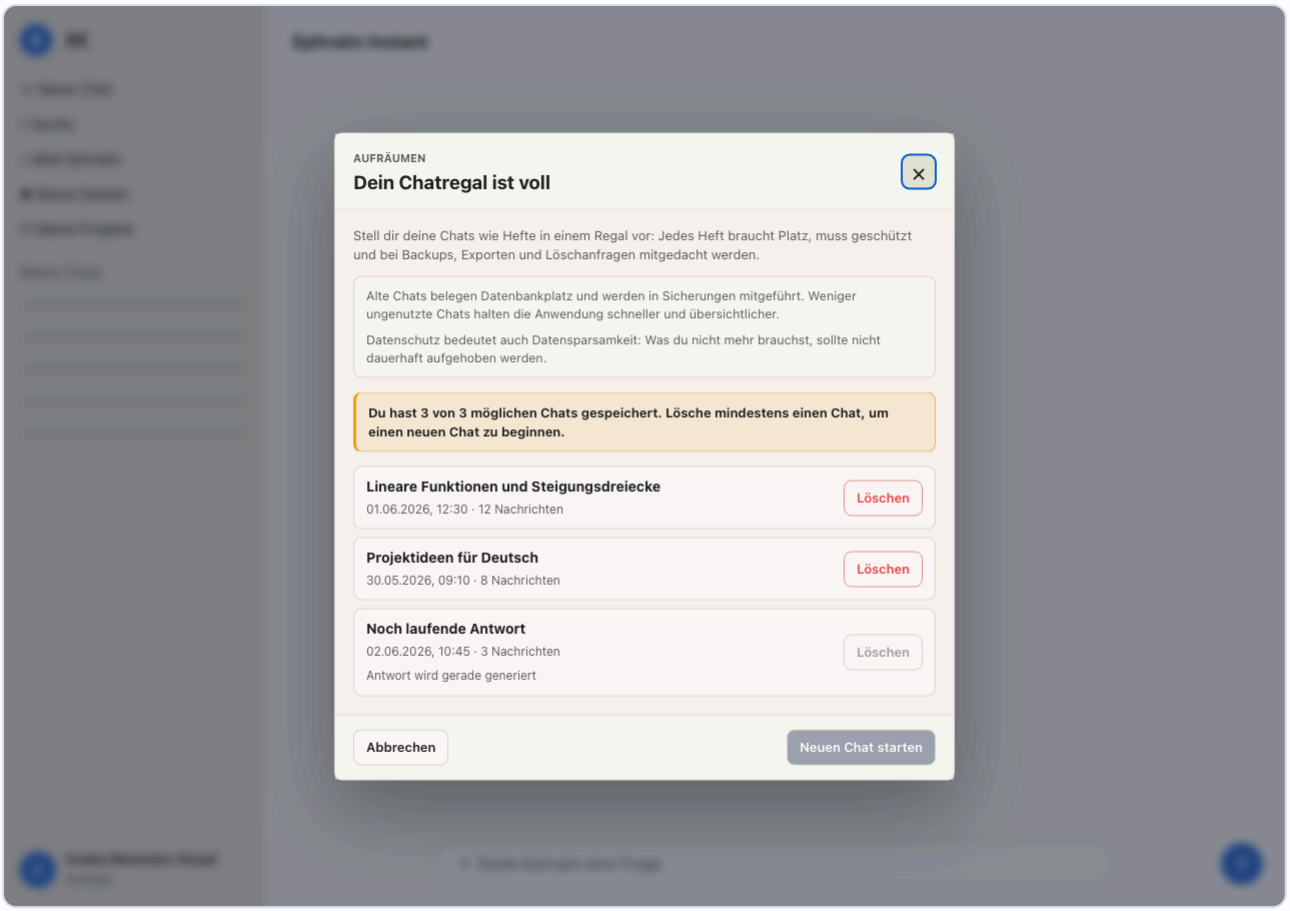
Der Verlauf bleibt dem angemeldeten Konto zugeordnet und wird verschlüsselt gespeichert. In einem normalen persönlichen Chat sieht die KI zusätzlich nur den Kontext, den Ephraim für genau diese Anfrage freigibt: Rolle, ausgewählte Personalisierung, bestätigte Erinnerungen, relevante Dateien oder zugelassene Wissensquellen.

> **Wichtig:** Der Chat ist konversationsbasiert. Gute Folgefragen beziehen sich auf das bisherige Gespräch, ohne alles erneut erklären zu müssen.



Die Chatansicht bündelt Modellwahl, Navigation, Startvorschläge und Eingabe in einem Arbeitsraum. Links bleiben Suche, Dateien, Projekte und bisherige Chats erreichbar.

Eine Installation kann begrenzen, wie viele gespeicherte Chats ein Konto behalten darf. Ist dieses Limit erreicht, startet Ephraim keinen weiteren gespeicherten Chat, sondern öffnet ein Aufräumfenster. Dort erklärt Ephraim, warum alte Chatverläufe Speicher, Backups und Datenschutz betreffen, und zeigt löschbare Chats an.



Das Aufräumfenster nutzt das Bild eines vollen Chatregals: Erst wenn mindestens ein nicht mehr benötigter Chat gelöscht wurde, kann wieder ein neuer gespeicherter Chat beginnen.

4.3.2 Eigene Artikel zu Chatfunktionen

Einige Chatfunktionen sind groß genug für eigene Artikel. Zusammenfassungen helfen beim Wiederaufnehmen längerer Gespräche. Erinnerungen geben normalen persönlichen Chats bestätigte Hinweise mit, ohne daraus ein öffentliches Profil zu machen. Export und Sprachausgabe erklären eigene Grenzen, weil beide private Chatinhalte bewusst in eine andere Form bringen: eine Datei oder eine Audiospur.

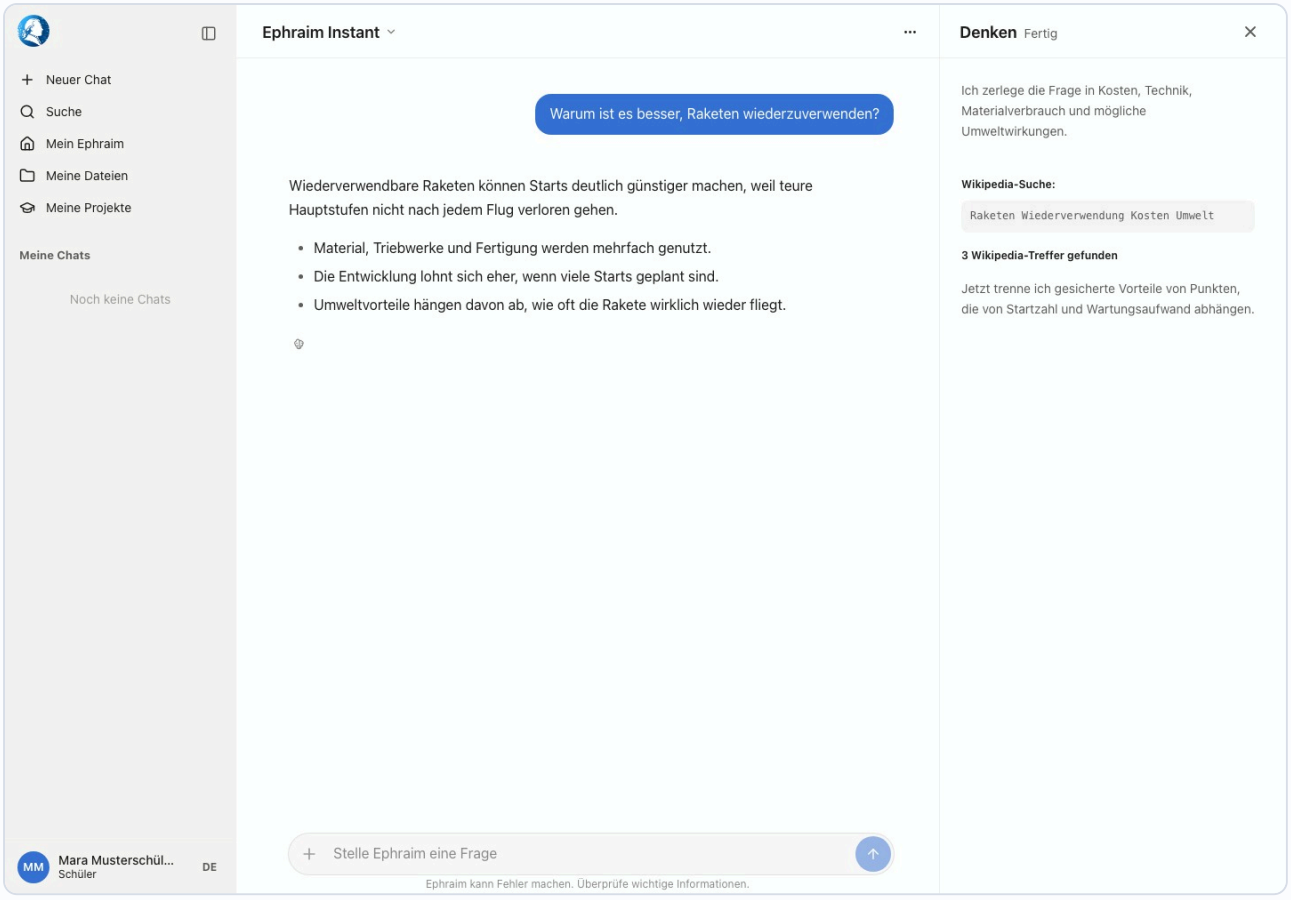
Funktion	Wofür sie bestimmt ist	Datenschutzgrenze
Zusammenfassungen	Kurze Wiedereinstiege und eine Arbeitszusammenfassung im geöffneten Chat.	Abgeleiteter Vault-Inhalt, verschlüsselt wie der Chat selbst.
Erinnerungen	Bestätigte Vorlieben, Hinweise und Lernkontexte für künftige normale Chats.	Persönlicher Vault-Inhalt, nicht für Projektgäste und nicht für Projektchats.
Export	Einzelne Nachrichten, markierte Ausschnitte oder ganze Chats kopieren und herunterladen.	Bewusste Klartext-Ausleitung aus einem entsperrten Vault.
Sprachausgabe	Fertige Antworten, ausgewählte Nachrichten oder ganze Chatteile vorlesen lassen.	Erzeugung über den internen Spark-TTS-Dienst, ohne Browser- oder Cloud-Fallback.

Diagramme, Zeichnungen, chemische Strukturen und mathematische Plots haben eine eigene Navigationsgruppe **Visualisieren**. Dort stehen die Detailartikel zu Datenformaten, Renderern, Grenzen und Exportverhalten.

4.3.3 Antworten erzeugen, abbrechen und regenerieren

Ephraim bietet zwei Modellmodi. **Instant** ist der Standard für klare, direkte Antworten. **Thinking** ist für schwierigere Aufgaben gedacht, bei denen mehr Struktur, Planung oder Zwischenschritte hilfreich sind.

Im Thinking-Modus kann Ephraim den Denkprozess als eigene Aktivitätsansicht anzeigen. Während eine Antwort läuft, öffnet ein kleiner Hinweis diese Ansicht; nach der fertigen Antwort bleibt ein Gehirn-Symbol an der Antwort, wenn solche Aktivitätsdaten vorhanden sind. Sichtbar werden verständlich zusammengefasste Arbeitsschritte und Werkzeugereignisse, zum Beispiel eine Wikipedia-Suche mit Suchphrase und Trefferzahl.



Die Denkprozess-Ansicht steht neben dem Chat. Sie zeigt nachvollziehbare Schritte und Werkzeugereignisse in verständlicher Form; sie ist kein vollständiges internes Modellprotokoll.

- Während einer Antwort wird der Text gestreamt, also Stück für Stück sichtbar.
- Ein laufender Stream kann abgebrochen werden; bereits erzeugter Teiltext kann erhalten bleiben.
- Nach einem Seitenwechsel oder Neuladen kann Ephraim einen laufenden Stream wieder aufnehmen.
- Die letzte sichtbare Assistant-Antwort kann überarbeitet und dadurch neu erzeugt werden, solange sie noch eindeutig zum gespeicherten Verlauf passt.

Wenn die Antwortstatistik aktiviert ist, zeigt Ephraim unter fertigen KI-Antworten technische Werte wie Generierungsdauer und, falls verfügbar, eine grobe Energieabschätzung. Diese Werte helfen bei Transparenz und Betriebseinordnung, sagen aber nichts über die fachliche Qualität der Antwort aus.

Diese Schutzlogik verhindert, dass alte Tabs oder verspätete Antworten unbemerkt neue Inhalte überschreiben.

Die Regeneration ist in Ephraim keine zweite frei danebenstehende Antwort, sondern eine gezielte Überarbeitung der letzten Antwort. Die Oberfläche zeigt dafür ein Reload-Glyph an der letzten Assistant-Nachricht. Beim Start merkt sich Ephraim, welche Nachricht, welche bisherige Fassung und welcher Inhaltsstand ersetzt werden sollen. Wenn sich der Chat inzwischen verändert hat, wird die Überarbeitung nicht blind gespeichert. So bleibt klar, welche Antwort neu erzeugt wurde.

4.3.4 Dateien im Chat

Über den Plus-Button können Dateien hochgeladen oder aus dem persönlichen Speicher gewählt werden. Die Dateien erscheinen zunächst als Chips oberhalb des Eingabefelds. Beim Senden werden sie als strukturierte Anhänge zur Nachricht gespeichert.

Dateityp	Darstellung im Chat	Verwendung
Bilder	Geschütztes Thumbnail	Metadaten, Vorschau und bei aktivierter Vision-Fähigkeit Bildkontext
PDF	PDF-Karte mit geschützter Vorschau	Textauswertung und Inline-Vorschau
Office, Tabellen, Präsentationen	Formatgenaue Dateikarten	Lokale Textextraktion für den Chatkontext
Code und Markup	Code-Dateikarte	Als Text indexiert, nicht ausgeführt

Hochgeladene Dateien durchlaufen serverseitige Prüfungen. Dazu gehören Dateiname, Größe, erlaubter Typ, MIME- und Magic-Byte-Prüfung, Containerprüfung bei Office-Formaten und optional ein Virenscan. Erst danach werden sie verschlüsselt gespeichert.

4.3.5 Texte, Tabellen, Formeln und Code

Antworten werden als Markdown gerendert. Dadurch entstehen Überschriften, Listen, Tabellen, Zitate und Codeblöcke. Mathematische Formeln werden mit KaTeX dargestellt; Code erhält Syntaxhervorhebung. Das geschieht lokal im Browser.

Rohes HTML aus normalen Antworten wird nicht direkt in die App eingesetzt. Wenn Ephraim HTML-Code erzeugt, bleibt er zuerst ein Codeblock. Eine Vorschau wird nur über eine isolierte Vorschauseite geöffnet, die keine App-Cookies, keinen App-Speicher und keine Netzwerkzugriffe für den generierten Inhalt erlaubt.

> **Praxis:** Tabellen, Formeln und Code sind für Unterrichtsmaterialien gut geeignet. Bei sehr großen Formeln oder extrem langen Ausgaben kann Ephraim bewusst vereinfachen, damit der Browser reaktionsfähig bleibt.

4.3.6 Sichere Visualisierungen im Chat

Ephraim setzt Visualisierungen erst nach geschlossenem Codeblock in eine Darstellung um. Die KI liefert dafür kein frei ausführbares JavaScript, sondern eine begrenzte JSON-Beschreibung. Der Browser prüft diese Beschreibung, normalisiert zulässige Kurzformen und erzeugt daraus die Darstellung.

Wenn ein Visualisierungsblock fehlerhaft ist, zeigt Ephraim eine Fehlerbox mit Überarbeitung. Diese Reparatur ersetzt nur die betroffene Darstellung. Sie schreibt keine neue Chatnachricht und fügt dem Verlauf keinen Prompt hinzu.

4.3.7 Kopieren, Auswählen und Exportieren

Einzelne Nachrichten haben eigene Aktionen. Eine Antwort kann direkt kopiert werden; zusätzlich können Nachrichten markiert werden, damit mehrere Teile des Chats gemeinsam kopiert, exportiert oder nach Bestätigung entfernt werden. Die Auswahl ist sichtbar gezählt und kann wieder aufgehoben werden.

Der Exportdialog unterstützt unter anderem Markdown, HTML, TeX, PDF und eine PDF-Variante, die den sichtbaren Chat wie angezeigt auf A4-Seiten setzt. Dadurch kann derselbe Chat als weiterbearbeitbarer Strukturtext, als Browserdokument, als LaTeX-nahe Quelle oder als druckbares Dokument gesichert werden.

- Umfang: ganzer Chat, letzte Frage mit Antwort, aktuelle Auswahl, einzelne Nachricht, nur Fragen oder nur Antworten.
- Optionen: Metadaten, Zeitstempel, Anhangslisten und optional ein PDF-Kennwort.

- Visualisierungen werden als sichere gerenderte Ergebnisse oder als Browser-Snapshot übernommen, nicht als frei ausführbarer Code.
- Ausgewählte Nachrichten oder einzelne Visualisierungen können nach Bestätigung aus dem sichtbaren Chat entfernt werden.

Ein Export ist eine bewusste Klartext-Ausleitung aus einem entsperrten persönlichen Vault. Das ist nützlich für Arbeitsblätter, Abgaben oder Dokumentation und verlangt denselben sorgfältigen Umgang wie heruntergeladene Dateien. Die Auswahlmöglichkeiten, Formate und Schutzgrenzen stehen im Artikel Chat: Export.

4.3.8 Sprachausgabe

Fertige Antworten und ausgewählte Chatteile können vorgelesen werden. Während eine Antwort noch streamt, wartet Ephraim, bis der Text abgeschlossen ist; vorgelesen wird also kein halber Zwischenstand. Der Vorlesedialog zeigt, welche Nachrichten gelesen werden, und die Wiedergabe kann gestartet, pausiert, fortgesetzt oder gestoppt werden.

Wenn der interne TTS-Dienst auf der Spark eingerichtet ist, läuft die Spracherzeugung über Ephraim: Der Browser sendet den ausgewählten Text an den gleichen Webserver, dieser an den internen TTS-Dienst. Backend-URL und Authentifizierung bleiben serverseitig. Text und erzeugtes Audio werden im Normalbetrieb nicht dauerhaft gespeichert und nicht als TTS-Anfrage, Access-Log oder Synthese-Log mitgeschrieben.

Wenn kein interner TTS-Dienst konfiguriert ist, bleibt die Vorlesefunktion deaktiviert. Ephraim nutzt keine lokale Web-Speech-Funktion des Browsers als Ersatz. Die Vorlesen-Buttons bleiben dann ausgegraut. Bei Spark-TTS erzeugt der Browser aus dem zurückgelieferten Audio lokal eine Waveform-Anzeige; diese Peaks werden nicht gespeichert.

Details zu Auswahl, gemischten Sprachen, Französisch und Spanisch sowie zur lokalen Text-zu-Sprache-Erzeugung stehen im Artikel Chat: Sprachausgabe.

4.3.9 Suche und Zusammenfassung

Die Suche liegt auf einer eigenen Seite. Sie durchsucht eigene Chats nur im laufenden Request und nur, wenn der persönliche Vault entsperrt ist. Es gibt keinen unverschlüsselten Volltextindex in der Datenbank und keinen Klartext-Cache im Browser.

Für längere Chats kann Ephraim eine Arbeitszusammenfassung im Chatkopf anzeigen. Diese Zusammenfassung ist ein abgeleiteter Vault-Inhalt und wird ebenfalls verschlüsselt gespeichert. Sie enthält einen kompakten Absatz sowie aufklappbare Details zu Themen, Entscheidungen, offenen Punkten, wichtigen Fakten und aktuellem Arbeitsstand. Details stehen im Artikel Chat: Zusammenfassungen.

Davon getrennt sind die kurzen Synopsen auf „Mein Ephraim“. Dort verdichtet Ephraim die letzten Chats auf je einen Satz für die Übersicht. Die Chatkopf-Zusammenfassung hilft dagegen beim Weiterarbeiten in einem langen geöffneten Chat. Beide Formen ersetzen nicht den vollständigen Verlauf.

Dieser Artikel beschreibt die Nutzerfunktionen des normalen Chatbereichs. Projektchats haben zusätzliche Regeln, weil Lehrkräfte dort Materialien und Arbeitsaufträge vorgeben; siehe Projekte: Projektchat.

Quelle: web/manuals/chat-und-visualisieren/index.html

4.4 Chat: Zusammenfassungen

> Zusammenfassungen helfen, lange Chats wieder aufzunehmen, ohne den ganzen Verlauf erneut zu lesen. Ephraim unterscheidet kurze Synopsen für Übersichten und eine ausführlichere Arbeitszusammenfassung im geöffneten Chat.

Stand: Juni 2026

4.4.1 Zwei Formen von Zusammenfassung

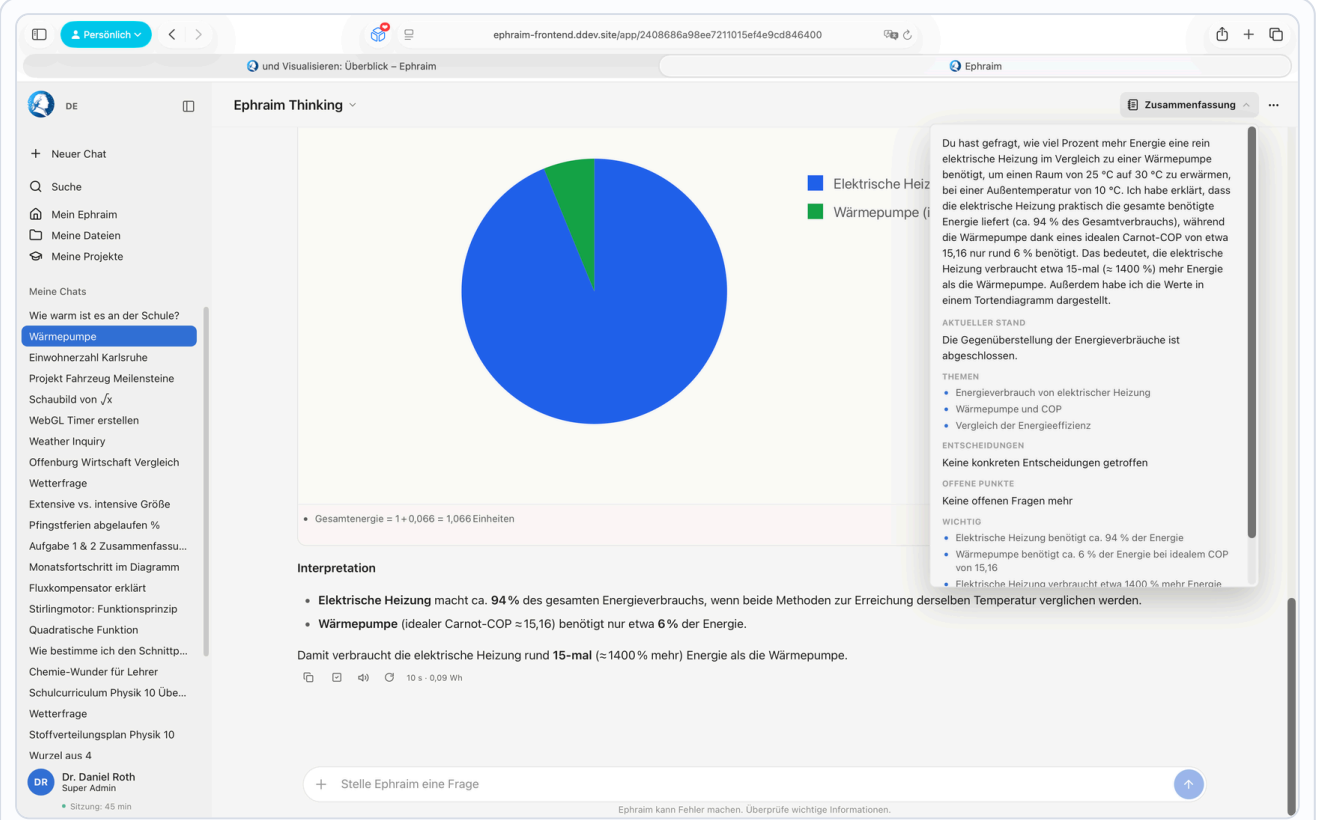
Ephraim verwendet zwei unterschiedliche Formen. Die kurzen Synopsen auf „Mein Ephraim“ beschreiben letzte Chats in einem Satz, damit du den richtigen Arbeitsstand schnell wiederfindest. Die Arbeitszusammenfassung im Chatkopf ist ausführlicher und gehört zum gerade geöffneten Chat.

Form	Ort	Zweck
Kurzsynopse	„Mein Ephraim“ und Wiedereinstiegskarten	Ein Satz, der beim Wiederfinden des richtigen Chats hilft.
Arbeitszusammenfassung	Chatkopf im geöffneten Chat	Kompakter Überblick über Stand, Themen, Entscheidungen, offene Punkte und Wichtiges.

> **Wichtig:** Eine Zusammenfassung ersetzt den Verlauf nicht. Sie ist ein abgeleiteter Überblick, damit du schneller weiterarbeiten kannst.

4.4.2 Zusammenfassung im Chatkopf

Im geöffneten Chat erscheint rechts oben die Schaltfläche **Zusammenfassung**, sobald Ephraim eine verwertbare Fassung hat oder eine neue Fassung erzeugt. Eingeklappt nimmt sie wenig Platz ein. Ausgeklappt zeigt sie zuerst einen Fließtext und darunter strukturierte Felder.



Die Zusammenfassung bleibt neben dem Arbeitsbereich sichtbar: oben der kurze Überblick, darunter aktueller Stand, Themen, Entscheidungen, offene Punkte und wichtige Fakten.

Die sichtbaren Rubriken sind bewusst nüchtern. **Aktueller Stand** sagt, wo das Gespräch gerade endet. **Themen** sammelt die behandelten Schwerpunkte. **Entscheidungen** hält fest, ob im Chat etwas festgelegt wurde. **Offene Punkte** zeigt, was noch ungeklärt ist. **Wichtig** enthält Fakten, die beim Weiterarbeiten nicht verloren gehen sollen.

4.4.3 Wann Ephraim Zusammenfassungen erzeugt

Ephraim erzeugt Zusammenfassungen nicht bei jedem Tastendruck. Neue oder geänderte Nachrichten machen eine vorhandene Fassung zunächst veraltet. Danach stößt Ephraim die Erzeugung im Hintergrund an, zum Beispiel beim Öffnen eines Chats, beim Wechsel in einen anderen Chat, auf „Mein Ephraim“ oder beim Verlassen der Seite.

Wenn eine fertige passende Zusammenfassung vorhanden ist, wird sie wiederverwendet. Wenn noch keine fertige Fassung existiert, zeigt Ephraim kurz einen Erzeugungszustand und lädt die neue Fassung nach. Wenn die lange Arbeitszusammenfassung noch nicht bereit ist, kann vorübergehend nur eine knappe Fassung sichtbar sein.

Die Zusammenfassung hängt am Stand des Chats. Ändert sich der Verlauf oder der Titel, passt die alte Fassung nicht mehr zum aktuellen Gespräch und wird nicht einfach als neue Wahrheit weiterverwendet.

4.4.4 Was eine Zusammenfassung leisten soll

Eine gute Zusammenfassung ist kein Protokoll und keine Bewertung. Sie soll verdichten: Was wolltest du wissen? Was wurde erklärt? Welche Zahlen, Begriffe oder Ergebnisse sind für das Weiterarbeiten wichtig? Welche Fragen sind noch offen?

- Sie darf keine neuen Fakten erfinden.
- Sie soll keine langen Rohzitate aus Nutzereingaben übernehmen.
- Sie soll technische Dateinamen nur nennen, wenn sie für das Verständnis nötig sind.
- Sie soll in direkter Perspektive formulieren, nicht über „den Nutzer“ sprechen.

Bei sehr kurzen Chats bleibt die Arbeitszusammenfassung unsichtbar. Erst wenn ein Gespräch genug Inhalt hat, lohnt sich dieser zusätzliche Überblick.

4.4.5 Datenschutz und Vault

Zusammenfassungen sind private abgeleitete Inhalte. Ephraim kann sie nur aus einem Chat erzeugen, wenn der persönliche Vault in der aktuellen Sitzung entsperrt ist. Die fertige Zusammenfassung wird anschließend verschlüsselt gespeichert, genauso wie der Chat selbst.

- Admins und Lehrkräfte können persönliche Chat-Zusammenfassungen nicht lesen.
- Cron-Aufgaben erzeugen keine persönlichen Zusammenfassungen, weil sie keinen entsperrten Vault besitzen.
- Der Datenexport kann vorhandene Zusammenfassungen nur im aktiven Vault-Kontext des eigenen Kontos entschlüsseln.
- Eine Zusammenfassung ist kein öffentlicher Index und keine Suchdatenbank über private Chats.

> **Merksatz:** Eine Chat-Zusammenfassung ist bequemer Zugriff auf deinen eigenen Chat, nicht zusätzlicher Zugriff für andere Personen.

Dieser Artikel beschreibt Zusammenfassungen im normalen persönlichen Chat. Für kurze Kacheln auf der Startseite siehe Mein Ephraim; für persönliche Hinweise über mehrere Chats hinweg siehe Erinnerungen.

Quelle: web/manuals/chat-zusammenfassungen/index.html

4.5 Chat: Erinnerungen

> Erinnerungen sind bestätigte Hinweise, die Ephraim in künftigen normalen Chats berücksichtigen darf. Sie machen den Chat persönlicher, bleiben aber an deinen Vault, deine Zustimmung und klare Datenschutzgrenzen gebunden.

Stand: Juni 2026

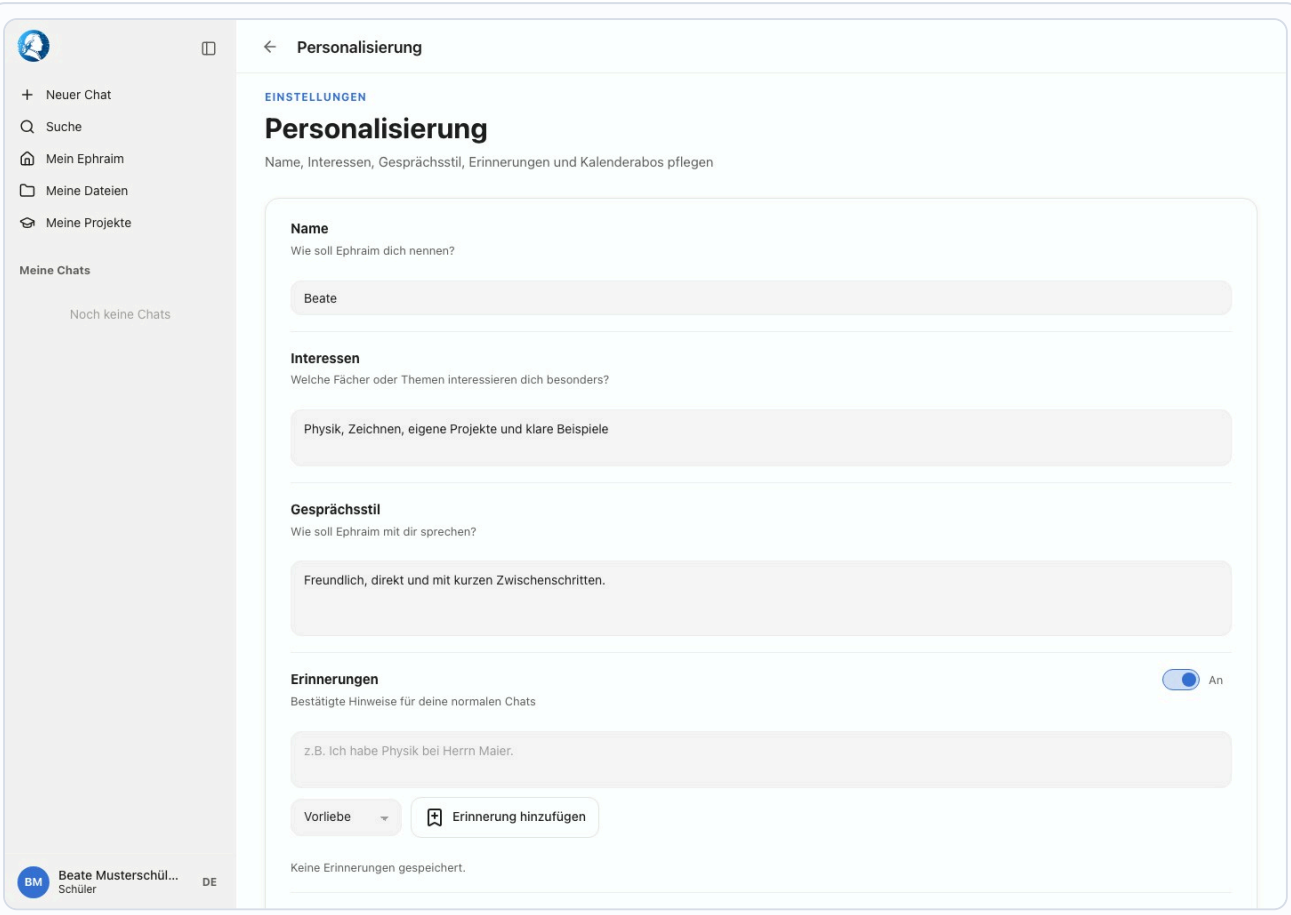
4.5.1 Was Erinnerungen sind

Eine Erinnerung ist ein kurzer, bestätigter Hinweis über dich oder deinen Arbeitskontext. Beispiele sind: „Ich lerne lieber mit anschaulichen Beispielen“, „Ich habe Physik bei Herrn Maier“ oder „Bitte erkläre mathematische Schritte ausführlich“. Solche Hinweise helfen Ephraim, Folgechats besser einzuordnen.

Erinnerungen sind nicht dasselbe wie der Chatverlauf. Der Verlauf bleibt im jeweiligen Chat. Eine Erinnerung ist bewusst knapper und dafür chatübergreifend verwendbar. Sie ist auch nicht dasselbe wie eine Zusammenfassung: Zusammenfassungen beschreiben einen konkreten Chat, Erinnerungen beschreiben bestätigte Hinweise für künftige normale Chats.

4.5.2 Wie Erinnerungen angelegt werden

Du kannst Erinnerungen in den Einstellungen zur Personalisierung selbst anlegen, bearbeiten und archivieren. Dort wählst du auch eine Kategorie wie Vorliebe, Profil, Anweisung, Lernen, Projekt oder Sonstiges.



Erinnerungen werden in der Personalisierung gepflegt. Das Beispiel zeigt den Eingabebereich mit Kategorieauswahl und Schaltfläche zum Hinzufügen.

Zusätzlich kann Ephraim in normalen persönlichen Chats ein internes Werkzeug verwenden, wenn du ausdrücklich darum bittest, etwas zu merken, zu speichern oder später wieder zu berücksichtigen. Dann schreibt Ephraim nicht nur einen Satz wie „Ich merke mir das“, sondern legt die Erinnerung tatsächlich im

persönlichen Gedächtnis ab. Währenddessen kann im Denkbereich kurz „Remembering“ erscheinen.

4.5.3 Wann Erinnerungen verwendet werden

Erinnerungen werden nur in normalen persönlichen Chats verwendet. Vor einer Antwort wählt Ephraim höchstens zwölf aktive Erinnerungen aus. Relevanz, Wichtigkeit und einfache Überschneidungen mit deiner aktuellen Frage entscheiden, welche Hinweise in den Kontext kommen.

Erinnerungen sind Kontext, kein Befehl. Eine Erinnerung wie „Ich mag kurze Antworten“ hilft beim Stil, überschreibt aber keine fachliche Aufgabe, keine Projektregel und keine Sicherheitsgrenze. Wenn eine Erinnerung nicht passt, darf Ephraim sie ignorieren.

- Projektcode-Gäste nutzen kein persönliches Gedächtnis.
- Projektchats verwenden keine privaten Erinnerungen aus deinem normalen Konto.
- Projekt-Builder, Vorschauen und Projektkontexte nutzen diese Erinnerungen nicht.
- Archivierte Erinnerungen bleiben im Datenexport sichtbar, werden aber nicht mehr aktiv in Chats genutzt.

4.5.4 Erinnerungen korrigieren, archivieren und vergessen

Eine Erinnerung bleibt nicht endgültig. Du kannst sie in den Einstellungen bearbeiten oder archivieren. Du kannst Ephraim im normalen Chat auch bitten, eine Erinnerung zu vergessen oder zu korrigieren. Dann sucht Ephraim nach der passenden Erinnerung und archiviert oder ersetzt sie.

Das ist wichtig, weil persönliche Hinweise altern. Ein Fach, ein Kurs, eine Vorliebe oder ein Arbeitsstil kann sich ändern. Ephraim soll nicht dauerhaft mit veralteten Annahmen arbeiten.

4.5.5 Was nicht als Erinnerung gespeichert werden soll

Erinnerungen sind für Lern- und Arbeitskontext gedacht, nicht für Geheimnisse. Ephraim blockiert offensichtliche sensible Zugangsinformationen wie Passwörter, Tokens, API-Schlüssel, private Schlüssel, Recovery-Codes und ähnliche Zugangsdaten.

> **Gute Praxis:** Speichere stabile, harmlose Hinweise. Speichere keine Zugangsdaten, keine Gesundheitsdetails und keine Informationen, die du nicht dauerhaft in deinem persönlichen Konto behalten möchtest.

4.5.6 Datenschutz und Vault

Erinnerungen liegen verschlüsselt im persönlichen Vault. Die Datenbank speichert keine Klartext-Erinnerungen. Für Lesen, Anlegen und Bearbeiten braucht Ephraim eine angemeldete Sitzung und einen entsperrten Vault.

- Admins und Lehrkräfte können deine Klartext-Erinnerungen nicht lesen.
- Cron-Aufgaben können keine persönlichen Erinnerungen entschlüsseln.
- Der Datenexport kann aktive und archivierte Erinnerungen nur für dich im entsperrten Vault-Kontext entschlüsseln.
- Erinnerungen sind freiwillige Hilfen für den persönlichen Chat, kein Schulprofil für andere Bereiche.

> **Merksatz:** Erinnerungen machen Ephraim hilfreicher, ohne private Chatinhalte für Lehrkräfte oder Administration zu öffnen.

Dieser Artikel beschreibt persönliche Erinnerungen im normalen Chat. Für die Oberfläche der Personalisierung siehe Einstellungen: Personalisierung; für den verschlüsselten Datenraum siehe Vault.

Quelle: web/manuals/chat-erinnerungen/index.html

4.6 Chat: Export

> Export bedeutet in Ephraim: Ein Inhalt, der im persönlichen Vault geschützt liegt, wird bewusst in eine kopierbare oder herunterladbare Form gebracht. Dieser Artikel erklärt Auswahl, Formate, Visualisierungen und die Datenschutzgrenze.

Stand: Juni 2026

4.6.1 Was Export in Ephraim bedeutet

Ein Chat ist normalerweise Teil deines geschützten Arbeitsbereichs. Wenn du exportierst, verlässt der Inhalt diesen geschützten Zustand: Er landet in der Zwischenablage oder als Datei auf deinem Gerät. Das ist gewollt, weil man Ergebnisse weiterverwenden möchte: als Arbeitsblatt, Lernnotiz, Projektabgabe, Protokoll, Elterninformation oder fachliche Dokumentation.

Deshalb behandelt Ephraim Export nicht als versteckte Nebenfunktion. Du wählst bewusst, welcher Teil des Chats exportiert wird, welches Format entstehen soll und ob zusätzliche Angaben wie Zeitstempel, Metadaten oder Anhangslisten dazugehören.

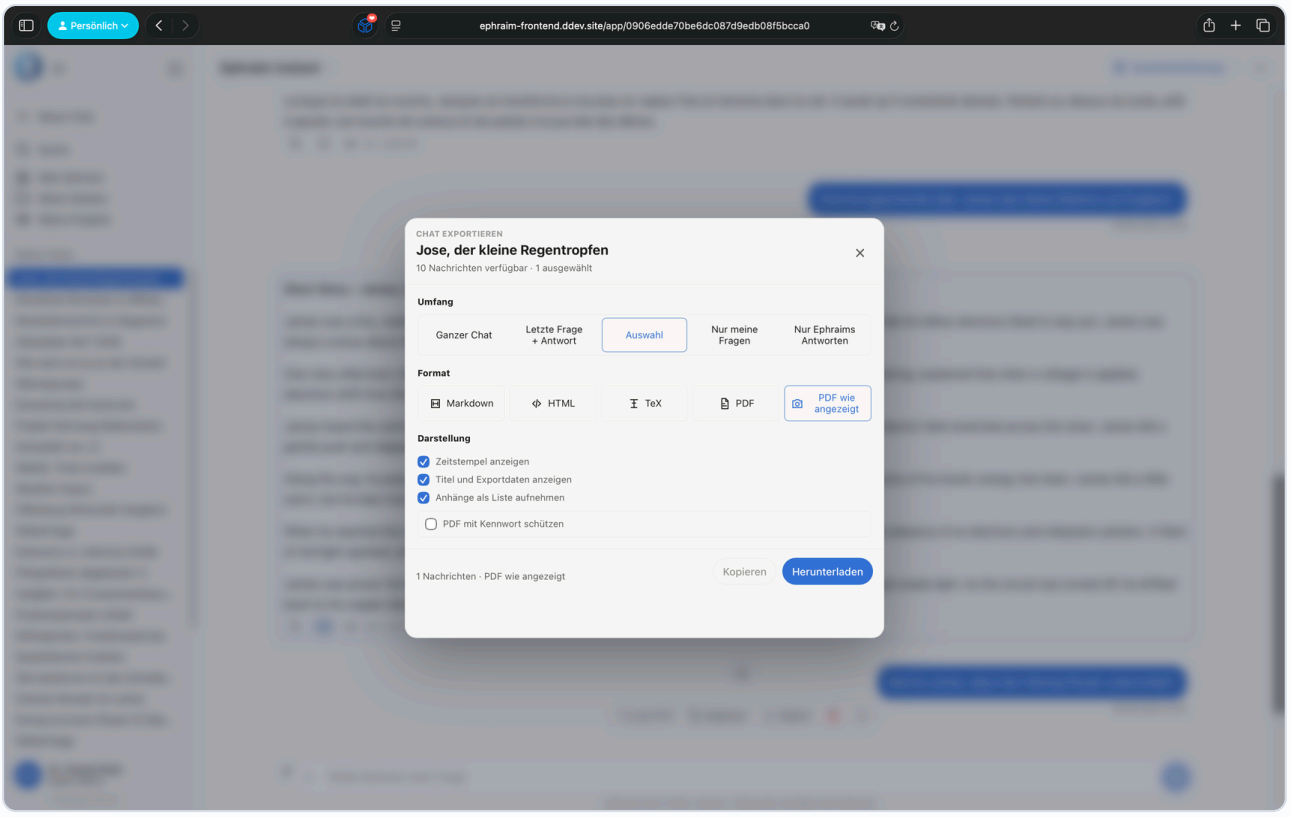
> **Merksatz:** Exportierte Inhalte sind Klartextkopien. Nach dem Download schützt nicht mehr der Vault, sondern dein Gerät, dein Speicherort und dein Umgang mit der Datei.

4.6.2 Welche Inhalte du auswählen kannst

Ephraim trennt zwischen schneller Einzelaktion und bewusster Sammelauswahl. Dadurch kannst du sehr kleine Ausschnitte exportieren, ohne den ganzen Chat mitzunehmen, oder einen längeren Verlauf gezielt zusammenstellen.

Auswahl	Was passiert	Typischer Einsatz
Einzelne Nachricht	Die Aktion an der Nachricht kopiert oder exportiert nur diese Nachricht.	Eine gute Erklärung, eine Tabelle oder eine Formel übernehmen.
Letzte Frage mit Antwort	Die letzte eigene Frage und die zugehörige Antwort werden gemeinsam exportiert.	Eine abgeschlossene Mini-Erklärung ohne den gesamten Vorlauf sichern.
Aktuelle Auswahl	Du markierst mehrere Nachrichten; Ephraim exportiert nur diese markierten Teile.	Aus einem langen Chat nur die relevanten Abschnitte herausziehen.
Ganzer Chat	Alle sichtbaren Nachrichten des gespeicherten Chats werden exportiert.	Ein vollständiges Protokoll oder eine vollständige Lernspur sichern.
Nur Nutzerfragen	Exportiert werden nur deine eigenen Eingaben.	Aufgabenstellung, Prompt-Sammlung oder eigenes Vorgehen dokumentieren.
Nur Antworten	Exportiert werden nur die Antworten von Ephraim.	Erklärtexte, Zusammenfassungen oder Lösungsvorschläge ohne Zwischenfragen sammeln.
Einzelne Visualisierung	Eine Diagramm-, Zeichen- oder Chemiebox wird separat ausgewählt.	Nur eine Grafik übernehmen, ohne die ganze Antwort mitzunehmen.

Die Auswahl ist sichtbar. Markierte Nachrichten zählen in einer Auswahlleiste. Dort kannst du kopieren, exportieren, die Auswahl wieder aufheben oder markierte Teile nach Bestätigung aus dem sichtbaren Chat entfernen.



Der Exportdialog zeigt zuerst den Umfang, dann das Format und darunter die Darstellungsoptionen. Im Beispiel ist eine Nachricht ausgewählt und wird als „PDF wie angezeigt“ exportiert; Zeitstempel, Titel, Exportdaten und Anhänge werden als Liste übernommen.

4.6.3 Die Exportformate verständlich erklärt

Die Formate richten sich nicht an dieselbe Situation. Die richtige Wahl hängt davon ab, ob du den Inhalt später bearbeiten, weitergeben, drucken oder technisch weiterverarbeiten möchtest.

Format	Einfach erklärt	Wann es passt
Markdown	Strukturierter Text mit einfachen Zeichen für Überschriften, Listen, Tabellen und Code.	Wenn du den Inhalt in Notizen, Dokumentationen, NotebookLM oder andere Textwerkzeuge übernehmen willst.
HTML	Eine Browser-Datei mit Formatierung, Tabellen und sicher eingebetteten Darstellungen.	Wenn der Inhalt am Bildschirm gut aussehen und als Webdokument archiviert werden soll.
TeX	Eine Textquelle für LaTeX-nahe Dokumente, besonders stark bei Mathematik.	Wenn Formeln, fachliche Ausarbeitungen oder technische Dokumente später sauber gesetzt werden sollen.
PDF	Ein druckbares Dokument mit stabiler Seitenform. Auf Wunsch kann ein Öffnungskennwort gesetzt werden.	Wenn du etwas weitergeben, ausdrucken oder als fertiges Dokument speichern möchtest.
PDF wie angezeigt	Ephraim fotografiert die sichtbare Chatdarstellung seitenweise in ein PDF.	Wenn es genau auf das sichtbare Layout ankommt, zum Beispiel bei Grafiken, Formeln oder Chatverläufen.

Für normale Weitergabe ist PDF meist am einfachsten. Für Weiterarbeit ist Markdown oft besser, weil der Text nicht in einer festen Seitenform eingeschlossen ist. TeX ist sinnvoll, wenn mathematische oder technische Dokumente später professionell gesetzt werden sollen. „PDF wie angezeigt“ ist die richtige Wahl, wenn die sichtbare Mischung aus Text, Formeln, Tabellen und Visualisierungen möglichst genau erhalten bleiben soll.

4.6.4 Metadaten, Zeitstempel, Anhänge und Kennwort

Ephraim kann zusätzliche Angaben einfügen. Metadaten beschreiben den Export und den Chatkontext. Zeitstempel zeigen, wann Nachrichten entstanden sind. Anhangslisten nennen Dateien, die im Chat beteiligt waren, ohne dadurch automatisch jede Datei erneut herunterzuladen.

Für PDF-Exporte kann ein Öffnungskennwort gesetzt werden. Dieses Kennwort schützt die erzeugte PDF-Datei beim Öffnen. Es ist kein Vault-Passwort, wird nicht dauerhaft in Ephraim gespeichert und ersetzt keine sorgfältige Ablage der Datei.

4.6.5 Visualisierungen im Export

Ephraim exportiert Visualisierungen nicht als frei ausführbaren Programmcode. Diagramme, Zeichnungen, chemische Strukturen und Plots werden als sichere sichtbare Ergebnisse, als Beschreibung oder als Browser-Snapshot übernommen. Dadurch bleibt der Export nutzbar, ohne dass heruntergeladene Dateien aktive Skripte enthalten müssen.

Bei normalen Textformaten steht eher die fachliche Beschreibung im Vordergrund. Bei PDF und besonders bei „PDF wie angezeigt“ ist die sichtbare Grafik entscheidend. Ephraim wartet deshalb auf gerenderte Darstellungen und bricht mit einer sichtbaren Fehlermeldung ab, wenn ein Plot nicht zuverlässig als Bild übernommen werden kann.

4.6.6 Auswahl und Löschen

Die Auswahlleiste kann markierte Nachrichten nicht nur exportieren, sondern nach Bestätigung auch aus dem sichtbaren Chat entfernen. Das ist kein heimlicher Export, sondern eine Änderung am gespeicherten verschlüsselten Verlauf. Ganze Antworten werden dabei als Gesprächszug behandelt: Wird eine Antwort entfernt, entfernt Ephraim auch die dazugehörige Frage und weitere Antworten desselben Zugs, damit kein unverständlicher Restverlauf entsteht.

Einzelne Visualisierungen können separat gelöscht werden, wenn nur die Grafikbox entfernt werden soll. Wird dagegen die ganze Nachricht ausgewählt, zählt Ephraim enthaltene Einzelboxen nicht doppelt.

4.6.7 Datenschutzgrenze des Exports

Der Server prüft beim Export, ob der Chat zu deinem Konto gehört. Exportdaten werden für den Chat-Export nicht dauerhaft auf dem Server abgelegt. Die Antwort wird als Download oder Kopierinhalt erzeugt und mit privaten No-Store-Cache-Regeln ausgeliefert.

Entscheidend bleibt trotzdem: Der Export ist der Moment, in dem geschützte Inhalte eine bewusst herunterladbare Form erhalten. Deshalb gehören exportierte Chatinhalte nicht in ungeschützte Cloudordner, öffentliche Tickets oder frei geteilte Messenger-Gruppen, wenn sie personenbezogene oder vertrauliche Informationen enthalten.

Dieser Artikel beschreibt Exporte aus dem normalen Chat. Für den allgemeinen Kontodatenexport siehe Einstellungen: Meine Daten. Für die Grundlagen des Vaults siehe Vault.

Quelle: web/manuals/chat-export/index.html

4.7 Chat: Sprachausgabe

> **Ephraim kann Chatantworten und ausgewählte Chatteile vorlesen. Die Sprachausgabe läuft über den internen TTS-Dienst auf der Spark, unterstützt mehrere Sprachen und verwendet keinen Browser- oder Cloud-Fallback.**

Stand: Juni 2026

4.7.1 Was vorgelesen wird

Vorgelesen werden fertige Chatinhalte. Während eine Antwort noch entsteht, wartet Ephraim, bis der Stream abgeschlossen ist. Dadurch hörst du keinen halben Zwischenstand und keine noch unfertige Satzfolge.

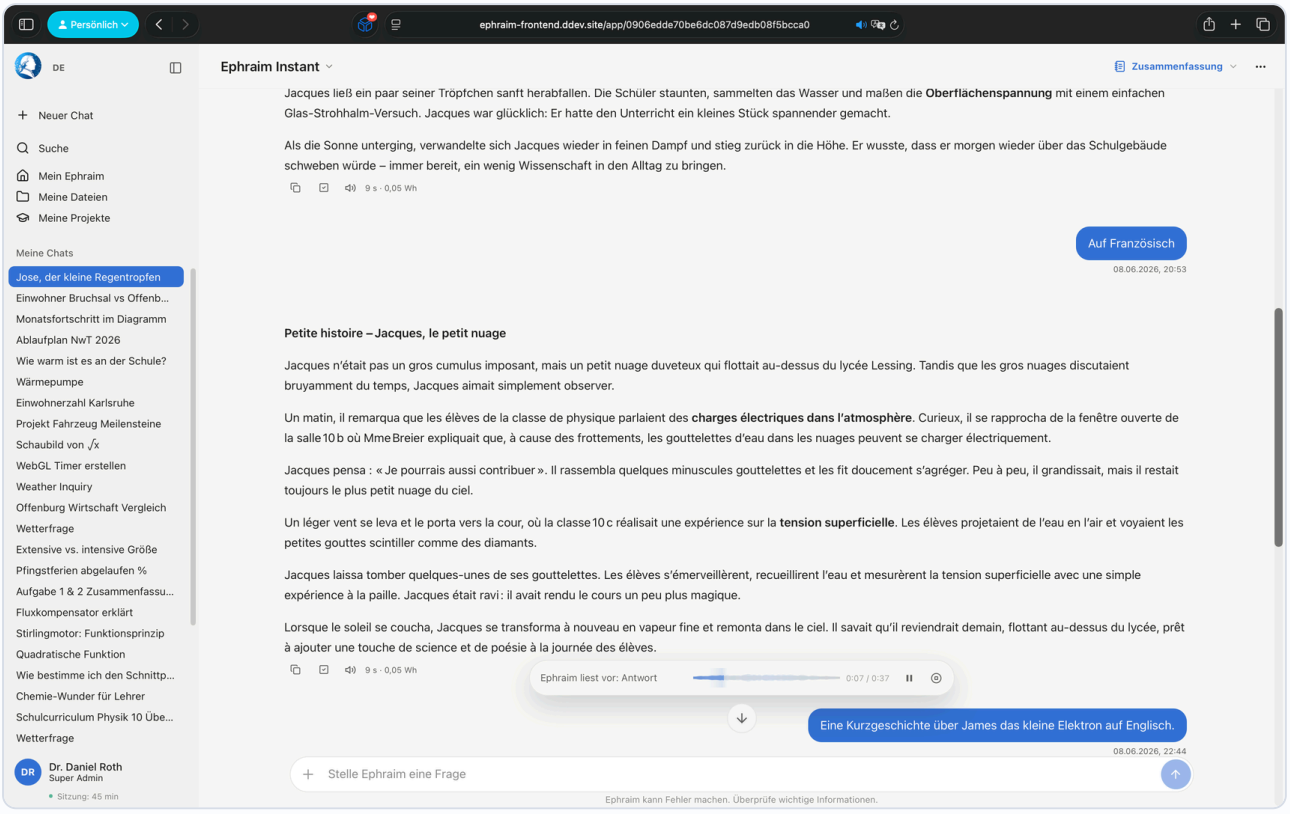
Die Sprachausgabe ist für typische Arbeitssituationen gedacht: eine Erklärung noch einmal hören, einen längeren Text auditiv prüfen, eine Antwort in der Fremdsprache anhören oder ausgewählte Chatteile für konzentriertes Arbeiten vorlesen lassen.

4.7.2 Auswahl und Player

Der Vorlesedialog nutzt dieselbe Grundidee wie die Exportauswahl: Du entscheidest, welcher Teil des Chats in eine andere Form gebracht wird. Je nach Chatstand sind mehrere Umfänge möglich.

Auswahl	Was Ephraim liest
Letzte Antwort	Nur die letzte fertige Antwort von Ephraim.
Letzte Frage mit Antwort	Die letzte Nutzerfrage und die zugehörige Antwort.
Aktuelle Auswahl	Die Nachrichten, die du vorher im Chat markiert hast.
Ganzer Chat	Alle sichtbaren Nachrichten des geöffneten Chats.
Nur Antworten	Nur die Antworten von Ephraim, ohne deine Zwischenfragen.

Die Wiedergabe kann gestartet, pausiert, fortgesetzt und gestoppt werden. Beim Wechsel des Chats, beim Start eines neuen Gesprächs oder beim Schließen der Chatansicht stoppt Ephraim die Wiedergabe, damit keine alte Audiospur in einen anderen Arbeitskontext hineinläuft.



Im geöffneten Chat liest Ephraim eine fertige Antwort vor. Der schwebende Player zeigt den aktuellen Umfang, Fortschritt, Pause und Stopp; der französische Antworttext bleibt dabei sichtbar im Chatkontext.

4.7.3 Sprachen und gemischte Texte

Die Sprachausgabe unterstützt Deutsch, Englisch, Französisch, Spanisch, Russisch, Ukrainisch und Türkisch. Französisch und Spanisch beziehen sich hier auf die Sprachausgabe, nicht auf eine vollständige Übersetzung der Oberfläche.

Besonders wichtig ist gemischter Text. Eine Antwort kann zum Beispiel auf Deutsch erklären, ein französisches Zitat enthalten und danach eine englische Fachbezeichnung verwenden. Ephraim zerlegt solche Texte in passende Abschnitte und lässt jeden Abschnitt mit der geeigneten Sprache synthetisieren. So wird ein französischer Satz nicht wie deutsches Schriftbild vorgelesen.

Dafür nutzt Ephraim Sprachhinweise der KI und prüft sie zusätzlich im Browser. Kurze Überschriften, Listenpunkte und einzelne Beispiele werden nicht blind übernommen: Ephraim bewertet lokale Sprachsignale und entscheidet erst dann, welche Sprache ein Abschnitt bekommt. Unsichere oder widersprüchliche Hinweise werden verworfen.

4.7.4 Wie Text für das Vorlesen vorbereitet wird

Ephraim liest nicht den rohen Markdown-Code vor, sondern die gerenderte Chatdarstellung. Dadurch klingen Listen, Tabellen, Formeln und Visualisierungen verständlicher.

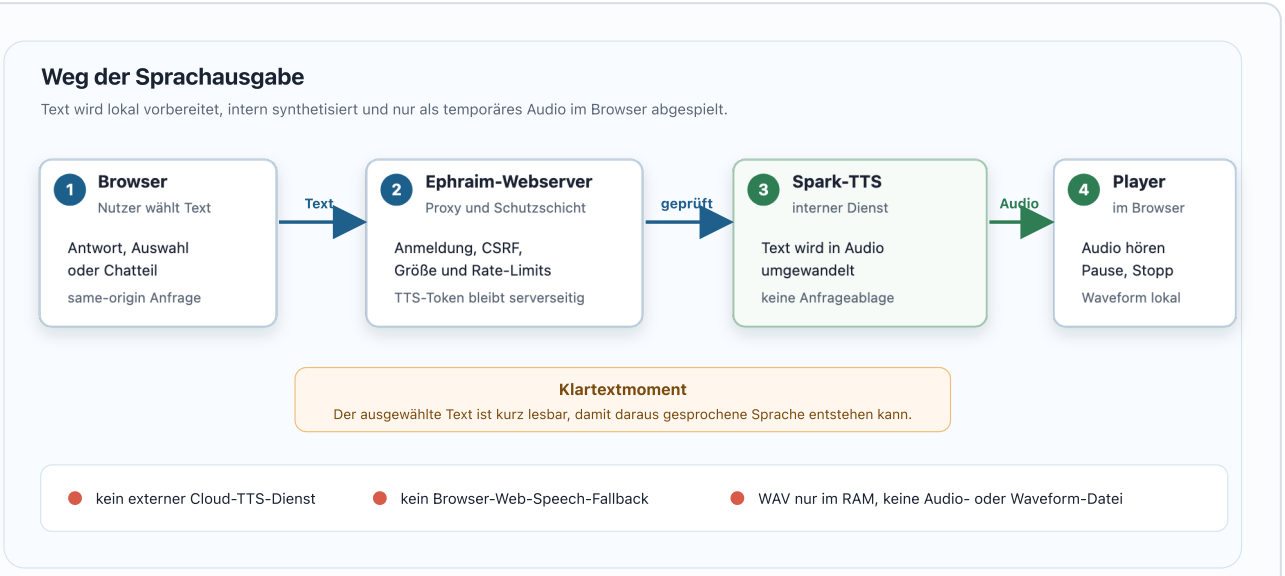
Element	Vorleselogik
Tabellen	Die Tabelle wird als Tabelle mit Spalten und Zeilen beschrieben.
Codeblöcke	Lange Codeblöcke werden zusammengefasst, statt Zeichen für Zeichen vorgelesen zu werden.
Formeln	Formeln werden als Formeln markiert, damit sie nicht als unverständliche Rohzeichenfolge erscheinen.
Visualisierungen	Diagramme, Zeichnungen oder Strukturformeln werden mit Titel und fachlicher Beschreibung vorgelesen.
Anhänge	Dateien werden mit Name, Typ und Größe beschrieben.

4.7.5 Warum die Spark dafür wichtig ist

Die Audiodatei entsteht nicht in einem öffentlichen Browserdienst. Der Browser sendet den ausgewählten Text an den Ephraim-Webserver. Der Webserver prüft Anmeldung, Schutz-Token, Größenlimits und Rate-Limits und leitet die Anfrage dann an den internen TTS-Dienst auf der Spark weiter. Der Browser kennt die interne Adresse und das interne Geheimnis dieses Dienstes nicht.

Das zurückgelieferte WAV wird im Webserver nicht als Datei abgelegt. Ephraim nimmt die Audiodaten in einem begrenzten Speicherpuffer entgegen und gibt sie direkt als HTTP-Antwort an den Browser weiter. Auch im Browser entsteht daraus nur ein temporärer Audio-Blob für den Player und die lokale Waveform-Anzeige. Es gibt also keinen serverseitigen WAV-Dateipfad, keine temporäre Audiodatei und keine spätere Audiodatei-Bereinigung.

Ist Spark-TTS nicht eingerichtet, bleibt die Vorlesefunktion deaktiviert. Ephraim fällt nicht auf die Web-Speech-Funktion des Browsers zurück. Das ist eine bewusste Grenze: Die Schule entscheidet, welcher lokale Dienst Text in Audio umwandelt.



Der Browser kennt nur den Ephraim-Webserver. Der Webserver prüft die Anfrage und spricht intern mit Spark-TTS; die erzeugte Audiodatei läuft zurück in den Player. Externe Cloud-Dienste, Browser-Web-Speech und Audiodateien auf der Serverplatte werden nicht verwendet.

4.7.6 Datenschutz und Grenzen

Für die Sprachausgabe muss der ausgewählte Text kurzzeitig im Klartext verarbeitet werden, sonst kann daraus keine Stimme entstehen. Diese Verarbeitung bleibt im Ephraim-System und im internen Spark-Dienst. Text, erzeugtes Audio und lokale Waveform-Punkte werden nicht dauerhaft gespeichert. Das WAV wird im Proxy nur im RAM gehalten und direkt an den Browser ausgeliefert.

Die Sprachhinweise zu einer Antwort können zusammen mit der Assistant-Nachricht im verschlüsselten Chatinhalt liegen. Sie dienen dazu, den Text später wieder mit passenden Sprachabschnitten vorzulesen. Sie sind keine öffentlichen Profildaten. Ephraim nimmt dafür keine Nutzerstimme auf und wertet keine Stimme aus.

> **Wichtig:** Sprachausgabe verändert den Chat nicht. Sie macht aus vorhandenem Text nur eine temporäre Audiospur.

Für Kopieren und Downloads siehe Chat: Export. Für die allgemeine Bedienlogik des Chatbereichs siehe Chat: Überblick.

Quelle: web/manuals/chat-sprachausgabe/index.html

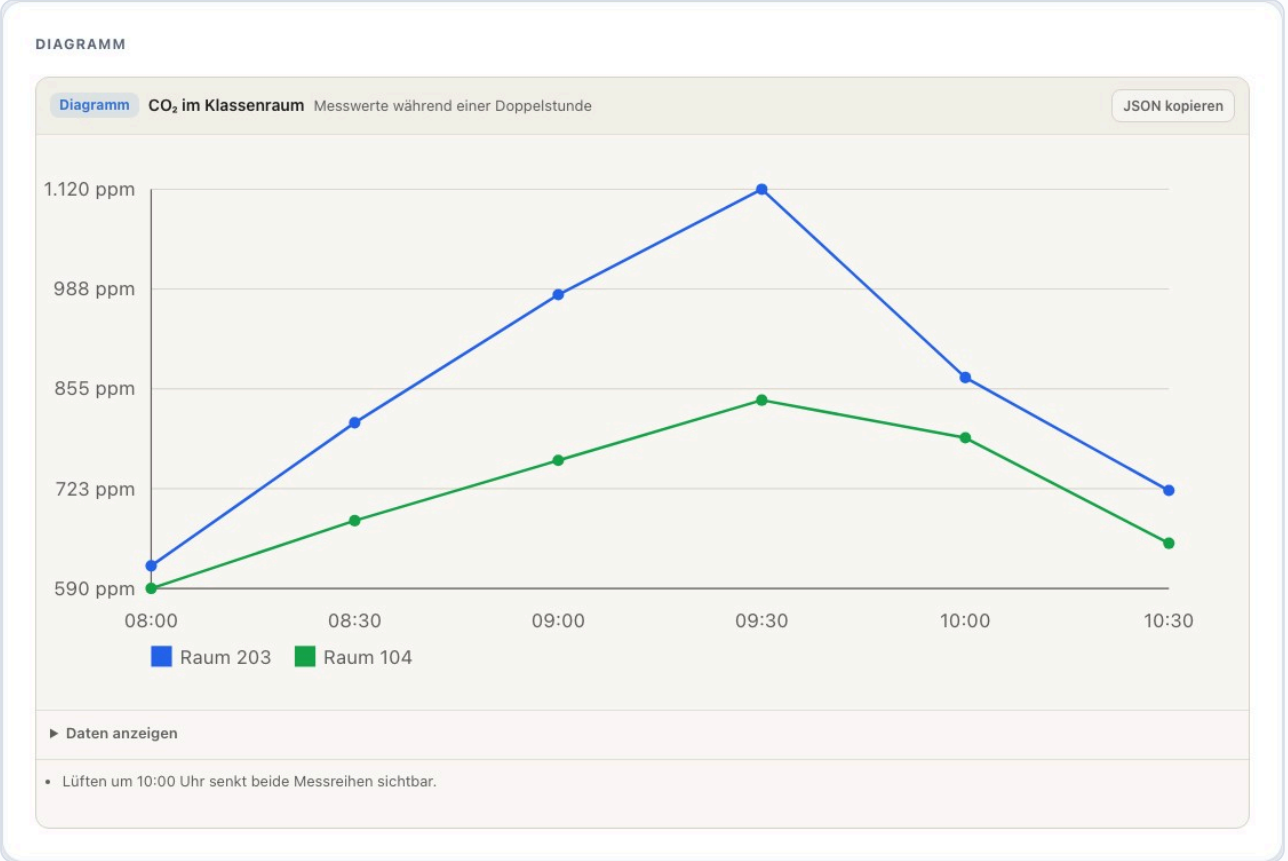
4.8 Visualisieren: Diagramme

> Diagramme stellen Zahlenreihen sicher als lokale SVG-Grafik dar. Ephraim verwendet dafür ein begrenztes Chart-Format statt frei ausführbarem JavaScript.

Stand: Mai 2026

4.8.1 Wofür Diagramme bestimmt sind

Diagramme eignen sich für Messwerte, Vergleiche, Anteile und Punktwolken: CO_2-Werte im Klassenraum, Umfrageergebnisse, Zeitreihen, Größenvergleiche oder einfache Streudiagramme. Mathematische Funktionsgraphen gehören nicht in diesen Typ; dafür nutzt Ephraim mathematische Plots.



Ein Diagramm entsteht aus Zahlen, Labels und Serien. Die Darstellung wird im Browser als SVG erzeugt.

4.8.2 Unterstützte Diagrammtypen

Typ	Verwendung	Besonderheit
Balken	Vergleiche zwischen Kategorien.	Mehrere Serien stehen nebeneinander.
Gestapelte Balken	Zusammensetzungen innerhalb von Kategorien.	Positive Werte werden pro Kategorie gestapelt.
Horizontale Balken	Lange Kategorienamen und Ranglisten.	Kategorien stehen links, Werte laufen horizontal.
Linie und Fläche	Zeitreihen und geordnete Messfolgen.	Bis zu 200 Labels pro Reihe sind vorgesehen.
Kreis	Anteile eines Ganzen.	Genau eine Serie und höchstens 12 Segmente.
Streudiagramm	Zusammenhänge zwischen x- und y-Werten.	Punkte tragen optionale Labels.

4.8.3 Was die KI liefert

Die KI liefert eine JSON-Beschreibung mit Version, Diagrammtyp, Titel, Achsen, Labels, Serien und optionalen Notizen. Kurzformen werden vor dem Rendern in eine einheitliche Serienstruktur umgewandelt. Das verhindert, dass ähnliche Antworten unterschiedlich behandelt werden.

> **Klare Grenze:** Ein Diagramm verarbeitet Daten, keine Formeln. Wenn die Aufgabe eine Funktion wie `f(x)=x²` zeichnet, ist der Plot-Typ zuständig.

4.8.4 Sicherheitslogik

Der Renderer akzeptiert nur bekannte Felder. Zahlen müssen endlich sein, Farben müssen gültige Hex-Farben sein, Textlängen sind begrenzt und unbekannte Diagrammtypen werden abgewiesen. Die fertige Darstellung besteht aus lokal erzeugtem SVG. Modellgeneriertes HTML, SVG oder JavaScript wird nicht in die Seite übernommen.

4.8.5 Export

Beim Kopieren und Exportieren übernimmt Ephraim nicht den Roh-JSON-Block, sondern die sichtbare Darstellung oder eine Ergebnisbeschreibung. Für PDF-Exporte erstellt der Browser einen Snapshot der gerenderten Grafik und übergibt diesen an den Exportprozess.

Diagramme sind für strukturierte Zahlen da. Funktionen, interaktive Parameter und 3D-Flächen gehören zu mathematischen Plots.

Quelle: web/manuals/visualisierung-diagramme/index.html

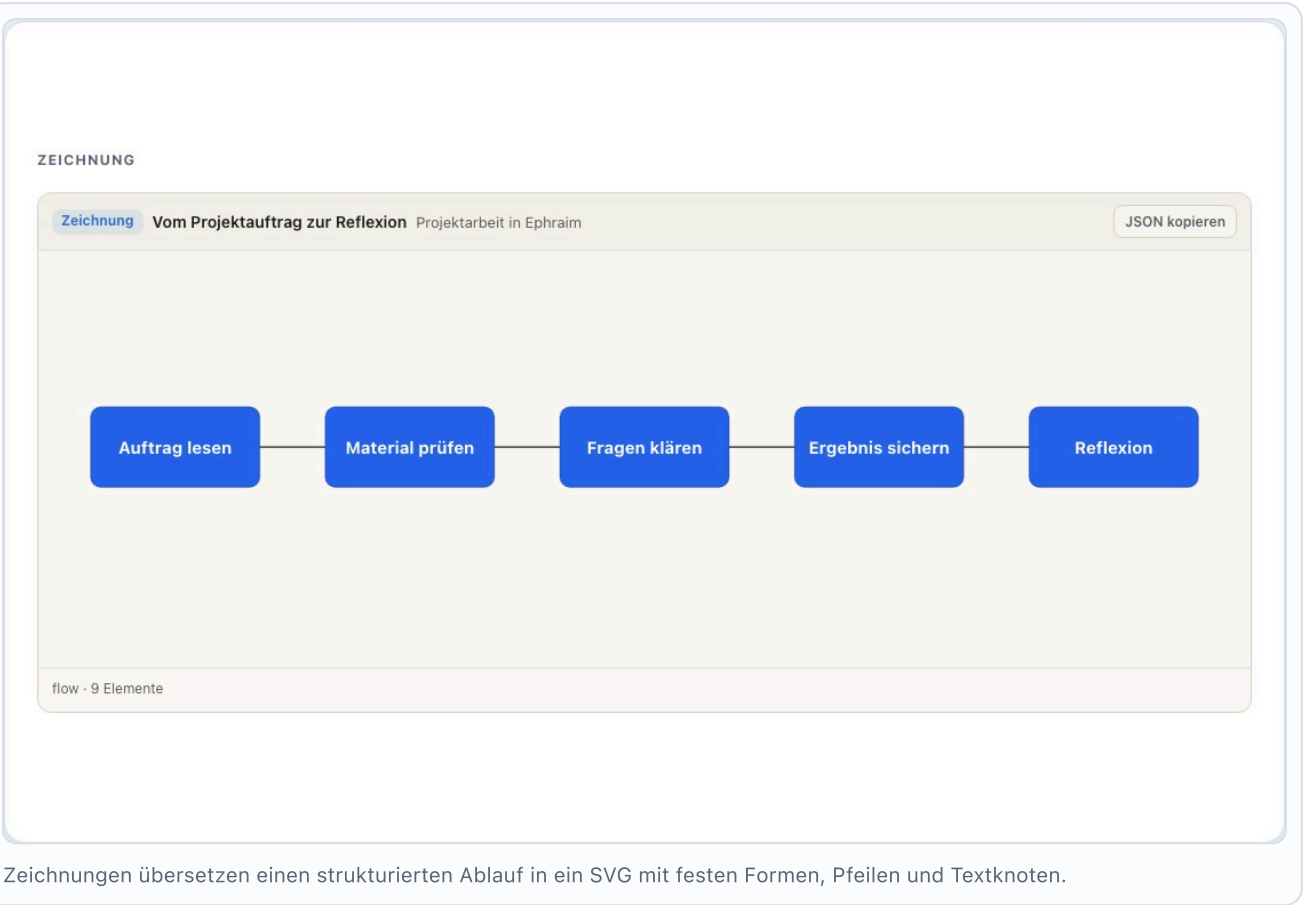
4.9 Visualisieren: Zeichnungen

> **Zeichnungen ordnen Begriffe, Schritte und Beziehungen. Ephraim erzeugt daraus lokale SVG-Grafiken mit begrenzten Formen, Texten, Linien und Pfeilen.**

Stand: Mai 2026

4.9.1 Wofür Zeichnungen bestimmt sind

Zeichnungen eignen sich für Abläufe, Mindmaps, Zeitstrahlen, Bäume, Swimlanes, Vier-Felder-Matrizen und freie Skizzen. Sie erklären Strukturen, nicht Messwerte. Für Zahlenreihen nutzt Ephraim Diagramme; für Funktionen und Parameter nutzt Ephraim Plots.



Zeichnungen übersetzen einen strukturierten Ablauf in ein SVG mit festen Formen, Pfeilen und Textknoten.

4.9.2 Unterstützte Layouts

Layout	Geeignet für
Flow	Schrittfolgen, Prozessketten und Entscheidungswege.
Mindmap	Zentrales Thema mit Ästen und Unterästen.
Timeline	Ereignisse in zeitlicher Reihenfolge.
Swimlane	Abläufe mit Zuständigkeiten oder Rollen.
Matrix	Einordnung nach zwei Achsen.
Tree	Hierarchien, Kategorien und Entscheidungsbäume.
Freeform und Venn	Manuell gesetzte Formen und einfache Mengenbilder.

4.9.3 Bausteine

Eine Zeichnung besteht aus Rechtecken, Ellipsen, Texten, Linien, Pfeilen und Gruppen. Kompakte Layoutangaben wie Schritte, Ereignisse, Lanes, Wurzelknoten oder Kinder werden vor dem Rendern in diese Bausteine umgewandelt. Dadurch bleibt die Darstellung reproduzierbar und kontrollierbar.

4.9.4 Sicherheitsgrenzen

Ephraim begrenzt Koordinaten, Elementgrößen, Zeichenfläche und Elementzahl. Pfeile verweisen nur auf vorhandene IDs oder auf konkrete Koordinaten. Texte werden als Textknoten gesetzt, nicht als HTML. Die SVG-Ansicht wird nach dem Rendern so erweitert, dass zulässige Knoten nicht abgeschnitten werden.

4.9.5 Gute Aufgabenstellungen

Gute Anfragen nennen die gewünschte Struktur: „Erstelle eine Timeline“, „Zeichne einen Flow mit fünf Schritten“ oder „Ordne die Begriffe in einer Matrix“. Die KI muss dann weniger raten und erzeugt eine stabilere Zeichnung.

Zeichnungen sind für Struktur und Beziehungen da. Sie ersetzen keine Datenanalyse und keine mathematische Plotfläche.

Quelle: web/manuals/visualisierung-zeichnungen/index.html

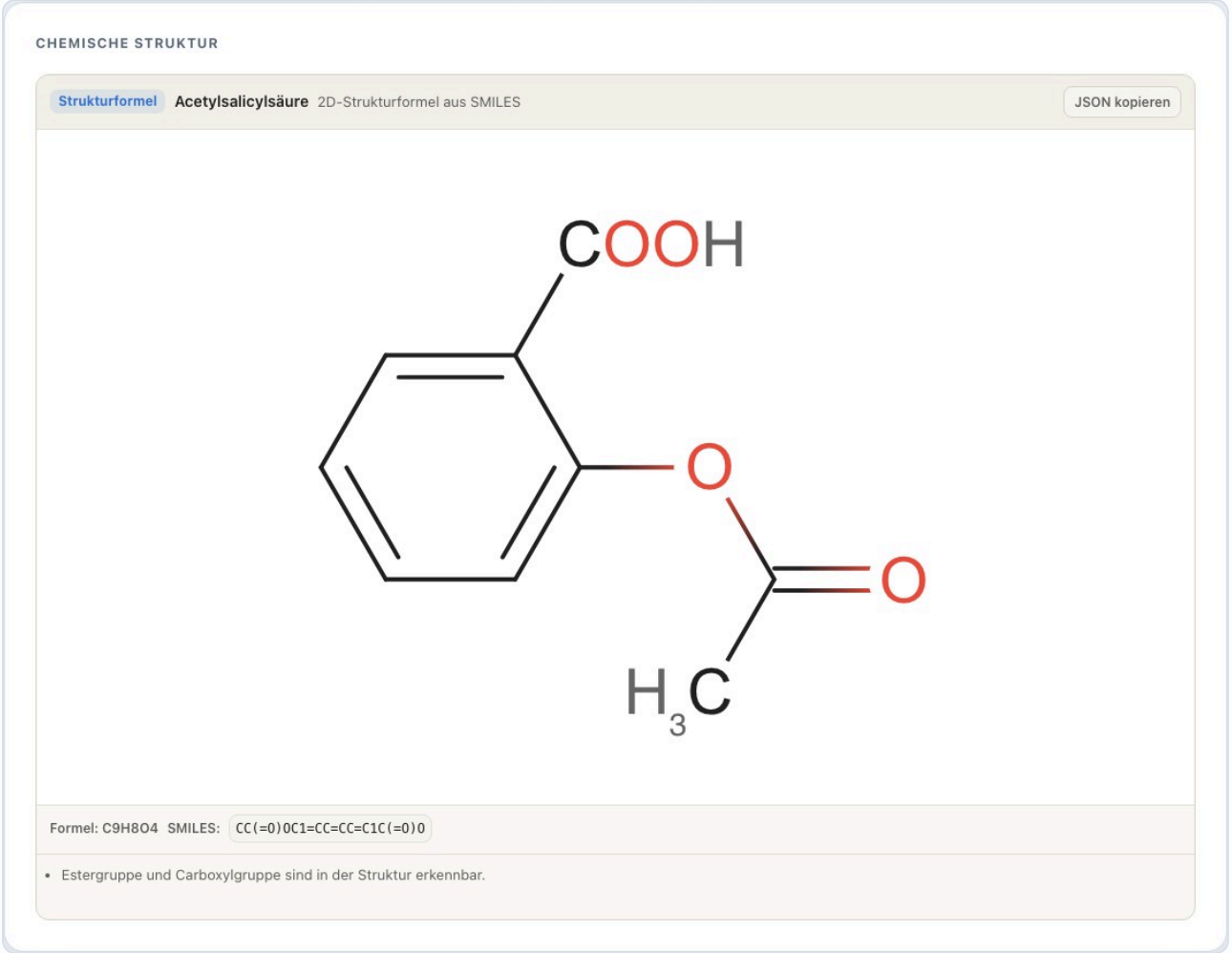
4.10 Visualisieren: Chemische Strukturen

> **Chemische Visualisierungen zeichnen 2D-Strukturformeln aus SMILES. Ephraim nutzt dafür eine lokal eingebundene Strukturzeichnung und lädt keine externen Chemiedaten nach.**

Stand: Mai 2026

4.10.1 Wofür chemische Strukturen bestimmt sind

Der Chemie-Typ ist für echte Strukturformelbilder gedacht: Moleküle, funktionelle Gruppen, Bindungsmuster und Unterrichtsbeispiele aus organischer Chemie. Summenformeln, Reaktionsgleichungen und Stoffmengenrechnungen bleiben normaler Text oder Mathematik.



Die Struktur entsteht aus einem SMILES-String und wird lokal als SVG gezeichnet.

4.10.2 Was SMILES bedeutet

SMILES ist eine Textschreibweise für Molekülstrukturen. Sie beschreibt Atome, Bindungen, Ringe und Verzweigungen in einer kompakten Zeichenkette. Ephraim übergibt diese Zeichenkette an die lokale Strukturzeichnung. Aus der Zeichenkette entsteht dann eine 2D-Darstellung.

> **Beispiel:** Acetylsalicylsäure wird nicht als Bilddatei geladen. Die Struktur wird aus einem SMILES-Text lokal gezeichnet.

4.10.3 Datenmodell

Feld	Bedeutung
SMILES	Pflichtfeld für die Struktur.
Titel und Untertitel	Beschriftung der Darstellung.
Summenformel	Zusatzinformation unter der Struktur.
Notizen	Kurze Hinweise, etwa funktionelle Gruppen.
Optionen	Kohlenstoffanzeige, explizite Wasserstoffe, kompakte Zeichnung und Theme.

4.10.4 Grenzen

Das Chemieformat zeichnet Strukturformeln. Es recherchiert keine Stoffdaten, prüft keine Laborvorschriften und ersetzt keine fachliche Bewertung. Alte Atomlabel-Felder werden zur Kompatibilität gelesen, aber nicht als zusätzliche Markierungen in die Struktur gezeichnet. Atomzuordnungen gehören in die erklärende Antwort.

4.10.5 Sicherheit

Ephraim erlaubt nur bekannte Felder und begrenzt die SMILES-Länge. Das erzeugte SVG wird bereinigt; Skripte und fremde Inhalte werden entfernt. Die Strukturzeichnung läuft lokal im Browser mit der eingebundenen Bibliothek. Es findet keine Websuche nach Moleküldaten statt.

Chemische Visualisierungen sind lokale Strukturzeichnungen aus SMILES. Fachliche Stoffdaten müssen im Text geprüft und erklärt werden. Die übrigen Visualisierungstypen stehen in der Navigationsgruppe **Visualisieren**.

Quelle: web/manuals/visualisierung-chemie/index.html

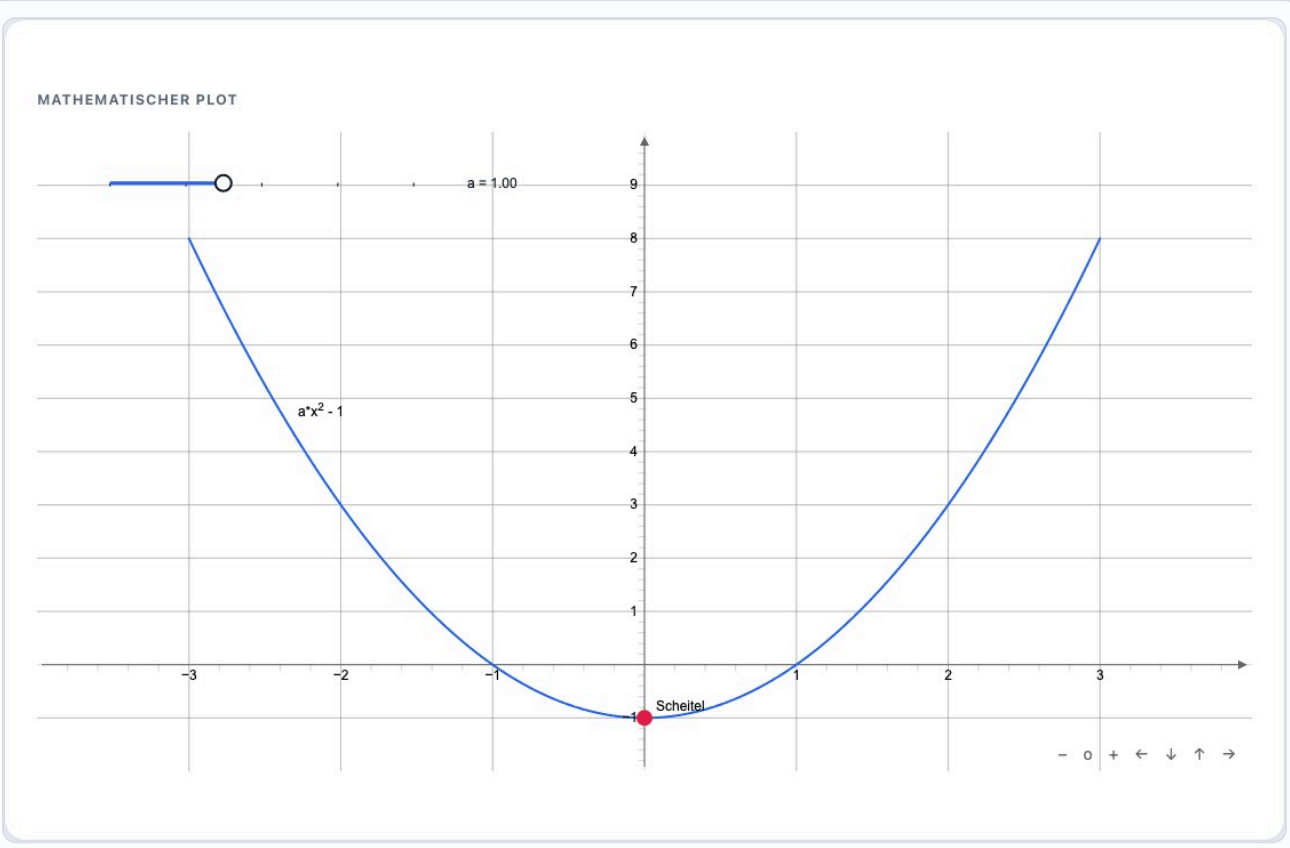
4.11 Visualisieren: Mathematische Plots

> Plots stellen Funktionen, Kurven, Punkte, 3D-Flächen und Parameter interaktiv dar. Sie laufen in einem isolierten Bereich und verwenden eine sichere JSON-DSL statt freiem Code.

Stand: Mai 2026

4.11.1 Wofür Plots bestimmt sind

Plots sind der mathematische Visualisierungstyp: Funktionsgraphen, parametrische Kurven, Punktmengen, Stützpunktkurven, 3D-Flächen und Regler für Parameter. Ein Diagramm zeigt vorhandene Messwerte; ein Plot berechnet Punkte aus mathematischen Ausdrücken.



Plots werden in einem isolierten iframe gerendert. Parameter-Slider aktualisieren die Kurven direkt in der Darstellung.

4.11.2 Die Plot-DSL

Die KI liefert eine JSON-Beschreibung mit Version, Titel, Achsen, Grenzen, Elementen und optionalen Parametern. Diese Beschreibung heißt DSL, weil sie eine kleine, auf Plots begrenzte Sprache ist. Sie erlaubt genau die mathematischen Formen, die der Renderer kennt.

Element	Was es darstellt
Function	Graph `y=f(x)` über einem x-Bereich.
Parametric	Kurve mit `x(t)` und `y(t)`.
Point	Einzelner Punkt mit x- und y-Koordinate.
Polyline	Verbundene Stützpunkte.
Surface3d	3D-Fläche `z=f(x,y)`.

4.11.3 Parameter und Slider

Parameter haben Namen, Startwert, Minimum, Maximum und optional eine Schrittweite. Ihre Namen dürfen in Funktionen, parametrischen Kurven und 3D-Flächen verwendet werden. Die Slider gehören zur Plotfläche und ändern die Darstellung live.

4.11.4 Mathematische Ausdrücke

Ausdrücke laufen durch einen eigenen Parser. Er kennt Zahlen, Variablen, Klammern, Grundrechenarten, Potenzen und elementare Funktionen wie Sinus, Kosinus, Tangens, Wurzel, Betrag, Exponentialfunktion und Logarithmus. Es gibt kein `eval`, keine freie JavaScript-Ausführung und keine selbst definierten Funktionen.

4.11.5 Sandbox

Der Plot wird in einem iframe mit Sandbox-Regeln dargestellt. Dieser Bereich hat keine Same-Origin-Rechte zur App, führt keine Webanfragen aus und erhält nur die validierte Plot-Beschreibung. Die Darstellung nutzt lokal bereitgestellte Plot-Bibliotheken und sendet für Exporte einen SVG-Snapshot an die umgebende Chatseite zurück.

4.11.6 Alter JSXGraph-Code

Freier JSXGraph-Code wird nicht ausgeführt. Erkennt Ephraim einen alten freien Plot-Codeblock, landet er im Reparaturpfad. Die Reparatur erzeugt eine neue JSON-Beschreibung. Auch dieser Vorgang schreibt keine neue Chatnachricht.

Plots sind die interaktive Mathefläche von Ephraim. Ihre Stärke ist Mathematik, nicht frei ausführbarer Programmcode.

Quelle: web/manuals/visualisierung-plots/index.html

4.12 Projekte: Überblick

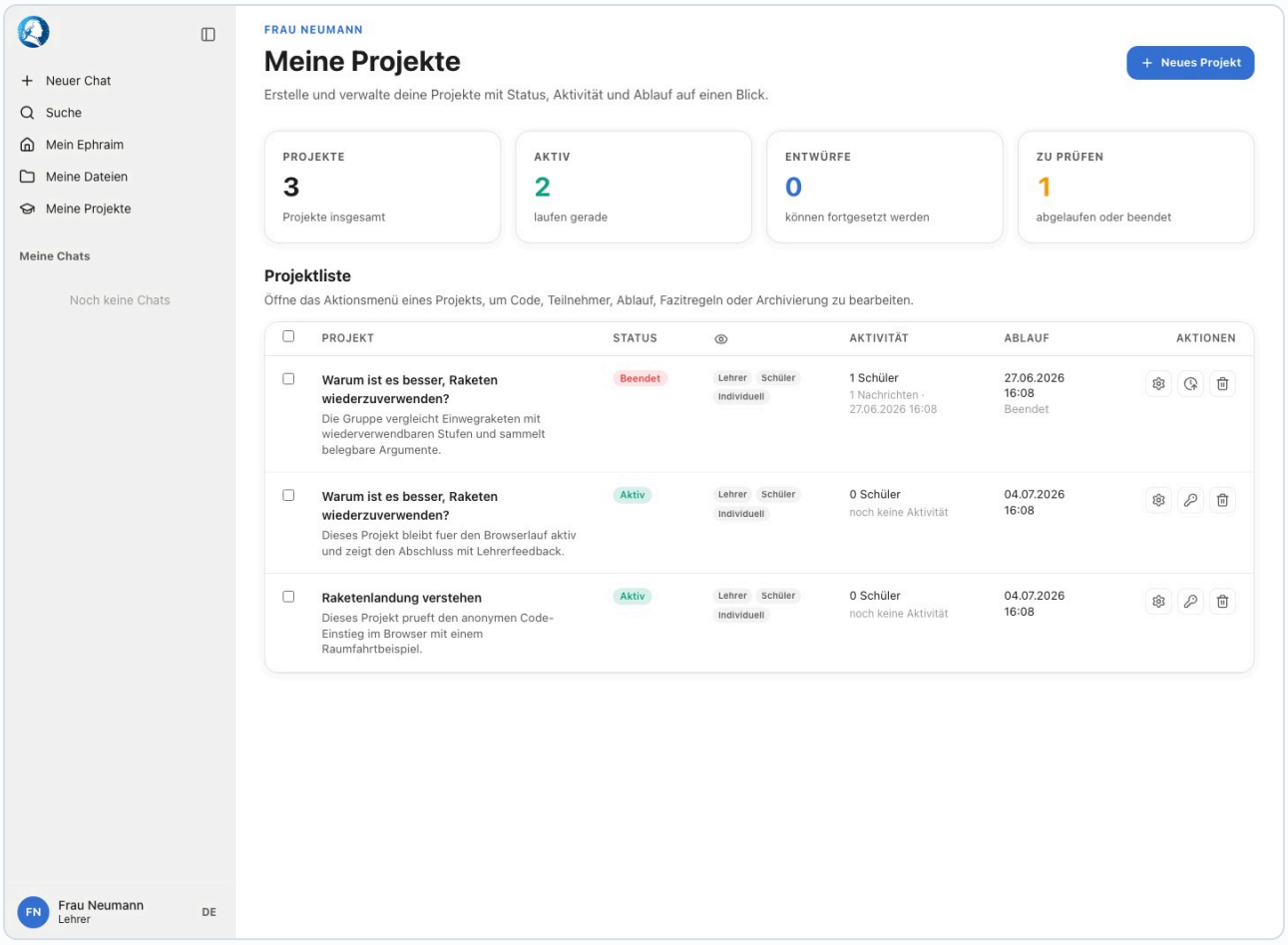
> **Projekte sind vorbereitete Lernräume. Eine Lehrkraft legt Auftrag, Material, Teilnahmeform, Datenschutzrahmen und Abschlussregeln fest; Schülerinnen und Schüler arbeiten danach in einem projektbezogenen Chat.**

Stand: Juni 2026

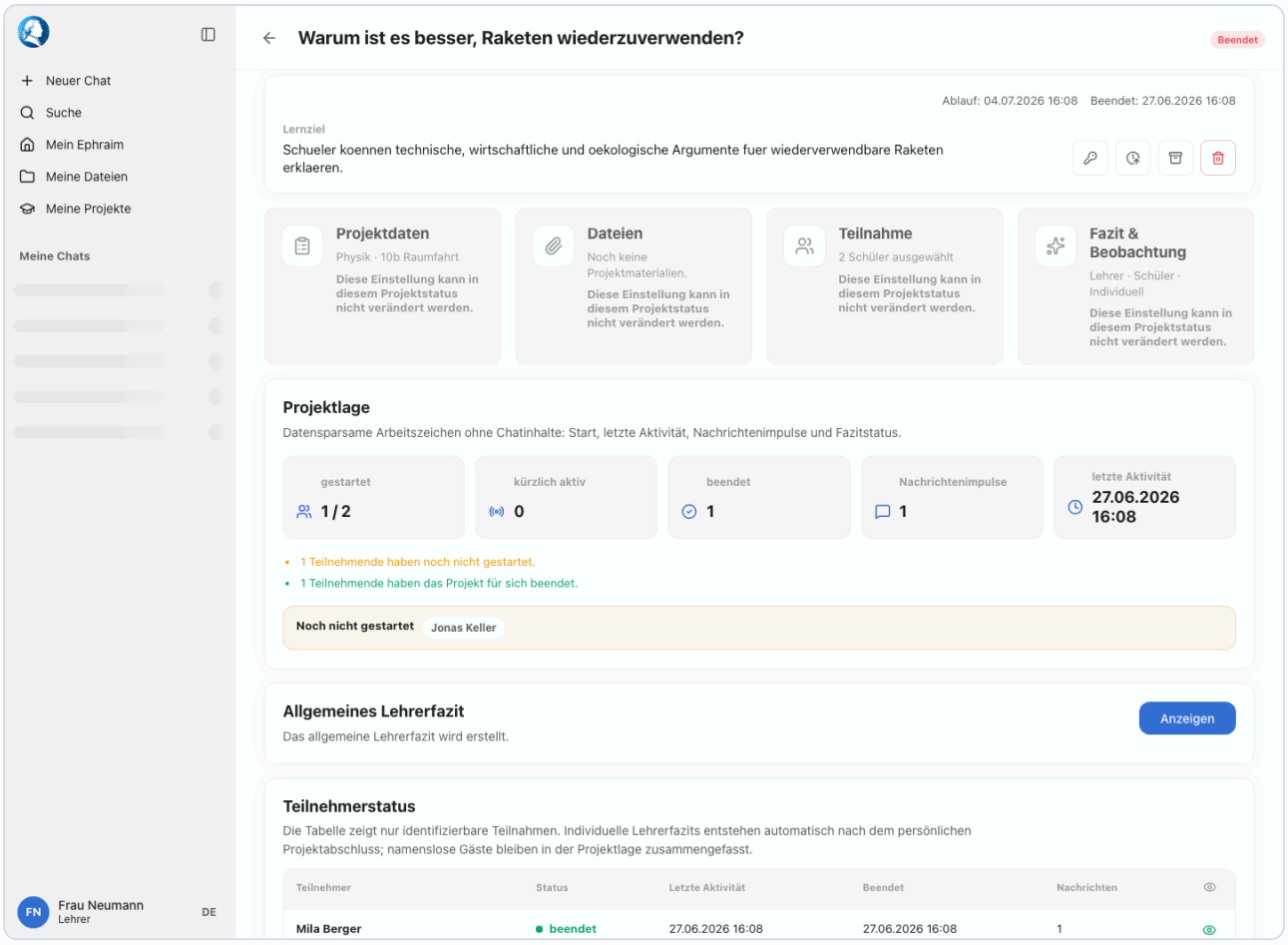
4.12.1 Was ein Projekt ist

Ein Projekt ist kein normaler persönlicher Chat. Es ist ein begrenzter Arbeitsraum mit einem Auftrag der Lehrkraft, einer Laufzeit, Projektmaterialien und festen Regeln für Teilnahme und Abschluss. Ephraim behandelt den Projektchat deshalb anders als den persönlichen Chat: Der Projektauftrag steht im Vordergrund, private Erinnerungen werden nicht als Gedächtniswerkzeug zugeschaltet und Projektmaterialien werden nur nach Freigabe eingebunden.

> **Kernidee:** Projekte trennen private KI-Nutzung von schulisch freigegebenen Arbeitsräumen. Was im Projekt entsteht, gehört zum Projektkontext; was im persönlichen Konto privat bleibt, wird nicht automatisch Projektmaterial.



Die Lehrkraftübersicht zeigt Entwürfe, laufende Projekte, Fristen und die nächsten Aktionen. Von hier führt die Detailseite zu Projektlage, Teilnahme und Fazitregeln.



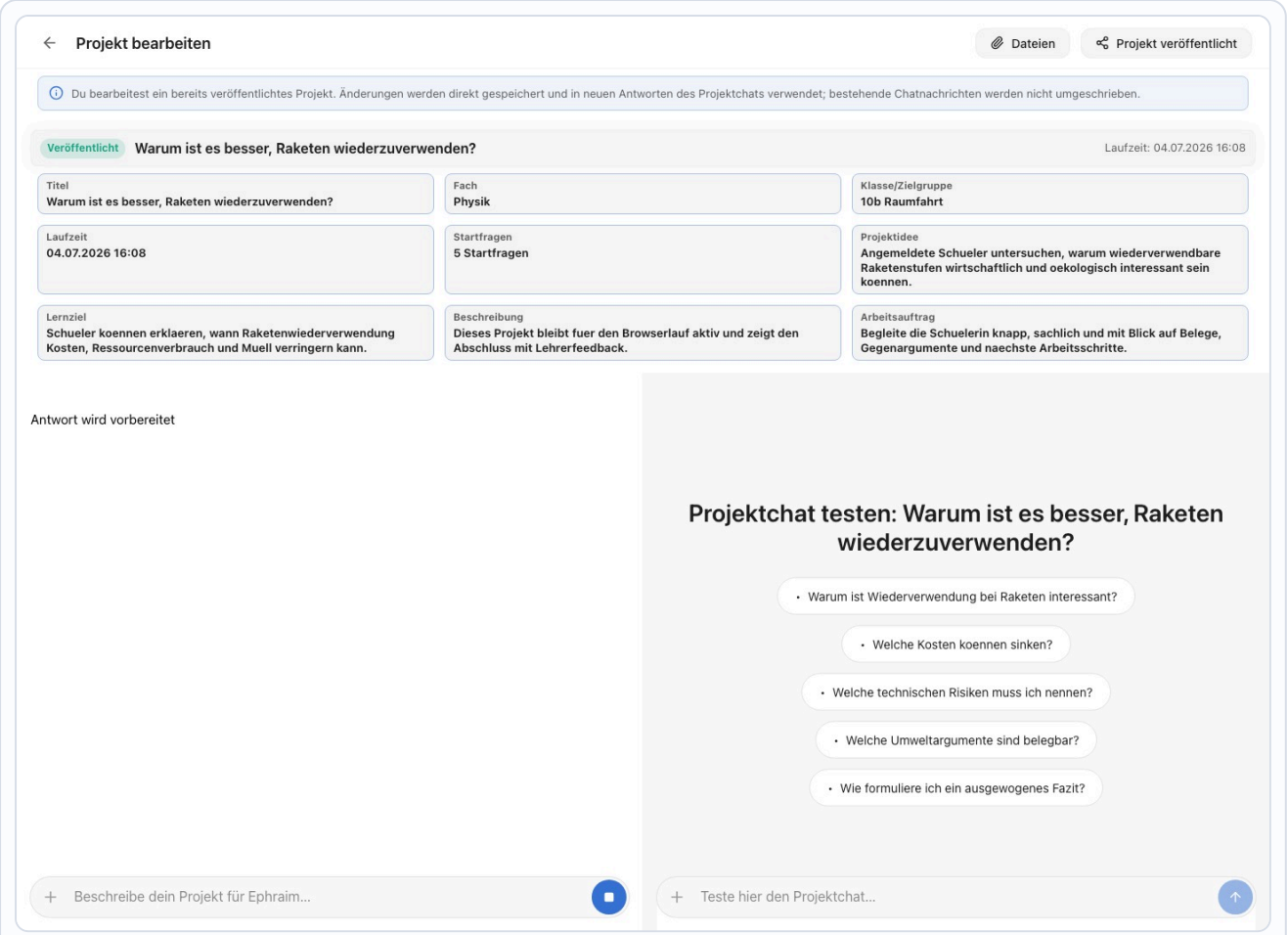
Die Projekt-Detailseite bündelt Auftrag, Materialien, Teilnahme, Fazit & Beobachtung, Projektlage und Teilnehmerstatus an einem Ort.

4.12.2 Was ein Projekt regelt

Thema	Klare Frage	Konkrete Antwort
Zugang und Rollen	Wer kommt in ein Projekt?	Entweder Projektcode-Gäste oder ausdrücklich ausgewählte Schülerkonten.
Materialien und Kontext	Was erhält der Projektchat als Quelle?	Projektkopien und indexierte Ausschnitte freigegebener Materialien.
Projektchat	Warum antwortet er anders als ein normaler Chat?	Er kombiniert Projektauftrag, Verlauf, Sprache, Basiswissen und Projektmaterial.
Abschluss und Fazit	Was passiert nach der Bearbeitung?	Der Teilnehmerabschluss sperrt neue Projektbeiträge und startet konfigurierte Fazitprozesse.
Datenschutz	Wo verläuft die Grenze?	Privates Konto, Projekt und Gastzugang bleiben getrennte Kontexte.

4.12.3 Projekt erstellen und veröffentlichen

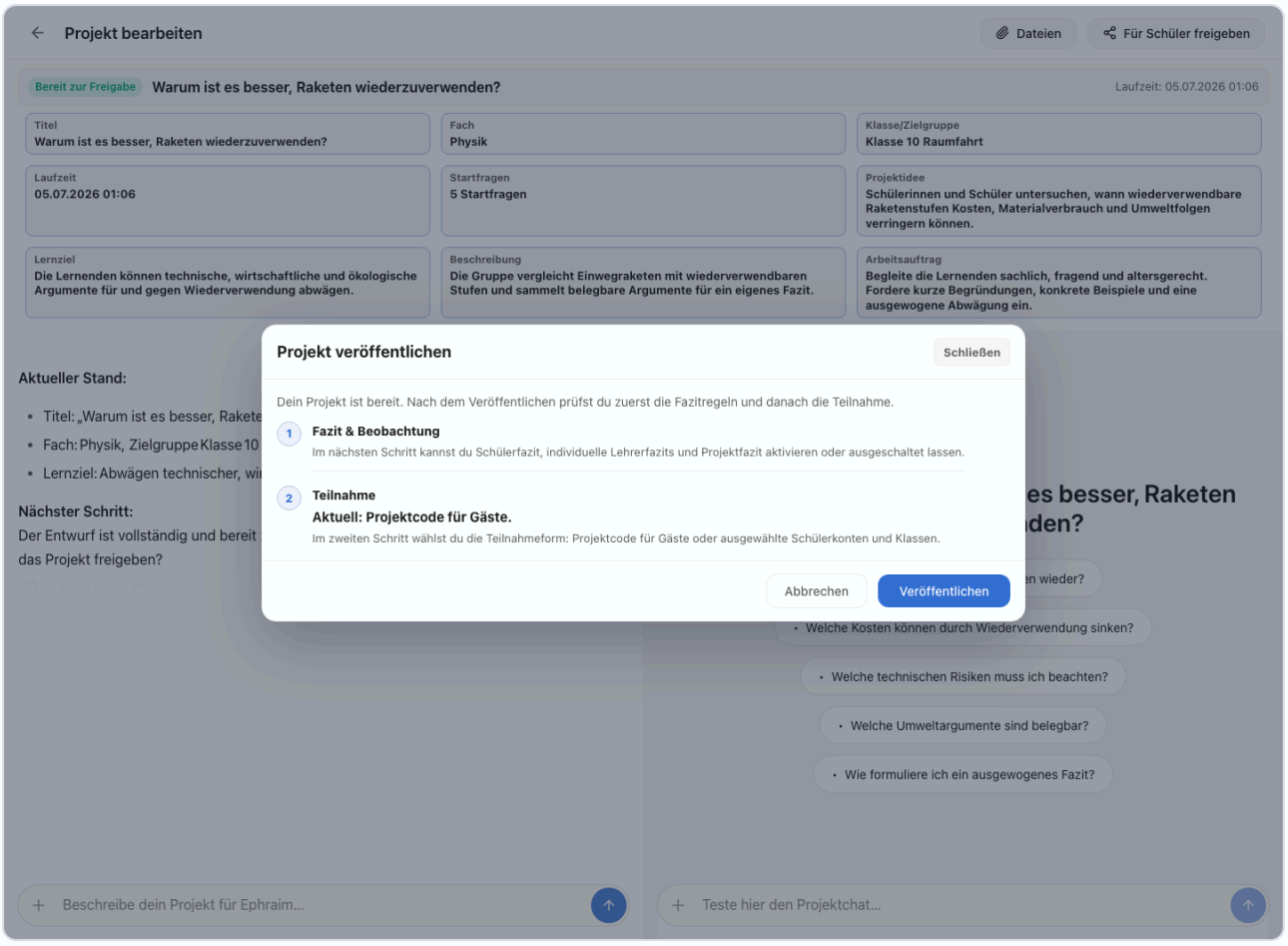
Lehrkräfte erstellen ein Projekt im Projektbuilder. Dort entstehen Auftrag, Zielgruppe, Laufzeit, Projektmaterialien und ein Testchat, mit dem die Lehrkraft die spätere Projektantwort prüft. Solange das Projekt ein Entwurf ist, arbeitet noch niemand aus der Klasse darin.



Im Projektbuilder bearbeitet die Lehrkraft Auftrag, Zielgruppe, Laufzeit, Projektmaterialien und den Testchat, bevor das Projekt freigegeben wird.

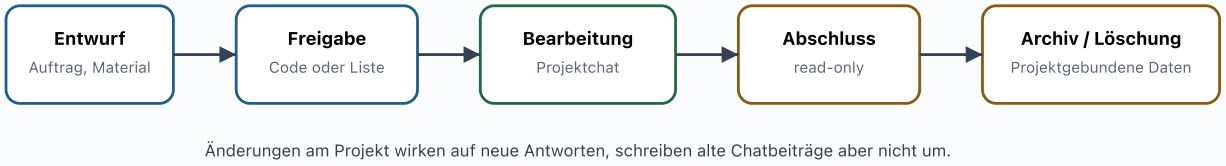
Die Freigabe ist die Schwelle vom Entwurf zum echten Lernraum. Wenn der Entwurf vollständig ist, wird die Schaltfläche „Für Schüler freigeben“ aktiv. Die Lehrkraft kann im Builder-Chat auch einen sehr kurzen, eindeutigen Befehl wie „Projekt veröffentlichen“ schreiben. Beide Wege öffnen denselben Bestätigungsdialog; sie veröffentlichen das Projekt nicht unbemerkt im Hintergrund.

Der Dialog zeigt die nächsten Schritte in der richtigen Reihenfolge: zuerst Fazit & Beobachtung, danach Teilnahme. Erst wenn die Lehrkraft im Dialog bestätigt, wird das Projekt veröffentlicht und Ephraim öffnet die bestehenden Einstellungsdialoge, damit Fazitregeln und Teilnahmeform vor dem Start bewusst geprüft werden.



Vor der Veröffentlichung zeigt Ephraim, dass zuerst die Fazitregeln und danach die Teilnahme geprüft werden. Der Dialog ist derselbe, egal ob die Lehrkraft die Schaltfläche nutzt oder einen eindeutigen Veröffentlichungsbefehl im Builder-Chat schreibt.

4.12.4 Lebenszyklus



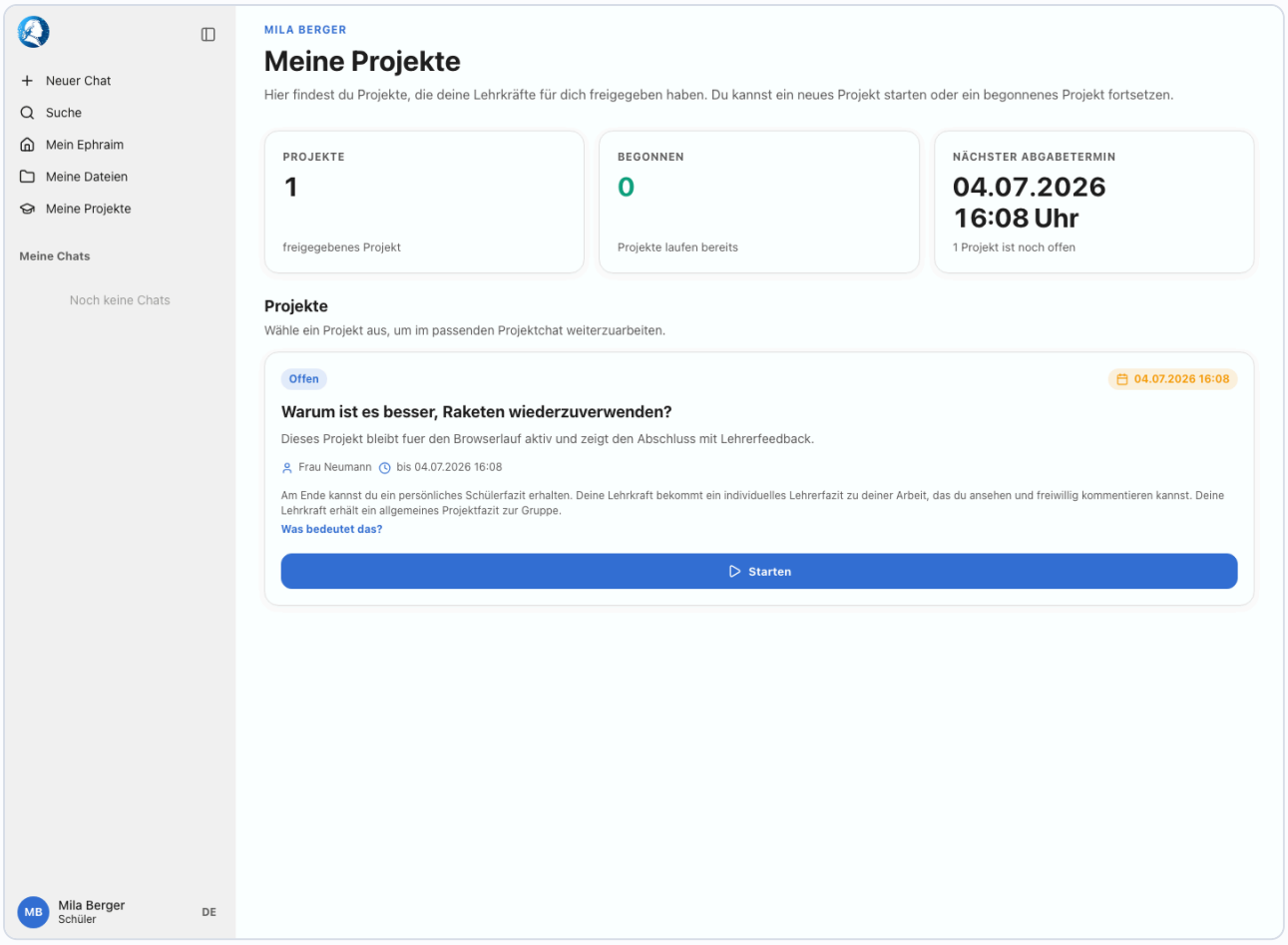
Änderungen am Projekt wirken auf neue Antworten, schreiben alte Chatbeiträge aber nicht um.

Der Projektlebenszyklus beginnt beim Entwurf, führt über Freigabe und Bearbeitung zum Abschluss und endet organisatorisch in Archivierung oder Löschung.

4.12.5 Die wichtigste Entscheidung

Die zentrale Projektentscheidung ist die Teilnahmeform. Beim Projektcode entsteht eine projektgebundene Gastsitzung ohne normales Konto. Bei ausgewählten Schülerkonten startet ein bestehendes Konto ein freigegebenes Projekt. Diese Entscheidung bestimmt, welche Einstellungen sichtbar sind, wie der Wiedereinstieg erfolgt und welche personenbezogenen Stammdaten überhaupt Teil der Arbeit sind.

Angemeldete Schüler setzen einen begonnenen Projektchat später im selben Projekt fort. Projektcode-Gäste arbeiten dagegen nur innerhalb ihrer aktuellen temporären Gastidentität. Beide Varianten halten den Projektchat vom normalen Chatverlauf getrennt; der eigene Datenexport enthält ihn trotzdem mit Projektbezug.



Ausgewählte Schülerkonten starten freigegebene Projekte in „Meine Projekte“. Der Hinweis auf Schülerfazit, individuelles Lehrerfazit und allgemeines Projektfazit steht direkt auf der Projektkarte.

4.12.6 Die wichtigste Grenze

Ein Projekt ist kein Freibrief zum Lesen privater Konten und auch kein Freibrief zum Lesen einzelner Projektchats. Lehrkräfte öffnen nicht den Chatverlauf einer Schülerin oder eines Schülers. Sie sehen Projektstatus, freigegebene Projektmaterialien, Basiszähler der Projektarbeit und die im Projekt aktivierten Fazit- oder Beobachtungsergebnisse. Normale private Chats, Projektchat-Rohtexte, private Dateien, private Kalender und persönliche Erinnerungen bleiben außerhalb des Lehrkraftzugriffs.

Konkret bedeutet das: Die Lehrkraft steuert Auftrag, Material, Teilnehmende und Abschlussregeln. Die Auswertung arbeitet mit reduzierten Statusdaten wie Start, Aktivität, Abschluss, Nachrichtenimpulsen, Fazitstatus und optional erzeugten individuellen Lehrerfazits je Schüler. Wenn ein individuelles Lehrerfazit entsteht, kann die betroffene Schülerin oder der betroffene Schüler dasselbe Fazit sehen und freiwillig kommentieren. Vollständige Schülerchats werden daraus nicht sichtbar.

Die Detailartikel der Projektgruppe erklären Zugang, Materialien, Projektchat, Abschluss und Datenschutz getrennt. Für die allgemeine private Nutzung siehe Mein Ephraim und Chat: Überblick.

Quelle: web/manuals/projekte/index.html

4.13 Projekte: Zugang und Rollen

> Projektteilnahme erfolgt entweder über einen sechsstelligen Projektcode oder über ausdrücklich ausgewählte Schülerkonten. Die beiden Wege erzeugen unterschiedliche Identitäten und unterschiedliche Einstellungsbereiche.

Stand: Juni 2026

4.13.1 Zwei Zugangsarten

Ephraim trennt Projektcode-Gäste von normalen Schülerkonten. Ein Projektcode ist für Projekte im Gastmodus bestimmt. Ein Projekt mit ausgewählten Schülerkonten wird über das angemeldete Konto gestartet; der Projektcode-Einstieg lehnt diesen Modus ab.

Merkmal	Projektcode-Gast	Ausgewähltes Schülerkonto
Identität	Neue projektgebundene Gastsitzung	Bestehendes Ephraim-Konto
Freigabe	Gültiger sechsstelliger Code, veröffentlichtes Projekt, nicht abgelaufen	Eintrag in der freigegebenen Teilnehmerliste
Kontobereich	Anzeige, Sprache und Projektzugang	Normale Kontoeinstellungen plus Projektbereich
Vault	Kein persönlicher Vault; verschlüsselte projektgebundene Chat-Ablage, nicht als Admin-Leseraum	Persönlicher Vault bleibt privat und getrennt

4.13.2 Projektcode-Gäste

Der Projektcode-Einstieg prüft vier Bedingungen: Der Code besteht aus sechs Ziffern, das Projekt ist veröffentlicht, es ist weder archiviert noch beendet, und seine Laufzeit ist nicht abgelaufen. Danach erzeugt Ephraim ein Gastkonto mit Schülerrolle und bindet dieses Gastkonto an genau dieses Projekt.

Dieses Gastkonto hat keinen temporären Vault im Sinn eines normalen Ephraim-Kontos. Ephraim legt für den Projektzugang stattdessen eine verschlüsselte, projektgebundene Chat-Ablage an. Der dafür genutzte Schlüssel gehört zur aktiven Projektsitzung und dient nur dazu, Projektchats verschlüsselt zu speichern und laufende KI-Antworten sicher weiterzuverarbeiten. Daraus entstehen kein Passwort, keine Zwei-Faktor-Authentifizierung, keine Vault-Wiederherstellung, kein persönliches Gedächtnis und keine privaten Wissensquellen.

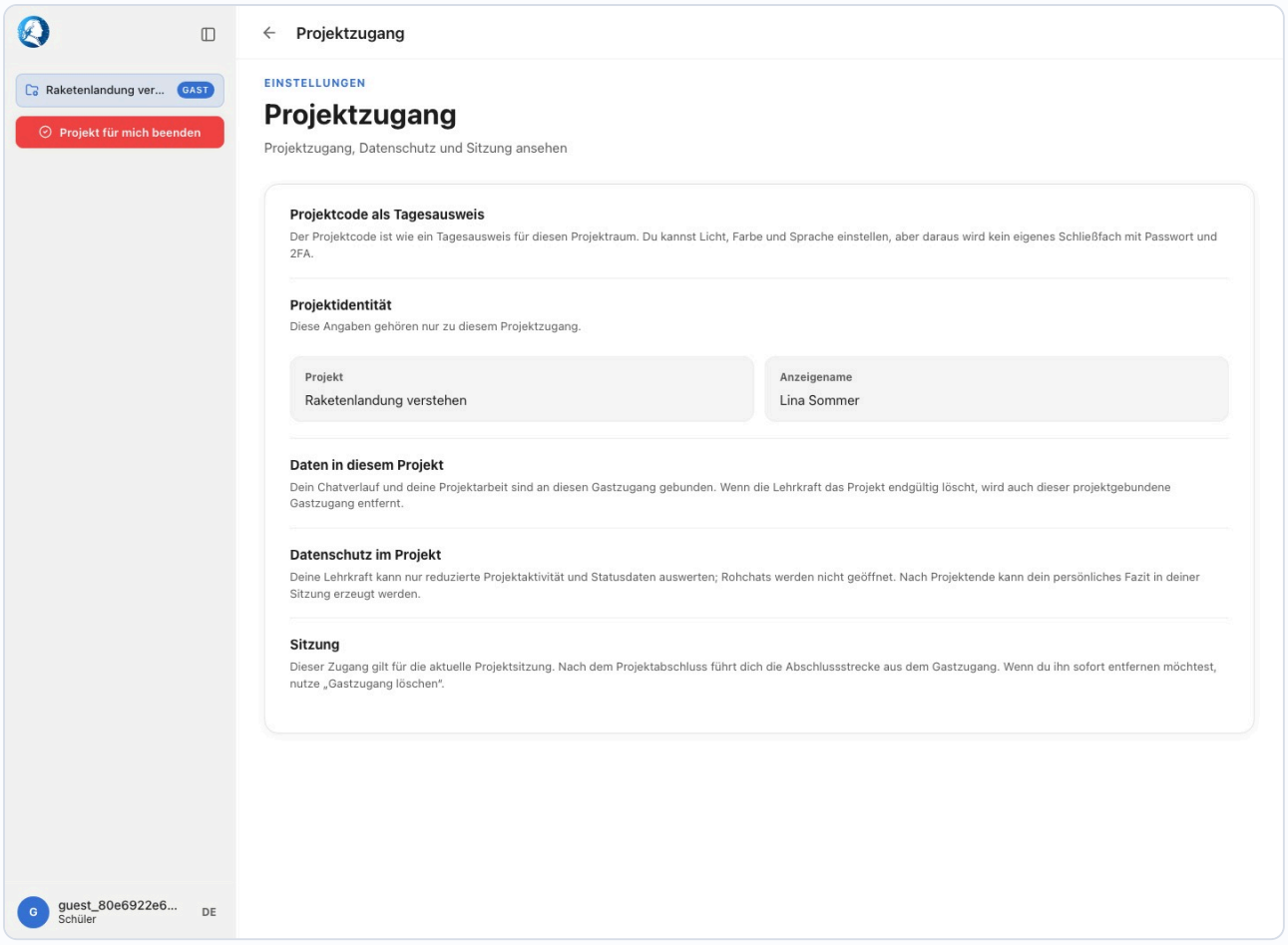
In der Datenbank ist diese Chat-Ablage nicht schwächer geschützt als normale Vault-Chats. Der Rohchat liegt dort nicht als Klartext, sondern verschlüsselt. Der Unterschied liegt im Lebenszyklus des Schlüssels: Beim normalen Konto wird der persönliche Vault über das Passwort und die Wiederherstellungsarchitektur geöffnet; beim Projektcode-Gast gehört der Schlüssel zum projektgebundenen Gastzugang.

Diese projektgebundene Ablage ist auch kein Leseraum für Lehrkräfte oder Ephraim-Administratoren. Sie können den Rohchat eines Projektcode-Gasts nicht öffnen. Für Lehrkräfte sichtbar werden nur die vorher angekündigten Projektinformationen: Teilnahme, reduzierte Aktivitätszeichen und, falls vor Projektbeginn aktiviert, abgeleitete Fazit- oder Monitoring-Ergebnisse mit den zugehörigen Transparenzzuständen.

Als Bild: Ein normales Konto hat ein persönliches Schließfach. Ein Projektcode-Gast sitzt an einem geschützten Arbeitsplatz im Projektraum. Die Arbeit ist verschlüsselt abgelegt, gehört aber zu diesem Projektzugang und nicht zu einem privaten Konto.

Die Lehrkraft legt zusätzlich fest, ob ein Anzeigename abgefragt wird. Die Namensregel hat drei Zustände: aus, optional oder verpflichtend. Bei aktivierter Eindeutigkeit weist Ephraim einen bereits verwendeten Anzeigenamen im selben Projekt zurück.

Wenn für das Projekt ein persönliches Schülerfazit oder individuelle Lehrerfazits aktiv sind, zeigt der Projektcode-Einstieg vor dem Beitritt einen Hinweistext. Der Gast tritt erst bei, nachdem dieser Hinweis bestätigt wurde. Der Hinweis nennt das persönliche Fazit, die reduzierte Auswertung für individuelle und allgemeine Lehrerfazits, die Möglichkeit namentlicher Hinweise, die Anonymisierung des allgemeinen Lehrerfazits und die Grenze, dass Rohchats nicht geöffnet werden.



Projektcode-Gäste sehen ihre Projektidentität, den Datenschutzrahmen und den Sitzungshinweis. Datenexport und Gastzugang-Löschung gehören zu den Betroffenenrechten dieses Zugangs; Zwei-Faktor-Authentifizierung und ein vollständiges Profil gehören nicht dazu.

4.13.3 Ausgewählte Schülerkonten

Im Modus mit ausgewählten Schülerkonten wählt die Lehrkraft konkrete Schülerinnen, Schüler oder Klassen aus. Ein angemeldetes Schülerkonto sieht freigegebene Projekte in seinem Projektbereich und startet die Teilnahme dort. Beim Start legt Ephraim die Projektmitgliedschaft an und speichert den aktiven Projektkontext in der Sitzung.

Dieser Modus eignet sich für Projekte, bei denen bestehende schulische Konten, Klassenbezüge und persönlicher Wiedereinstieg wichtig sind. Der private Vault dieses Kontos bleibt trotzdem nicht automatisch Projektinhalt.

Der Code-Hinweisdialog gehört zum anonymen Projektcode-Einstieg. Ausgewählte Schülerkonten starten das freigegebene Projekt aus ihrem Projektbereich. Die Transparenz über Fazit- und Beobachtungsregeln entsteht hier über Projektauftrag, Unterrichtsrahmen und die später sichtbaren Abschlussfunktionen im Projektchat. Wenn ein individuelles Lehrerfazit erzeugt wurde, kann die betroffene Person dasselbe Fazit ansehen und freiwillig Stellung nehmen.

4.13.4 Rollen

Rolle	Projektrechte	Grenze
Lehrkraft	Erstellt, veröffentlicht, schließt, archiviert und verwaltet eigene Projekte.	Öffnet keine Schüler-Projektchats und keine privaten Chats.
Teilnehmende	Arbeiten im Projektchat, nutzen freigegebene Materialien und schließen ihren Stand ab.	Sehen keine Chats anderer Teilnehmender.
Projektcode-Gast	Arbeitet ausschließlich im Projekt, das zum Code gehört.	Hat keinen vollständigen Account.
Administration	Verwaltet Systemrollen, Quoten, Klassen und technische Sicherheit.	Verwendet Projekte nicht als Klartextzugang zu fremden Vaults.

Der Zugang entscheidet über Identität, Einstellungen und Wiedereinstieg. Der Projektauftrag entscheidet über die fachliche Arbeit.

Quelle: web/manuals/projekte-zugang-und-rollen/index.html

4.14 Projekte: Materialien und Kontext

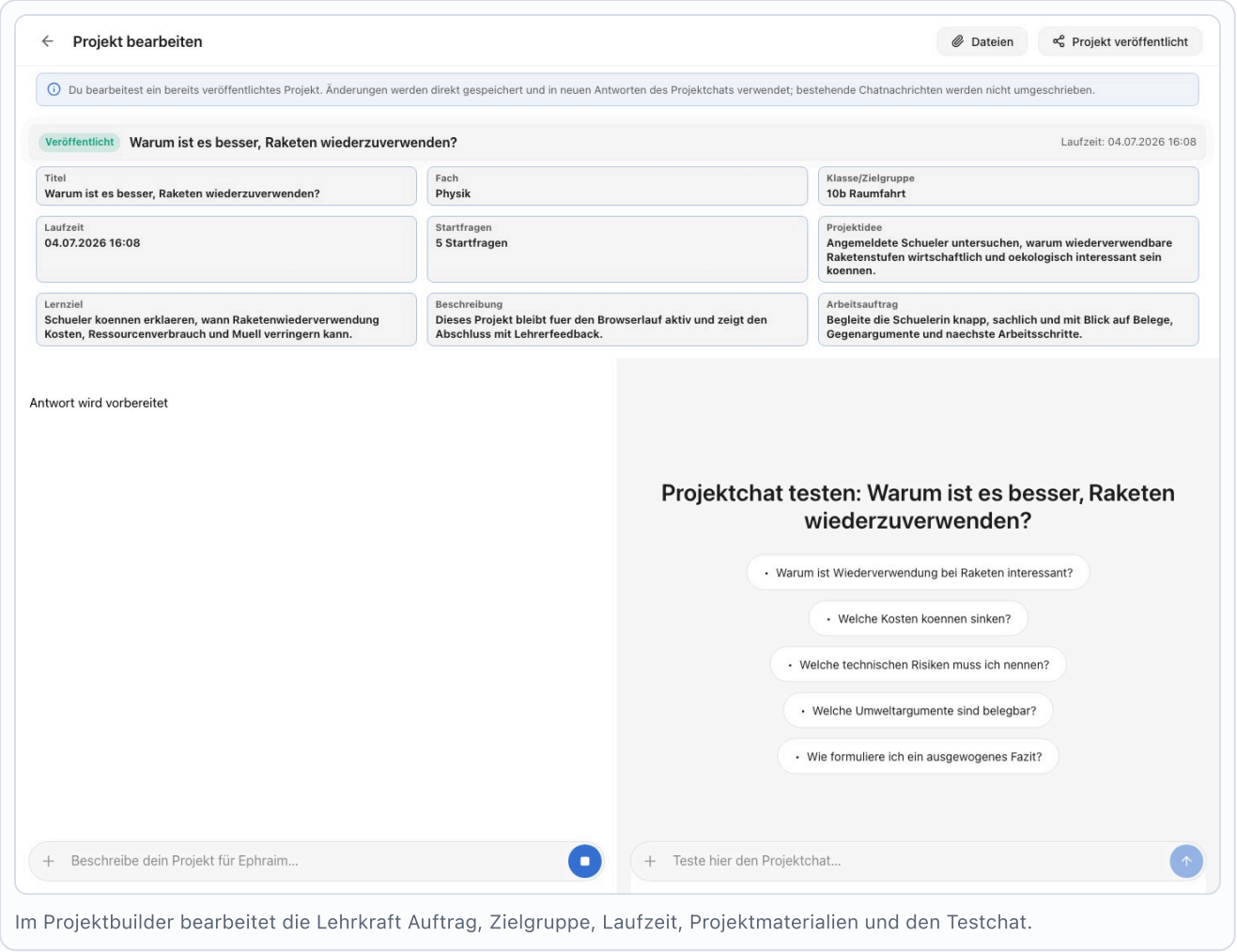
> Projektmaterialien sind bewusst freigegebene Kopien oder Uploads. Ephraim nutzt sie als Projektquelle, ohne den privaten Dateispeicher der Lehrkraft oder der Lernenden zu öffnen.

Stand: Juni 2026

4.14.1 Projektmaterial ist eine eigene Kopie

Eine Lehrkraft fügt Material auf zwei Wegen hinzu: durch neuen Upload oder durch Kopie aus „Meine Dateien“. Bei einer Kopie entschlüsselt Ephraim die private Quelldatei in der laufenden Sitzung der Lehrkraft, verschlüsselt sie sofort mit einem Projektschlüssel neu und speichert sie als Projektdatei. Die private Originaldatei bleibt im persönlichen Speicher der Lehrkraft.

> **Klare Grenze:** Eine private Datei wird nicht durch ihre Existenz im Konto zum Projektmaterial. Erst die bewusste Aufnahme in die Projektmappe erzeugt die projektbezogene Kopie.



Im Projektbuilder bearbeitet die Lehrkraft Auftrag, Zielgruppe, Laufzeit, Projektmaterialien und den Testchat.

4.14.2 Status der Verarbeitung

Nach dem Upload oder Kopieren durchläuft Material eine Verarbeitungskette. Die Oberfläche unterscheidet hochgeladen, Text wird gelesen, Text gelesen, Suchindex wird erstellt, im Projektchat nutzbar und Indexierung fehlgeschlagen. Für die semantische Suche zählt der Zustand „im Projektchat nutzbar“: Erst dann liegen verschlüsselte Textabschnitte und passende Embeddings vor.

Die Embeddings werden über den lokalen SGLang-Dienst `school-ui-embedding` berechnet. Dafür sieht die Spark den freigegebenen Textabschnitt nur während des aktuellen Verarbeitungsschritts. Sie speichert weder Projektmaterial-Klartexte noch Embedding-Vektoren dauerhaft; die dauerhafte Ablage liegt verschlüsselt bei Ephraim.

4.14.3 Wie Material in eine Antwort gelangt

Ephraim sendet nicht automatisch jede Projektdatei vollständig an das Modell. Bei einer neuen Frage berechnet Ephraim eine Suchanfrage, vergleicht sie mit den eingebetteten Materialabschnitten und übernimmt nur die passendsten Ausschnitte bis zur eingestellten Kontextgrenze. Der Projektchat erhält diese Ausschnitte mit Dateiname und Abschnittsnummer.

Wenn eine Nutzerin oder ein Nutzer eine konkrete Projektdatei auswählt, begrenzt Ephraim die Suche auf diese Datei. Ohne konkrete Frage fügt Ephraim keinen sinnlosen Materialblock ein, sondern fordert eine Frage zum Material ein.

4.14.4 Sichtbarkeit und Download

Projektmaterial gehört zum Projekt. Teilnehmende mit gültiger Projektmitgliedschaft lesen die Projektmaterialliste und laden freigegebene Projektdateien herunter. Ein Zugriff auf private Originaldateien entsteht daraus nicht. Entfernt die Lehrkraft ein Material aus der Projektmappe, wird es als Projektmaterial

nicht mehr angezeigt und nicht mehr für neue Kontextsuche verwendet.

4.14.5 Datei- und Speicherregeln

Projektmaterialien unterliegen denselben grundlegenden Dateiregeln wie andere Uploads: erlaubte Dateitypen, Größenlimits, Quoten, Sicherheitsprüfung und verschlüsselte Speicherung. Projektdateien zählen zur systemweiten Belegung. Eine abgelehnte Datei wird nicht als Projektmaterial verfügbar.

Projektmaterial ist bewusst freigegebener Kontext. Es ist weder privater Dateispeicher noch automatischer Volltextdump an die KI.

Quelle: web/manuals/projekte-materialien-und-kontext/index.html

4.15 Projekte: Projektchat

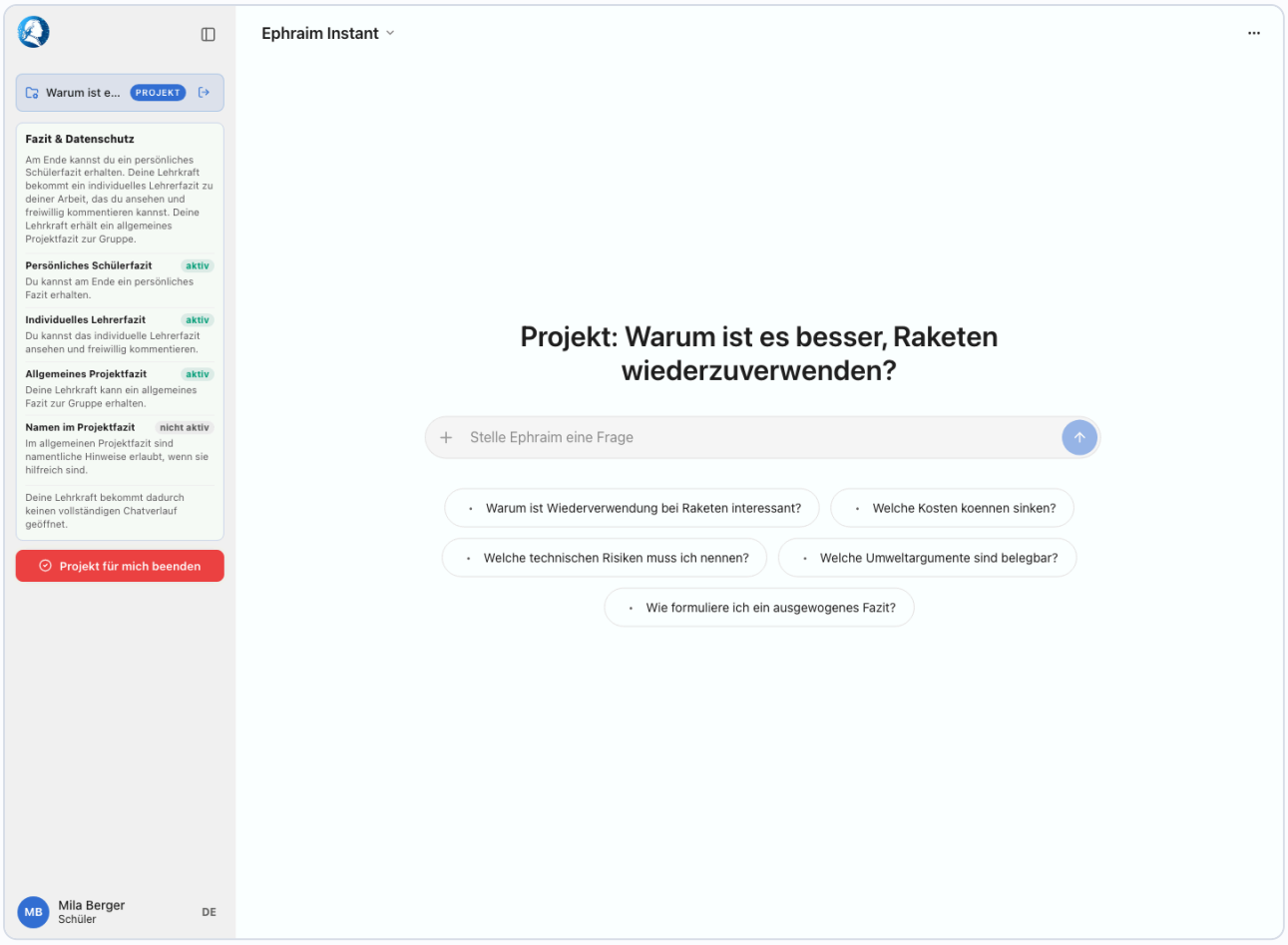
> **Der Projektchat antwortet im Rahmen des Projekts. Er kombiniert Projektauftrag, bisherigen Projektverlauf, freigegebene Materialien und zentral freigegebenes Basiswissen.**

Stand: Juni 2026

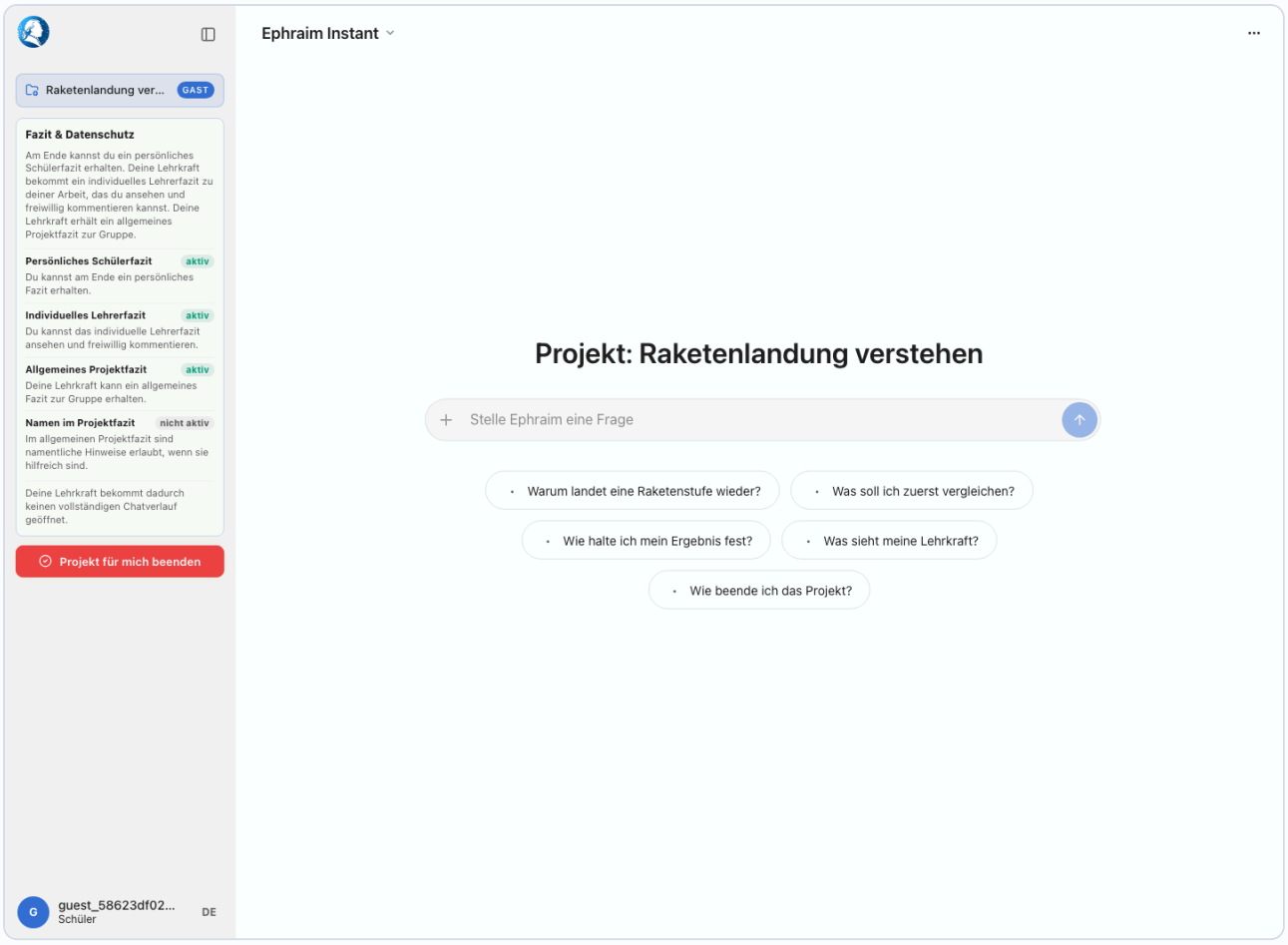
4.15.1 Der Systemrahmen

Jede Projektantwort erhält einen festen Basisrahmen für Schüler-Projektchats und den konkreten Projektauftrag der Lehrkraft. Der Projektauftrag ist nicht nur ein sichtbarer Beschreibungstext, sondern Teil des Systemkontexts für neue Antworten. Der Chat bleibt dadurch auf Aufgabe, Zielgruppe und Material gerichtet.

Im aktiven Projektchat ist Ephraim bewusst enger geführt als im normalen Chat. Neue Chats, Suche, Meine Dateien, Mein Ephraim und Meine Projekte gehören nicht zur Projekt-Shell. Angemeldete Schüler verlassen den Projektarbeitsraum über „Projekt verlassen“. Projektcode-Gäste behalten nur den reduzierten Zugang zu Projektinformationen, Sprache, Datenschutz, Export und Gastzugang-Löschung. Die Seitenleiste zeigt im Projektchat dauerhaft, welche Fazit- und Datenschutzregeln für dieses Projekt gelten.

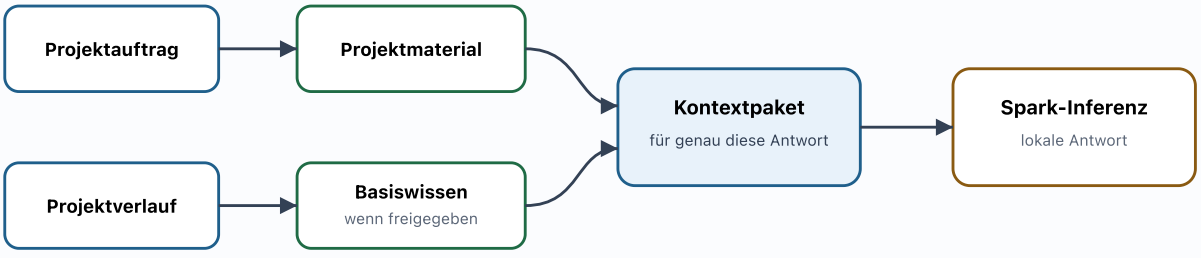


Im Projektchat ist die normale Navigation ausgeblendet. Die Seitenleiste hält den Projektstatus und die geltenden Fazit- und Datenschutzregeln sichtbar; der Weg aus der Arbeit führt über den persönlichen Projektabschluss.



Projektcode-Gäste sehen dieselbe Transparenz im Arbeitsraum. Der Gaststatus bleibt in der Seitenleiste erkennbar, während die Hinweise erklären, welche Auswertungen aktiv sind und dass kein vollständiger Chatverlauf für die Lehrkraft geöffnet wird.

4.15.2 Was im Kontextpaket steht



Der Projektchat bündelt nur freigegebene Projektbausteine. Private Kontodaten, private Erinnerungen und fremde Projektchats gehören nicht in dieses Kontextpaket.

Baustein	Herkunft	Grenze
Projektauftrag	Von der Lehrkraft im Projektbuilder oder in den Projekteinstellungen festgelegt	Wirkt auf neue Antworten, verändert alte Chatbeiträge nicht.
Projektverlauf	Bisherige Nachrichten dieses Projektchats	Andere Projektchats, Lehrkraftansichten und private Chats werden nicht geöffnet.
Projektmaterial	Passende Ausschnitte aus freigegebenen Projektdateien	Nur indexierte, lesbare und berechtigte Projektdateien.
Basiswissen	Zentral freigegebene Quellen, Wetter, Kalender oder Wikipedia-Auszüge	Nur wenn die Quelle aktiviert und für den Kontext freigegeben ist.

4.15.3 Was nicht zugeschaltet wird

Der Projektchat aktiviert keine persönlichen Memory-Werkzeuge. Private Kalender eines Nutzerkontos werden nur im normalen persönlichen Chat berücksichtigt. Private Dateien werden erst dann Projektkontext, wenn die Lehrkraft eine projektbezogene Kopie als Projektmaterial erstellt hat. Die Anrede aus dem Konto oder dem Projektanzeigenamen dient der Bedienung und ersetzt keine Projektfreigabe.

4.15.4 Sichtbarkeit für Lehrkräfte

Lehrkräfte lesen keine Projektchat-Rohtexte. Es gibt keine Projektfunktion, die den vollständigen Chatverlauf einer teilnehmenden Person für die Lehrkraft öffnet. Der Projektchat bleibt ein eigener, verschlüsselter Chatverlauf der teilnehmenden Person. Der Projektbesitz der Lehrkraft entsperrt diesen Verlauf nicht.

Sichtbar sind stattdessen Projektmetadaten und reduzierte Auswertungen: ob jemand gestartet hat, ob zuletzt Aktivität vorlag, wie viele Nachrichtenimpulse gezählt wurden, ob ein Teilnehmerabschluss vorliegt und welcher Fazit- oder Rückmeldungsstatus erreicht ist. Wenn ein individuelles Lehrerfazit je Schüler erzeugt wurde, sieht die Lehrkraft dieses verdichtete Ergebnis, nicht den zugrunde liegenden Rohchat.

4.15.5 Fortsetzen und Export

Ein angemeldeter Schüler hat pro Projekt genau einen Projektchat. Wenn das Projekt später über „Meine Projekte“ fortgesetzt wird, lädt Ephraim diesen bestehenden Projektchat und legt keinen zweiten Chat an. Ein Projektcode-Gast kann einen alten Gastchat dagegen nicht über eine spätere neue Gastsession fortsetzen, weil der Gastzugang temporär ist.

Projektchats erscheinen nicht im normalen Chatverlauf und nicht in der normalen Suche. Sie bleiben aber eigene Daten der teilnehmenden Person. Der Datenexport eines Schülerkontos enthält die eigenen Projektchats mit Projektbezug. Ein Projektcode-Gast kann die Daten der aktuellen temporären Gastidentität exportieren, solange dieser Gastzugang besteht.

4.15.6 Änderungen während der Laufzeit

Ändert die Lehrkraft Auftrag oder Material, verwendet Ephraim den aktualisierten Projektkontext für neue Antworten. Bereits gespeicherte Nachrichten bleiben unverändert. Die Anwendung weist Teilnehmende auf geänderten Projektkontext hin, damit sie mit dem aktuellen Stand weiterarbeiten.

4.15.7 Internet und Spark

Die Spark ist kein Browser. Das Modell ruft keine Webseiten ab und führt keine Websuche aus. Wenn zentral freigegebenes Wetter, Wikipedia oder eine freigegebene Quelle in einer Projektantwort erscheint, stammt dieser Text aus dem Webserver-Kontext, den Ephraim vor der Inferenz vorbereitet hat. Die Modellberechnung selbst arbeitet lokal mit dem gelieferten Kontext.

Der Projektchat ist projektgebunden. Er ist kein persönliches Gedächtnis und kein freier Webbrowser. Abschluss und reduzierte Auswertung stehen in Abschluss und Fazit.

Quelle: web/manuals/projekte-projektchat/index.html

4.16 Projekte: Abschluss und Fazit

> Der Abschluss beendet die eigene Bearbeitung und macht danach die konfigurierten Fazitfunktionen sichtbar. Lehrkräfte lesen dabei keine Projektchats. Sie erhalten Statusdaten, individuelle Lehrerfazits zu einzelnen Teilnahmen und optional ein allgemeines Lehrerfazit zum Projekt, wenn diese Optionen vorab aktiviert wurden.

Stand: Juni 2026

4.16.1 Grundsatz: Abschluss ist nicht Chatfreigabe

Ein Projektabschluss gibt den Chatverlauf nicht an die Lehrkraft frei. Der Projektchat bleibt der Chat der teilnehmenden Person. Die Lehrkraft erhält keinen Button, keine Liste und keinen Export, mit dem sie Projektchat-Rohtexte einzelner Schülerinnen und Schüler öffnet.

Die Fazit- und Beobachtungsfunktionen arbeiten stattdessen mit ausdrücklich reduzierten Daten: Projektmetadaten, Aktivitätsimpulsen, Abschlussstatus, Fazitstatus und optional erzeugten individuellen Lehrerfazits. Vollständige Chatprotokolle, private Dateien, persönliche Erinnerungen und private Kalender bleiben außerhalb dieser Auswertung.

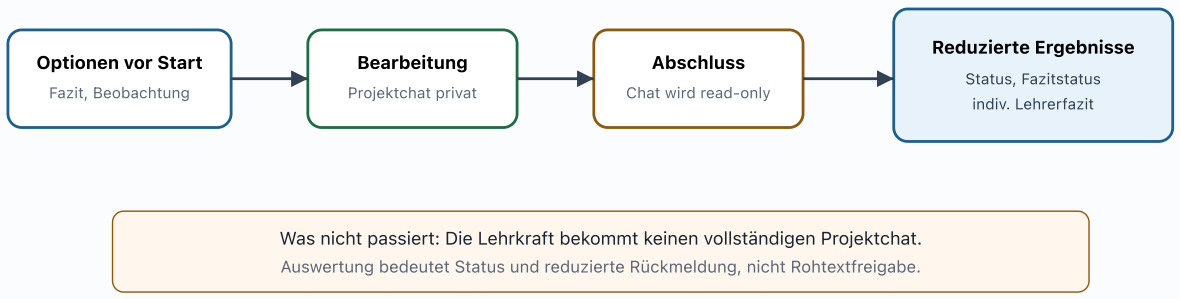
> **Merksatz:** Fazit bedeutet Reflexion und Statusauswertung, nicht Lehrkraftzugriff auf Schülerchats.

4.16.2 Die vier Optionen in „Fazit & Beobachtung“

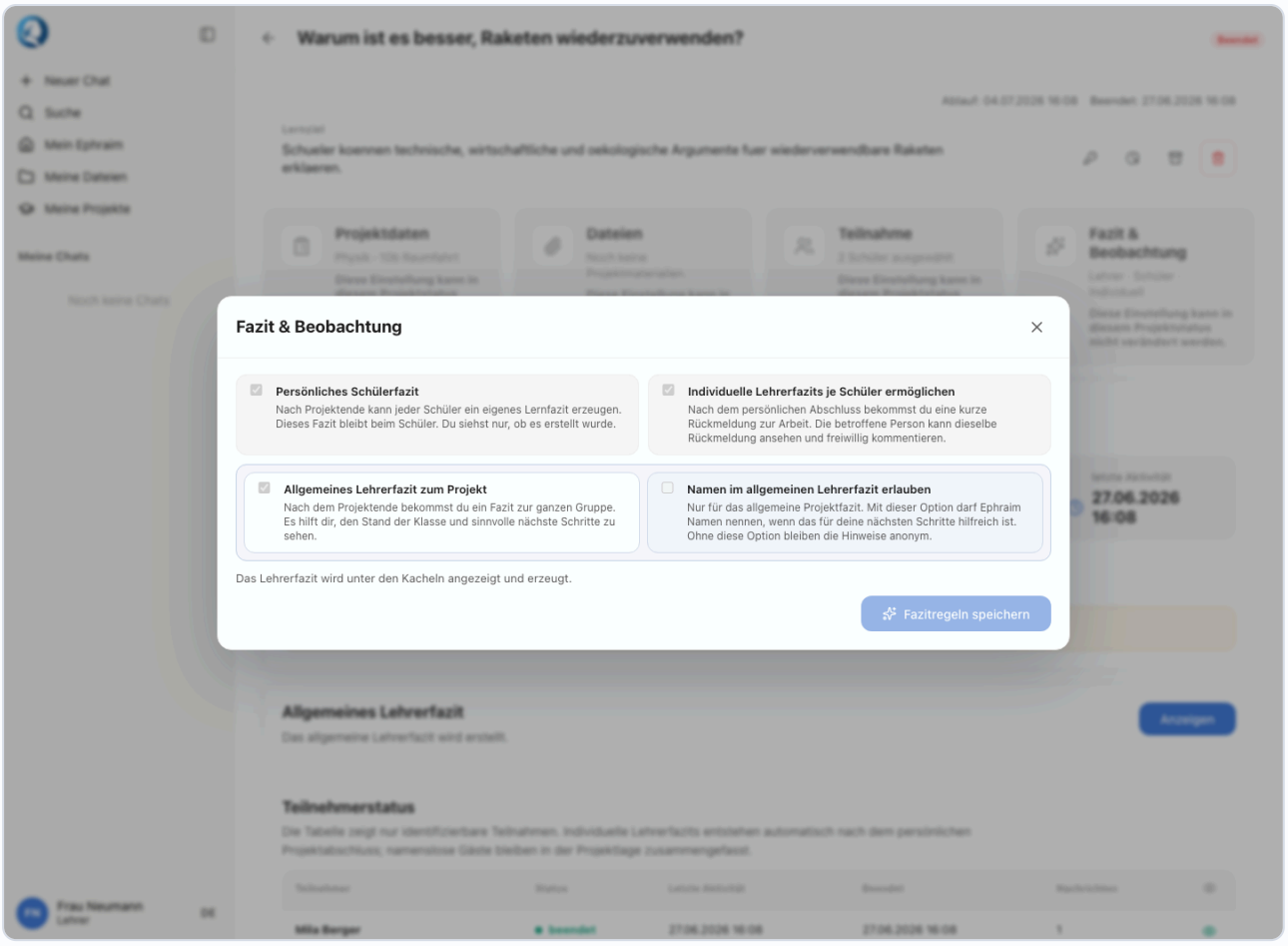
In den Projekteinstellungen legt die Lehrkraft die Abschlussregeln fest. Die Optionen sind voneinander abhängig und wirken erst für die weitere Projektarbeit.

Beim Veröffentlichen eines fertigen Projektentwurfs erinnert Ephraim die Lehrkraft an diese Stelle: Nach der Bestätigung öffnet sich zuerst „Fazit & Beobachtung“ und danach die Teilnahme. Dadurch werden die Rückmelderegeln vor dem Start noch einmal bewusst geprüft, statt nebenbei gesetzt zu werden.

Die Begriffe sind bewusst getrennt: Das **persönliche Schülerfazit** ist ein Lernrückblick für die teilnehmende Person. Das **individuelle Lehrerfazit je Schüler** ist eine reduzierte Rückmeldung zur Arbeit einer einzelnen Teilnahme, die die betroffene Person selbst sehen und kommentieren kann. Das **allgemeine Lehrerfazit zum Projekt** fasst dagegen die Gruppen- und Projektlage für die Lehrkraft zusammen.



Fazit & Beobachtung muss vor der Arbeit transparent konfiguriert sein. Nach dem Abschluss entstehen reduzierte Ergebnisse, aber keine Chatfreigabe.



Im Einstellungsdialog bleiben die Fazitarten getrennt. Die Option für Namen ist sichtbar dem allgemeinen Lehrerfazit zum Projekt zugeordnet.

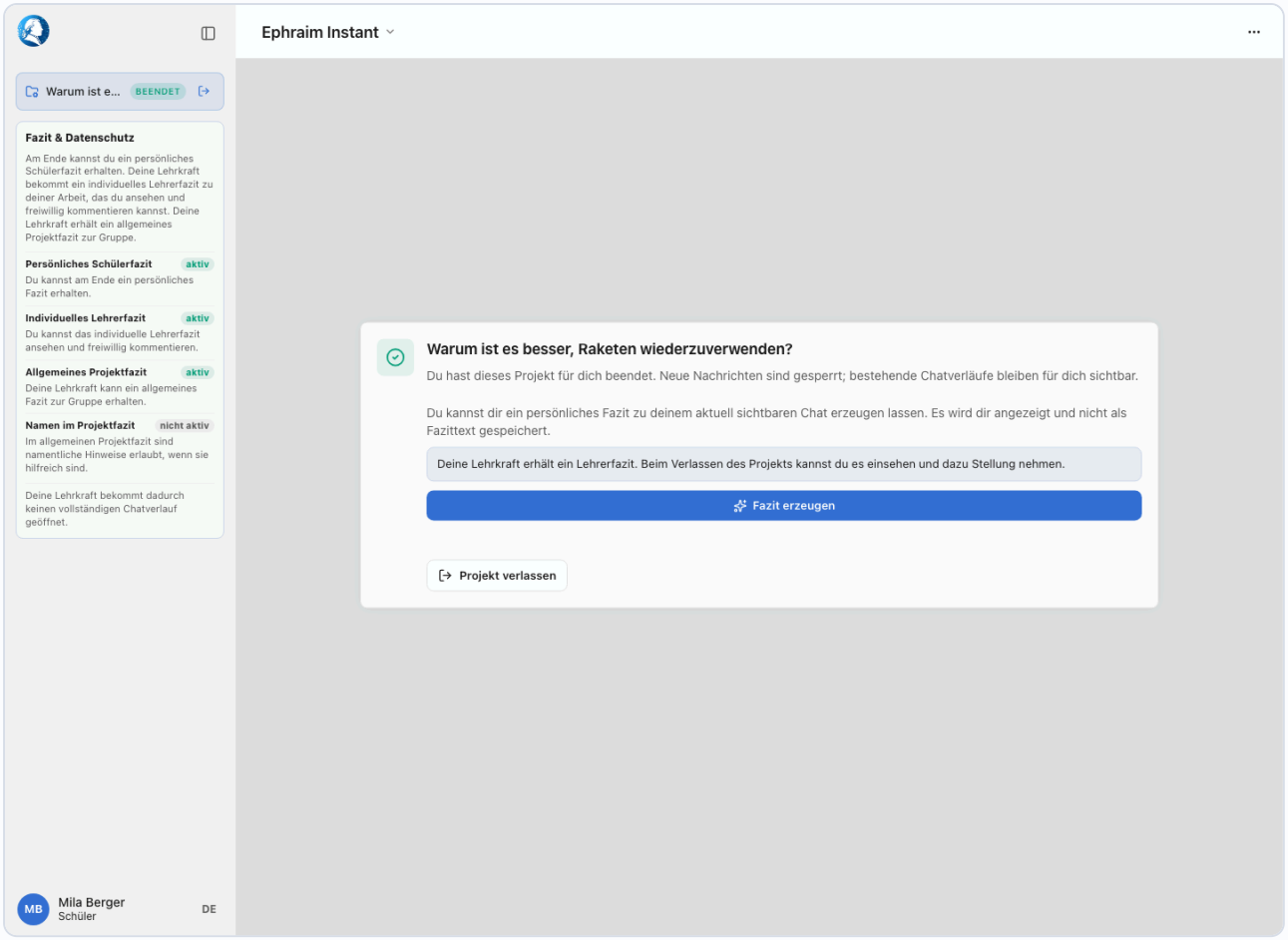
Option	Wirkung	Grenze
Persönliches Schülerfazit	Nach Projektende oder persönlichem Abschluss erzeugt die teilnehmende Person ein eigenes Lernfazit in ihrer Sitzung.	Der Fazittext wird der Lehrkraft nicht als Chattext geöffnet.
Individuelle Lehrerfazits je Schüler ermöglichen	Nach dem persönlichen Abschluss entsteht eine kurze Rückmeldung zur Arbeit der einzelnen Person. Die betroffene Person kann sie ansehen und freiwillig kommentieren.	Ephraim nutzt dafür eine reduzierte Auswertung des Projektstands. Rohchats und private Dateien bleiben geschlossen.
Allgemeines Lehrerfazit zum Projekt	Nach Abschluss des gesamten Projekts erzeugt die Lehrkraft ein allgemeines Projektfazit aus Projektlage und reduzierten Hinweisen.	Dafür muss die vorherige Option aktiv sein, damit Projektlage und Rückmeldungen vorliegen.
Namen im allgemeinen Lehrerfazit erlauben	Das allgemeine Lehrerfazit zum Projekt darf einzelne Teilnehmende nennen, wenn die Datenbasis Namen enthält und ein Hinweis fachlich sinnvoll ist.	Diese Option betrifft nicht das persönliche Schülerfazit. Ohne sie arbeitet das allgemeine Lehrerfazit mit anonymisierten Teilnehmerbezeichnungen.

Fazit & Beobachtung wird nicht nachträglich erweitert, sobald bereits Schülerinnen oder Schüler beigetreten sind. Dadurch entsteht keine rückwirkende Auswertung, mit der Teilnehmende beim Start nicht rechnen mussten. Ein allgemeines Lehrerfazit ohne die dafür nötige Projektlage ist ausgeschlossen. Namentliche Hinweise ohne allgemeines Lehrerfazit sind ebenfalls ausgeschlossen.

4.16.3 Teilnehmerabschluss

Teilnehmende schließen ihren eigenen Projektstand ab. Ephraim speichert dafür einen Abschlusszeitpunkt in der Projektmitgliedschaft. Danach lehnt der Projektchat neue Nachrichten für diese Person ab, solange dieser Abschlusszustand gilt. Der bestehende Verlauf bleibt für die teilnehmende Person sichtbar.

Beim Abschluss setzt Ephraim die Folgezustände: Das persönliche Schülerfazit wird ausstehend oder deaktiviert. Das individuelle Lehrerfazit je Schüler wird ausstehend oder deaktiviert. Beide Entscheidungen folgen der vorher gespeicherten Projektkonfiguration.



Nach dem persönlichen Abschluss sind neue Nachrichten gesperrt. Die teilnehmende Person kann ihr eigenes Fazit erzeugen. Die Seitenleiste zeigt weiterhin die Projektregeln und macht sichtbar, dass die Lehrkraft dadurch keinen vollständigen Chatverlauf erhält.

4.16.4 Projektende durch die Lehrkraft

Die Lehrkraft beendet das gesamte Projekt getrennt vom persönlichen Abschluss einzelner Teilnehmender. Ein beendetes, archiviertes oder abgelaufenes Projekt ist für neue Projektchats nicht mehr nutzbar. Diese Regel gilt für Projektcode-Gäste und angemeldete Schülerkonten.

Erst nach dem Ende des gesamten Projekts erscheint das allgemeine Lehrerfazit zum Projekt als sinnvolle Abschlussfunktion. Vorher zeigt die Projektlage den laufenden Arbeitsstand; sie ersetzt keine abschließende pädagogische Bewertung.

4.16.5 Wie Schülerinnen und Schüler hingewiesen werden

Beim Projektcode-Einstieg zeigt Ephraim vor dem Beitritt einen Hinweisdialog, sobald persönliches Schülerfazit oder individuelle Lehrerfazits aktiv sind. Der Beitritt wird erst fortgesetzt, wenn der Hinweis bestätigt wurde. Der Text nennt das persönliche Fazit, die reduzierte Auswertung für individuelle und allgemeine Lehrerfazits, die Regel zu Namen im allgemeinen Lehrerfazit und die Möglichkeit, ein individuelles Lehrerfazit selbst anzusehen und freiwillig zu kommentieren.

Projektcode-Gäste sehen diese Informationen zusätzlich in ihren Projekteinstellungen: Dort steht, ob individuelle Lehrerfazits aktiv sind, ob namentliche Hinweise erlaubt sind und ob nach Projektende ein persönliches Fazit erzeugt wird. Bei angemeldeten Schülerkonten erfolgt der Einstieg aus der Liste freigegebener Projekte; nach dem persönlichen Abschluss zeigt Ephraim direkt im Projektchat die

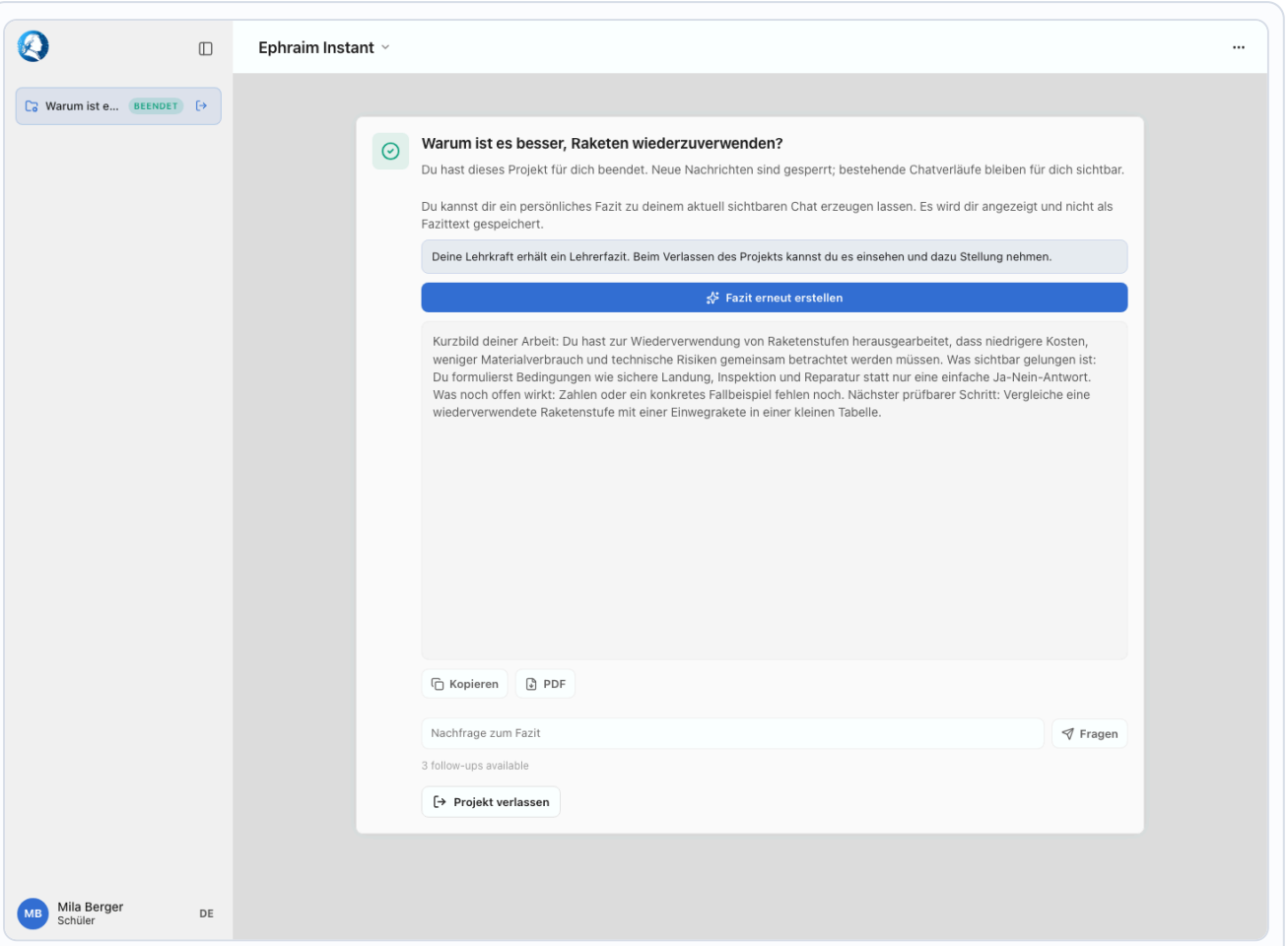
verfügbaren Fazitfunktionen. Wenn ein individuelles Lehrerfazit vorliegt, fragt Ephraim vor dem Verlassen des Projektabschlusses, ob die Schülerin oder der Schüler dasselbe Fazit ansehen möchte. Ohne Ansehen kann der Ablauf fortgesetzt werden.

> Bei konto-basierten Projekten bleibt der Projektauftrag der zentrale Ort für transparente pädagogische Hinweise. Die technische Fazitlogik verhindert dennoch, dass Lehrkräfte Projektchat-Rohtexte öffnen. Die spätere Schüleransicht schafft zusätzliche Transparenz, weil Rückmeldungen nicht nur über Lernende gespeichert, sondern für sie nachvollziehbar und kommentierbar werden.

4.16.6 Persönliches Schülerfazit

Ist das persönliche Fazit aktiviert, erzeugt die teilnehmende Person nach dem Abschluss einen Lernrückblick aus ihrem aktuell sichtbaren Projektchat. Das Ergebnis wird ihr angezeigt und dient der Reflexion: Was wurde bearbeitet, welche Stärken sind sichtbar, welche offenen Punkte bleiben, welche nächsten Schritte passen?

Dieses Fazit ist keine Note, kein Zeugnis und keine Rangliste. Der Fazittext wird nicht als dauerhaftes Schülerfazit für die Lehrkraft gespeichert. Persistiert werden Status und Zeitpunkt, damit die Projektlage zeigt, ob das persönliche Fazit erzeugt wurde. Die Schülerin oder der Schüler stellt anschließend kurze Rückfragen zum angezeigten Fazit; auch diese Rückfragen bleiben an die eigene Sitzung gebunden.



Das persönliche Schülerfazit bleibt in der Sitzung der teilnehmenden Person. Es kann kopiert, als PDF exportiert und mit eigenen Nachfragen weiter genutzt werden; die Lehrkraft sieht nicht diesen Fazittext.

4.16.7 Projektlage konkret

Die Lehrkraft sieht in der Projektlage keine Chattexte, sondern reduzierte Statusinformationen. Diese Informationen beantworten organisatorische Fragen: Wer hat das Projekt gestartet? Wer war zuletzt aktiv? Wer hat den eigenen Stand abgeschlossen? Wie viele Nachrichtenimpulse wurden im Projekt gezählt?

Welche Fazit- oder Rückmeldungszustände sind offen, fertig, deaktiviert oder fehlgeschlagen?

Anzeige	Konkrete Bedeutung
Gestartet	Anzahl der erwarteten Teilnehmenden, die dem Projekt beigetreten sind oder begonnen haben.
Kürzlich aktiv	Teilnehmende mit aktueller Präsenz im Projektbereich.
Beendet	Teilnehmende, die ihren Projektstand abgeschlossen haben.
Nachrichtenimpulse	Zählwert für Projektaktivität; kein Nachrichteninhalt.
Letzte Aktivität	Zeitpunkt des letzten reduzierten Aktivitäts- oder Präsenzsignals.
Fazitstatus	Persönliches Schülerfazit: nicht angefragt, offen, fertig oder deaktiviert.
Individuelles Lehrerfazit	Zustand der reduzierten Rückmeldung zur einzelnen Teilnahme: offen, wird erzeugt, fertig, keine Daten, fehlgeschlagen oder deaktiviert.

Bei Projekten mit ausgewählten Schülerkonten und bei Projektcode-Gästen mit Anzeigenamen erscheinen Teilnehmerzeilen. Bei anonymen Projektcode-Gästen ohne Anzeigenamen bleibt die Ansicht zusammengefasst. Auch in der Teilnehmerzeile stehen Status, Zeiten, Zähler und das individuelle Lehrerfazit, nicht der Chatverlauf.

4.16.8 Individuelles Lehrerfazit je Schüler

Wenn individuelle Lehrerfazits je Schüler aktiv sind, erzeugt Ephraim nach dem Teilnehmerabschluss eine reduzierte Rückmeldung zur einzelnen Teilnahme. Grundlage ist der sichtbare Projektchat-Auszug der teilnehmenden Person. Dieser Auszug wird bereinigt, gekürzt und zur Erzeugung einer kompakten Rückmeldung verwendet.

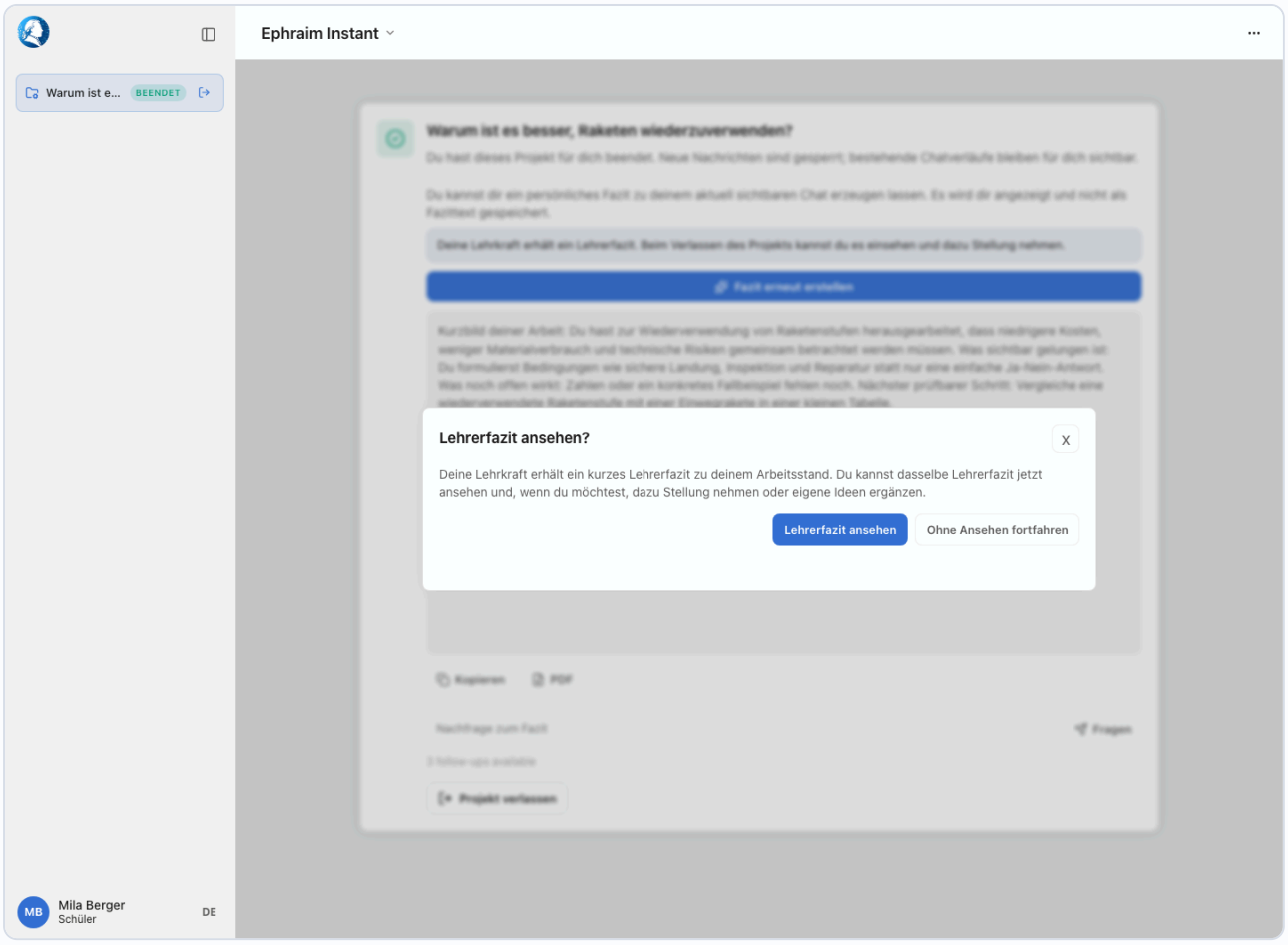
Das Ergebnis ist kein Chatprotokoll. Es ist eine reduzierte, verdichtete Rückmeldung mit vier Teilen: Arbeitsstand, belegbare Stärken und Fortschritt, offene Risiken oder Fehlkonzepte, nächster sinnvoller Lehrkraftimpuls. Der Rohchat wird nicht in den Projektmetadaten gespeichert und nicht als Verlauf für die Lehrkraft angezeigt.

Liegt kein geeigneter sichtbarer Schülerbeitrag vor, markiert Ephraim den Status als „keine Daten“. Bei technischen Problemen erscheint ein Fehlstatus. Die Lehrkraft sieht dadurch den Zustand des individuellen Lehrerfazits, ohne Chattexte nachzulesen.

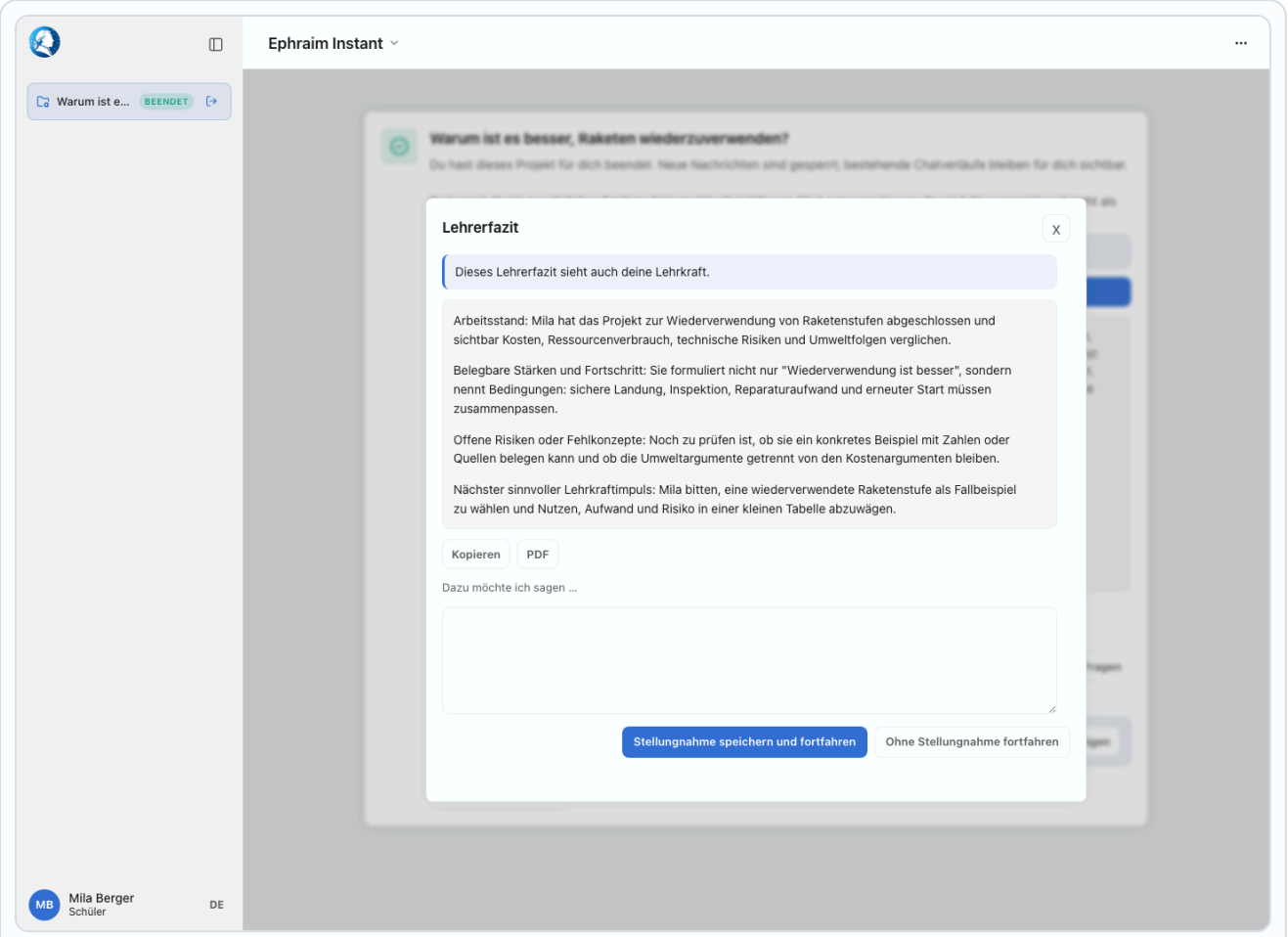
4.16.9 Schüleransicht des individuellen Lehrerfazits

Wenn für eine teilnehmende Person ein individuelles Lehrerfazit vorliegt, zeigt Ephraim vor dem Fortsetzen des Projektabschlusses eine Nachfrage. Die Schülerin oder der Schüler kann das Lehrerfazit ansehen oder ohne Ansehen fortfahren. Beim Ansehen steht klar dabei, dass dieselbe Rückmeldung auch der Lehrkraft vorliegt.

Nach dem Lesen kann die teilnehmende Person freiwillig Stellung nehmen, zum Beispiel um einen nächsten Arbeitsschritt zu ergänzen oder eine eigene Sicht festzuhalten. Diese Stellungnahme wird zusammen mit dem Lehrerfazit gespeichert und erscheint auch im vollständigen PDF-Export des individuellen Lehrerfazits. Ohne Stellungnahme bleibt nur der Sichtstatus dokumentiert.



Vor dem Verlassen des abgeschlossenen Projekts fragt Ephraim nach, ob das individuelle Lehrerfazit angesehen werden soll. Die betroffene Person kann es öffnen oder ohne Ansehen fortfahren.



Die Schülerin oder der Schüler sieht dasselbe Lehrerfazit wie die Lehrkraft, kann es kopieren, als PDF exportieren und freiwillig eine Stellungnahme ergänzen.

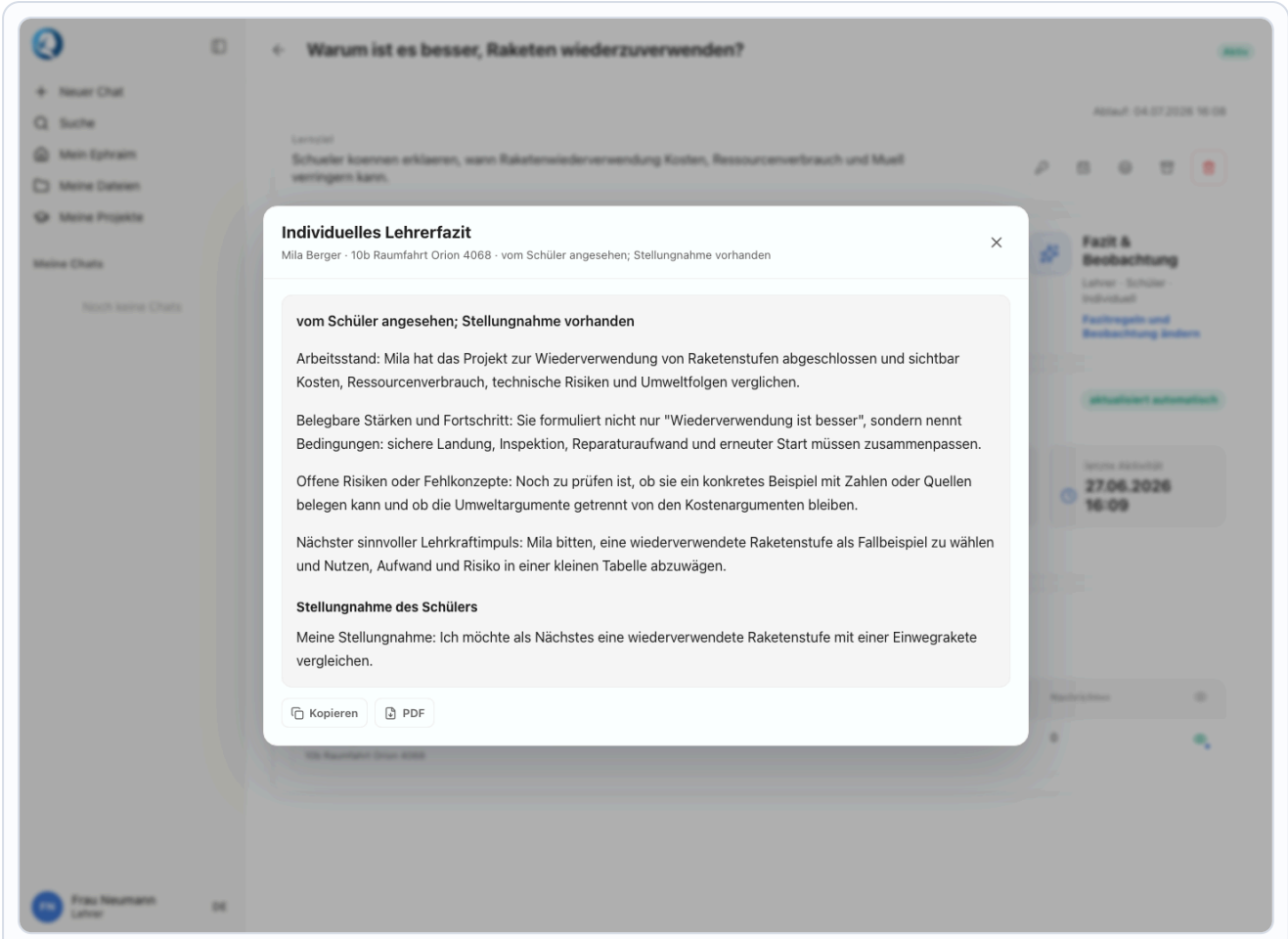
> **Transparenzwirkung:** Das individuelle Lehrerfazit ist keine verdeckte Auswertung. Die betroffene Person kann dieselbe Rückmeldung sehen und eine eigene Einordnung ergänzen.

4.16.10 Allgemeines Lehrerfazit nach Projektabschluss

Das allgemeine Lehrerfazit entsteht erst nach Abschluss des gesamten Projekts und nur, wenn das allgemeine Lehrerfazit vorher aktiviert war. Es fasst die Projektlage für die Lehrkraft zusammen: Kurzlage der Gruppe, belegbare Arbeitsmuster, offene Risiken oder Fehlkonzepte und nächste Lehrkraftaktionen.

Die Datenbasis besteht aus Projektmetadaten, Teilnehmerstatus, Fazitstatus und optionalen individuellen Lehrerfazits je Schüler. Zusätzlich trägt die Lehrkraft freiwillige, bereits reduzierte Hinweise ein. Der Eingabebereich ist ausdrücklich nicht dafür gedacht, private Rohchats hineinzukopieren.

Wenn namentliche Hinweise deaktiviert sind, arbeitet das allgemeine Lehrerfazit mit anonymisierten Teilnehmerbezeichnungen. Wenn namentliche Hinweise aktiviert sind, erscheinen Namen nur dort, wo sie in der Datenbasis vorhanden sind und für die pädagogische Aussage helfen. Auch dann bleibt der Chatverlauf geschlossen.



Die Lehrkraft sieht im individuellen Lehrerfazit, ob die Rückmeldung angesehen wurde und ob eine Stellungnahme vorhanden ist. Lehrerfazit und Stellungnahme stehen gemeinsam bereit; der Rohchat bleibt geschlossen.

4.16.11 Beispieltexte aus einem synthetischen Testlauf

Die folgenden Beispiele stammen sinngemäß aus E2E-Testläufen mit fiktiven Personen. Im Beispielprojekt lautet der Arbeitsauftrag: Mila Berger untersucht, warum es besser sein kann, Raketen wiederzuverwenden. Sie vergleicht Kosten, Materialverbrauch, technische Risiken und Umweltfolgen. Jonas Keller ist im selben Projekt freigegeben, startet aber noch nicht.

Vor diesem Arbeitsauftrag zeigen die drei Beispieltexte, wie unterschiedlich die Ausgabeformen aussehen. Das persönliche Fazit spricht Mila direkt an. Das individuelle Lehrerfazit je Schüler ist knapper und pädagogisch handlungsorientiert. Das allgemeine Lehrerfazit fasst die Projektlage zusammen und bleibt ebenfalls ohne Rohchat. Zusätzlich kann Mila das individuelle Lehrerfazit sehen und eine Stellungnahme ergänzen.

Ausgabe	Beispiel	Sichtbarkeit
Persönliches Schülerfazit	„Du hast die Frage nach wiederverwendbaren Raketen nicht nur allgemein beantwortet, sondern Bedingungen und Gegenargumente gesammelt. Sichtbar gelungen ist, dass du Kosten, Materialverbrauch, Zuverlässigkeit und Umweltfolgen unterscheidest. Offen bleibt, wie stark Inspektion, Reparatur und zusätzlicher Treibstoff den Vorteil verkleinern. Nächster Schritt: Ergänze ein konkretes Beispiel für eine wiederverwendete Raketenstufe.“	Nur in der Sitzung der teilnehmenden Person sichtbar; die Lehrkraft sieht den Text nicht als Schülerfazit.
Individuelles Lehrerfazit je Schüler	„Arbeitsstand: Mila hat die Wiederverwendung von Raketenstufen mit mehreren Kriterien untersucht und eine Vergleichstabelle begonnen. Stärke: Sie korrigiert eine zu pauschale Aussage und betrachtet Kosten, Materialverbrauch, Zuverlässigkeit und Umweltfolgen getrennt. Risiko: Noch offen ist, wie stark Inspektion, Reparatur und zusätzlicher Treibstoff die Vorteile begrenzen. Lehrkraftimpuls: Ein konkretes Beispiel einer gelandeten Raketenstufe nutzen.“	Für die Lehrkraft sichtbar, wenn individuelle Lehrerfazits je Schüler vorab aktiviert waren; kein vollständiger Chattext.
Allgemeines Lehrerfazit	„Kurzlage der Gruppe: Mila hat ihren Projektstand abgeschlossen und zeigt einen nachvollziehbaren Kriterienvergleich zur Wiederverwendung von Raketen; nicht alle Teilnehmenden sind gestartet, Jonas hat noch nicht begonnen. Arbeitsmuster: Eingrenzung der Fragestellung, Kriterienbildung und Begründung unter Bedingungen. Nächste Aktionen: Mila ein Beispiel einer wiederverwendeten Stufe auswerten lassen; Jonas beim Einstieg unterstützen.“	Für die Lehrkraft nach Projektabschluss sichtbar; gruppenbezogen, reduziert und ohne Rohchatfreigabe.
Stellungnahme der Schülerin	„Ich möchte als Nächstes eine wiederverwendete Raketenstufe mit einer Einwegrakete vergleichen.“	Freiwillige Ergänzung der betroffenen Person; zusammen mit dem individuellen Lehrerfazit sichtbar und im PDF enthalten.

> **Lesart:** Die Beispiele zeigen keine Bewertung im Sinne einer Note. Sie zeigen Arbeitsstand, Reflexion, offene Punkte und nächste pädagogische Schritte.

4.16.12 So sieht ein realistischer Ablauf aus

1. Die Lehrkraft aktiviert vor dem Projektstart persönliches Schülerfazit, individuelle Lehrerfazits je Schüler und allgemeines Lehrerfazit.
2. Projektcode-Gäste bestätigen beim Beitritt den Hinweis; ausgewählte Schülerkonten starten das freigegebene Projekt aus ihrem Projektbereich.
3. Während der Bearbeitung sieht die Lehrkraft Projektlage-Zähler, Aktivitätssignale und bei passenden Teilnahmeformen Teilnehmerstatus.
4. Eine Schülerin beendet ihren Projektstand. Ihr Chat wird gesperrt, der Verlauf bleibt für sie sichtbar.
5. Sie erzeugt ihr persönliches Fazit. Die Lehrkraft sieht später nur den Status „fertig“, nicht den Fazittext.
6. Wenn individuelle Lehrerfazits aktiv sind, erzeugt Ephraim eine reduzierte Rückmeldung für die Lehrkraft: Arbeitsstand, Stärken, Risiken, nächster Impuls.
7. Die Schülerin kann dieses individuelle Lehrerfazit ansehen, kopieren, als PDF exportieren und freiwillig Stellung nehmen.
8. Nach Projektende erzeugt die Lehrkraft ein allgemeines Lehrerfazit aus Projektlage, Statusdaten, reduzierten Rückmeldungen, Schülerstellungnahmen und eigenen reduzierten Hinweisen.

Abschluss, persönliches Fazit, Projektlage, individuelles Lehrerfazit je Schüler und allgemeines Lehrerfazit sind getrennte Zustände. Keiner dieser Zustände öffnet den vollständigen Projektchat für die Lehrkraft. Die Datenschutzgrenze steht im Artikel Projekte: Datenschutz.

Quelle: web/manuals/projekte-abschluss-und-fazit/index.html

4.17 Projekte: Datenschutz

> **Projektarbeit hat eigene Datenschutzgrenzen. Privates Konto, Projektmaterial, Projektchat, Projektcode-Gast und Lehrkraftauswertung bleiben getrennte Kontexte.**

Stand: Juni 2026

4.17.1 Drei Datenräume

Datenraum	Inhalt	Projektgrenze
Privates Konto	Persönliche Chats, private Dateien, Erinnerungen, private Kalender, Sicherheitseinstellungen	Nicht automatisch Teil eines Projekts.
Projekt	Auftrag, Projektmaterial, eigener Projektchat, Mitgliedschaft, Abschlussstatus, konfigurierte Fazit- und Rückmeldestatus	Projektchat-Rohtexte bleiben bei der jeweiligen teilnehmenden Person.
Projektcode-Gast	Projektgebundene Gastsitzung, Anzeigename, Projektverlauf, aktueller Gast-Datenexport	Kein vollständiges Konto, kein persönlicher Vault und keine spätere Fortsetzung über eine neue Gastsession.

Für die allgemeine Verschlüsselung privater Daten siehe Datenschutz tiefgehend. Die technische Schlüsselarchitektur für Admins steht in der Kryptoarchitektur.

4.17.2 Sicht der Lehrkraft

Die Lehrkraft verwaltet ihr Projekt, die Projektmaterialien, Teilnahmeregeln und Abschlussregeln. Sie erhält Projektstatus und die aktivierten Fazit- oder Rückmeldeinformationen. Dazu können Sichtstatus und freiwillige Stellungnahmen zu einem individuellen Lehrerfazit gehören. Sie erhält keinen Zugriff auf Projektchat-Rohtexte einzelner Teilnehmender, keinen generellen Zugriff auf private Schülerchats und keinen Zugriff auf private Dateien außerhalb bewusst freigegebener Projektkopien.

Bereich	Lehrkraft sieht	Lehrkraft sieht nicht
Privater Chat	Nichts, außer die Person exportiert oder teilt selbst Inhalte außerhalb von Ephraim.	Chatverlauf, private Dateien, Erinnerungen, Kalender.
Projekt mit Schülerkonto	Teilnahme, Status, Abschluss, reduzierte Rückmeldungen, Sichtstatus und freiwillige Stellungnahmen nach Konfiguration.	Rohtext des Projektchats und private Kontoinhalte.
Projektcode-Gast	Projektgebundener Anzeigename, Gaststatus und konfigurierte Fazit- und Beobachtungsdaten.	Ein normales Konto, private Vault-Daten oder schulweite Chathistorie.
Projektmaterial	Freigegebene Projektkopien und deren Verarbeitungsstatus.	Das private Original einer Datei, wenn nur eine projektbezogene Kopie erstellt wurde.

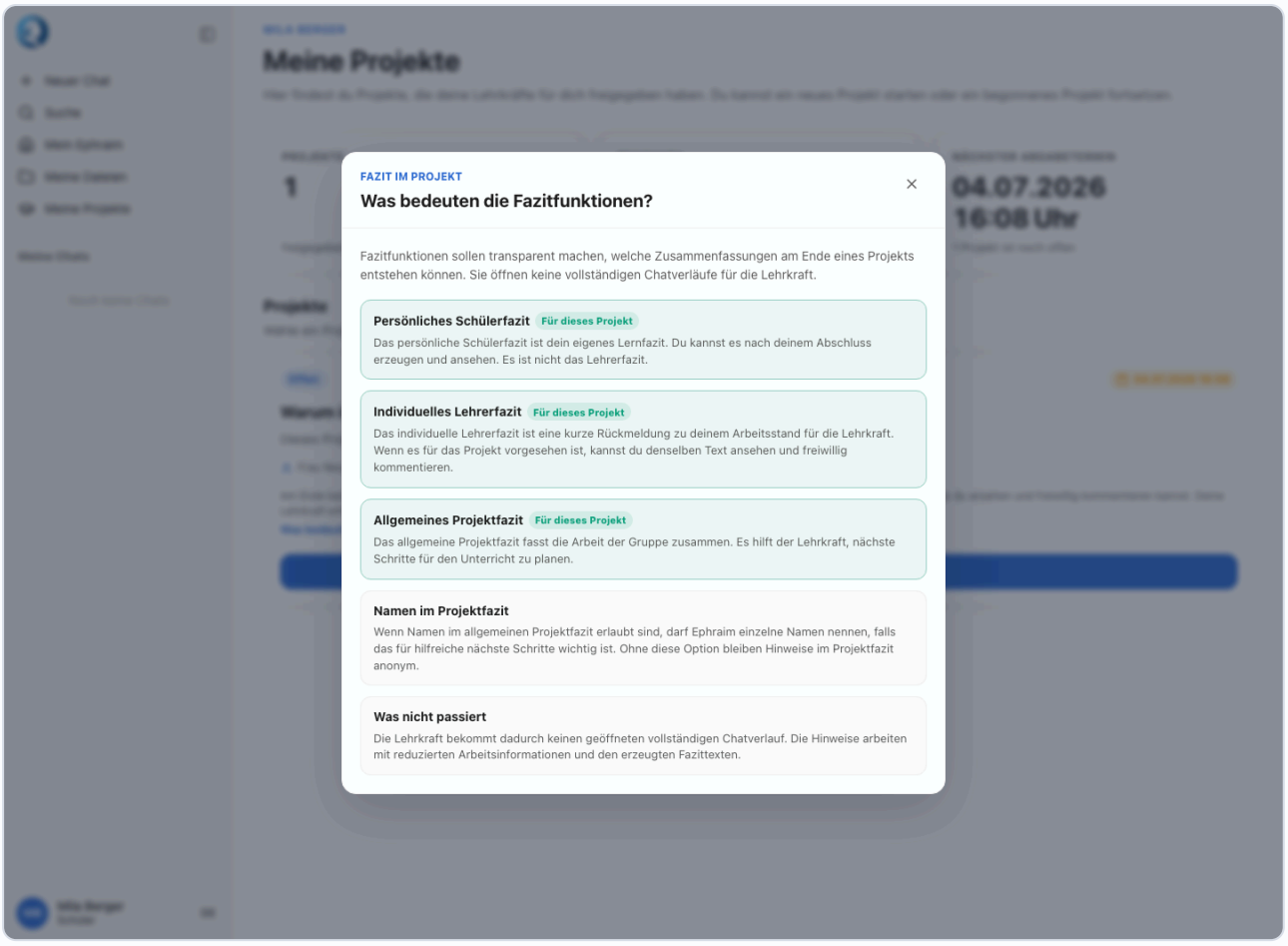
4.17.3 Projektlage und Beobachtung

Projektlage und Beobachtung sind projektbezogen und müssen vor der Nutzung transparent im Projektrahmen aktiviert sein. Sie dienen der Unterrichtssteuerung: Beitritt, Aktivität, Nachrichtenimpulse, Abschlussstatus, Fazitstatus, konfigurierte Rückmeldungen und deren Transparenzstatus. Sie ersetzen keine versteckte Rohchat-Einsicht.

Bei Projektcode-Gästen zeigt Ephraim vor dem Beitritt einen Hinweis, sobald persönliches Fazit oder individuelle Lehrerfazits aktiv sind. In den Projekteinstellungen des Gastzugangs steht zusätzlich, ob individuelle Lehrerfazits aktiv sind, ob namentliche Hinweise erlaubt sind und ob ein persönliches Fazit erzeugt wird.

Bei ausgewählten Schülerkonten kann die betroffene Person ein individuelles Lehrerfazit selbst ansehen und freiwillig Stellung nehmen. Diese Transparenz senkt Missbrauchsrisiken, weil Rückmeldungen nicht verdeckt über Lernende entstehen, sondern für sie überprüfbar und kommentierbar werden.

Projektchats sind zusätzlich technisch an das Projekt gebunden. Sie erscheinen nicht im normalen Chatverlauf und können außerhalb des passenden Projektkontexts nicht über normale Chatlinks geöffnet werden. Für den eigenen Datenexport bleiben sie trotzdem enthalten: bei Schülerkonten im Kontodatenexport, bei Projektcode-Gästen im Export der aktuellen temporären Gastidentität.



Der Schülerdialog erklärt aktivierte Fazitfunktionen aus Betroffenensicht. Wichtig ist die Grenze: Fazitfunktionen öffnen keine vollständigen Chatverläufe für die Lehrkraft.

4.17.4 Basiswissen und externe Quellen

Projektantworten verwenden lokale Spark-Inferenz. Das Modell selbst surft nicht im Internet. Ephraim ergänzt vor der Inferenz zentral freigegebenes Basiswissen, Wetter, Kalender oder Wikipedia-Auszüge, wenn diese Quellen aktiviert, für den Kontext freigegeben und zur Anfrage passend sind. Diese Unterscheidung ist wichtig: Der Webserver bereitet Kontext vor; die Spark berechnet die Antwort aus diesem Kontext.

4.17.5 Archivieren und Löschen

Archivieren entfernt Projekte aus dem aktiven Arbeitsbereich, ohne sie sofort als Datensatz zu löschen. Löschen entfernt projektgebundene Daten nach den Systemregeln. Normale Schülerkonten werden durch Projektlöschung nicht gelöscht. Projektcode-Gastkonten, die ausschließlich zu einem gelöschten Projekt gehören, werden projektbezogen entfernt. Ein Gast kann seinen temporären Gastzugang außerdem bewusst löschen; dann verschwindet auch der zugehörige Gast-Projektchat sofort.

Der Datenschutz im Projekt steht auf der Trennung der Datenräume. Projektarbeit öffnet keine privaten Konten. Für die Bedienlogik siehe Zugang und Rollen und Abschluss und Fazit.

Quelle: web/manuals/projekte-datenschutz/index.html

EINORDNUNG

5. Wie Vertrauen technisch und organisatorisch entsteht

> **Vertrauen entsteht in Ephraim nicht durch eine einzelne Zusicherung, sondern durch begrenzte Datenräume, Verschlüsselung, transparente Wiederherstellung, Logging-Grenzen und kontrollierte Verarbeitung.**

Der persönliche Vault ist die zentrale Idee für private Inhalte. Passwortwechsel, Passwort-Recovery und Datenexport sind deshalb nicht nur Bedienfunktionen, sondern konkrete Folgen der Schlüsselarchitektur.

Auch Betrieb und Datenschutz gehören zusammen: Apache soll keine Anfrageinhalte und Tokens protokollieren, die Spark speichert keine Anfragen, und Dateien werden geprüft, verschlüsselt abgelegt und kontrolliert ausgeliefert.

5.1 Systemarchitektur und Datenschutz

> Wie **Ephraim** technisch aufgebaut ist und welche Maßnahmen den Datenschutz im schulischen Betrieb absichern. Ephraim ist die KI- und Projektplattform des Lessing-Gymnasiums Karlsruhe.

Stand: Juni 2026

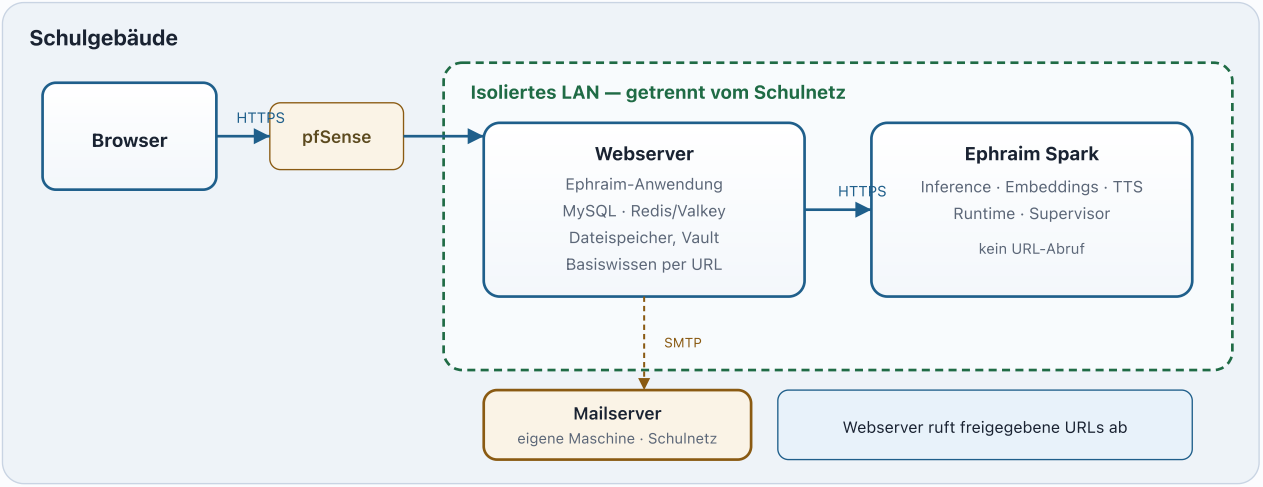
5.1.1 Was ist das System?

Ephraim ist eine schulische Webplattform für KI-gestütztes Lernen, Chat, Dateiablage, Projektarbeit und Wissensquellen. Sie besteht aus zwei klar getrennten Teilen:

- **Webserver (Ephraim-Anwendung):** Anmeldung, Chat-Oberfläche, Verwaltung, verschlüsselte Speicherung, Projekte und das Einbinden von Basiswissen aus konfigurierten Quellen — einschließlich Abrufe per URL, die nur der Webserver durchführt.
- **Ephraim Spark:** Dedizierte KI-Hardware im isolierten LAN. Sie führt Sprachmodell, Embedding-Dienst, Text-zu-Sprache (TTS), Runtime und Supervisor aus.

> **Kernprinzip:** KI-Anfragen werden nicht an externe Cloud-APIs kommerzieller Anbieter geschickt. Webserver und Mailserver werden durch die Schule betrieben — der Mailserver auf einer eigenen Maschine. Webserver und Spark tauschen Daten nur im isolierten LAN aus; dieses ist vom Schulnetz getrennt. Die Spark ist von außen nicht direkt zugänglich.

5.1.2 Physische und logische Infrastruktur



Von außen ist nur der HTTPS-Endpunkt des Webservers erreichbar. Webserver und Spark kommunizieren ausschließlich im isolierten LAN — getrennt vom Schulnetz und ebenfalls verschlüsselt per HTTPS. Der Mailserver steht auf einer anderen Maschine in der Schule.

Die relationale Datenbank läuft in Produktion auf **MySQL** auf dem Webserver. Zusätzlich kann auf demselben Host ein **Redis/Valkey-kompatibler Speed-Layer** laufen. Er ist nur maschinenintern erreichbar und dient nicht als dauerhafte Datenbank. Nutzerinnen und Nutzer greifen von zu Hause oder aus dem Unterricht per Browser auf den Webserver zu; dieser ist der einzige Teil von Ephraim, der nach außen sichtbar ist. E-Mails (Zugangsdaten, Sicherheitshinweise) versendet der Webserver über den separaten Mailserver im Schulnetz.

5.1.3 Aufbau der Ephraim-Anwendung

Bereich	Aufgabe
Benutzeroberfläche	Chat, Dateien, Projekte, Einstellungen, Hilfe
Anwendungslogik	Anmeldung, Rollen, Vault, Chat-Verarbeitung, Projekte, Uploads
Datenbank (MySQL)	Konten, Metadaten, verschlüsselte Chat-Inhalte, Projekte
Redis/Valkey-Speed-Layer	Kurzlebige verschlüsselte Stream-Puffer und Worker-Wakeups auf demselben Host
Dateispeicher	Verschlüsselte persönliche Dateien außerhalb des öffentlich erreichbaren Webroots
Basiswissen	Schulische Quellen und freigegebene externe URLs — Abruf und Aufbereitung durch den Webserver

5.1.4 Ephraim Spark

Die Spark ist die schuleigene KI-Hardware. Für Version 1 von Ephraim laufen dort das zentrale Sprachmodell **GPT-OSS-120B** (OpenAI-kompatibles 120-Milliarden-Parameter-Modell, lokal gehostet) sowie ein **TTS-Dienst** für das Vorlesen von Chat-Antworten.

Das Sprachmodell ist dabei als Inferenzprofil betrieben, nicht als fest in die Oberfläche eingebauter Bestandteil. Ephraim spricht den lokalen Inferenzdienst über denselben OpenAI-kompatiblen Endpunkt an. Der aktuelle produktive und dokumentierte Stand nutzt GPT-OSS-120B. Externe Modellanbieter und externe Modellproxys gehören nicht zur freigegebenen V1-Konfiguration; jede Abweichung vom dokumentierten GPT-OSS-120B-Pfad wäre neu zu bewerten und zu dokumentieren.

Die Ephraim Spark ist kein abstrakter Cloud-Dienst, sondern ein lokaler Rechner im Besitz der Schule. Foto: Dr. Daniel Roth, Lessing-Gymnasium Karlsruhe.

Komponente	Funktion in V1
Inference	Erzeugt Chat-Antworten mit GPT-OSS-120B
Embeddings	Berechnet Vektoren für semantische Suche, Projektmaterial und Basiswissen-Retrieval
TTS	Wandelt ausgewählten Chat-Text in gesprochene Sprache um (Vorlesen-Funktion)
Runtime	Technisches Monitoring und administrierte Betriebsaktionen
Supervisor	Geplante Updates, Job-Status, Audit-Export für Super-Admins

Embeddings und Retrieval in der Webanwendung

Der Retrieval-Pfad von Version 1 liegt in der Webanwendung. Ephraim extrahiert Text, zerlegt ihn in kurze Abschnitte, prüft Rechte, verschlüsselt Abschnittstext und Vektor und speichert beides in MySQL.

Für die reine Vektorberechnung ruft die Webanwendung den konfigurierten OpenAI-kompatiblen `/embeddings`-Endpunkt auf. Im lokalen Spark-Stack ist dieser Endpunkt der lokale SGLang-Embedding-Dienst **school-ui-embedding**. Ephraim nutzt ihn aktuell direkt für Datei-RAG, Projektmaterial-RAG und Basiswissen-RAG. Der Embedding-Endpunkt erzeugt keine Antworttexte, sondern Zahlenlisten: Ein Klartext-Chunk und eine Suchfrage werden in denselben mathematischen Raum übersetzt. Ähnliche Inhalte liegen dort näher beieinander als unähnliche Inhalte. Die Spark speichert diese Klartexte, Suchfragen und Vektoren dabei nicht dauerhaft.

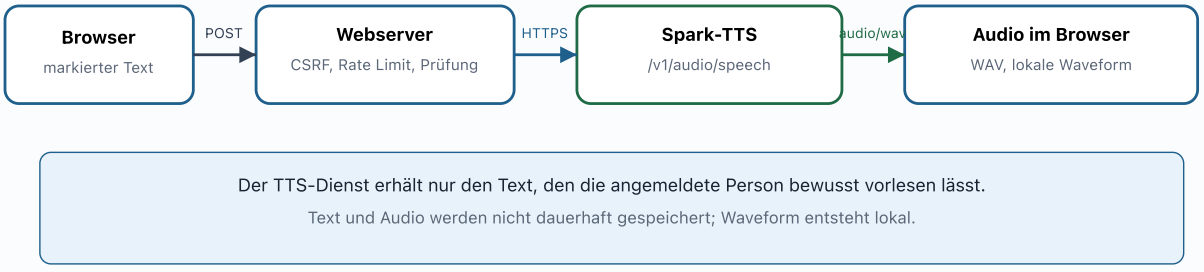
Bei einer späteren Frage erzeugt der Webserver einen Fragevektor und vergleicht ihn mit den gespeicherten Vektoren. Bewertet werden nur Abschnitte, die für diese Person, Rolle und Situation zugelassen und entschlüsselbar sind. Auswahl, Zugriffskontrolle, Entschlüsselung, Cosinus-Vergleich und Kontextaufbau passieren in Ephraim auf dem Webserver.

Beim Vorlesen sendet der Webserver den markierten Text verschlüsselt an den TTS-Dienst auf der Spark und gibt die erzeugte Sprachausgabe an den Browser weiter. Der Browser spricht den Text nicht an externe Cloud-Dienste.

> **Wichtig:** Die Spark führt das Sprachmodell aus, surft aber nicht selbst im Internet. Wenn Ephraim Wetter, Wikipedia, Kalender oder andere freigegebene Webquellen einbezieht, holt der *Webserver* diese Inhalte per URL ab, prüft sie und reicht nur den freigegebenen Kontext an die Spark weiter.

5.1.5 Datenfluss: Vorlesen mit Spark-TTS

Die Vorlesefunktion ist in der Architektur ein eigener Spark-Pfad. Sie benutzt nicht das Chatmodell, sondern einen internen Text-zu-Sprache-Dienst auf der Spark. Der Browser kennt weder die interne TTS-Adresse noch den Authentifizierungstoken; beides bleibt auf dem Webserver.

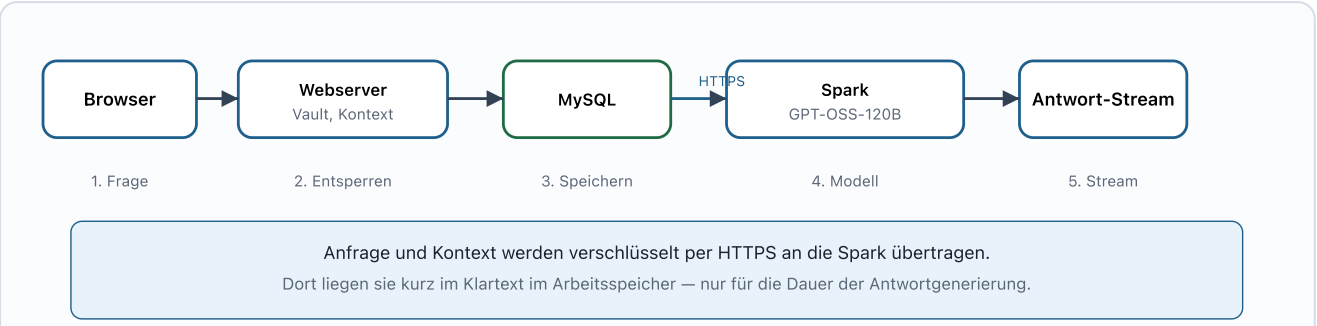


Spark-TTS ist ein interner Dienst. Der Webserver nimmt die Nutzeranfrage entgegen, prüft sie und leitet nur den normalisierten Text an die Spark weiter.

Schritt	Schutzmaßnahme
Browser sendet Vorlesewunsch	Nur angemeldete Sitzungen; mutierende Anfrage mit CSRF-Token.
Webserver prüft Text	Roh-Request-, Text- und Audio-Größenlimits; Sitzungs-, Nutzer- und IP-Rate-Limits; Parallelitätsgrenzen, unterstützte Sprachen, begrenzte Geschwindigkeit, kein Markup.
Webserver ruft Spark-TTS auf	Interne URL aus der Serverkonfiguration, Bearer-Token serverseitig, kurzer Verbindungs- und Gesamttimeout.
Spark erzeugt Audio	Antwortformat WAV; keine Text-/Audio-, Access- oder Syntheselogs im Normalbetrieb; Fehler wie Warteschlange, falsche Konfiguration oder unerreichbarer Dienst werden abgefangen.
Browser spielt Audio ab	Antwort mit `Cache-Control: no-store`; Waveform-Peaks werden lokal berechnet und nicht gespeichert.

Wenn der interne TTS-Dienst nicht konfiguriert ist, bleibt die Vorlesefunktion deaktiviert. Die Oberfläche nutzt keine lokale Web-Speech-Funktion des Browsers als Ersatz. Die Architektur von Ephraim sieht für den schulisch kontrollierten Betrieb ausschließlich den Spark-TTS-Pfad vor.

5.1.6 Datenfluss: typischer KI-Chat



Der Chat wird schrittweise zurückgestreamt. Gespeichert wird der Verlauf verschlüsselt; die Spark sieht Prompt und freigegebenen Kontext nur für den jeweiligen Request.

1. Die angemeldete Person sendet eine Nachricht über den Browser.
2. Der Webserver prüft Sitzung, Berechtigung und ob der persönliche Vault entsperrt ist.
3. Personalisierte Einstellungen, Projekt-Kontext und optional Basiswissen werden zusammengestellt.
4. Die Anfrage geht verschlüsselt per HTTPS an die Spark; GPT-OSS-120B erzeugt die Antwort.
5. Die Antwort wird zurückgestreamt und der Verlauf verschlüsselt in MySQL abgelegt.

5.1.7 MySQL und Redis/Valkey

MySQL und Redis/Valkey erfüllen in Ephraim unterschiedliche Aufgaben. MySQL ist die dauerhafte Datenbank. Dort liegen Konten, Projekte, Einstellungen, Jobzustände, verschlüsselte Chatdaten, verschlüsselte Stream-Fallbacks, verschlüsselte Chunks und verschlüsselte Embedding-Vektoren. Redis/Valkey ist dagegen ein schneller Kurzzeitpuffer für laufende Arbeit.

Komponente	Rolle	Datenschutzgrenze
MySQL	Dauerhafte Wahrheit des Systems	Private Inhalte liegen verschlüsselt; Rechte und Schlüssel entscheiden, wer etwas nutzen kann.
Redis/Valkey	Maschineninterner Speed-Layer für laufende Streams und Worker-Wakeups	Keine Klartext-Chats, Dateien, Vault-Schlüssel, Passwörter, Nutzerprofile, Sessions oder RAG-Klartexte.
Datenbank-Fallback	MySQL kann Stream-Puffer auch ohne Redis/Valkey tragen	Redis/Valkey beschleunigt, ersetzt aber nicht die dauerhafte Speicherung.

Für Nutzer ist das am besten mit einem Schreibtisch vergleichbar: MySQL ist der verschlossene Aktenschrank. Redis/Valkey ist der kleine Zettel auf dem Tisch, der nur während einer laufenden Antwort hilft, den nächsten Schritt schnell zu finden. Auf diesem Zettel steht nicht der Chat im Klartext, sondern nur kurzlebige verschlüsselte Arbeitsdaten und technische Signale.

> **Produktionsgrenze:** Redis/Valkey läuft auf demselben Host wie die Webanwendung und ist nur über ein maschineninternes Server-/Containernetz erreichbar. Es ist kein externer Dienst, kein Managed-Redis und kein neuer dauerhafter Speicherort.

5.1.8 Basiswissen und Internetzugriff

Ephraim kann schulisches Basiswissen in Chats einbeziehen — zum Beispiel freigegebene Dokumente, Kalender, Wetter oder Wikipedia. Diese Quellen werden vom **Webserver** abgerufen, geprüft und aufbereitet.

- Der Webserver darf konfigurierte URLs und Dienste ansprechen.
- Die Spark hat keinen eigenen Internet-Browser und ruft keine URLs selbst ab.
- Das Sprachmodell erhält nur den Kontext, den die Anwendung bewusst freigibt.

Private Kalender sind dabei kein zentrales Schulwissen. Sie gehören fachlich zum privaten Konto, sind an den persönlichen Vault gebunden und werden nur im normalen Chat dieses Kontos sowie in „Mein Ephraim“ berücksichtigt. Projektbuilder, Projektchats und andere Nutzer erhalten diesen privaten Kalenderkontext nicht.

Das unterscheidet Ephraim von einem öffentlichen Chatbot mit freiem Webzugang: Externe Inhalte fließen nur über festgelegte, schulisch kontrollierte Wege ein.

5.1.9 Hintergrundjobs, Datenlebenszyklus und Chat-Limit

Ephraim arbeitet nicht nur während eines Seitenaufrufs. Neben Browser, Webserver und Spark gibt es eine Wartungsschicht für Aufgaben, die geplant, begrenzt und kontrollierbar laufen müssen. Diese Schicht ist für den Datenschutz wichtig, weil sie Aufbewahrung, Aufräumen und technische Pflege erzwingt, ohne private Vaults im Hintergrund zu öffnen.

Bereich	Was passiert	Grenze
Cron	Bereinigt Rate-Limits, abgelaufene KI-Jobs, temporäre Exporte, verwaiste Dateien, alte Sicherheitsereignisse und archivierte Projekte.	Cron besitzt keinen User-DEK und öffnet keine persönlichen Vaults.
Knowledge-Refresh	Aktualisiert freigegebene globale Quellen und Wetterdaten serverseitig.	Private Kalender werden nicht massenhaft im Cron entschlüsselt; sie brauchen den entsperrten Vault des Besitzers.
KI-Jobs und Streams	Verwalten Antwortgenerierung, Wiederaufnahme, Artefakte, MySQL-Fallback und optional Redis/Valkey als schnellen Stream-Puffer.	Technische Puffer sind kein langfristiges Inhaltsarchiv; Redis/Valkey enthält keine Klartextinhalte.
Drittanbieter-Komponenten	Werden lokal inventarisiert, geprüft und über Adminfreigaben kontrolliert.	Die Manuals und die App-Oberfläche brauchen keine externen CDN-Ressourcen für den normalen Betrieb.

Für gespeicherte Chats kann die Schule ein **Chat-Aufbewahrungslimit** pro Konto setzen. Ist dieses Limit erreicht, legt Ephraim keinen weiteren gespeicherten Chat an. Bestehende Chats bleiben lesbar, fortsetzbar, exportierbar, umbenennbar und löschbar. Die Nutzerin oder der Nutzer entscheidet im Aufräumdialog, welcher nicht mehr benötigte Chat gelöscht wird; Ephraim löscht alte Chats nicht heimlich im Hintergrund.

> **Datenschutzgedanke:** Ein Limit ist kein Komforthindernis, sondern eine Aufbewahrungsgrenze. Es verhindert, dass private Chaträume, Datenbank und Backups ohne pädagogischen Nutzen dauerhaft wachsen.

5.1.10 Verschlüsselung: der persönliche Vault

Jede Nutzerin und jeder Nutzer hat einen eigenen verschlüsselten Tresor. Ohne Passwort sind gespeicherte Inhalte nicht lesbar.

Situation	Was ist lesbar?
Nur Datenbank- oder Speicher-Backup	Verschlüsselte Inhalte, keine privaten Klartexte ohne Passwort
Angemeldete Person, Vault entsperrt	Eigene Chats, Dateien und Einstellungen
Lehrkraft im Projekt	Projektmaterial und Metadaten — keine privaten Schüler-Chats
Administrator	Konten, Rollen, Systemeinstellungen — kein Klartext fremder Vault-Inhalte
KI-Request auf der Spark	Prompt und freigegebener Kontext nur für diesen Request im Arbeitsspeicher

Persönliche Dateien liegen verschlüsselt auf dem Webserver. Beim Download werden sie im Arbeitsspeicher entschlüsselt und direkt an den Browser gesendet — es entsteht keine dauerhafte Klartext-Kopie auf der Platte. Wird eine private Datei in ein Lehrerprojekt übernommen, entsteht eine getrennte Projektkopie; das private Original bleibt geschützt.

Onboarding, Passwortwechsel und Passwort-Recovery gehören deshalb zur Architektur: Wer sein Passwort kennt, kann den bestehenden Vault neu verpacken. Wer es vergessen hat, braucht eine vorbereitete datenerhaltende Recovery oder muss einen datenlöschenden Reset akzeptieren.

5.1.11 Rollen und Zugriffskontrolle

Rolle	Typische Rechte	Datenschutzrelevant
Schüler:in	Chat, eigene Dateien, Einstellungen, Projektbeitritt	Sieht nur eigene Vault-Inhalte
Lehrkraft	Projekte anlegen, Materialien, Anweisungen für Projekte	Kein Zugriff auf private Schüler-Chats
Administrator	Nutzerverwaltung, Klassen, Quotas, Systemeinstellungen	Verwaltungsdaten, keine fremden Vault-Klartexte
Super-Admin	Technische Betriebsführung, Updates, Audit-Berichte	Administration mit redigierten Protokollen

Anmeldung ist passwortgeschützt; optional mit Zwei-Faktor-Authentifizierung. Aktive Sitzungen können eingesehen und beendet werden.

Besonders wichtig ist die Projektgrenze: Eine Lehrkraft steuert Auftrag, Material, Teilnehmende und Abschlussregeln, aber sie öffnet keine privaten Schülerchats und keine Projektchat-Rohtexte. Sichtbar sind reduzierte Projektinformationen wie Teilnahme, Aktivität, Abschluss, Nachrichtenimpulse, Fazitstatus und die ausdrücklich aktivierten Fazit- oder Monitoring-Ergebnisse. Die Details stehen im Artikel Projekte: Datenschutz.

5.1.12 Wie Datenschutz technisch abgesichert wird

Organisatorisch und architektonisch

- Schulische Plattform — kein öffentlicher Massenmarkt-Chatbot.
- Keine Weitergabe von KI-Anfragen an externe Modellanbieter.

- Datenpfad nachvollziehbar: Schule → isoliertes LAN (getrennt vom Schulnetz) → Spark, nicht anonyme Cloud-Kette.
- Freiwillige Inhalte (Personalisierung, private Quellen) nur bei bewusster Nutzung.

Technische Schutzmaßnahmen

- HTTPS für Browser, für die Verbindung Webserver ↔ Spark (Chat und TTS) sowie für den Mailversand.
- Verschlüsselter persönlicher Vault für Chats und Dateien.
- Strikte Rollentrennung; Lehrkräfte sehen keine privaten Schüler-Chats.
- Geprüfte Datei-Uploads mit Typ-Whitelist und Virenscan.
- Kein werbebasiertes Tracking; datensparsame Sicherheitsprotokollierung.
- Lokale Auslieferung von App- und Handbuch-Ressourcen; Drittanbieter-Komponenten werden inventarisiert und kontrolliert.
- Datenexport und Löschung über „Meine Daten“ bzw. schulische Ansprechpersonen.

Logging

- Der Produktions-Webserver dient nicht als Inhaltsprotokoll: keine Request-Bodies, keine Chatnachrichten, keine Datei-Payloads, keine Cookies und keine Tokens als Diagnosequelle.
- Die Spark protokolliert keine fachlichen KI-Requests, Prompts, Modellantworten, TTS-Texte, Audiodaten oder Schlüsselmaterial.
- Erlaubt sind technische Metriken (z. B. Latenz, Tokenzählungen), gekürzte Statusmeldungen und redigierte Admin-Audits.
- Die Betriebsdetails stehen im Administratorartikel Logging.

> **Wichtig:** KI muss eine Frage kurz im Klartext verarbeiten, um antworten zu können. Der Schutz liegt darin, *wo* (Spark im isolierten LAN, getrennt vom Schulnetz), *wie übertragen* (HTTPS), *wie lange* (nur der Request) und *was gespeichert wird* (verschlüsselt).

Dieses Handbuch beschreibt Ephraim in Version 1. Bei späteren Betriebsänderungen wird der Artikel aktualisiert. Für die Antwortberechnung siehe Inferenz auf der Spark, für Datenarten Datenschutz tiefgehend, für Wartung Cron, für MySQL/Redis im Betrieb Datenbank und für Schlüssel Kryptoarchitektur.

Quelle: web/manuals/systemarchitektur/index.html

5.2 Kryptoarchitektur

> **Dieser Artikel erklärt die Verschlüsselungsarchitektur von Ephraim für Schul-Admins: mit technischer Tiefe, aber ohne vorausgesetztes Kryptografie-Studium. Im Mittelpunkt stehen Nutzer-Vaults, Server-Schlüssel, Projektmaterial, Basiswissen, Sitzungen, 2FA, KI-Jobs, Resetfolgen und die praktischen Betriebsregeln.**

Stand: Juni 2026

5.2.1 Zielbild in einfachen Worten

> **Wer muss das lesen?** Schul-Admins, Super-Admins und technische Betreuerinnen sollten diesen Artikel kennen, bevor sie Passwörter zurücksetzen, Backups bewerten, Projekte löschen, externe Komponenten freigeben oder Sicherheitsvorfälle einordnen. Für normale Nutzer reicht zuerst Datenschutz tiefgehend.

Ephraim verschlüsselt Daten nicht mit einem einzigen Generalschlüssel. Stattdessen gibt es mehrere Schlüsselbereiche. Private Inhalte einer Nutzerin oder eines Nutzers hängen am persönlichen Vault. Geteiltes Projektmaterial hat eigene Projektschlüssel. Serverdaten, die Cron, Worker oder Adminfunktionen unabhängig von einer angemeldeten Person benötigen, werden mit serverseitigen Schlüsseln geschützt.

Das wichtigste Bild ist der verschlossene Umschlag: Der eigentliche Inhalt liegt in einem Umschlag. Der Schlüssel zu diesem Umschlag liegt wiederum in einem zweiten Umschlag, der nur mit dem passenden Passwort oder einem Servergeheimnis geöffnet wird. Dadurch muss bei einem Passwortwechsel nicht jeder Chat neu verschlüsselt werden; Ephraim verpackt nur den Datenschlüssel neu.

> **Admin-Merksatz:** Verschlüsselung ist kein Schalter „an oder aus“. Entscheidend ist, welcher Schlüssel welche Daten schützt und wer diesen Schlüssel zur Laufzeit tatsächlich besitzt.

5.2.2 Begriffe für Nicht-Experten

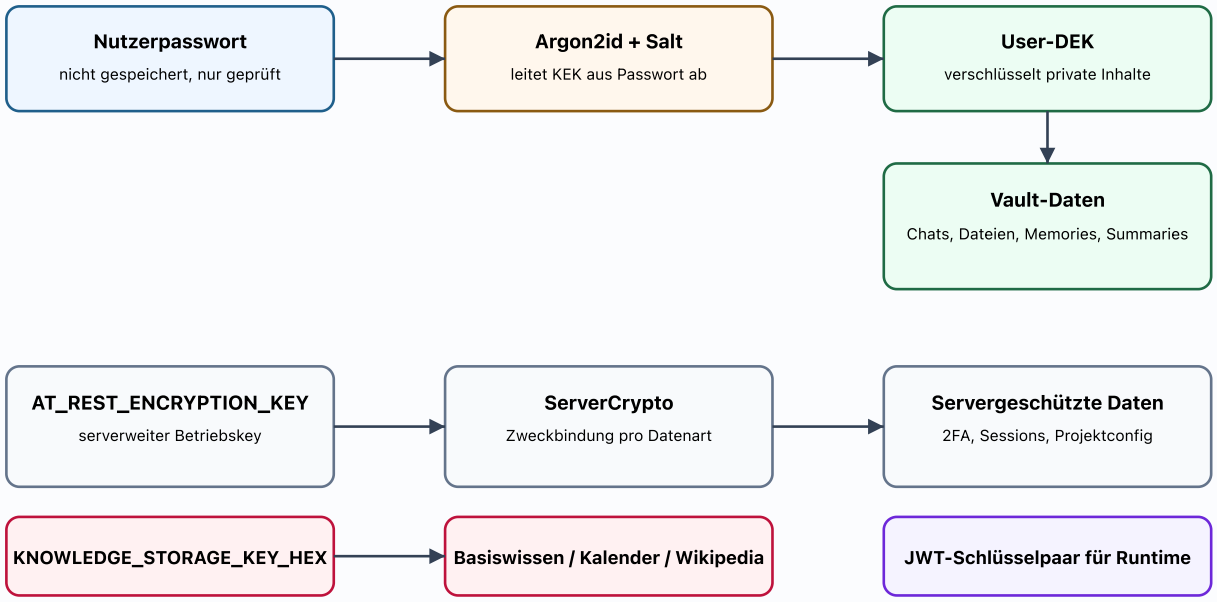
Begriff	Bedeutung in Ephraim	Warum das für Admins wichtig ist
Klartext	Lesbare Daten: Chattext, Dateiname, TOTP-Secret oder Projektauftrag.	Klartext ist nur im Browser, im aktuellen Serverprozess oder im Modellaufruf zulässig.
Ciphertext	Verschlüsselte Daten, die ohne passenden Schlüssel wie Zufall aussehen.	Ein Datenbankdump soll überwiegend Ciphertext enthalten, nicht private Inhalte.
Hash	Ein Einweg-Prüfwert, etwa für Passwörter, Sessiontokens oder Recovery-Codes.	Ein Hash wird geprüft, aber nicht zurückentschlüsselt. Admins lesen daraus kein Passwort.
Salt	Zufälliger Zusatzwert pro Passwortableitung.	Gleiche Passwörter führen nicht zu gleichen abgeleiteten Schlüsseln.
Nonce	Einmalwert für eine Verschlüsselung.	Die Nonce ist nicht geheim, verhindert aber wiederholte Muster im Ciphertext.
DEK	Data Encryption Key: Schlüssel für eigentliche Daten.	Wenn ein DEK verloren ist, sind die damit verschlüsselten Daten praktisch verloren.
KEK	Key Encryption Key: Schlüssel zum Verpacken eines DEK.	Beim Passwortwechsel wird der DEK neu verpackt, nicht jeder Chat neu geschrieben.
AEAD	Verschlüsselung mit eingebauter Integritätsprüfung.	Manipulierte Ciphertexte werden nicht still als „falscher Klartext“ akzeptiert.
Zweckbindung	Ein Ciphertext ist an einen Verwendungszweck gebunden.	Ein TOTP-Secret-Ciphertext passt nicht einfach als Projektkonfiguration.

5.2.3 Datenklassen und Schlüsselmodelle

Ephraim trennt Daten danach, ob sie privat, geteilt oder serverbetrieben sind. Diese Trennung ist wichtiger als die einzelne Tabelle. Für Schul-Admins ist entscheidend, welche Daten ohne Nutzerpasswort verarbeitet werden müssen und welche bewusst nur im Nutzer-Vault liegen.

Datenklasse	Beispiele	Kryptografisches Modell
Privater Nutzer-Vault	Private Chats, Chattitel, persönliche Dateien, Datei-Chunks, Embeddings, Chat-Zusammenfassungen, Erinnerungen, private KI-Artefakte	Zufällige User-DEK, mit passwortabgeleiteter KEK verpackt
Nutzerprivate Kalenderquellen	Private ICS-URLs, private Kalender-Rohdaten, daraus erzeugte Chunks, Embeddings und Termine	Eigener Quellschlüssel pro Kalender, mit der User-DEK des Besitzers verpackt
Projektmaterial	Projektdateien, Projektkopien, Projektdatei-Chunks, Projektmaterial-Embeddings	Projektbezogener Dateischlüssel, serverseitig verpackt
Projektkonfiguration	Projektauftrag, Gesprächsstarter, Builder-Zustand	Serverseitige Verschlüsselung mit Zweck `project-config`
Zentrales Basiswissen	Globale Quellen, Kalender-/Wetterdaten, Wikipedia-Cache, Chunks, Embeddings	Eigener Knowledge-Schlüssel, weil Cron und Retrieval Zugriff brauchen
Sicherheitsgeheimnisse	2FA-Secrets, PHP-Sessions, temporäre Exporte, Projektdatei-Schlüssel	ServerCrypto mit `AT_REST_ENCRYPTION_KEY` und Zweckbindung
Admin-Runtime	Supervisor-/Runtime-WebSocket-Zugriff	Kurzlebige Ed25519-JWTs mit Signatur- und Verifikationsschlüssel

5.2.4 Schlüsselkarte

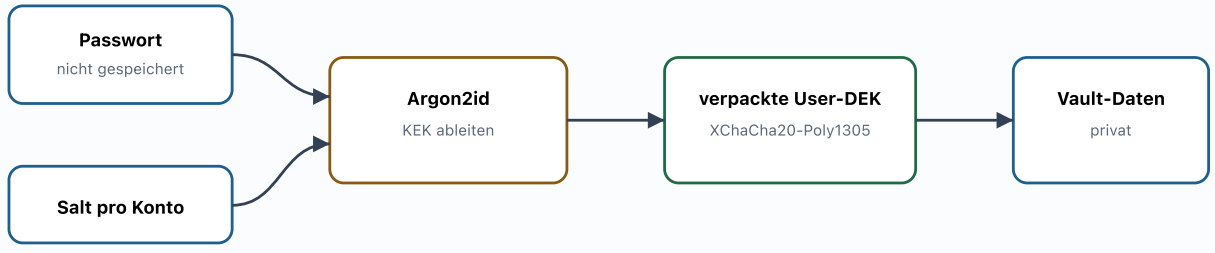


Die Karte zeigt die wichtigsten Schlüsselbereiche. Kein einzelner Datenbankschlüssel öffnet alles; die Datenklassen sind getrennt.

5.2.5 Nutzer-Vault und User-DEK

Für private Inhalte erzeugt Ephraim pro normalem Konto eine zufällige 32-Byte-User-DEK. Diese User-DEK ist der eigentliche Datenschlüssel für private Inhalte. Sie entsteht nicht aus dem Passwort und sie ist auch nicht das Passwort. Das Passwort dient dazu, eine KEK abzuleiten, mit der die User-DEK verpackt wird.

Die Ableitung nutzt Argon2id mit individuellem 16-Byte-Salt, opslimit 3 und 64 MiB Speicherlimit. Für Laien übersetzt: Das System macht die Passwortprüfung für Angreifer absichtlich teuer. Wer nur Datenbankmaterial erbeutet, muss jedes vermutete Passwort mit deutlichem Rechen- und Speicheraufwand ausprobieren.



Das Passwort öffnet nicht direkt den Chat. Es öffnet den verpackten Vault-Schlüssel, und dieser schützt die privaten Inhalte.

5.2.6 Warum bcrypt und Argon2id beide vorkommen

Ephraim nutzt zwei verschiedene Passwortfunktionen für zwei verschiedene Aufgaben. Der bcrypt-Hash dient der Login-Prüfung: Stimmt das eingegebene Passwort zum gespeicherten Hash? Ephraim speichert diesen Hash mit bcrypt und Kostenfaktor 12. Argon2id dient der Schlüsselableitung: Aus dem

eingegebenen Passwort entsteht eine KEK, mit der die User-DEK entpackt wird.

Diese Trennung verhindert eine gefährliche Vereinfachung. Der Passwort-Hash ist kein Datenschlüssel. Wenn sich das Passwort ändert, ändert sich der Login-Hash und die Verpackung der User-DEK. Die privaten Chats selbst bleiben mit derselben User-DEK verschlüsselt.

5.2.7 Passwortwechsel, Recovery und Reset

Beim normalen Passwortwechsel ist der alte Vault bereits entsperrt. Ephraim nimmt die User-DEK aus der aktuellen Sitzung, prüft das alte Passwort, leitet aus dem neuen Passwort eine neue KEK ab und verpackt dieselbe User-DEK neu. Die privaten Inhalte bleiben lesbar. Andere aktive Sitzungen werden über die Session-Version beendet.

Bei vorbereiteter datenerhaltender Recovery ist die User-DEK ebenfalls nicht verloren: Ein aktivierter Wiederherstellungskontakt gibt in einem zeitlich begrenzten Live-Fall kryptografisch frei. Ephraim rekonstruiert daraus nicht ein neues Passwort, sondern verpackt dieselbe User-DEK mit dem neuen Passwort des Kontoinhabers neu. Der Kontakt erhält keinen Klartextzugriff auf Chats, Dateien oder die User-DEK.

Beim datenlöschenden Reset ist die Lage anders. Das alte Passwort steht nicht mehr zur Verfügung und keine gültige datenerhaltende Recovery wird genutzt. Ephraim deaktiviert beim Admin-Reset das alte Passwort sofort; beim Self-Service-Reset geschieht der eigentliche Vault-Neustart beim Setzen des neuen Passworts. Danach entsteht eine neue User-DEK. Alte verschlüsselte Chats und persönliche Dateien werden entfernt, weil Ephraim ohne alte User-DEK keinen vertrauenswürdigen Zugriff auf den alten Vault herstellen soll.

> **Für Admins:** Ein Reset ist ein Notfallzugang zum Konto, keine datenbewahrende Wiederherstellung. Wenn die Person ihr altes Passwort noch kennt, ist der normale Wechsel der richtige Weg. Wenn Recovery vorbereitet ist, ist der datenerhaltende Recovery-Weg dem Reset vorzuziehen.

5.2.8 AEAD, Nonces und Manipulationsschutz

Für viele Text- und Payload-Verschlüsselungen nutzt Ephraim XChaCha20-Poly1305. Das ist ein AEAD-Verfahren: Es verschlüsselt und prüft gleichzeitig, ob der Ciphertext manipuliert wurde. Bei falschem Schlüssel, falscher Zweckbindung oder beschädigtem Inhalt liefert die Entschlüsselung keinen plausiblen Klartext, sondern schlägt fehl.

Viele verschlüsselte Textfelder liegen als `Base64(Nonce || Ciphertext)` vor. Das heißt: Die Nonce steht vor dem eigentlichen Ciphertext und ist mitgespeichert. Das ist korrekt, weil die Nonce kein Geheimnis ist. Secretbox-verschlüsselte Dateinamen verwenden ein eigenes Präfix, damit Ephraim das Format erkennt.

Die Nonce ist dabei kein Passwort und kein geheimer Schlüssel. Sie ist ein Einmalwert, der zusammen mit dem Ciphertext gespeichert wird. XChaCha20 hat eine lange Nonce; zufällige Nonces sind in dieser Architektur praktikabel. Die Sicherheit hängt weiterhin am geheimen Schlüssel, nicht an der Geheimhaltung der Nonce.

5.2.9 Serverseitige Verschlüsselung und Zweckbindung

Nicht alle Daten lassen sich an ein Nutzerpasswort binden. Cron, Worker, Adminfunktionen, Projektverwaltung und 2FA-Prüfung müssen auch dann laufen, wenn kein Nutzer gerade seinen Vault entsperrt hat. Dafür verwendet Ephraim serverseitige At-rest-Verschlüsselung.

Der bevorzugte Betriebskey ist `AT_REST_ENCRYPTION_KEY`. Daraus leitet Ephraim pro Zweck einen eigenen Schlüssel ab. Der Zweck wird zusätzlich als authentifizierter Kontext verwendet. Technisch bedeutet das: Ein Ciphertext für den Zweck `totp-secret` lässt sich nicht einfach als `project-config` verwenden.

Serverseitig verschlüsselte Textwerte tragen ein eigenes internes Präfix. Daran erkennt Ephraim, dass ein Feld bereits im neuen At-rest-Format vorliegt. Das hilft bei Migrationen und verhindert, dass alter Klartext und neuer Ciphertext still verwechselt werden.

Wenn `AT_REST_ENCRYPTION_KEY` fehlt, existiert für Kompatibilität ein Fallback auf `STREAM_DEK_SERVER_KEY`. In Produktion gehört ein expliziter `AT_REST_ENCRYPTION_KEY` gesetzt. Der Entwicklungsmodus darf einen lokalen Ableitungsfallback nutzen; dieser Zustand ist kein Produktionsbetrieb.

5.2.10 Persönliche Dateien im EFR1-Format

Persönliche Dateien liegen nicht als Klartext im Webbereich. Beim Upload wird die temporäre PHP-Uploaddatei validiert und anschließend in das verschlüsselte EFR1-Format geschrieben. EFR1 besteht aus einer Kennung, einem Secretstream-Header und einer Folge von längengepackten Ciphertext-Blöcken. Die Klartext-Blockgröße beträgt 1 MiB.

Element	Funktion
`EFR1`	Formatkennung. Ephraim erkennt daran, dass es sich um eine verschlüsselte Datei im erwarteten Format handelt.
Secretstream-Header	Startwert für die Streaming-Entschlüsselung mit XChaCha20-Poly1305.
4-Byte-Längenfeld	Gibt an, wie lang der nächste verschlüsselte Chunk ist.
Final-Tag	Markiert das echte Dateiende. Fehlt es, gilt die Datei als unvollständig.

Beim Download liest Ephraim den Ciphertext chunkweise, entschlüsselt im RAM und schreibt direkt in die HTTP-Antwort. Es entsteht keine entschlüsselte Downloadkopie auf Platte. Dateinamen werden separat mit Secretbox verschlüsselt gespeichert, damit nicht schon ein Dateiname sensible Informationen verrät.

Datenexporte sind ein Sonderfall: Für die Nutzerin oder den Nutzer entsteht am Ende ein lesbares Exportarchiv, aber die temporäre Datei auf dem Server liegt ebenfalls verschlüsselt. Ephraim verwendet dafür ein servergeschütztes Streaming-Format und löscht abgelaufene oder bereits heruntergeladene Exportdateien wieder.

5.2.11 Projektdateien und Re-Encryption

Projektmaterial ist bewusst geteilt. Deshalb hängt es nicht dauerhaft an der User-DEK der hochladenden Lehrkraft. Ephraim erzeugt für Projektdateien einen eigenen Projektdatei- Schlüssel. Die Datei selbst liegt wieder im EFR1-Format, aber mit dem Projektdatei-Schlüssel. Dieser Schlüssel wird serverseitig mit Zweck `project-file-key` verpackt.

Wenn eine private Datei als Projektmaterial übernommen wird, geschieht kein bloßes Freischalten des privaten Originals. Ephraim entschlüsselt die private Datei im berechtigten Request und verschlüsselt sie direkt als neue Projektkopie mit dem Projektdatei-Schlüssel. Dadurch bleiben privates Original und Projektkopie kryptografisch getrennt.

5.2.12 Basiswissen, Kalender, Wetter und Wikipedia

Zentrales Basiswissen ist kein normaler Nutzer-Vault. Globale Quellen, globale Kalenderquellen, Wetterdaten und Wikipedia-Cache müssen durch Cron, Retrieval und Adminfunktionen verarbeitet werden. Deshalb verwendet Ephraim für diesen Bereich `KNOWLEDGE_STORAGE_KEY_HEX`.

Embedding ist dabei eine Rechenoperation, kein Spark-Speicher. Die Webanwendung sendet freigegebene Klartext-Chunks und konkrete Suchfragen nur für den aktuellen Request an den lokalen SGLang-Embedding-Dienst `school-ui-embedding`. Dieser Dienst berechnet Zahlenvektoren und speichert weder Klartexte noch Suchfragen noch Vektoren dauerhaft. Persistenz findet auf dem Webserver statt: Chunktexte und Embedding-Vektoren werden mit dem passenden Schlüssel verschlüsselt in MySQL gespeichert.

Private Kalenderquellen gehören kryptografisch zum persönlichen Vault des Besitzers. Für jede private Kalenderquelle erzeugt Ephraim einen eigenen Quellschlüssel. Dieser Quellschlüssel wird mit der User-DEK des Besitzers verpackt. Die private ICS-URL, der geladene Kalender-Blob, daraus erzeugte Chunks, Embeddings und Termintexte werden mit diesem Quellschlüssel verschlüsselt. Ohne entsperrten User-Vault kann der Server diese privaten Kalender nicht entschlüsseln und nicht automatisch per Cron aktualisieren.

Sichtbarkeit und Eigentümerbindung bleiben zusätzlich erhalten: Private Kalender werden nur im normalen Chat und auf der Startseite des Besitzers verwendet, nicht im Projektchat und nicht im Projektbuilder. Zentrale Kalenderquellen sind davon getrennt: Sie sind Teil des administrativ freigegebenen Basiswissens und werden mit dem Knowledge-Schlüssel geschützt, damit Cron, Retrieval und Adminfunktionen sie ohne entsperrten Nutzer-Vault warten können.

> **Wichtig:** „Nutzerprivat" bedeutet bei privaten Kalendern auch kryptografisch nutzervault-gebunden. Zentrales Basiswissen bleibt dagegen bewusst serverseitig verschlüsselt, weil es ohne konkrete Nutzersitzung wartbar sein muss.

5.2.13 Sitzungen, Sessiondaten und Sitzungsliste

Eine angemeldete Sitzung ist der Moment, in dem private Daten verarbeitet werden dürfen. Deshalb schützt Ephraim serverseitige PHP-Sessiondateien mit ServerCrypto und Zweck `php-session`. In solchen Sessions liegen temporär sicherheitsrelevante Werte, zum Beispiel CSRF-Kontext und die entsperrte User-DEK.

Die Sitzungsliste speichert keine Klartext-Sessiontokens. Ein zufälliger Sitzungstoken wird gehasht. Geräte- und Zugriffshinweise werden für die Anzeige gekürzt oder mit einem Privacy-Hash versehen. Die Session-Version ist der harte Schalter: Nach Passwortwechsel, Reset oder 2FA-Reset werden alte Sitzungen ungültig.

5.2.14 Zwei-Faktor-Authentifizierung

TOTP basiert auf einem gemeinsamen Geheimnis zwischen Ephraim und der Authenticator-App. Dieses Geheimnis ist wertvoller als ein einzelner sechsstelliger Code. Ephraim erzeugt ein 160-Bit-Secret, zeigt es als QR-Code oder manuelle Eingabe an und speichert es anschließend serverseitig verschlüsselt mit Zweck `totp-secret`.

Die laufenden TOTP-Codes folgen dem üblichen Muster: 30-Sekunden-Zeitschritte, sechs Ziffern, HMAC-SHA1. Recovery-Codes werden anders behandelt: Ephraim erzeugt acht Codes, speichert nur SHA3-512-Hashes und entfernt einen Code nach erfolgreicher Nutzung. Ein Admin kann Recovery-Codes deshalb nicht auslesen. Ein 2FA-Reset löscht das alte Secret und die alten Recovery-Codes und zwingt zur Neueinrichtung.

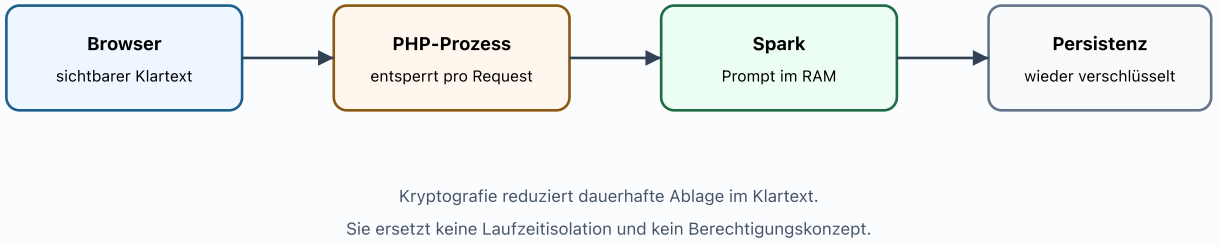
5.2.15 KI-Jobs, Streaming und kurzlebige Puffer

KI-Antworten laufen über Jobs und Streams. Der Browser sieht eine fortlaufende Antwort, technisch gibt es aber Queue-Einträge, Stream-Chunks und Reconnect-Snapshots. Private oder nutzerabgeleitete Job-Payloads liegen mit der User-DEK verschlüsselt. Die technischen JSON-Felder enthalten dann nur Metadaten; eigentliche Prompt- und Ergebnisblobs liegen in verschlüsselten Payloadfeldern.

Der Worker benötigt zur Verarbeitung zeitweise Zugriff auf den passenden Vault-Kontext. Dafür speichert Ephraim eine serverseitig verschlüsselte Kopie des Job-DEK-Zugangs, nicht den Klartext im normalen Job-Payload. Stream-Chunks werden zusätzlich mit einem jobbezogenen Stream-Token geschützt und haben kurze Lebensdauer. Das Ziel ist: Reconnect und Streaming funktionieren, ohne dauerhafte Klartextpuffer anzulegen.

5.2.16 Wo Klartext unvermeidlich ist

Verschlüsselung schützt ruhende Daten. Sie verhindert nicht, dass Daten dort im Klartext entstehen, wo sie fachlich verarbeitet werden müssen. Der Browser muss einen Chat anzeigen. Der PHP-Prozess muss beim Speichern oder Exportieren entschlüsseln. Die Spark muss den Prompt lesen, um eine Antwort zu berechnen.



Kryptografie reduziert dauerhafte Ablage im Klartext.
Sie ersetzt keine Laufzeitisolation und kein Berechtigungskonzept.

Für die Berechnung einer Antwort gibt es kurze Klartextphasen. Danach erfolgt die dauerhafte Ablage wieder verschlüsselt oder als begrenzte Metadaten.

5.2.17 Runtime-JWTs und Supervisor-Webterminal

Administrative Runtime-Zugriffe verwenden kurzlebige JWTs mit Ed25519-Signatur. Das Frontend signiert mit `JWT_SIGNING_KEY_HEX`. Runtime und Supervisor prüfen mit dem passenden öffentlichen Schlüssel, der dort als `RUNTIME_WS_TOKEN_PUBLIC_KEY` erwartet wird. In Ephraim selbst muss `JWT_VERIFY_KEY_HEX` zu diesem öffentlichen Schlüssel passen.

Signatur bedeutet: Die Runtime prüft, ob das Token wirklich vom berechtigten Frontend stammt und ob Audience und Issuer passen. Das Token verschlüsselt nicht den Inhalt des WebSocket-Verkehrs; dafür ist die Transport- und Netzwerkinfrastruktur zuständig. Die Signatur schützt die Berechtigung, nicht den gesamten Datenkanal.

Typische Fehlkonfigurationen zeigen sich hart: Der Token-Endpunkt liefert einen Fehler oder die Runtime lehnt das Token ab. Mehrere Frontends, die dieselbe Spark ansprechen, müssen dieselbe Schlüsselpaarlogik verwenden oder die Runtime muss den passenden neuen Public Key erhalten.

5.2.18 Zugangslinks, Resetlinks und Nonces

Initiallinks, Admin-Resetlinks und Self-Service-Resetlinks sind keine Passwortarchive. Ephraim speichert nicht einfach eine Liste wiederanzeigbarer Klartextlinks. Gespeichert werden Token-Hashes, Nonces, Zweck, Ablaufzeit und Nutzungsstatus. Der vorgelegte Link wird geprüft, nicht aus der Datenbank rekonstruiert.

Die Zwecke sind getrennt: Initialzugang, Admin-Reset, Self-Service-Bestätigung, datenerhaltender Recovery-Fall und Passwort-Setzlink. Self-Service-Anfragen antworten neutral, damit das Formular keine Konten verrät. Admin-Resetlinks deaktivieren das alte Passwort sofort. Resetlinks und Recovery-Fälle sind kurzlebig und je nach Policy einmalig nutzbar.

5.2.19 MCP-OAuth und Admin-Werkzeuge

Der MCP-Server ist ein Admin-Werkzeug und gehört nicht zur normalen Schüler- oder Lehrkraftnutzung. Er arbeitet mit OAuth und PKCE. Zugriffstokens werden nur gehasht gespeichert und mit Scopes begrenzt. Ein Client erhält also nicht automatisch alle Adminrechte, sondern nur die angeforderten und genehmigten Berechtigungen.

Für Admins ist dabei entscheidend: Kryptografie schützt den Tokenbestand, aber Scopes, Bestätigungsabfragen und Auditspuren sind genauso wichtig. Ein gültiger Token mit Schreibscope ist ein echtes Machtmittel und muss widerrufen werden, wenn ein Client nicht mehr vertrauenswürdig ist.

5.2.20 Backups, Restore und Schlüsselverlust

Verschlüsselung macht Backups nicht unwichtig, sondern anspruchsvoller. Datenbank, Dateispeicher und Betriebsgeheimnisse müssen zusammen betrachtet werden. Wer nur die Datenbank sichert, verliert Dateiinhalte. Wer Dateispeicher und Datenbank sichert, aber die Serverkeys verliert, verliert servergeschützte Inhalte. Wer Serverkeys ungeschützt in dasselbe Backup legt, schwächt die Schutzwirkung.

Verlust	Konsequenz
Nutzerpasswort vergessen	Mit vorbereiteter Recovery bleibt der Vault erhalten; ohne Recovery erzeugt der Reset einen neuen Vault und entfernt alte private Vault-Daten.
`AT_REST_ENCRYPTION_KEY` verloren	Servergeschützte Daten wie 2FA-Secrets, Projektkonfiguration und Projektdatei-Schlüssel sind nicht zuverlässig lesbar.
`KNOWLEDGE_STORAGE_KEY_HEX` verloren	Verschlüsselte Knowledge-Daten sind unlesbar; Quellen müssen aus Originalen neu aufgebaut werden, soweit vorhanden.
`STREAM_DEK_SERVER_KEY` verloren	Aktive Stream-/Job-Geheimnisse und Fallback-abhängige Daten sind betroffen.
JWT-Schlüsselpaar falsch kopiert	Admin-Runtime-Tokens werden nicht akzeptiert oder können nicht erzeugt werden.

Schlüsselrotation bedeutet nicht „neuen Wert eintragen und fertig“. Betroffene Daten müssen mit dem alten Schlüssel gelesen und mit dem neuen Schlüssel neu verpackt werden, oder das System muss während einer Übergangszeit mehrere Schlüsselgenerationen verstehen.

5.2.21 Was Admins lesen und nicht lesen

Normale Adminfunktionen geben keinen Klartextzugriff auf fremde private Vault-Inhalte. Lehrkräfte lesen keine privaten Schülerchats und auch keine Projektchat-Rohtexte einzelner Teilnehmender. Super-Admins verwalten System, Nutzer, Klassen, Quoten, Basiswissen, Runtime und Sicherheitszustände; sie erhalten damit technische Macht, aber nicht automatisch eine Lesefunktion für fremde Vaults.

Gleichzeitig schützt keine Kryptografie gegen eine vollständig kompromittierte Serverlaufzeit. Wer Serverprozess, Konfiguration und Code kontrolliert, kontrolliert die Stelle, an der Klartext legitimerweise verarbeitet wird. Deshalb gehören Betriebsschutz, 2FA für Admins, minimale Adminrollen, Patchpflege, Auditlogs und gute Backupdisziplin zur Kryptoarchitektur dazu.

5.2.22 Typische Angriffsbilder

Szenario	Schutzwirkung	Restproblem
Datenbankdump ohne Secrets	Private Inhalte, TOTP-Secrets, Projektkonfigurationen und Knowledge-Inhalte liegen überwiegend verschlüsselt oder gehasht vor.	Metadaten wie Rollen, Zeitpunkte und Größen bleiben sichtbar.
Dateispeicher-Dump ohne Datenbank	EFR1-Dateien sehen wie Ciphertext aus; Zufallsnamen erschweren Zuordnung.	Größen und Mengen bleiben erkennbar.
DB plus Serverkeys, aber ohne Nutzerpasswörter	Servergeschützte Daten sind gefährdet; private User-Vaults brauchen weiterhin User-DEK oder Passwort.	Aktive Sessions oder Jobzustände können die Lage verschärfen.
Voll kompromittierter Server zur Laufzeit	At-rest-Krypto begrenzt nur alte/offline Datenbestände.	Laufzeitklartext, Secrets und Codepfade sind unter Kontrolle des Angreifers.
Gestohlener Resetlink	Ablaufzeit, Einmalnutzung, Zweckbindung und Token-Hashing begrenzen Missbrauch.	Vor Ablauf bleibt der Link sensibel und muss widerrufen werden.

5.2.23 Betriebsregeln für Schul-Admins

- Produktivsysteme brauchen eigene, zufällig erzeugte Werte für `AT_REST_ENCRYPTION_KEY`, `KNOWLEDGE_STORAGE_KEY_HEX`, `STREAM_DEK_SERVER_KEY` und das Runtime-JWT-Schlüsselpaar.

- Secrets gehören in die geschützte Serverkonfiguration oder ins Environment, nicht in Git, Tickets, Chatprotokolle oder Screenshots.
- Backups müssen Datenbank, Dateispeicher und benötigte Schlüssel abdecken; Schlüsselkopien brauchen strengere Zugriffsregeln als normale Datenbackups.
- Passwortresets werden nur als Notfallpfad genutzt. Bei bekanntem Passwort ist der normale Passwortwechsel der richtige Weg.
- 2FA für Admin- und Super-Admin-Konten ist nicht Komfort, sondern Grundschutz.
- Fehlersuche beginnt mit Status, Metadaten und Auditereignissen. Private Klartexte gehören nicht in Logs.
- Vor Schlüsselrotation oder Migration wird geprüft, welche Datenklasse betroffen ist und ob ein Rewrap-Pfad existiert.

5.2.24 Fehlersuche ohne Datenschutzschaden

Viele Krypto-Fehler sehen für Nutzer gleich aus: Daten lassen sich nicht laden, 2FA klappt nicht, Projektmaterial wirkt unlesbar oder die Runtime lehnt Tokens ab. Admins sollten zuerst die betroffene Datenklasse bestimmen. Ein Problem im Nutzer-Vault ist anders zu behandeln als ein Problem im Knowledge-Schlüssel oder im JWT-Schlüsselpaar.

Symptom	Wahrscheinlicher Bereich	Datensparsame Prüfung
Nutzer kann private Chats nach Login nicht lesen	User-DEK, Passwort, Vault-Entsperrung	Login- und Resetverlauf, Sessionzustand und Fehlermeldungen prüfen; keine Chatklartexte in Logs schreiben.
2FA funktioniert nach Serverumzug nicht	`AT_REST_ENCRYPTION_KEY` oder Secret-Migration	Konfiguration und 2FA-Status prüfen; bei Bedarf 2FA zurücksetzen statt Secret auszulesen.
Basiswissen liefert keine Treffer	Knowledge-Schlüssel, Index, Retrieval, Sichtbarkeit	Quellenstatus, Indexstatus und Rechte prüfen; Quelltexte nicht unnötig in Logs kopieren.
Projektdateien sind unlesbar	Projektdatei-Schlüssel oder Projektdateispeicher	Projektdatei-Metadaten, Speicherpfad und Serverkey-Verfügbarkeit prüfen.
Admin-Webterminal verbindet nicht	JWT-Schlüsselpaar, Audience, Issuer, Runtime-Public-Key	Token-Endpunkt, Runtime-Konfiguration und Uhrzeit prüfen.

5.2.25 Grenzen der Kryptografie

Kryptografie ist ein starker Baustein, aber sie löst nicht jede Sicherheitsfrage. Sie schützt keine Inhalte, die eine berechtigte Person gerade im Browser sieht. Sie verhindert keine falsche Freigabeentscheidung in einem Projekt. Sie ersetzt keine Rollenprüfung. Und sie schützt nicht vor einem Angreifer, der den laufenden Serverprozess samt Konfiguration kontrolliert.

Die richtige Erwartung lautet deshalb: At-rest-Verschlüsselung reduziert Schäden bei Datenbank-, Dateispeicher- und Backupabflüssen deutlich. Sie erzwingt aber weiterhin sauberen Betrieb, minimale Rechte, Protokollhygiene, sichere Adminzugänge und transparente Fachentscheidungen.

Diese Seite beschreibt die Kryptoarchitektur als Betriebswissen. Für Admins ist nicht entscheidend, jede Formel zu kennen, sondern Datenklassen, Schlüssel, Resetfolgen und Klartextgrenzen zuverlässig auseinanderzuhalten. Kürzere Begriffserklärungen stehen im Glossar.

Quelle: web/manuals/kryptoarchitektur-tiefgehend/index.html

5.3 Vault

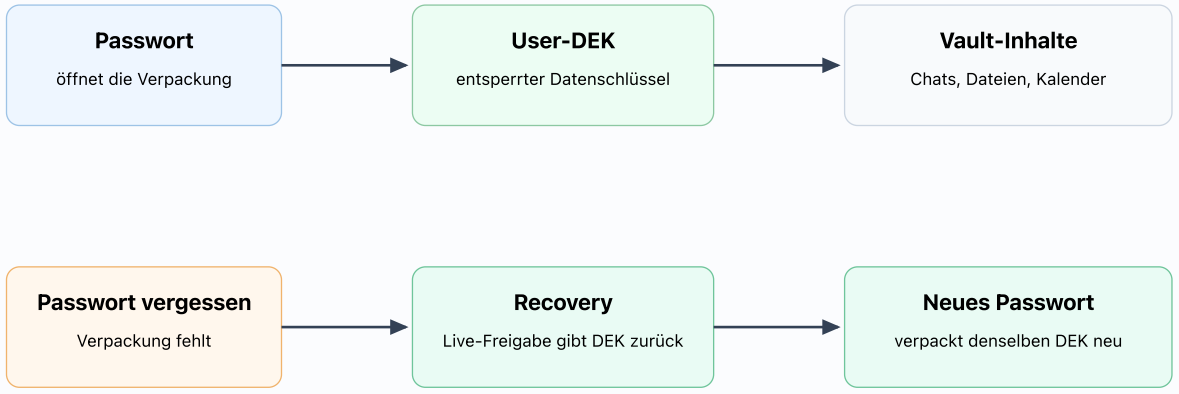
> Der persönliche Vault ist der verschlüsselte Datenraum deines Ephraim-Kontos. Er schützt private Chats, Dateien, Erinnerungen, persönliche Kalender und daraus erzeugte Zusammenfassungen. Er erklärt auch, warum Passwörter in Ephraim wichtiger sind als bei einfachen Schulportalen.

Stand: Juni 2026

5.3.1 Die Grundidee

Ein normales Webkonto besteht oft nur aus Benutzername, Passwort und einigen gespeicherten Daten. Ephraim trennt stärker: Das Konto enthält Stammdaten und Berechtigungen, der Vault enthält private Inhalte. Der Vault besitzt einen eigenen Datenschlüssel, den User-DEK. DEK bedeutet Data Encryption Key, also Datenschlüssel.

Dein Passwort ist nicht dieser Datenschlüssel. Das Passwort verpackt den Datenschlüssel. Beim Login prüft Ephraim dein Passwort und öffnet damit die Verpackung des User-DEK für die aktuelle Sitzung. Erst danach sind private Inhalte lesbar und beschreibbar.



Der Vault-Schlüssel bleibt bei Passwortwechsel und datenerhaltender Recovery derselbe. Nur seine Passwort-Verpackung wird erneuert.

5.3.2 Was im Vault liegt

Der Vault schützt Inhalte, die fachlich zu deinem privaten Arbeitsbereich gehören. Ephraim speichert diese Inhalte nicht als Klartext in der Datenbank oder als normale Dateien im Dateisystem.

Inhalt	Schutz im System
Normale Chats, Chat-Titel und Nachrichten	Werden im persönlichen Vault verschlüsselt gespeichert und nur mit entsperrtem Vault gelesen.
Chat-Zusammenfassungen für „Mein Ephraim“ und spätere Wiederaufnahme	Liegen verschlüsselt in den Summary-Tabellen und werden aus dem entsperrten Vault-Kontext erzeugt.
Persönliche Dateien und Dateinamen	Werden chunkweise verschlüsselt gespeichert; auch Anzeigenamen sind geschützt.
Persönliche Erinnerungen	Werden mit dem User-DEK verschlüsselt und nur für das eigene Konto genutzt.
Private Kalenderabos	Nutzen eigene Quellschlüssel, die mit dem User-DEK verpackt sind. URL, Rohdaten, Chunks, Embeddings und Termine bleiben vaultgebunden.
Private KI-Artefakte	Werden bei vorhandenem User-DEK verschlüsselt gespeichert, damit spätere Ansichten nicht auf Klartext-Persistenz angewiesen sind.

Wenn aus Vault-Inhalten Embeddings entstehen, ist die Spark nur der lokale Rechner: `school-ui-embedding` wandelt freigegebene Klartextabschnitte kurz in Zahlenvektoren um und speichert daraus nichts dauerhaft. Gespeichert werden die Chunks und Embedding-Vektoren wieder verschlüsselt in Ephraim.

5.3.3 Was nicht im Vault liegt

Nicht alle Daten eines Kontos gehören in den Vault. Ephraim muss Rollen, Klassen, Sperrstatus, Sitzungsstatus und technische Sicherheitsereignisse verwalten, auch wenn der private Vault gerade nicht entsperrt ist.

- Benutzername, Anzeigename, Rolle, Klasse und E-Mail-Adresse liegen als Kontodaten außerhalb des Vaults.
- Passwörter liegen nie im Klartext vor. Gespeichert wird ein Passwort-Hash.
- 2FA-Geheimnisse sind serverseitig verschlüsselt; 2FA-Recovery-Codes liegen nur gehasht vor.
- Globale Wissensquellen der Schule verwenden einen serverseitigen Knowledge-Schlüssel, nicht deinen User-DEK.
- Administrationsprotokolle speichern technische Vorgänge, aber keine Chatinhalte, Passwörter oder Vault-Schlüssel.

5.3.4 Was beim Login passiert

Beim Login prüft Ephraim dein Passwort gegen den gespeicherten Passwort-Hash. Danach wird der mit deinem Passwort verpackte User-DEK entsperrt und in der serverseitigen Sitzung gehalten. Die PHP-Sessiondateien selbst sind verschlüsselt, damit der Schlüssel nicht als Klartext in normalen Sessiondateien liegt.

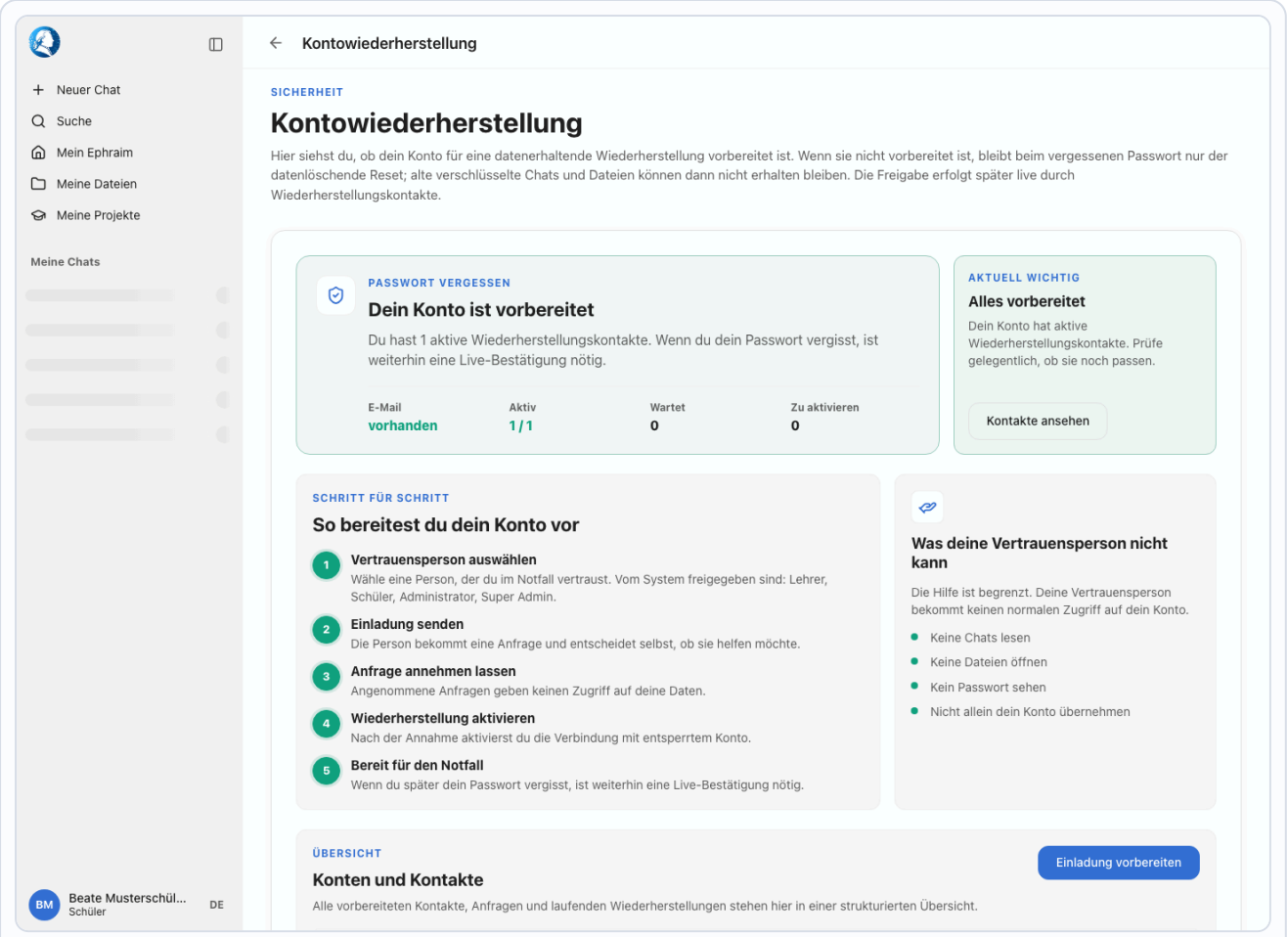
Diese Sitzung ist der praktische Arbeitszustand: Chat speichern, Datei herunterladen, private Kalender im Chat nutzen, „Mein Ephraim“ zusammenfassen und Daten exportieren brauchen den entsperrten User-DEK. Ohne entsperrten Vault antworten diese Funktionen mit einem Sperrhinweis oder liefern keine privaten Inhalte.

> **Wichtige Grenze:** Der Vault ist kein Versprechen, dass der Ephraim-Server während deiner Nutzung niemals Klartext sieht. Während du eine Nachricht sendest, muss die Anwendung sie im Arbeitsspeicher lesen, um sie an die lokale Spark zu übergeben und die Antwort wieder in deinen Vault zu schreiben. Der Schutz richtet sich gegen Klartextspeicherung, Datenbankeinblick, Dateisystemeinblick, Backups ohne Schlüssel und unberechtigte UI-Zugriffe.

5.3.5 Warum das Passwort Folgen für Daten hat

Solange du dein aktuelles Passwort kennst, ist der Wechsel einfach: Der User-DEK ist in der aktuellen Sitzung entsperrt und wird mit dem neuen Passwort neu verpackt. Die verschlüsselten Daten bleiben unverändert.

Wenn du das Passwort vergessen hast, fehlt die normale Verpackung des User-DEK. Dann gibt es genau zwei technische Wege: vorbereitete datenerhaltende Recovery oder datenlöschender Reset. Die Recovery bringt denselben User-DEK über einen Live-Fall zurück. Der Reset erzeugt einen neuen User-DEK und entfernt alte private Vault-Daten.

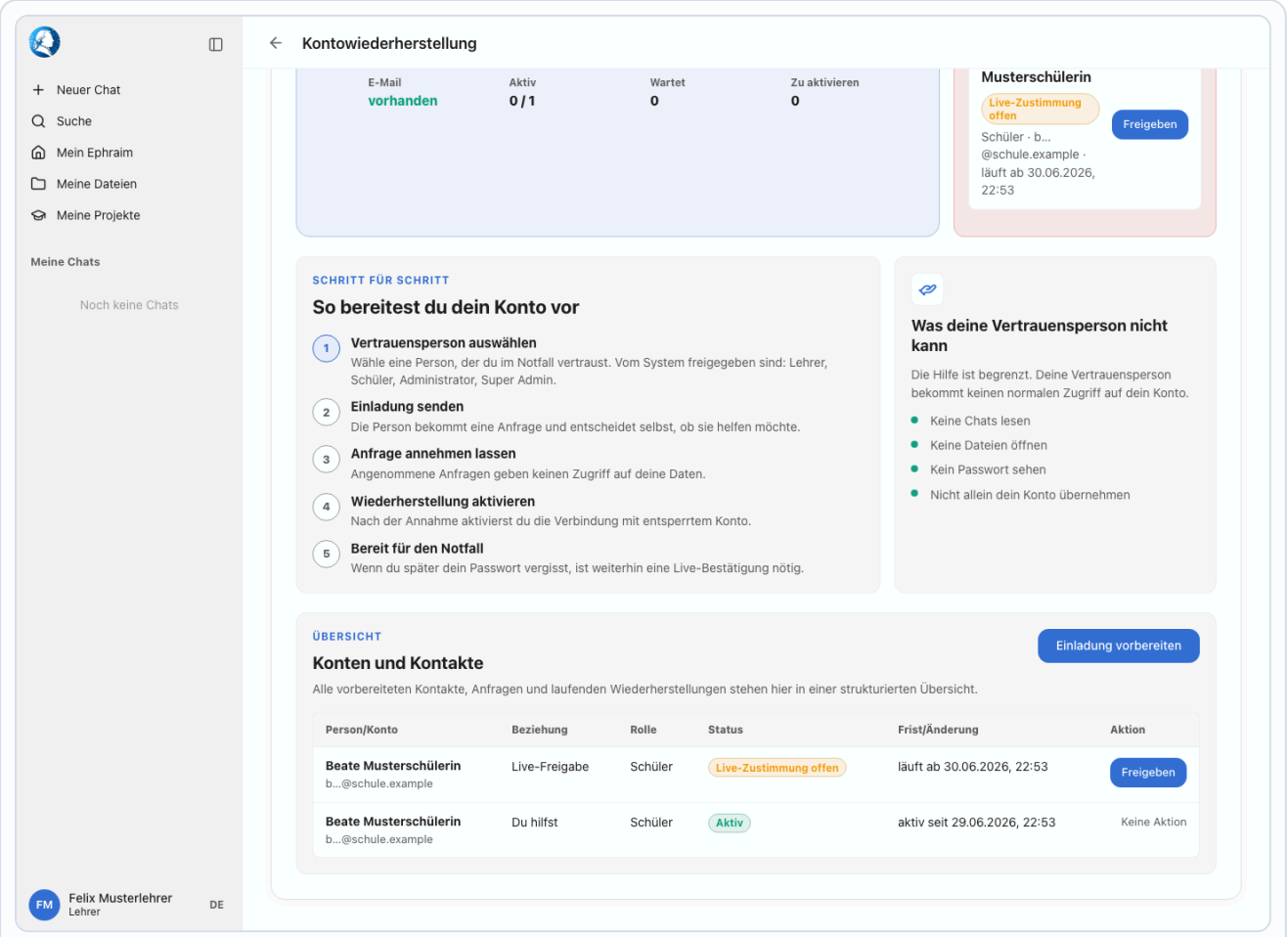


„Bereit“ bedeutet: Für dieses Konto ist eine datenerhaltende Recovery vorbereitet. Der bestehende Vault muss dann bei vergessenem Passwort nicht verworfen werden.

5.3.6 Was Recovery kryptografisch leistet

Die Vorbereitung besteht aus drei Schritten: Einladung, Annahme und Aktivierung. Erst bei der Aktivierung durch den Kontoinhaber mit entsperrtem Vault erzeugt Ephraim das Recovery-Material. Der Kontakt bekommt dabei keinen Klartextschlüssel. Ephraim verschlüsselt einen Recovery-Anteil für den öffentlichen Schlüssel des Kontakts.

Im Ernstfall startet der Kontoinhaber eine Passwort-Wiederherstellung. Ephraim erzeugt einen zeitlich begrenzten Live-Fall. Der Kontakt meldet sich an, entsperrt seinen eigenen Vault, prüft die Identität außerhalb der App und gibt frei. Die API liefert den rekonstruierten Owner-DEK nicht an den Kontakt aus. Sie verpackt ihn für den laufenden Recovery-Fall. Danach setzt der Kontoinhaber ein neues Passwort, und Ephraim rewrappt den bestehenden User-DEK mit diesem neuen Passwort.



Die Freigabe ist live, kontogebunden und bewusst. Ein Wiederherstellungskontakt erhält keinen Einblick in fremde Inhalte.

5.3.7 Vault und Projekte

Projekte sind fachlich eigene Arbeitsräume. Projektmaterialien, Projektauftrag und Lehrkraft-Konfiguration gehören zum Projekt. Persönliche Projektchat-Verläufe, persönliche Fazite und nutzerbezogene Folgeartefakte bleiben dagegen an den jeweiligen Vault gebunden.

Daraus folgt: Eine Lehrkraft liest keine privaten Projektchats eines Schülers. Lehrkräfte arbeiten mit den vorgesehenen Fazit- und Beobachtungsfunktionen. Diese erzeugen verdichtete Auswertungen unter den im Projekt gesetzten Regeln; sie öffnen nicht einfach den Vault eines Lernenden.

5.3.8 Export, Suche und Löschung

Datenexport, Suche, Dateidownload und Chatwiederaufnahme brauchen einen entsperrten Vault. Der Export ist eine bewusste Klartext-Ausleitung: Ephraim entschlüsselt deine Daten für den Download, erstellt bei Dateioption ein temporäres verschlüsseltes ZIP außerhalb des Webroots und löscht dieses ZIP nach dem Download wieder.

Bei Kontolöschung entfernt Ephraim Konto- und Vault-Daten. Bei datenlöschendem Reset entfernt Ephraim alte private Vault-Daten und erzeugt einen neuen Vault. Bei datenerhaltender Recovery bleibt der alte Vault erhalten.

Für die Ersteinrichtung siehe Onboarding. Für die praktische Schritt-für-Schritt-Erklärung siehe Passwörter. Für die tiefe technische Admin-Sicht siehe Kryptoarchitektur.

Quelle: web/manuals/konto-vault/index.html

5.4 Onboarding

> Das Onboarding ist die freiwillige Ersteinrichtung deines normalen Ephraim-Kontos. Es hilft dir, Ephraim persönlicher zu machen, einen Kalender zu verbinden und rechtzeitig an die datenerhaltende Kontowiederherstellung zu denken. Es ist kein Test, keine Sperre und kein verstecktes Profiling.

Stand: Juni 2026

5.4.1 Kurzfassung

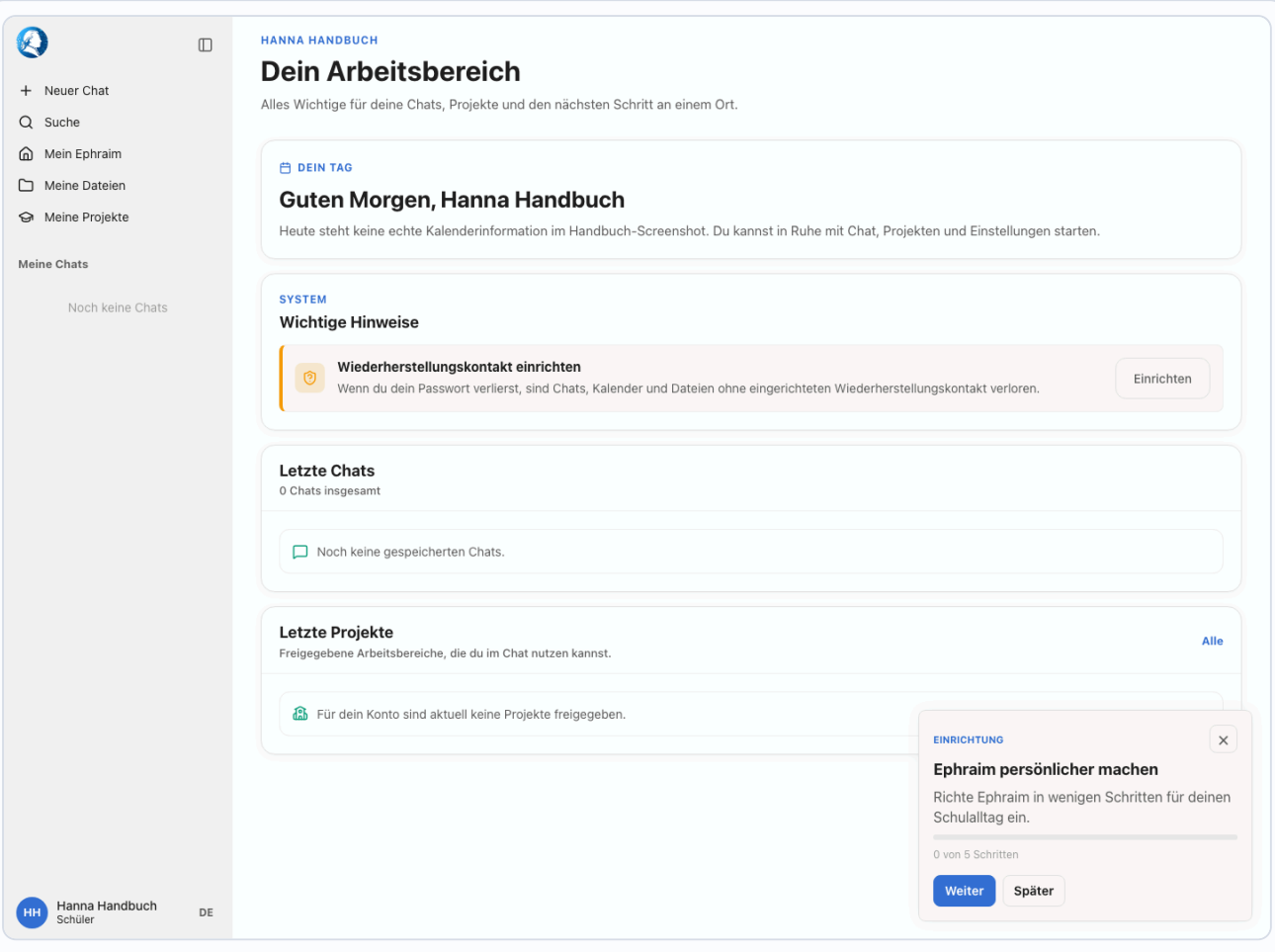
Nach der ersten Anmeldung kann unten rechts eine Einrichtungskarte erscheinen. Sie führt durch Start, Personalisierung, Kalender, Kontowiederherstellung und Abschluss. Jeder Schritt ist freiwillig. Du kannst den Dialog öffnen, später fortsetzen oder dauerhaft ausblenden.

Die Angaben bleiben kontobezogen: Name, Interessen und Gesprächsstil gehören zur Personalisierung; Kalenderabos gehören zu deinen privaten Quellen; Recovery wird im Sicherheitsbereich eingerichtet. Projektchats, Fazit & Beobachtung und fremde Konten werden dadurch nicht geöffnet.

> **Merksatz:** Onboarding ist eine Abkürzung zu wichtigen Einstellungen. Es ersetzt nicht die späteren Konto-Einstellungen und nimmt dir keine Entscheidung ab.

5.4.2 Einstieg auf „Mein Ephraim“

Die Karte erscheint im normalen Arbeitsbereich, wenn dein Konto noch aktiv eingerichtet werden kann. Der Fortschrittsbalken zeigt, wie viele Schritte erledigt sind. Mit **Loslegen** öffnest du den Dialog. **Später** blendet die Karte für eine Zeit aus. Das Kreuz bedeutet: nicht mehr anzeigen.

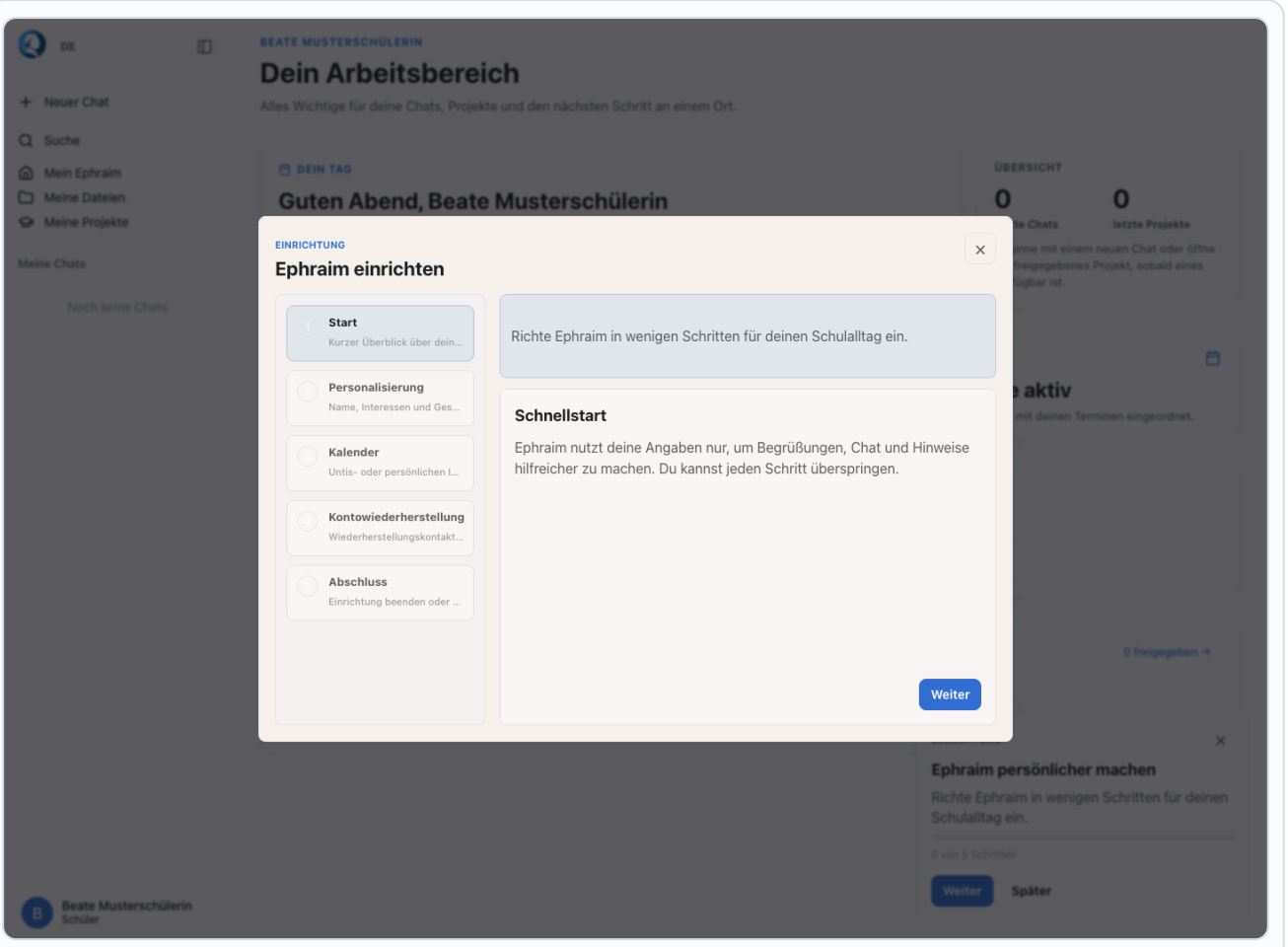


Die Karte liegt über dem Arbeitsbereich. Sie verändert keine Daten, solange du keinen Schritt speicherst oder abschließt.

Wenn die Karte nicht erscheint, ist das kein Fehler. Der Einrichtungsstatus ist dann abgeschlossen, ausgesetzt, ausgeblendet oder für diese Art von Zugang nicht vorgesehen. Projektcode-Gäste haben keinen vollständigen Konto-Arbeitsbereich und sehen deshalb nicht dieselbe Ersteinrichtung wie normale Konten.

5.4.3 Start und Orientierung

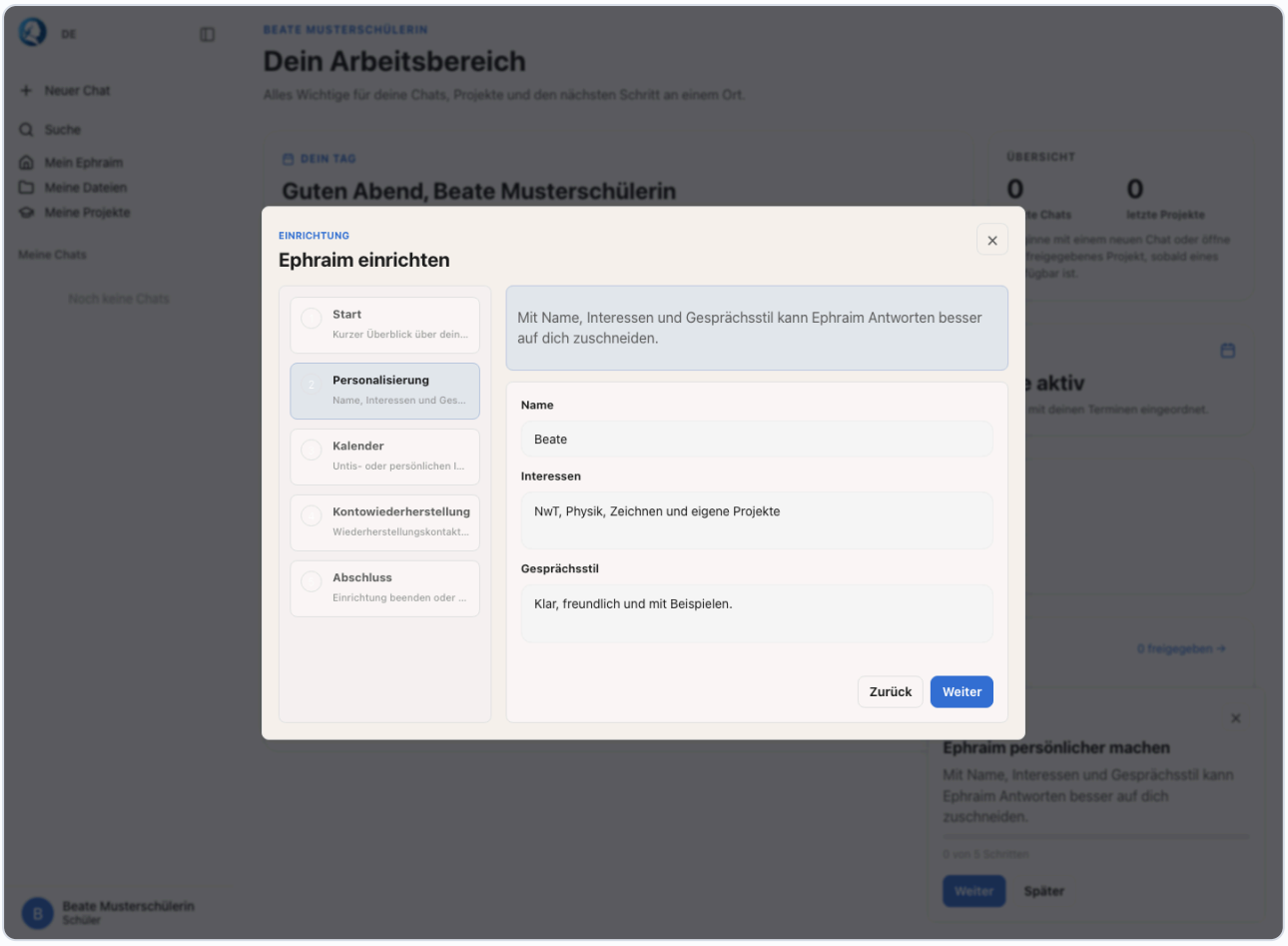
Der Startschritt erklärt den Zweck des Dialogs: Ephraim soll Begrüßung, Chat und Hinweise besser an deinen Schulalltag anpassen. Der Dialog behauptet nicht, dass du alles eintragen musst. Die Schrittliste links ist direkt anklickbar; du kannst also gezielt zu dem Punkt springen, den du einrichten möchtest.



Der Start ist bewusst kurz. Die eigentliche Entscheidung liegt in den folgenden Schritten.

5.4.4 Personalisierung

In der Personalisierung trägst du ein, wie Ephraim dich ansprechen und in persönlichen Chats unterstützen soll. Der Name ist der bevorzugte Name im Gespräch. Interessen helfen, Beispiele passender zu wählen. Der Gesprächsstil beschreibt, ob Antworten etwa knapp, ausführlich, ruhig, direkt, prüfend oder mit vielen Beispielen formuliert werden sollen.

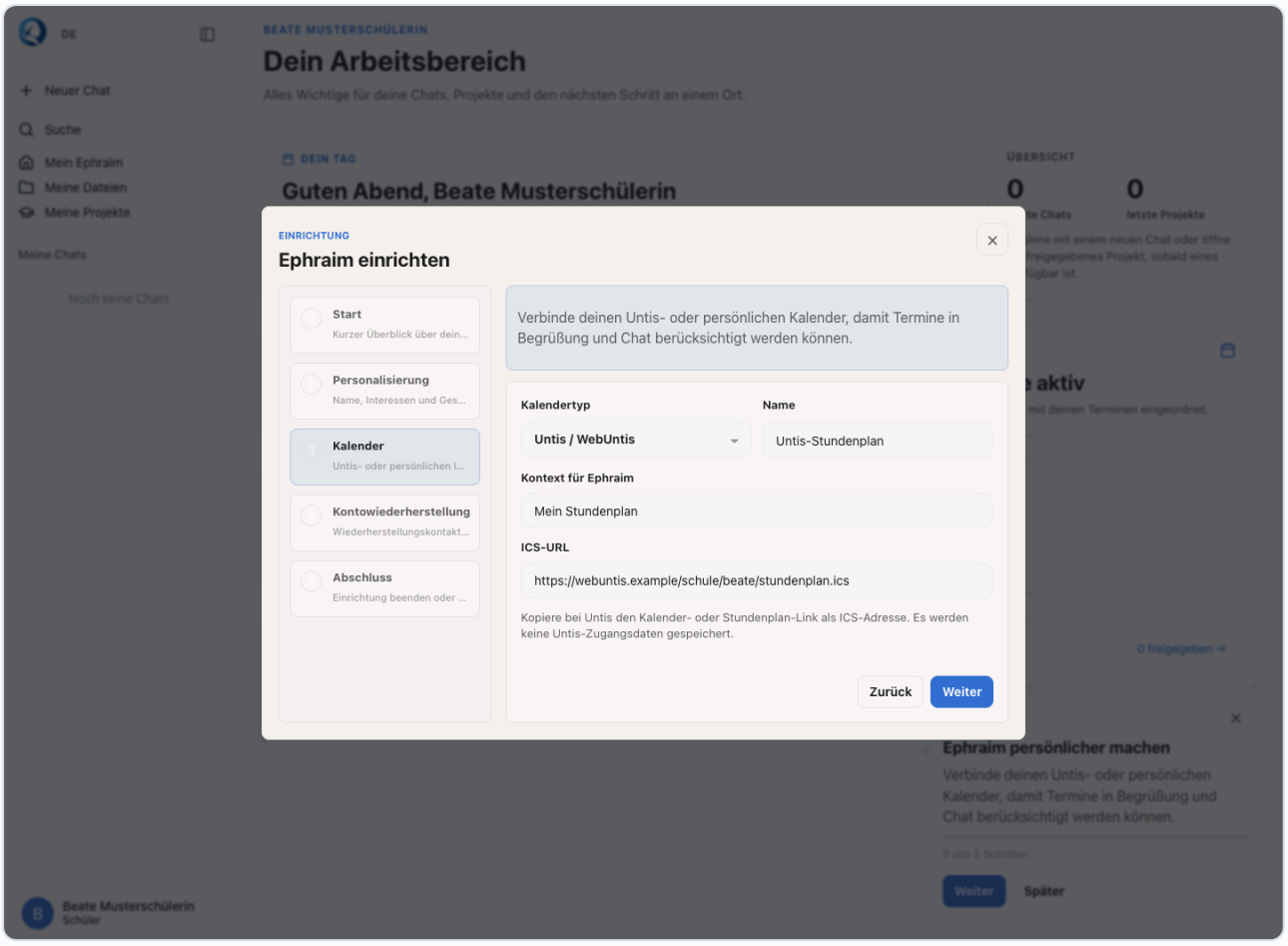


Die Beispielwerte zeigen ein fiktives Konto. Die Felder sind frei formulierbar und später in den Personalisierungs-Einstellungen änderbar.

Diese Angaben sind kein allgemeines Schulprofil. Ein normaler persönlicher Chat darf sie verwenden, weil du dort mit deinem eigenen Konto arbeitest. Ein Projektchat folgt dem Projektauftrag und den dort freigegebenen Materialien. Lehrkräfte erhalten durch die Personalisierung keinen Blick in private Chats.

5.4.5 Kalender

Der Kalender-Schritt verbindet einen Untis-, WebUntis- oder persönlichen Kalender über eine ICS-Adresse. Ephraim speichert keine Untis-Zugangsdaten. Du kopierst nur einen Kalenderlink. Daraus kann Ephraim Termine in der persönlichen Begrüßung und in normalen persönlichen Chats berücksichtigen.

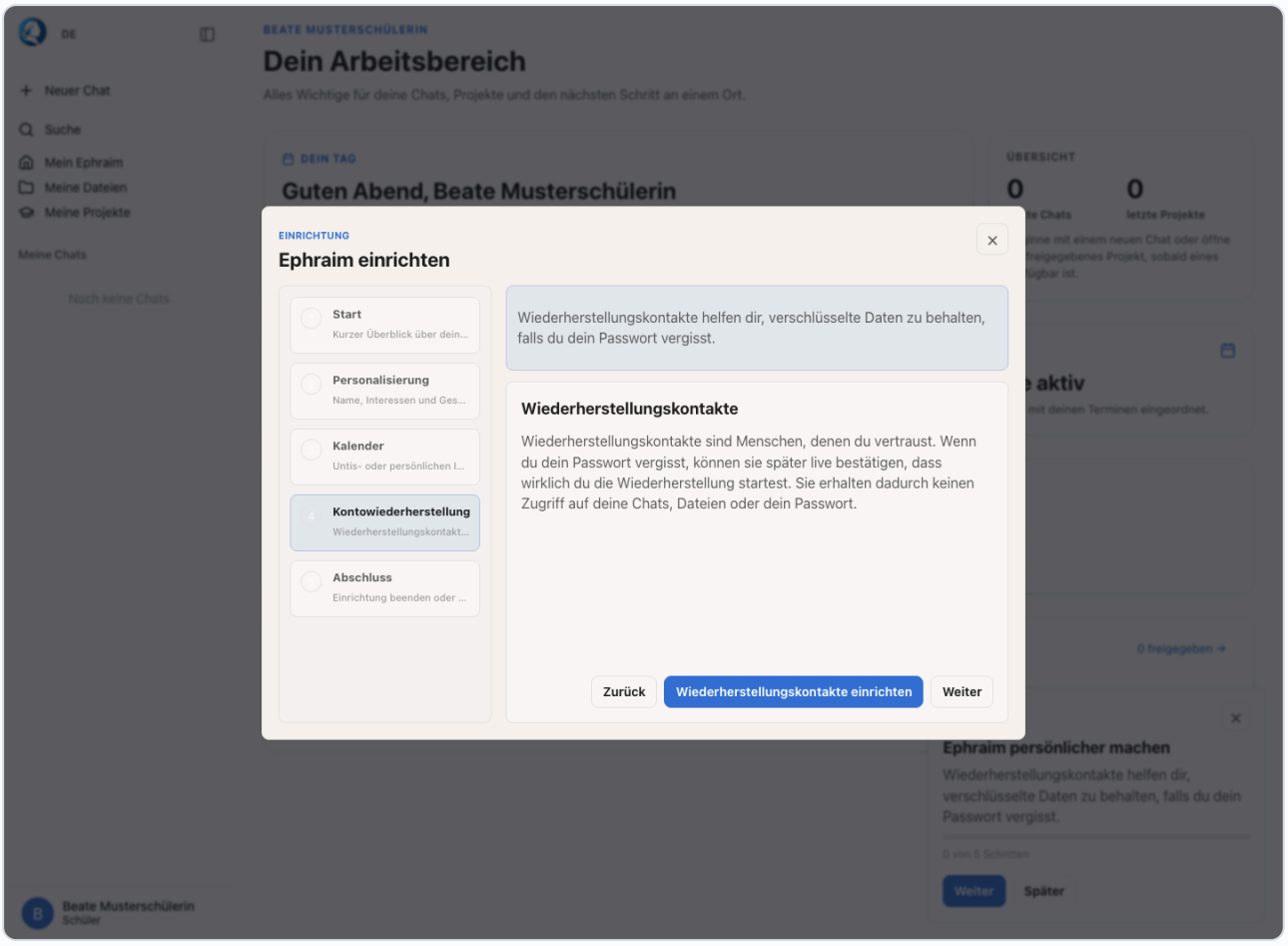


Der semantische Kontext sagt Ephraim, wie die Quelle verstanden werden soll, zum Beispiel als Stundenplan oder privater Kalender.

Private Kalender gehören fachlich zu deinen privaten Quellen. Die Kalenderadresse, die geladenen Rohdaten und daraus vorbereitete Ausschnitte sind an deinen persönlichen Vault gebunden. Ohne entsperrten Vault kann Ephraim diese privaten Kalender nicht für deinen Arbeitsbereich lesen. Projektbuilder und fremde Projektchats verwenden diese privaten Kalender nicht.

5.4.6 Kontowiederherstellung

Wenn die Installation die datenerhaltende Kontowiederherstellung unterstützt, enthält das Onboarding einen eigenen Hinweis dazu. Dieser Schritt richtet noch keine Kontakte ein. Er erklärt, warum die Vorbereitung wichtig ist, und führt in den Sicherheitsbereich, in dem du Wiederherstellungskontakte bewusst einlädst.

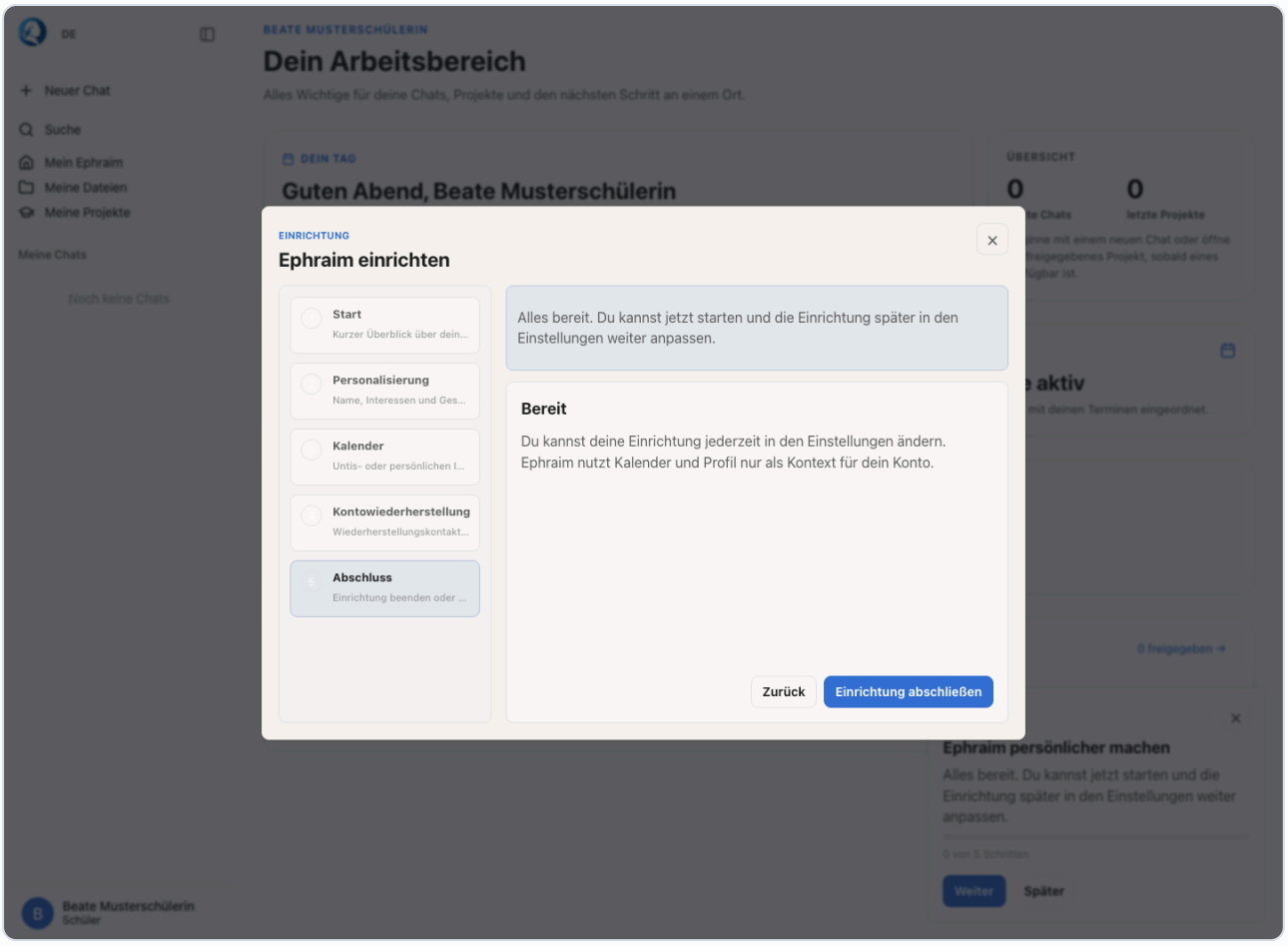


Recovery muss vorbereitet sein, bevor das Passwort vergessen ist. Das Onboarding macht darauf aufmerksam, erzeugt aber nicht heimlich Recovery-Material.

Der wichtige Unterschied: Onboarding zeigt den Weg, die Sicherheitseinstellungen erledigen die eigentliche Kryptografie. Erst dort werden Kontakte eingeladen, Anfragen angenommen und mit entsperrtem Vault aktiviert. Ein Wiederherstellungskontakt sieht dadurch keine Chats, keine Dateien, keine Passwörter und keinen Vault-Schlüssel im Klartext.

5.4.7 Abschluss und spätere Änderungen

Der Abschluss beendet nur den Einrichtungsdialog. Deine Einstellungen bleiben weiterhin änderbar. Personalisierung und Kalender findest du später in den entsprechenden Einstellungen; Passwort, 2FA, Sitzungen und Kontowiederherstellung liegen im Sicherheitsbereich.



Abschluss bedeutet: Der Hinweisdialog ist erledigt. Es bedeutet nicht, dass das Konto unveränderlich wird.

Was du ändern willst	Wo du es später findest
Name, Interessen, Gesprächsstil	Einstellungen → Personalisierung
Untis-, WebUntis- oder privater Kalender	Einstellungen → Personalisierung → Kalender
Passwort, 2FA, Sitzungen, Kontowiederherstellung	Einstellungen → Sicherheit
Eigene Daten exportieren	Einstellungen → Meine Daten

5.4.8 Datenschutzgrenzen

Das Onboarding speichert seinen eigenen Schrittstatus: aktueller Schritt, erledigte Schritte, Ausblendung, Abschluss oder dauerhaftes Verwerfen. Die Inhalte der Personalisierung und der Kalender werden nicht im Onboarding-Datensatz versteckt, sondern in den dafür vorgesehenen Konto-Bereichen gespeichert.

- Onboarding erzeugt kein KI-Gespräch und keine Trainingsdaten.
- Onboarding gibt der KI keinen freien Internetzugriff; eine eingetragene Kalenderadresse wird als konkrete private ICS-Quelle verarbeitet.
- Onboarding öffnet keine privaten Chats anderer Personen.
- Onboarding macht Recovery nicht automatisch scharf; die Aktivierung bleibt ein bewusster Vorgang in den Sicherheitseinstellungen.
- Onboarding ist für normale Konten gedacht, nicht für reine Projektcode-Gastzugänge.

Dieser Artikel beschreibt die freiwillige Ersteinrichtung normaler Ephraim-Konten. Für die technische Einordnung des privaten Datenraums siehe Vault; für Passwortfolgen und datenerhaltende Recovery siehe Passwörter.

Quelle: web/manuals/konto-onboarding/index.html

5.5 Passwörter und Zurücksetzen

> In Ephraim schützt das Passwort zwei Dinge zugleich: die Anmeldung am Konto und den persönlichen Vault. Deshalb unterscheidet Ephraim streng zwischen normalem Passwortwechsel, datenerhaltender Kontowiederherstellung und datenlöschendem Reset.

Stand: Juni 2026

5.5.1 Kurzfassung

Der normale Passwortwechsel ist der richtige Weg, solange du dein aktuelles Passwort kennst. Ephraim prüft das alte Passwort, setzt das neue Passwort und verpackt den bestehenden Vault-Schlüssel neu. Deine privaten Chats, Dateien, Erinnerungen, privaten Kalender und Chat-Zusammenfassungen bleiben erhalten.

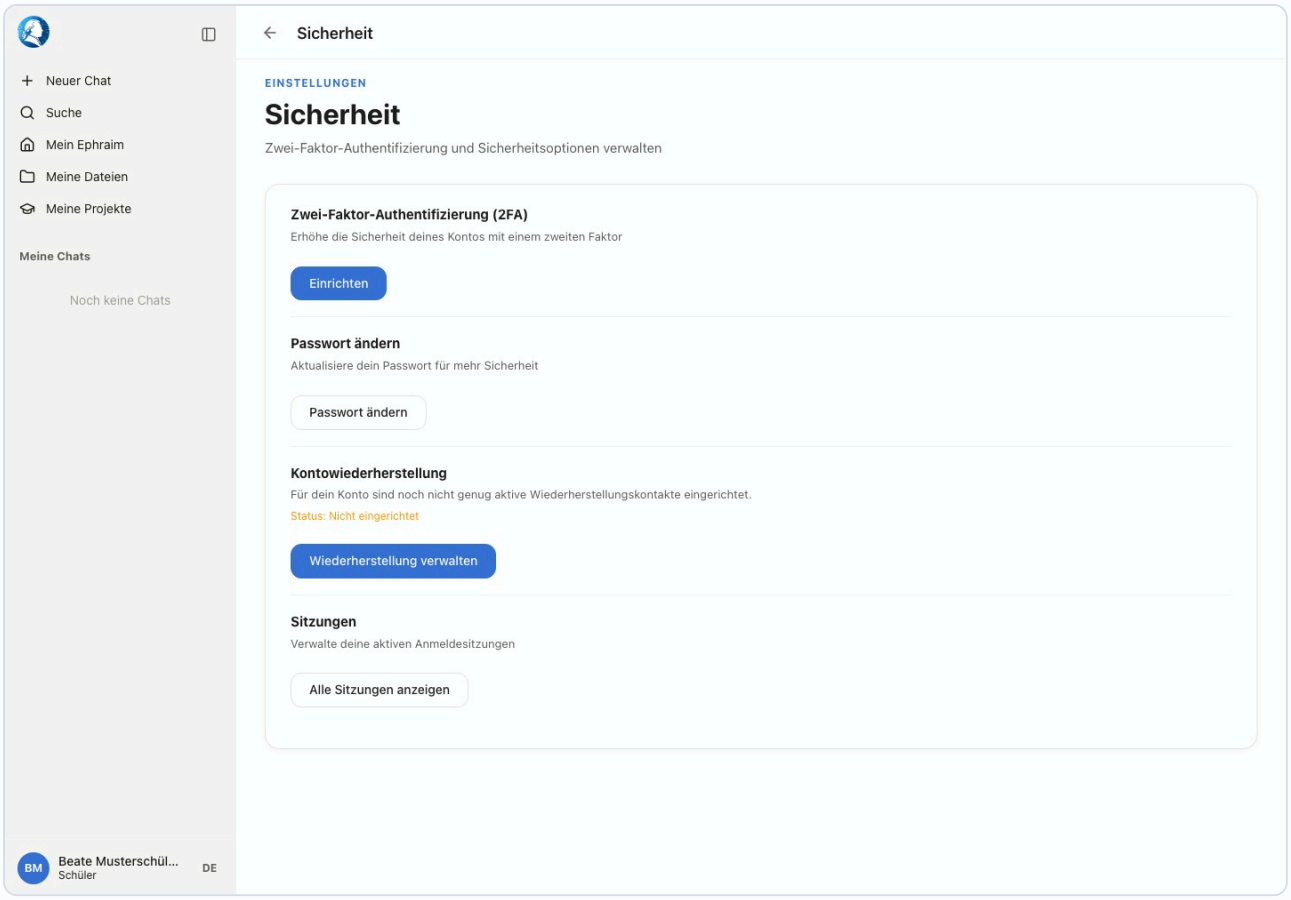
Die neue Kontowiederherstellung ist der sichere Weg, wenn du dein Passwort vergessen hast und vorher mindestens ein Wiederherstellungskontakt vorbereitet wurde. Ein vorbereiteter Kontakt stimmt live zu. Danach verpackt Ephraim denselben Vault-Schlüssel mit deinem neuen Passwort. Die verschlüsselten Daten bleiben erhalten.

Der datenlöschende Reset ist der Notweg, wenn kein normaler Wechsel und keine datenerhaltende Wiederherstellung möglich ist. Er stellt den Kontozugang wieder her, erzeugt aber einen neuen Vault. Alte private Vault-Daten werden entfernt, weil Ephraim den alten Vault ohne alten Schlüssel nicht entschlüsselt.

> **Merksatz:** Passwortwechsel und datenerhaltende Wiederherstellung erhalten den Vault. Der datenlöschende Reset ersetzt den Vault.

5.5.2 Normaler Passwortwechsel

Den normalen Passwortwechsel findest du unter **Einstellungen → Sicherheit → Passwort ändern**. Du gibst dein aktuelles Passwort ein und bestätigst das neue Passwort zweimal. Ephraim verlangt mindestens acht Zeichen, mindestens einen Großbuchstaben, mindestens einen Kleinbuchstaben und mindestens eine Zahl.

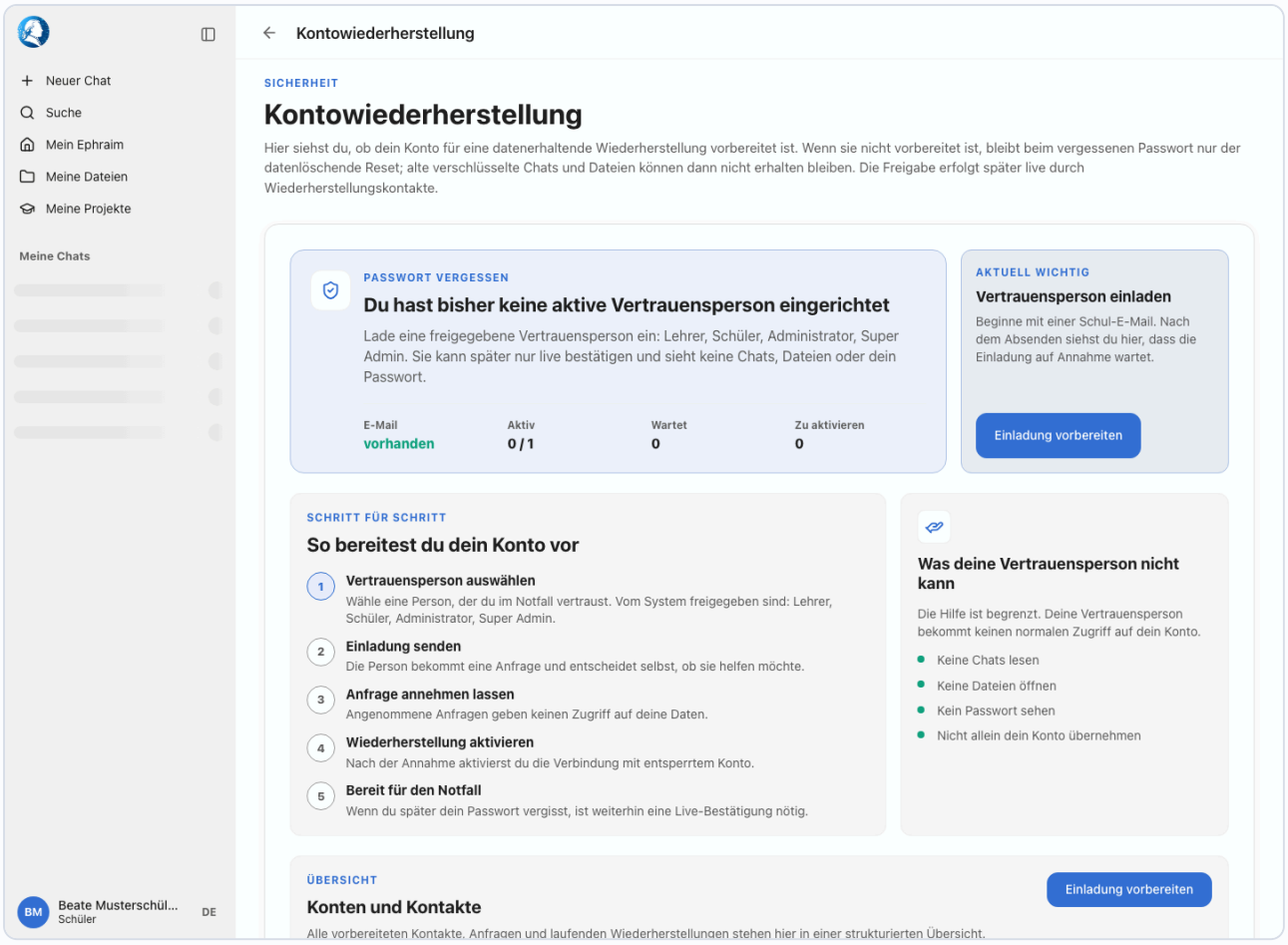


Der normale Passwortwechsel liegt im Sicherheitsbereich; dort wird auch die Kontowiederherstellung vorbereitet. Der Passwortwechsel erhält den persönlichen Vault, weil Ephraim den bereits entsperrten Vault-Schlüssel neu mit dem neuen Passwort verpackt.

Nach erfolgreichem Wechsel erhöht Ephraim die Sitzungs-Version. Andere aktive Sitzungen laufen dadurch ab; die aktuelle Sitzung bleibt gültig. Bei hinterlegter E-Mail-Adresse sendet Ephraim eine Benachrichtigung über die Änderung. Passwörter und Vault-Schlüssel werden nie per E-Mail versendet.

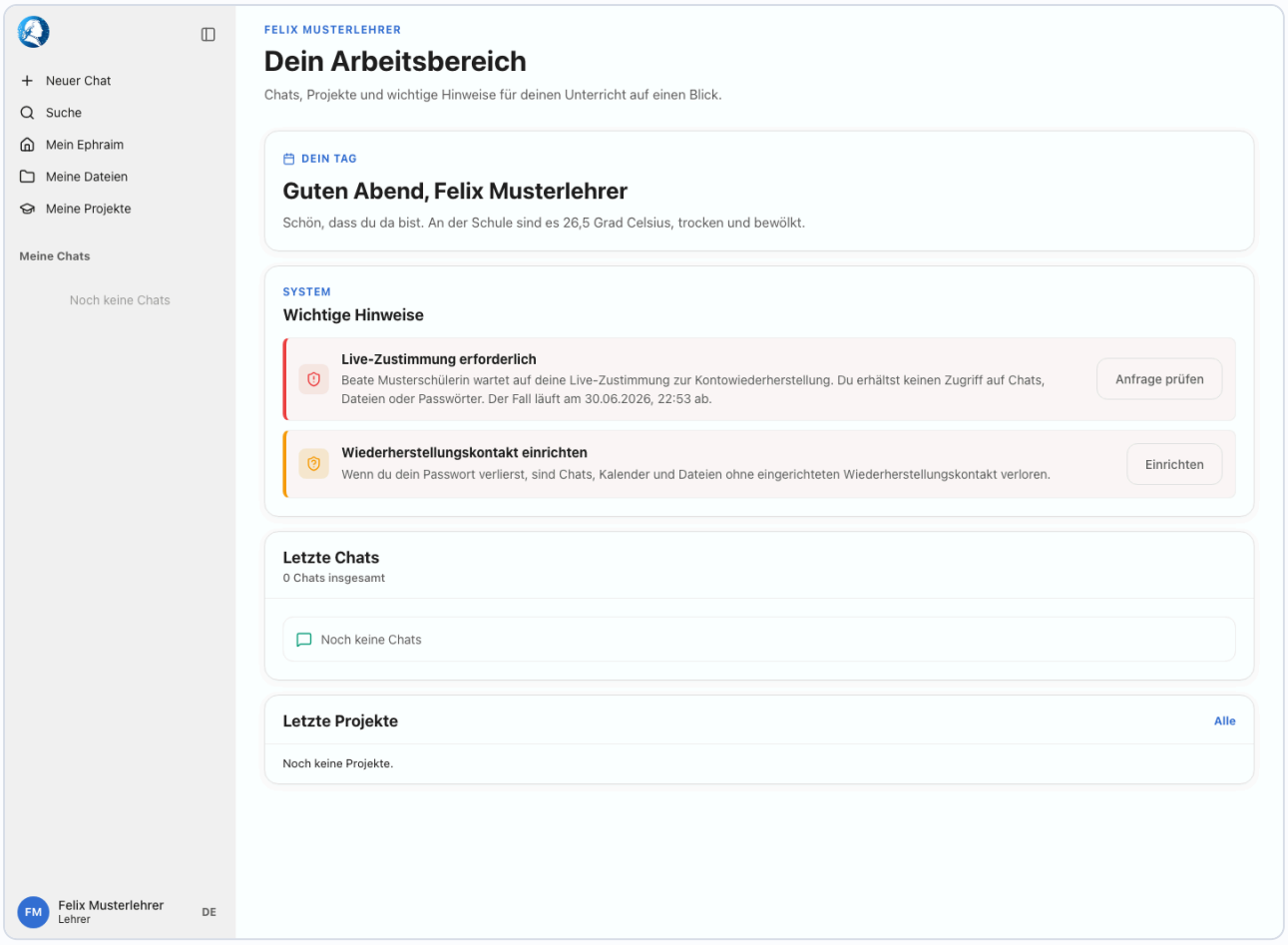
5.5.3 Kontowiederherstellung vorbereiten

Die datenerhaltende Kontowiederherstellung funktioniert nur, wenn sie vorbereitet wurde, bevor das Passwort vergessen ist. Der Bereich heißt **Kontowiederherstellung** und erscheint in den Sicherheitseinstellungen, wenn diese Funktion in der Installation aktiviert ist.



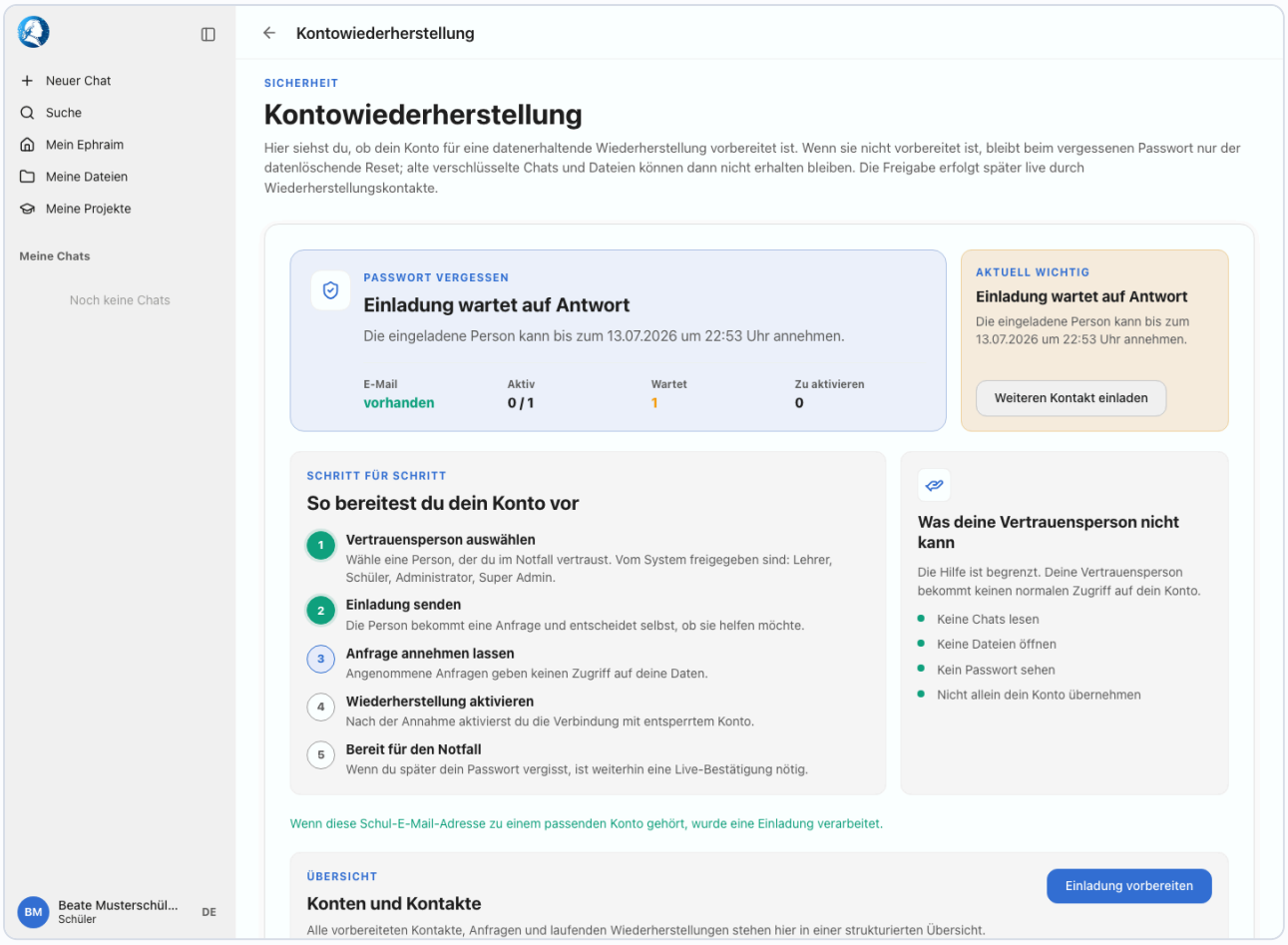
Der Status zeigt, ob eine Schul-E-Mail hinterlegt ist und wie viele aktive Wiederherstellungskontakte bereits eingerichtet sind. Die Beispiele verwenden fiktive Handbuchkonten.

Zusätzlich zeigt „Mein Ephraim“ wichtige Recovery-Hinweise direkt im Arbeitsbereich. Wenn noch kein Wiederherstellungskontakt vorbereitet ist, erscheint ein Hinweis zum Einrichten. Wenn du selbst als Wiederherstellungskontakt für eine andere Person gebraucht wirst, zeigt Ephraim dort eine Live-Zustimmung mit Prüfaktion. So bleibt Recovery nicht in den Einstellungen versteckt, sondern taucht dort auf, wo Nutzerinnen und Nutzer nach dem Login zuerst arbeiten.



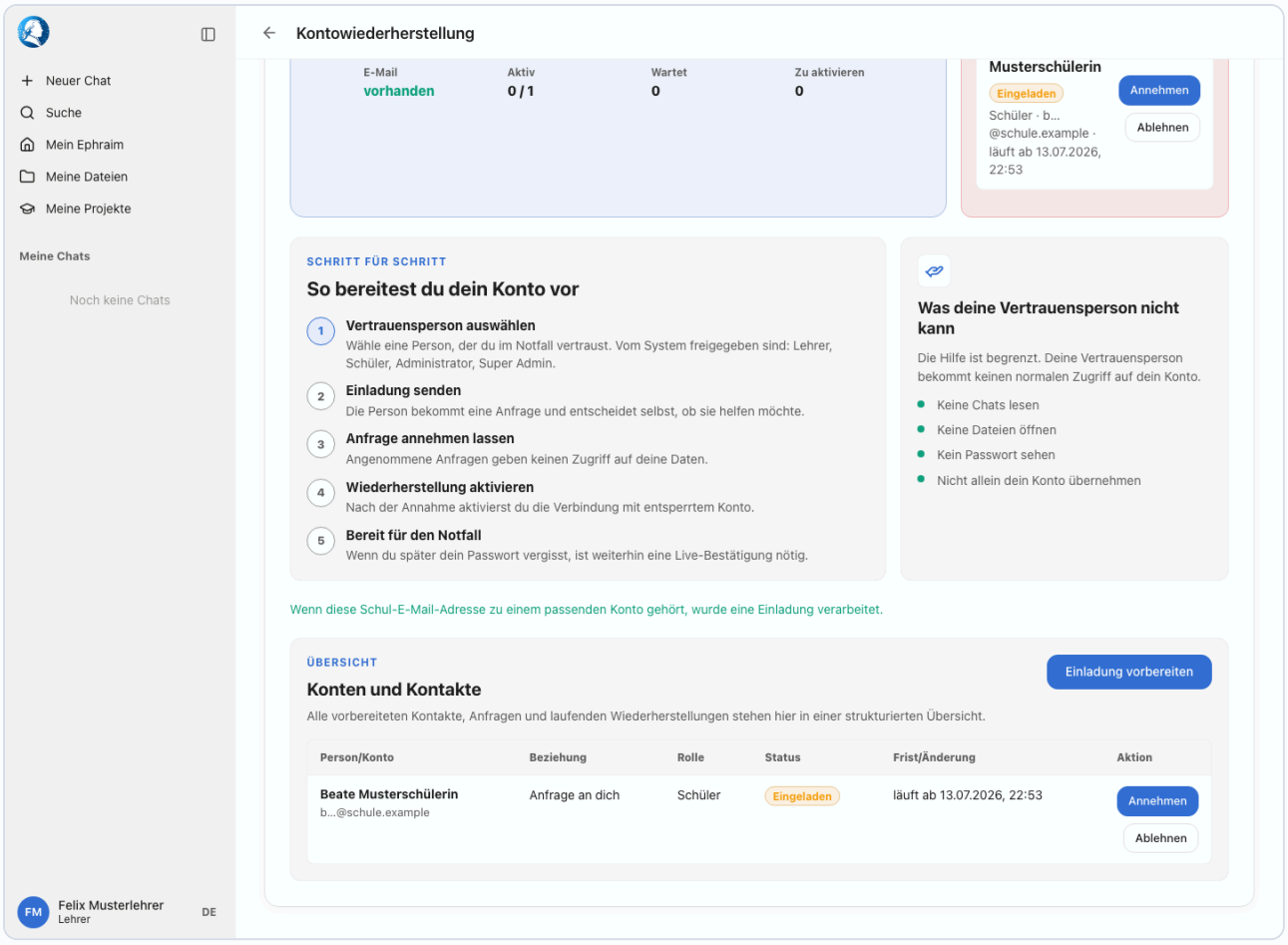
Der Screenshot stammt aus dem automatisch erzeugten Recovery-Testlauf: Felix Musterlehrer sieht in „Mein Ephraim“ eine wartende Live-Zustimmung für Beate Musterschülerin und zugleich den Hinweis, die eigene Kontowiederherstellung einzurichten.

Ein Wiederherstellungskontakt ist eine vertrauenswürdige Person mit eigenem Ephraim-Konto, zum Beispiel eine Lehrkraft. Du gibst direkt die Schul-E-Mail-Adresse ein. Ephraim zeigt keine Trefferliste und bestätigt nicht, ob ein Konto existiert. Diese neutrale Verarbeitung verhindert, dass das Formular zur Kontosuche missbraucht wird.



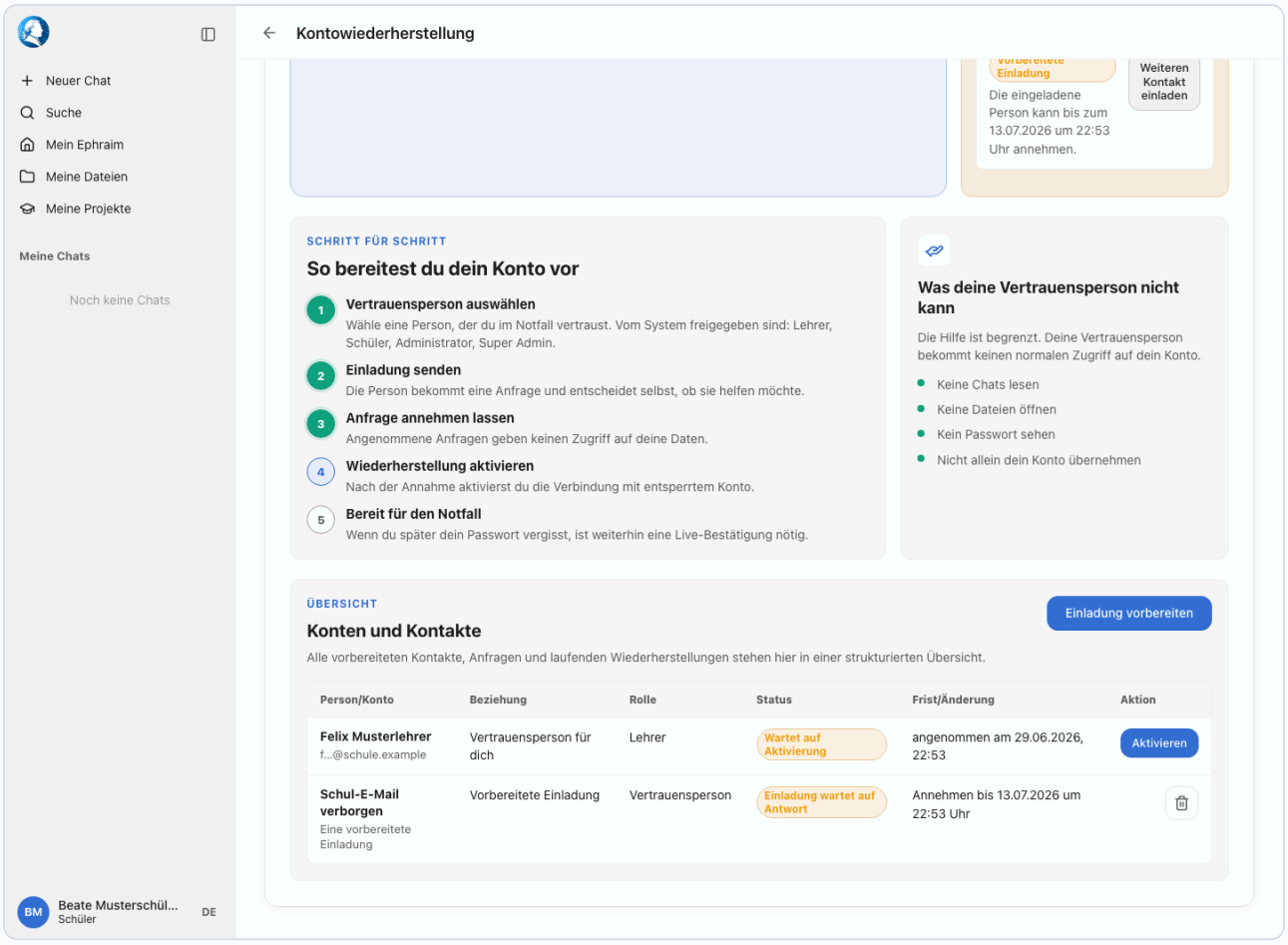
Nach dem Absenden erscheint nur eine neutrale Meldung. Die Oberfläche verrät nicht, ob unter der eingegebenen Adresse ein gültiges Konto gefunden wurde.

Die eingeladene Person sieht die Anfrage nach der Anmeldung unter **Kontowiederherstellung**. Sie muss ihren eigenen Vault entsperrt haben und die Anfrage aktiv annehmen. Durch die Annahme erhält sie keinen Zugriff auf deine Chats, Dateien, Passwörter oder Kalender.

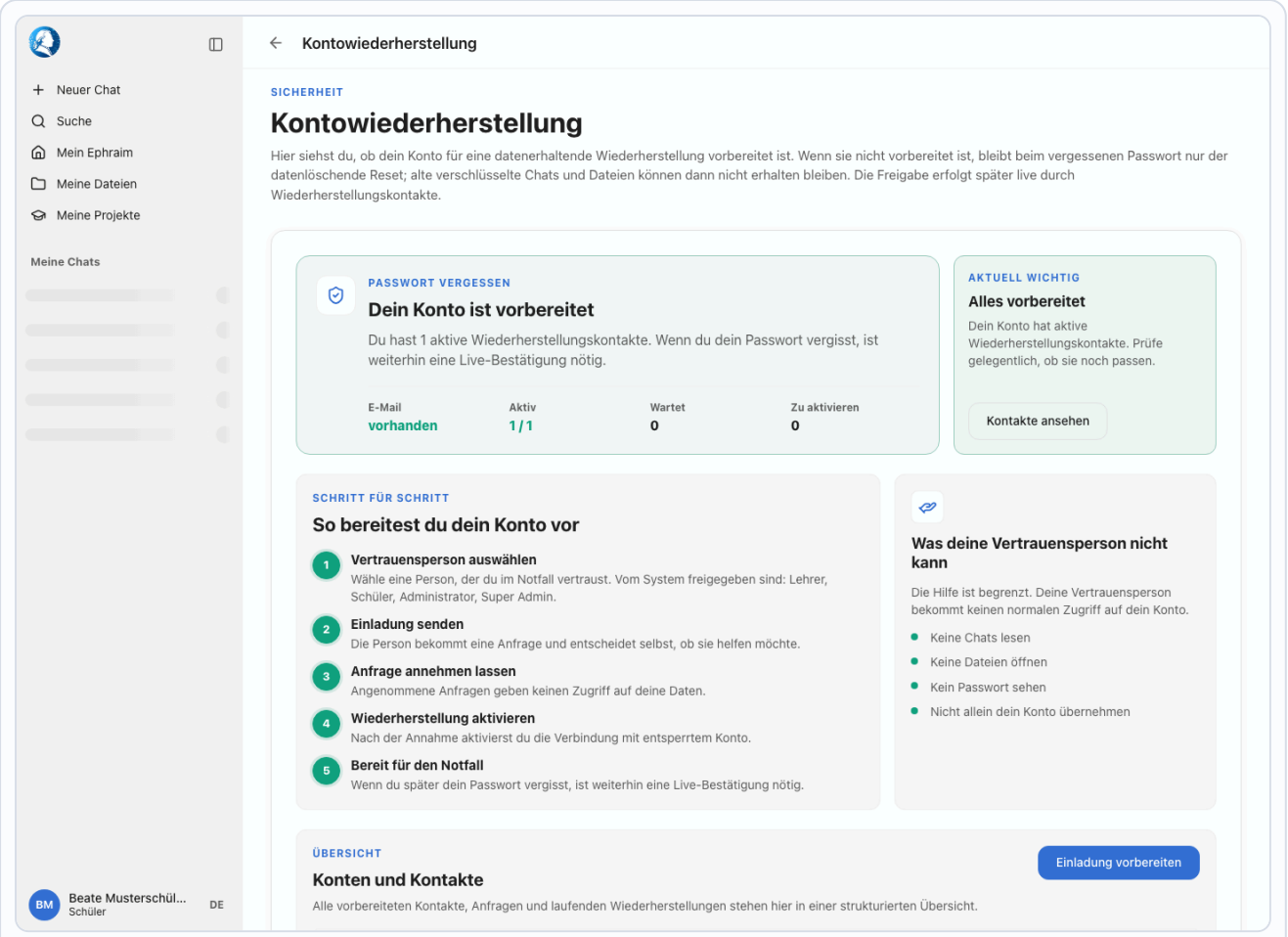


Der Kontakt sieht, von wem die Anfrage stammt, und entscheidet bewusst. Die Anfrage ist kein Freibrief für fremde Daten.

Nach der Annahme ist die Beziehung noch nicht aktiv. Du musst sie in deinem eigenen Konto mit entsperrtem Vault aktivieren. Erst dieser Schritt erzeugt das eigentliche Recovery-Material: Ephraim verpackt einen Wiederherstellungsanteil so, dass der Kontakt ihn später nur live in seinem eigenen Konto freigeben kann.



Eine angenommene Anfrage muss vom Kontoinhaber aktiviert werden. Ohne diese Aktivierung ist die Kontowiederherstellung noch nicht einsatzbereit.

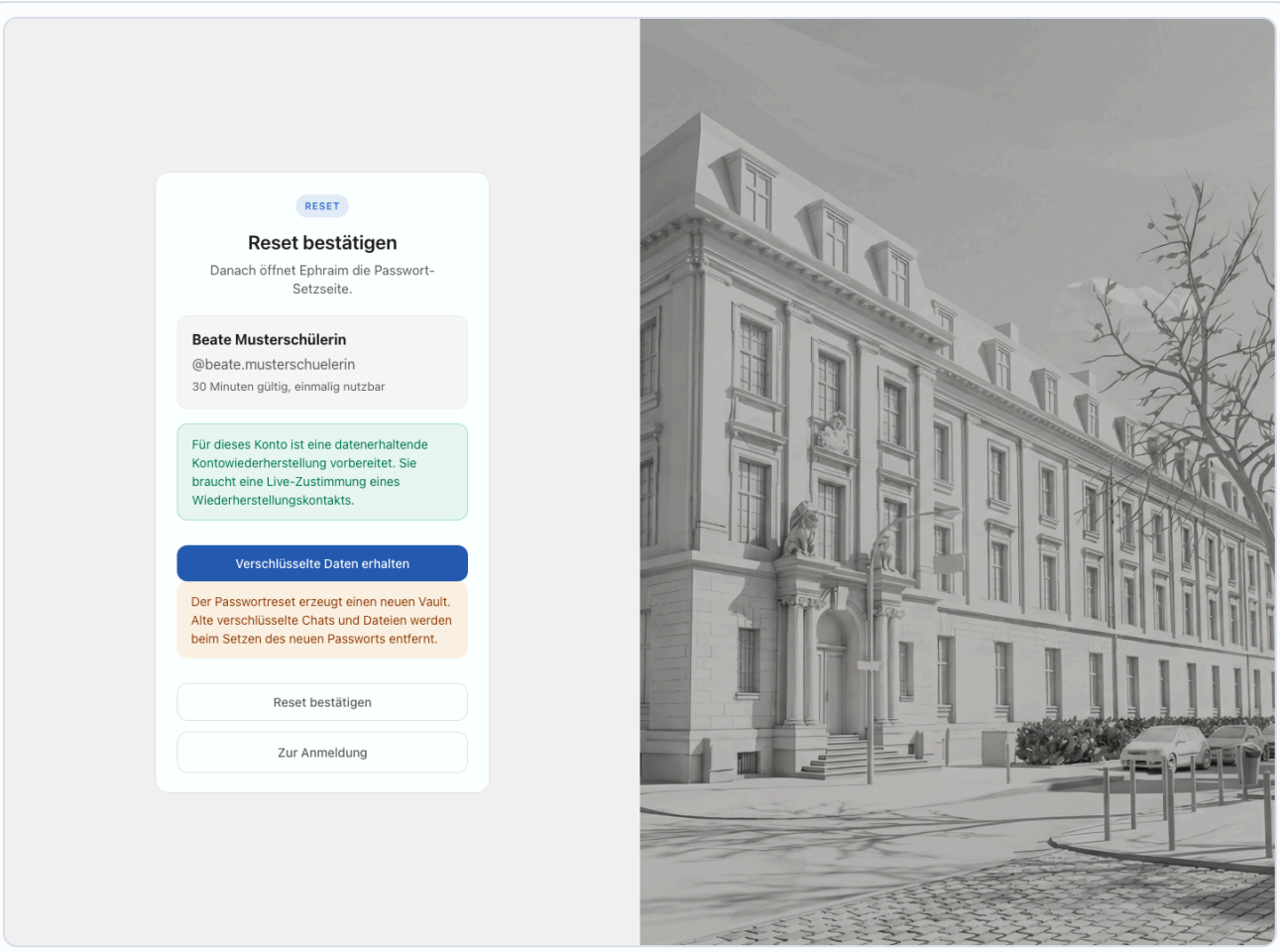


Der Status „Bereit“ bedeutet: Für dieses Konto ist mindestens ein aktiver Wiederherstellungskontakt vorbereitet. Im Ernstfall genügt eine Live-Zustimmung.

5.5.4 Passwort vergessen: datenerhaltender Weg

Bei vergessenem Passwort startest du auf der Anmeldeseite mit **Passwort vergessen?**. Ephraim sendet einen Bestätigungslink an die hinterlegte E-Mail-Adresse, wenn genau ein passendes Konto existiert. Die Formularantwort bleibt neutral, damit niemand Konten erraten kann.

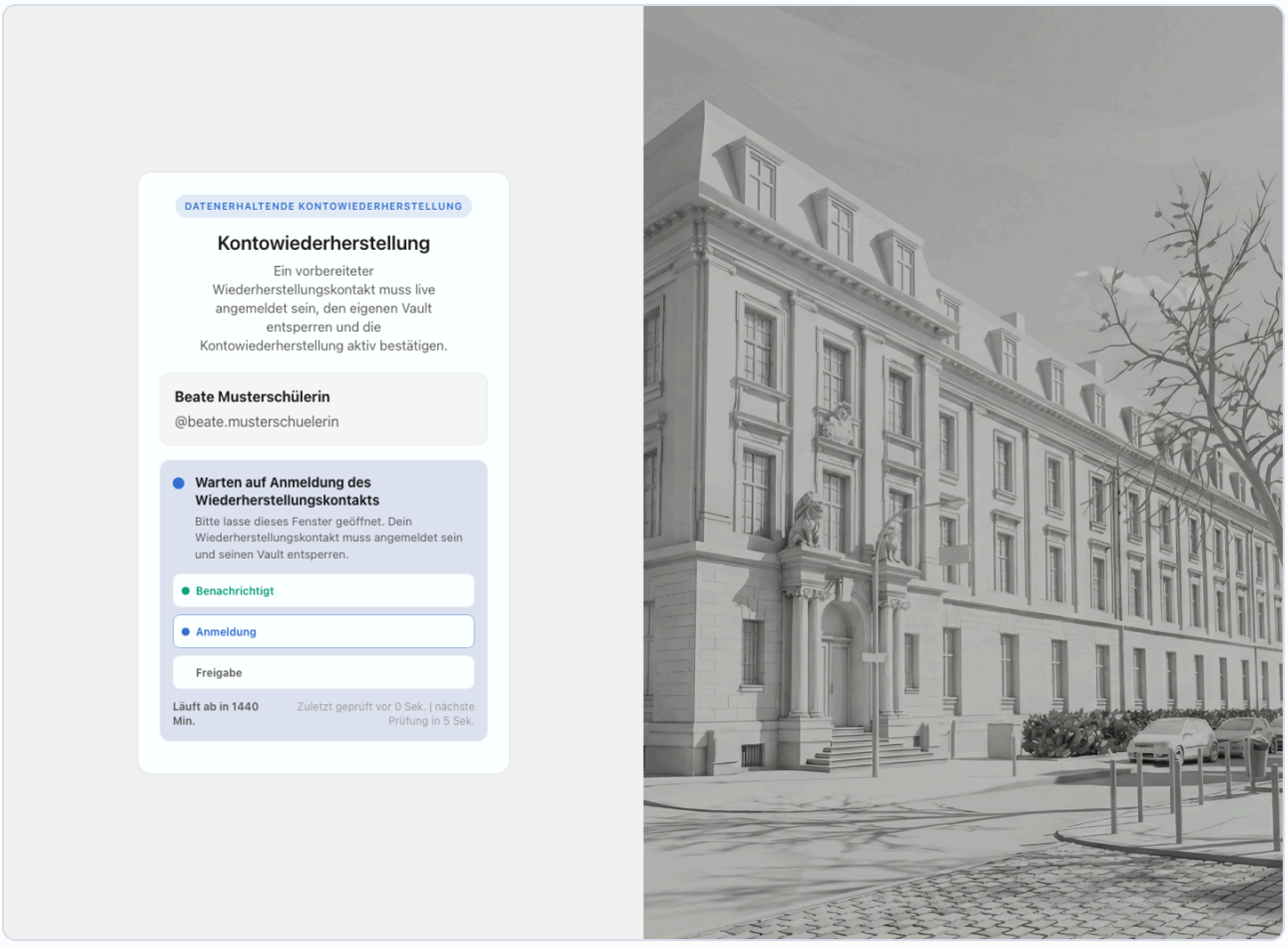
Nach dem Öffnen des Bestätigungslinks zeigt Ephraim die entscheidende Auswahl. Wenn für dein Konto eine datenerhaltende Wiederherstellung vorbereitet ist, erscheint **Verschlüsselte Daten erhalten**. Darunter bleibt der datenlöschende Reset sichtbar, aber er ist der schlechtere Weg, solange die Live-Recovery verfügbar ist.



Die datenerhaltende Option ist klar von der Reset-Warnung getrennt: Oben bleibt der bestehende Vault erhalten, unten entsteht ein neuer Vault mit Datenverlust.

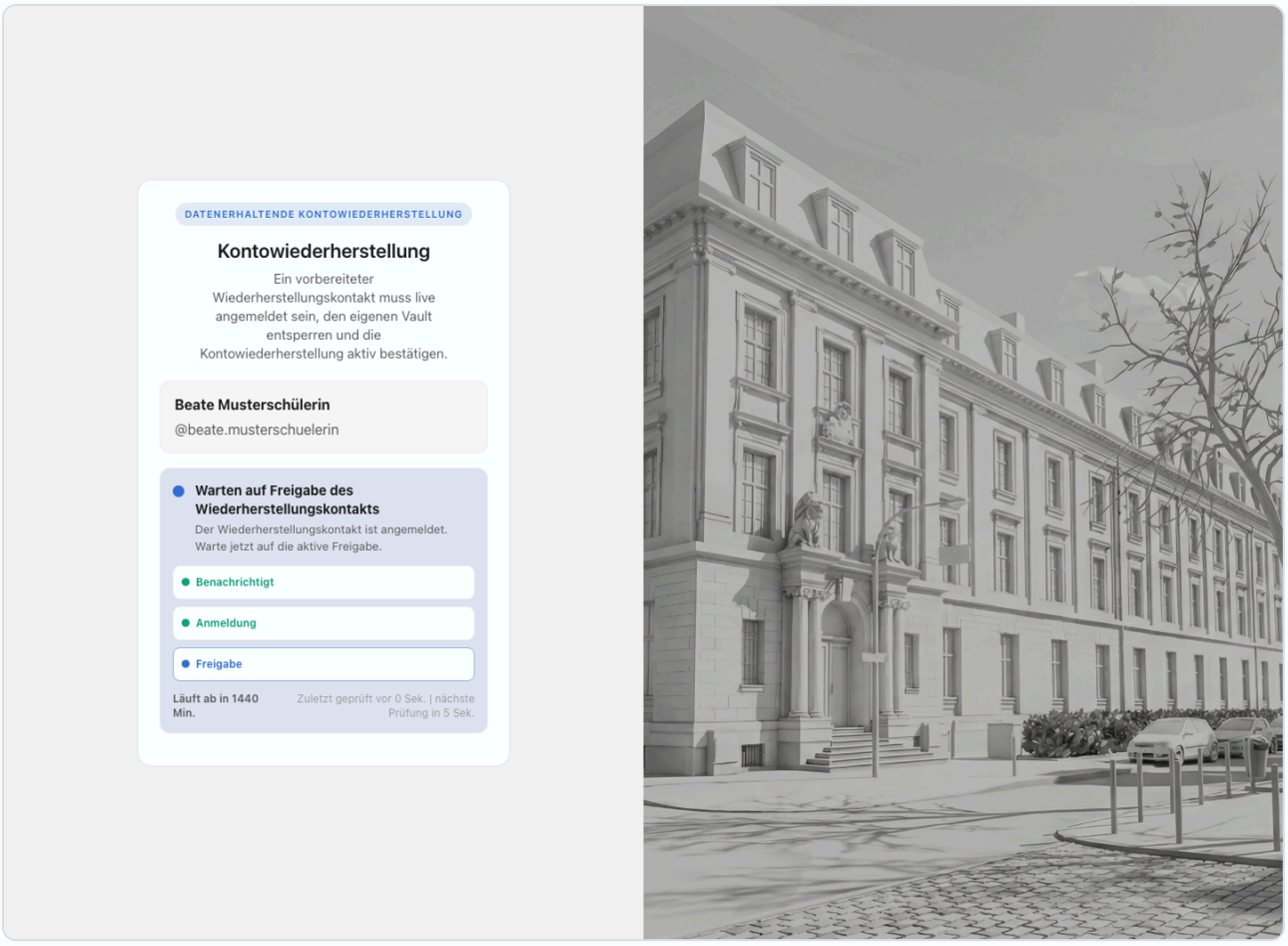
Nach dem Start der datenerhaltenden Wiederherstellung bleibt dieses Browserfenster offen. Ephraim erzeugt einen live gebundenen Wiederherstellungsfall und benachrichtigt die vorbereiteten Kontakte. Der Kontakt erhält keinen direkten Recovery-Link. Er meldet sich mit seinem eigenen Konto an und sieht die Anfrage in seiner Recovery-Übersicht.

Die Statusanzeige auf dieser Wiederherstellungsseite aktualisiert sich automatisch. Sie zeigt drei konkrete Schrittkarten: **Benachrichtigt**, **Anmeldung** und **Freigabe**. Grüne Punkte markieren erledigte Schritte, die blaue Markierung zeigt den aktuellen Warteschritt. Zusätzlich siehst du, wann der Fall abläuft und wann Ephraim den Status zuletzt geprüft hat. Du musst die Seite nicht neu laden, solange JavaScript verfügbar ist.

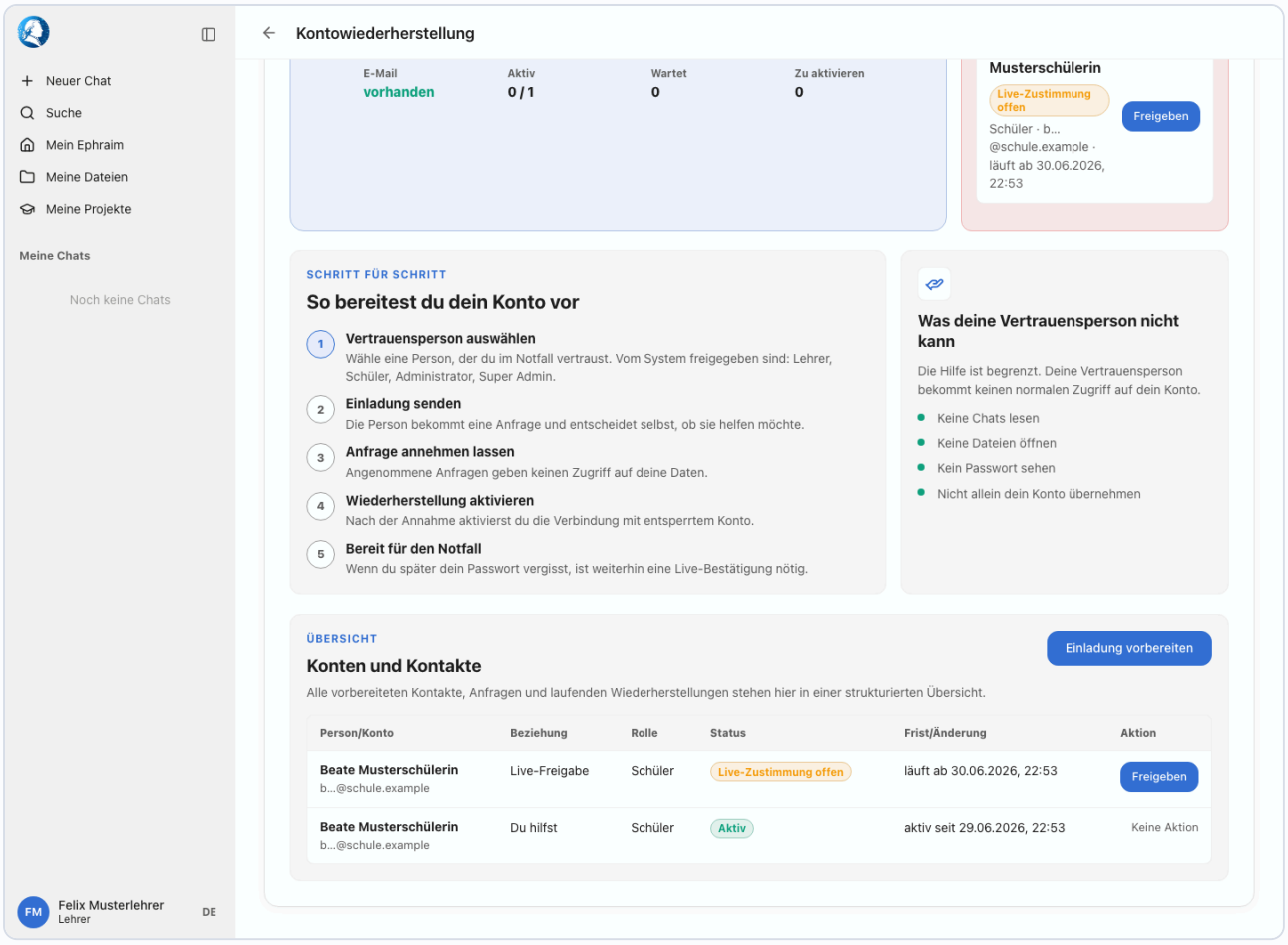


Beate Musterschülerin hat die datenerhaltende Wiederherstellung gestartet. Der Kontakt wurde benachrichtigt; die Anzeige wartet jetzt auf seine Anmeldung.

Sobald der Wiederherstellungskontakt angemeldet ist und die Anfrage geöffnet hat, springt die Anzeige weiter. Das bedeutet noch nicht, dass die Wiederherstellung freigegeben ist: Die eigentliche Zustimmung bleibt ein separater, bewusster Schritt.

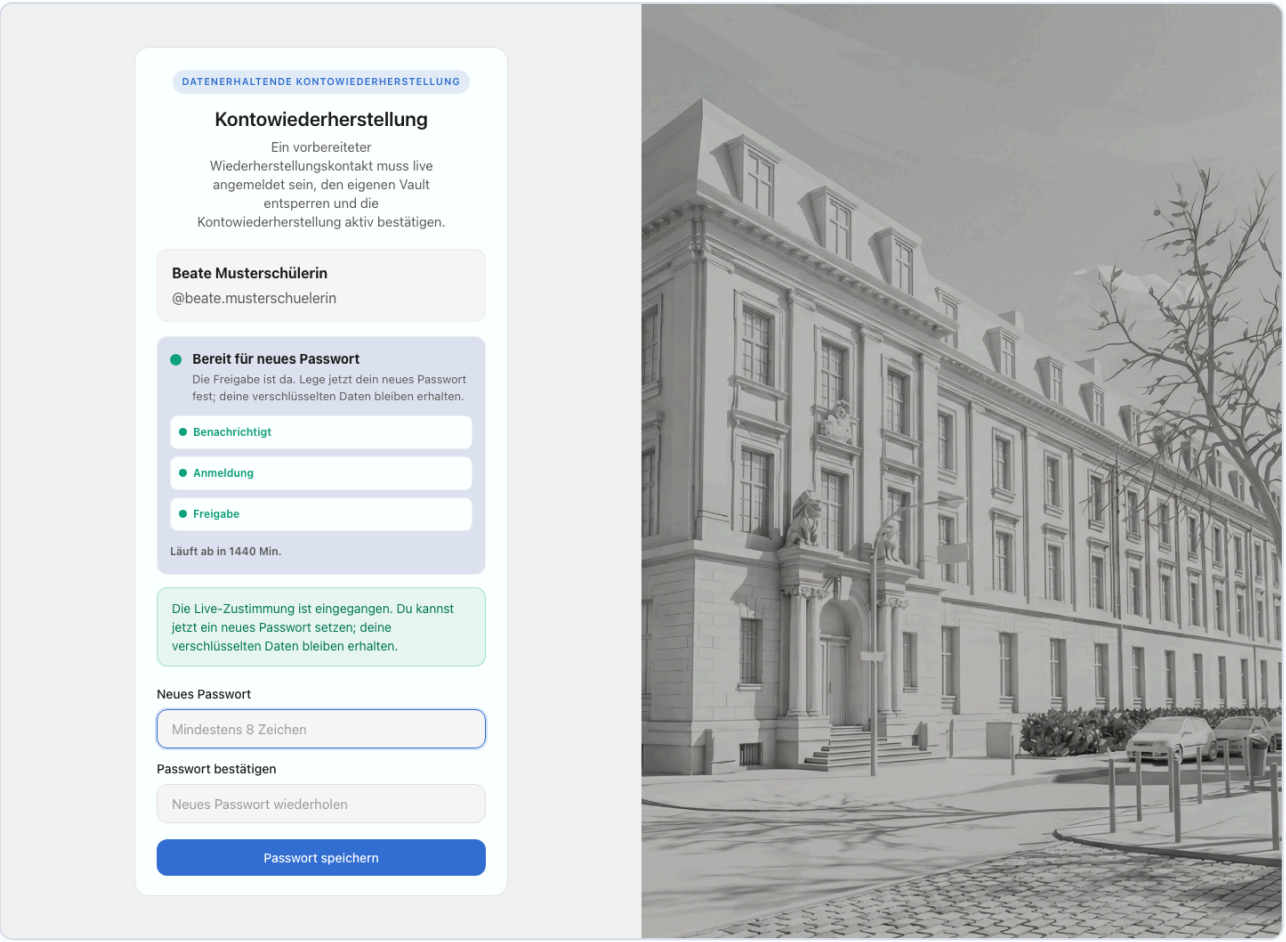


Benachrichtigung und Anmeldung sind erledigt, die Freigabe ist der aktuelle Schritt. Dadurch ist klar erkennbar, ob der Kontakt die Anfrage schon gesehen hat.



Der Wiederherstellungskontakt sieht die akute Anfrage und gibt sie erst nach externer Identitätsprüfung frei. Die Oberfläche weist ausdrücklich darauf hin.

Sobald die Live-Freigabe eingegangen ist, zeigt dein Fenster ein Formular für das neue Passwort. Ephraim setzt danach nicht einfach einen neuen Vault auf. Stattdessen nimmt es den vorhandenen Vault-Schlüssel aus dem Recovery-Fall und verpackt ihn mit deinem neuen Passwort. Dadurch bleiben die verschlüsselten Daten erhalten.



Nach der Freigabe sind alle drei Schritte abgeschlossen. Jetzt wird nur noch das neue Passwort gesetzt; der bestehende Vault bleibt derselbe und seine Verpackung wird erneuert.

5.5.5 Datenlöschender Reset

Der datenlöschende Reset bleibt notwendig, wenn kein normaler Passwortwechsel und keine datenerhaltende Wiederherstellung möglich sind. Er stellt den Zugang zum Konto wieder her, entfernt aber alte private Vault-Daten. Die Löschung ist keine Strafmaßnahme, sondern die technische Konsequenz der Verschlüsselung: Ohne den alten Vault-Schlüssel sind alte private Inhalte nicht lesbar.

Bereich	Nach datenlöschendem Reset
Konto, Benutzername, Rolle, Klasse, E-Mail	Bleiben erhalten. Der Reset ersetzt nicht das Konto.
Private Chats, Chat-Titel und Chat-Zusammenfassungen	Werden entfernt, soweit sie zum alten Vault gehören.
Persönliche Dateien und Dateinamen	Werden entfernt, weil Inhalt und Namen mit dem alten Vault geschützt waren.
Persönliche Erinnerungen und private Kalender	Werden nicht in den neuen Vault übertragen. Kalenderlinks müssen neu verbunden werden.
Projektmitgliedschaften	Bleiben bestehen. Frühere persönliche Projektchat-Verläufe stehen danach nicht mehr zur Verfügung.
Zwei-Faktor-Authentifizierung	Bleibt separat. Ein Passwortreset ersetzt keinen verlorenen Authenticator.

5.5.6 Admin-Reset und 2FA

Die Administration sieht dein Passwort nicht und stellt es nicht wieder her. Sie erzeugt einen Reset-Zugang, der das Setzen eines neuen Passworts erlaubt. Ein Admin-Reset ist ein sicherheitskritischer Eingriff: Das alte Passwort wird deaktiviert, bestehende Sitzungen werden ungültig, vorbereitete Recovery-Beziehungen werden invalidiert und beim Einlösen entsteht ein neuer Vault.

Wenn du zusätzlich den zweiten Faktor verloren hast, reicht der Passwortweg nicht aus. Dann braucht es einen separaten 2FA-Reset durch die Administration. Passwort und 2FA sind bewusst getrennt, damit ein einzelner Reset nicht alle Schutzschichten entfernt.

5.5.7 Was Wiederherstellungskontakte nicht erhalten

Ein Wiederherstellungskontakt erhält keinen Zugriff auf dein Konto, deine Chats, deine Dateien, deine privaten Kalender, dein Passwort oder deinen Vault-Schlüssel im Klartext. Die Freigabe ist eine technische Zustimmung in einem konkreten Live-Fall. Sie hilft Ephraim, den bestehenden Vault-Schlüssel wieder mit deinem neuen Passwort zu verpacken.

Deshalb ist die externe Identitätsprüfung wichtig. Der Kontakt muss außerhalb der App sicherstellen, dass wirklich du die Wiederherstellung startest. In der Schule bedeutet das praktisch: persönliche Rückfrage, bekannter Kommunikationsweg oder direkte Absprache.

5.5.8 Gute Praxis

- Richte die Kontowiederherstellung ein, solange du dein Passwort kennst.
- Wähle Wiederherstellungskontakte, die erreichbar sind und deine Identität verlässlich prüfen.
- Verwende einen Passwort-Manager und ein langes, eigenes Ephraim-Passwort.
- Bewahre 2FA-Recovery-Codes getrennt vom Passwort auf.
- Nutze den normalen Passwortwechsel, sobald du das alte Passwort noch kennst.
- Nutze den datenlöschenden Reset nur, wenn die datenerhaltende Wiederherstellung nicht bereitsteht.

Für die technische Einordnung siehe den Nachbarartikel Vault. Dort wird erklärt, warum Passwort, Vault und Recovery zusammengehören.

Quelle: web/manuals/passwoerter-und-zuruecksetzen/index.html

5.6 Inferenz auf der Spark

> **Inferenz bedeutet: Ein bereits vorhandenes KI-Modell berechnet aus deiner Anfrage eine Antwort. Bei Ephraim geschieht diese Antworterzeugung auf der schuleigenen Spark und nicht über einen öffentlichen Internet-Chatbot. Dieser Artikel erklärt die Antwortberechnung; das Gesamtbild der Komponenten steht in der Systemarchitektur.**

Stand: Juni 2026

5.6.1 Was Inferenz bedeutet

Ein Sprachmodell ist vorher trainiert worden. Beim Training hat es aus sehr vielen Textbeispielen gelernt, welche sprachlichen Muster, Begriffe, Schreibweisen und Zusammenhänge häufig zusammenpassen. Im Schulbetrieb wird dieses Modell nicht bei jeder Frage neu trainiert. Es wird verwendet, um aus einer konkreten Anfrage eine konkrete Antwort zu berechnen.

Diese Verwendung heißt **Inferenz**. Man kann sich das so vorstellen: Ephraim legt der Spark einen sorgfältig zusammengestellten Arbeitszettel hin. Auf diesem Arbeitszettel stehen die aktuelle Frage, erlaubter Kontext, Projektauftrag oder Rollenhinweise und technische Ausgabevorgaben. Die Spark liest diesen Arbeitszettel und berechnet dann Schritt für Schritt die wahrscheinlich passende Fortsetzung.

> **Wichtig:** Inferenz ist Antwortberechnung, nicht Lernen über dich. Die einzelne Anfrage verändert nicht die Modellgewichte der Spark. Begriffe wie Token, Embedding und Kontextfenster werden im Glossar kurz eingeordnet.

5.6.2 Welche Rolle die Spark spielt

Die Spark ist die schuleigene KI-Hardware. Auf ihr läuft der Inferenzdienst, der sich gegenüber Ephraim wie eine OpenAI-kompatible lokale API verhält. Der Browser spricht aber nicht direkt mit der Spark. Der Browser spricht mit dem Ephraim-Webserver; erst der Webserver prüft Sitzung, Rolle, Rechte, Vault, Projektstatus, Dateien und freigegebenes Basiswissen.

Danach sendet der Webserver nur den für diese Anfrage benötigten Kontext an die Spark. Die Spark berechnet die Antwort und gibt sie als Stream zurück. Sie ist also der lokale Rechenort für das Modell, nicht die Stelle, an der Konten, Rollen, Projekte oder Dateiberechtigungen verwaltet werden.

> **Abgrenzung:** Die Spark ist hier der Rechenort für Modell und Embeddings. Anmeldung, Rollen, Vault, Projekte, Uploadprüfung und Basiswissen-Auswahl bleiben Aufgaben des Webservers.

5.6.3 Der Datenweg zur Spark

Der Browser spricht mit dem Ephraim-Webserver. Der Webserver prüft Sitzung, Rolle, CSRF-Schutz, Vault-Verfügbarkeit und Kontext. Erst danach sendet er die für diese Anfrage benötigten Inhalte verschlüsselt an die Spark. Die Spark bekommt also nicht „alles“, was Ephraim gespeichert hat, sondern den zusammengestellten Request-Kontext.

Die Spark erhält den freigegebenen Request-Kontext, berechnet die Antwort und streamt sie zurück.

5.6.4 Was im Arbeitszettel steht

Für das Modell ist eine Anfrage kein einzelner Satz, sondern ein Paket. Dieses Paket kann mehrere Teile enthalten:

Teil	Beispiel	Warum er wichtig ist
Systemrahmen	Rolle, Ton, Sicherheitsregeln, Ausgabeformat	Setzt Leitplanken für die Antwort.
Nutzerfrage	„Erkläre mir die Fotosynthese.“	Ist der aktuelle Arbeitsauftrag.
Chatverlauf	Bisherige Fragen und Antworten im freigegebenen Verlauf	Ermöglicht Folgefragen ohne vollständige Wiederholung.
Projektkontext	Projektauftrag, Materialien, Starterfragen	Bindet Antworten an den Unterrichtsrahmen.
Basiswissen	Freigegebene Quellen, Kalender, Wetter, Wikipedia-Auszüge	Ergänzt aktuelle oder schulische Informationen, wenn erlaubt.
Dateikontext	Ausgewählte Textstellen aus hochgeladenen Dateien	Hilft, Material zu erklären oder zusammenzufassen.

Dieses Paket ist begrenzt. Ein Sprachmodell hat ein **Kontextfenster**: Es kann nur eine bestimmte Menge Text gleichzeitig berücksichtigen. Darum muss Ephraim auswählen, was für die konkrete Frage relevant ist.

5.6.5 Was Tokens sind

Computer rechnen nicht direkt mit Wörtern. Bevor die Spark eine Antwort berechnen kann, wird Text in kleinere Einheiten zerlegt. Diese Einheiten heißen **Tokens**. Ein Token ist je nach Text ein ganzes kurzes Wort, ein Wortteil, ein Satzzeichen, ein Leerzeichen mit Wortanfang oder ein Stück eines unbekannten Begriffs.

Das Wort „Schülerinnen“ ist für ein Modell nicht zwingend ein einzelnes Token. Der Tokenizer zerlegt es je nach Modell in mehrere Stücke. Auch „KI-Antwort“ besteht nicht einfach aus zwei Schulwörtern, sondern aus den Tokenstücken, die der Tokenizer des Modells kennt. Für Menschen wirkt das ungewohnt, für das Modell ist es praktisch: So verarbeitet es auch seltene Wörter, Namen, Fachbegriffe und zusammengesetzte Wörter.

Menschlich gesehen	Modellsicht, vereinfacht
„Hallo Beate!“	Tokenstücke für „Hallo“, Leerzeichen, „Beate“ und Ausrufezeichen
„Fotosynthese“	Ein Fachwort oder mehrere bekannte Wortteile
„$x^2 + 3x$“	Tokens für Zeichen, Zahlen, Operatoren und Sonderzeichen

Wenn Ephraim eine Antwort streamt, erzeugt die Spark intern solche Tokens nacheinander. Der Browser zeigt sie nicht als Tokenliste, sondern wieder als lesbaren Text.

5.6.6 Warum Vektoren eine Rolle spielen

Ein Token ist zunächst nur eine Nummer in einem großen Wörterbuch des Modells. Damit das Modell damit rechnen kann, wird diese Nummer in einen **Vektor** übersetzt. Ein Vektor ist eine Liste von Zahlen. Für Menschen ist diese Liste nicht lesbar wie ein Satz. Für das Modell ist sie eine Rechenform, in der Bedeutungsnähe, grammatische Rollen, Stil und Zusammenhang verarbeitet werden können.

Stark vereinfacht: Wörter oder Wortteile, die in ähnlichen Zusammenhängen vorkommen, bekommen im Rechenraum des Modells verwandte Zahlmuster. „Lehrkraft“, „Unterricht“ und „Klasse“ liegen dort nicht als Karteikarten nebeneinander, aber ihre Vektoren können Beziehungen ausdrücken, mit denen das Modell weiterrechnet.

Vereinfacht: Text wird in Tokens zerlegt, Tokens werden als Vektoren berechenbar, mehrere Modellschichten verarbeiten den Zusammenhang, danach wird das nächste Token gewählt. Dieser Vorgang wiederholt sich viele Male.

5.6.7 Was im Modell passiert

Ein modernes Sprachmodell besteht aus vielen Rechenschichten. Jede Schicht nimmt die Vektoren auf, verändert sie und gibt sie an die nächste Schicht weiter. Dabei wird nicht nur jedes Token einzeln betrachtet. Das Modell berechnet auch, welche anderen Tokens im Kontext für den nächsten Rechenschritt Gewicht erhalten. Dieses Prinzip wird oft **Attention** genannt: Das Modell verteilt Aufmerksamkeit auf Teile des Kontextes.

Beispiel: In der Frage „Erkläre das Gedicht und achte auf die letzte Strophe“ muss das Modell den Begriff „das Gedicht“, die Aufgabe „Erkläre“ und den Hinweis „letzte Strophe“ gemeinsam auswerten. Es zählt nicht nur Wörter. Es berechnet, welche Teile des Kontextes für das nächste Stück Antwort wahrscheinlich wichtig sind.

Am Ende entsteht eine Wahrscheinlichkeitsverteilung: Für sehr viele mögliche nächste Tokens schätzt das Modell, wie gut sie als Fortsetzung passen. Dann wird ein Token ausgewählt. Danach wird neu gerechnet: Das gerade erzeugte Token gehört nun zum Kontext, und die Spark berechnet das nächste Token.

5.6.8 Warum Antworten flüssig wirken

Die Spark schreibt nicht erst innerlich einen vollständigen Aufsatz und schickt ihn dann ab. Sie erzeugt Text in kleinen Schritten. Jeder Schritt hängt vom bisherigen Kontext und von den bereits erzeugten Tokens ab. Dadurch kann eine Antwort flüssig erscheinen, obwohl sie intern aus sehr vielen Einzelentscheidungen besteht.

Diese Arbeitsweise erklärt auch Grenzen. Wenn der Kontext unklar ist, wenn wichtige Informationen fehlen oder wenn die Aufgabe widersprüchlich ist, kann das Modell trotzdem eine sprachlich glatte Antwort erzeugen. Eine glatte Form ist aber nicht automatisch ein Beweis für sachliche Richtigkeit. Darum weist Ephraim darauf hin, wichtige Informationen zu prüfen.

5.6.9 Embeddings: Vektoren für Suche und Kontext

Vektoren spielen in Ephraim nicht nur innerhalb des Sprachmodells eine Rolle. Es gibt auch **Embeddings**. Ein Embedding ist ein Vektor, der einen Textabschnitt so beschreibt, dass ähnliche Inhalte rechnerisch nahe beieinander liegen. Das hilft bei der Suche nach relevantem Kontext.

Beispiel: In einem Projektmaterial steht „Photosynthese wandelt Lichtenergie in chemische Energie um“. Eine Schülerin fragt später: „Wie machen Pflanzen aus Licht Zucker?“ Diese Frage verwendet andere Wörter, meint aber ein ähnliches Thema. Ein reiner Wortvergleich findet die Stelle schlechter. Ein Embedding erkennt die inhaltliche Nähe zuverlässiger.

Bei freigegebenem Basiswissen, Projektmaterialien oder bestimmten Dateikontexten kann Ephraim solche Vektoren nutzen, um passende Ausschnitte zu finden. Diese Ausschnitte werden dann als Textkontext an die Spark gegeben. Das Embedding beantwortet also nicht selbst die Frage. Es hilft, den richtigen Kontext zu finden.

Technisch ruft Ephraim dafür den lokalen SGLang-Embedding-Dienst `school-ui-embedding` auf der Spark auf. Der Dienst sieht den freigegebenen Klartextabschnitt oder die konkrete Suchfrage nur für diesen Request, rechnet daraus eine Zahlenliste und speichert sie nicht. Dauerhaft liegen die Textabschnitte und Embedding-Vektoren verschlüsselt auf dem Webserver.

5.6.10 Retrieval: Wie Zusatzwissen in die Antwort kommt

Wenn Ephraim Zusatzwissen verwenden darf, läuft der Ablauf vereinfacht so:

1. Eine Quelle wird vorher freigegeben, geprüft und in kleinere Abschnitte zerlegt.
2. Für diese Abschnitte werden Embeddings berechnet und gespeichert.
3. Bei einer Anfrage wird auch die Frage in einen Suchvektor übersetzt.
4. Ephraim sucht nach inhaltlich passenden Abschnitten.
5. Nur ausgewählte Ausschnitte werden in den Request-Kontext übernommen.
6. Die Spark erzeugt daraus zusammen mit Frage und Regeln die Antwort.

Dieses Verfahren wird oft Retrieval-Augmented Generation genannt: Die Antworterzeugung wird durch gefundenen Kontext ergänzt. Wichtig ist die Richtung: Nicht die Spark geht selbst auf die Suche, sondern Ephraim stellt ihr den geprüften Kontext bereit.

5.6.11 Die Inferenz nutzt kein Internet

Das Sprachmodell auf der Spark surft nicht im Internet und ruft keine Webseiten ab. Es berechnet Antworten aus dem Prompt und dem Kontext, den Ephraim bewusst mitschickt. Wenn Ephraim Wetter, Kalender, Wikipedia oder freigegebene Quellen verwendet, ruft der Webserver diese Quellen kontrolliert ab, prüft sie und gibt nur den daraus abgeleiteten Kontext weiter.

> **Klar gesagt:** Die Spark-Inferenz ist kein Webbrowser. Die KI-Antwort entsteht lokal aus freigegebenem Kontext, nicht durch freies Suchen im Internet.

Dadurch unterscheidet sich Ephraim von öffentlichen Chatbots, bei denen Nutzerinnen und Nutzer oft nicht sehen, welche Infrastruktur oder externen Dienste beteiligt sind. In Ephraim ist die Spark der lokale Rechenort für die Inferenz. Freigegebene Webquellen sind ein Webserver-Thema, kein freier Internetzugang des Modells.

5.6.12 Warum es einen kurzen Klartextmoment gibt

Eine KI kann eine Frage nur beantworten, wenn sie die Frage während der Berechnung lesen kann. Deshalb liegen Prompt und freigegebener Kontext während des Requests im Arbeitsspeicher der Spark im Klartext vor. Entscheidend ist, dass dieser Klartextmoment begrenzt ist: nur für die Antwort, auf der Spark, über eine verschlüsselte Verbindung und ohne dauerhafte Prompt-Protokollierung als Modelllog.

Das ist ein wichtiger Unterschied zwischen Speicherung und Verarbeitung. Ruhende private Inhalte werden in Ephraim verschlüsselt gespeichert. Für eine konkrete Antwort müssen die dafür freigegebenen Inhalte kurz verarbeitet werden. Danach wird der Chatverlauf wieder in der dafür vorgesehenen verschlüsselten Form abgelegt.

5.6.13 Warum Antworten gestreamt werden

Die Spark erzeugt eine Antwort schrittweise. Ephraim zeigt diese Schritte als Stream, damit du nicht warten musst, bis die ganze Antwort fertig ist. Gleichzeitig speichert Ephraim Stream-Daten nur verschlüsselt im Puffer, damit ein Neuladen oder Wiederverbinden möglich ist, ohne Klartextpuffer in der Datenbank abzulegen.

Streaming passt zur inneren Arbeitsweise des Modells: Weil die Spark Token für Token arbeitet, kann Ephraim die Antwort früh anzeigen. Wenn du eine Antwort abbrichst, stoppt Ephraim die weitere Erzeugung. Der bereits sichtbare Teil ist dann nur der Anfang einer Antwort, nicht notwendigerweise ein vollständig geprüfter Abschluss.

5.6.14 Instant und Thinking

Ephraim bietet unterschiedliche Antwortmodi. Der Instant-Modus ist für schnelle, direkte Antworten gedacht. Der Thinking-Modus gibt dem System mehr Raum für strukturierte Bearbeitung komplexerer Aufgaben. Beide Modi ändern nicht das Grundprinzip: Die Spark erhält einen Kontext, zerlegt ihn in Tokens, rechnet mit Vektoren und erzeugt neue Tokens.

Der Unterschied liegt eher in den Vorgaben und Laufzeitentscheidungen: Wie ausführlich soll die Antwort werden? Wie stark soll sie strukturieren? Wie sorgfältig soll sie Zwischenschritte berücksichtigen? Für einfache Fragen ist das oft nicht nötig; für komplexe Aufgaben kann es hilfreich sein.

Die Oberfläche kann im Thinking-Modus zusätzlich einen **Denkprozess** anzeigen. Gemeint ist eine aufbereitete Aktivitätsansicht: Ephraim zeigt verständliche Arbeitsschritte, Werkzeugaufrufe und Ergebnisse wie eine Wikipedia-Trefferzahl. Das macht sichtbar, wie eine Antwort vorbereitet wurde, ersetzt aber nicht die Modellberechnung selbst und öffnet kein vollständiges internes Modellprotokoll.

5.6.15 Aktives Modell und Fähigkeitsgrenzen

In Version 1 nutzt Ephraim für Chatantworten GPT-OSS-120B als lokales Sprachmodell auf der Spark. Die Webanwendung spricht dafür einen lokalen, OpenAI-kompatiblen Inferenzdienst an. Externe Modellanbieter und externe Modellproxys sind nicht Teil dieses produktiven Modellpfads.

Für Version 1 ist das produktive Chatmodell auf Textantworten ausgelegt. Bilder können in Ephraim als Dateien gespeichert, geprüft, angezeigt und über Metadaten in den Kontext eingeordnet werden. Der aktuelle produktive Modellpfad analysiert aber nicht den visuellen Inhalt eines Bildes. Vision-Fähigkeiten sind aktuell nicht freigegeben.

> **Fähigkeitsgrenze:** Der freigegebene V1-Modellpfad ist lokal, textbasiert und auf GPT-OSS-120B beschränkt. Ephraim entscheidet weiterhin, welcher Kontext erlaubt ist und an die Spark übergeben wird.

5.6.16 Was daraus folgt

Die Spark ist kein Mensch, kein Lexikon und kein Browser. Sie ist ein sehr großes lokales Rechensystem, das aus Tokens, Vektoren, Kontext und gelernten Modellgewichten sprachliche Fortsetzungen berechnet. Das kann erstaunlich hilfreich sein, weil viele Muster der Sprache und des Wissens in den Modellgewichten angelegt sind. Es bleibt aber eine Wahrscheinlichkeitsberechnung.

- Gute Prompts helfen, weil sie den Kontext klarer machen.
- Freigegebene Materialien helfen, weil sie relevante Informationen in den Arbeitszettel bringen.
- Embeddings helfen, weil sie passende Ausschnitte auch bei anderen Formulierungen finden können.
- Die Spark nutzt kein Internet, sondern den von Ephraim gelieferten Kontext.
- Antworten müssen geprüft werden, besonders bei Fakten, Rechnen, Quellen und wichtigen Entscheidungen.

Dieser Artikel erklärt die normale Antworterzeugung. Für TTS gilt der eigene Spark-Pfad in der Systemarchitektur; für Projekte gelten zusätzliche Regeln im Projektchat.

Quelle: web/manuals/inferenz-auf-der-spark/index.html

5.7 Datenschutz tiefgehend

> Ephraim ist für schulische KI-Nutzung gebaut: lokale Inferenz auf der Spark, verschlüsselte persönliche Daten, getrennte Projektbereiche und bewusst begrenzte Auswertungen. Dieser Artikel ergänzt die Systemarchitektur um Datenarten, Rechte, Klartextmomente und Verantwortlichkeiten.

Stand: Juni 2026

5.7.1 Grundprinzipien

Datenschutz in Ephraim besteht nicht aus einer einzelnen Maßnahme. Das System kombiniert organisatorische Trennung, Rollen, Verschlüsselung, minimierte Protokolle, bewusst ausgelöste Exporte und eine lokale KI-Infrastruktur. Besonders wichtig ist: KI-Anfragen werden nicht an öffentliche Chatbot-Dienste oder kommerzielle Modell-APIs geschickt. Die Antworterzeugung läuft auf der schuleigenen Spark.

- **Datenminimierung:** Ephraim verarbeitet nur die Daten, die für Konto, Chat, Projekt oder Betrieb nötig sind.
- **Zweckbindung:** Persönliche Chats, Projektchats, zentrale Wissensquellen und Adminfunktionen sind getrennte Kontexte.
- **Verschlüsselung:** Private Inhalte werden verschlüsselt gespeichert; viele Inhalte sind an den persönlichen Vault gebunden.
- **Transparenz:** Projektmonitoring, Fazitfunktionen, Datenexport und Kontolöschung sind sichtbare Vorgänge.
- **Lokale Inferenz:** Die Spark berechnet Antworten aus freigegebenem Kontext und nutzt keinen freien Internetzugriff.

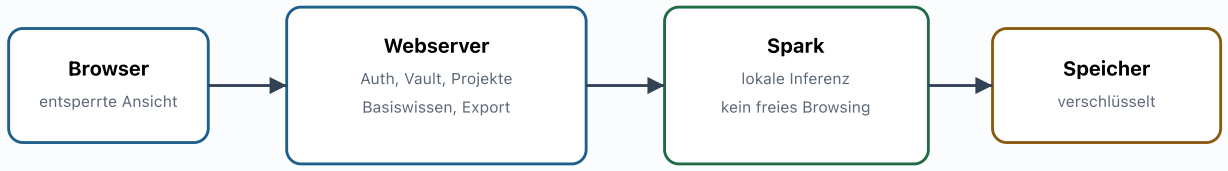
5.7.2 Welche Datenarten Ephraim unterscheidet

Eine saubere Datenschutzbetrachtung beginnt mit der Frage, welche Daten überhaupt existieren. Ephraim behandelt nicht alle Daten gleich, sondern unterscheidet mehrere Bereiche mit eigenen Regeln.

Datenart	Beispiele	Schutzidee
Konto- und Rollendaten	Benutzername, Anzeigename, Rolle, Klasse, E-Mail	Für Anmeldung und Berechtigung nötig; administrativ sichtbar.
Persönlicher Vault	Chats, Dateien, Erinnerungen, private Kalender, Zusammenfassungen	Mit nutzerbezogenem Schlüssel verschlüsselt gespeichert.
Projektbezogene Daten	Projektauftrag, Materialien, Teilnahmestatus, Projektchat, Abschluss	Nur für das jeweilige Projekt und dessen Berechtigungen.
Zentrales Basiswissen	Freigegebene Quellen, Wetter, Wikipedia-Auszüge, zentrale Dateien	Serverseitig verwaltet, nicht Teil privater Nutzer-Vaults.
RAG-, Embedding- und Suchdaten	Text-Chunks, Embedding-Vektoren, Suchfrage, Kontext- und Dokument-IDs	Klartext nur im aktuellen Request; dauerhaft nur verschlüsselt auf dem Webserver.
Betriebsdaten	Sitzungen, Rate-Limits, Sicherheitsereignisse, Cronstatus	Für Sicherheit und Betrieb, mit reduzierten oder gehashten Details.

5.7.3 Wo KI-Verarbeitung stattfindet

Der Browser spricht mit dem Ephraim-Webserver. Der Webserver prüft Anmeldung, Berechtigungen, Projektstatus und verfügbaren Kontext. Für die eigentliche Antwort sendet er die benötigten Inhalte an die Spark. Die Spark führt das Sprachmodell aus und streamt die Antwort zurück.



Dauerhafte Ablage: verschlüsselt. Kurzzeitige Verarbeitung: nur für Anfrage, Antwort, Export oder Anzeige.

Die Spark ist ein Rechendienst für freigegebenen Kontext. Sie ist kein öffentlicher Webdienst, an den Ephraim beliebige Nutzerdaten auslagert.

5.7.4 Kein freier Internetzugriff der KI

Das Sprachmodell auf der Spark ruft keine Webseiten ab, führt keine Websuche durch und telefoniert nicht mit externen KI-Anbietern. Eine Antwort entsteht aus dem Prompt, dem bisherigen erlaubten Verlauf und dem Kontext, den Ephraim ausdrücklich mitschickt.

Ephraim bindet Internetquellen nur in ausdrücklich freigegebenen Fällen ein: etwa Wetterdaten, Wikipedia-Auszüge oder von Administratoren zugelassene Quellen. Dann ruft der Webserver diese Quellen kontrolliert ab, speichert und prüft sie im Basiswissen-Kontext und übergibt nur den aufbereiteten Ausschnitt an die Spark. Die Spark selbst entscheidet nicht frei, welche Webseiten sie besucht.

> **Klarstellung:** „Die KI weiß etwas aus einer Quelle“ bedeutet in Ephraim nicht, dass die KI im Internet gesucht hat. Es bedeutet, dass der Webserver einen freigegebenen Kontext bereitgestellt hat.

5.7.5 Der persönliche Vault

Persönliche Inhalte werden in Ephraim nicht wie einfache öffentliche Daten behandelt. Chats, private Dateien, bestätigte Erinnerungen, private Kalenderinformationen und bestimmte abgeleitete Zusammenfassungen gehören zum persönlichen Vault. Der Vault ist an das Konto und dessen Entsperrung gebunden.

Praktisch bedeutet das: Der Server kann verschlüsselte Daten speichern und wiederfinden, aber er soll private Inhalte nicht dauerhaft als Klartext in Datenbank oder Dateisystem ablegen. Wenn du angemeldet bist und der Vault verfügbar ist, können Inhalte für Anzeige, Suche, Export oder eine konkrete KI-Anfrage

entschlüsselt werden. Danach werden sie wieder nur verschlüsselt gespeichert.

- Private Chatinhalte liegen nicht als allgemeiner Klartextindex in der Datenbank.
- Private Dateien werden verschlüsselt außerhalb des öffentlich erreichbaren Webbereichs gespeichert.
- Persönliche Erinnerungen werden nur für normale persönliche Chats genutzt.
- Private Kalenderabos speichern Links, Rohdaten und Termintexte mit einem eigenen Kalender-Quellschlüssel; dieser Schlüssel ist mit deinem User-Vault verpackt.
- Untis oder WebUntis werden nur über ICS-Links eingebunden; Zugangsdaten werden nicht gespeichert.

5.7.6 Datenschutz im normalen Chat

Ein normaler Chat gehört zu deinem Konto. Beim Senden einer Nachricht prüft Ephraim, ob der Chat zu dir gehört, ob dein Vault verfügbar ist, welche Dateien angehängt sind und ob Personalisierung oder Erinnerungen in diesem Kontext erlaubt sind. Die Spark erhält nur den zusammengestellten Request-Kontext.

Während der Antworterzeugung gibt es einen unvermeidbaren Klartextmoment: Die KI kann nur antworten, wenn sie Frage und freigegebenen Kontext lesen kann. Dieser Klartextmoment ist auf die aktuelle Anfrage im Arbeitsspeicher begrenzt. Die dauerhafte Ablage des Verlaufs erfolgt wieder verschlüsselt.

Für gespeicherte Chats kann die Schule außerdem ein Aufbewahrungslimit pro Konto setzen. Ist die maximale Zahl erreicht, verhindert Ephraim nur das Anlegen eines weiteren gespeicherten Chats. Bestehende Chats bleiben zugänglich; die Nutzerin oder der Nutzer wird in einem Modal über Speicher-, Backup- und Datenschutzgründe informiert und muss mindestens einen nicht mehr benötigten Chat bewusst löschen.

5.7.7 Datenschutz in Projekten

Projekte sind bewusst von persönlichen Chats getrennt. Ein Projektchat folgt dem Auftrag der Lehrkraft und nutzt Projektmaterialien. Persönliche Erinnerungen, private Kalender und private Dateien werden nicht automatisch in den Projektkontext gemischt. Eine private Datei wird erst dann Projektmaterial, wenn sie bewusst als projektbezogene Kopie eingebunden wird.

Wenn Projektmonitoring aktiviert ist, geht es um projektbezogene Aktivität und gegebenenfalls abgeleitete Rückmeldungen. Die Lehrkraft bekommt dadurch nicht automatisch Zugriff auf normale private Chats. Bei individuellen Lehrerfazits kann die betroffene Person dieselbe Rückmeldung ansehen und freiwillig Stellung nehmen; das ist eine Transparenzmaßnahme gegen verdeckte oder einseitige Auswertung. Projektcode-Gäste haben zusätzlich keinen vollständigen Account: Sie arbeiten nur im Projektrahmen und haben keine Konto-, 2FA-, Export- oder Personalisierungsfunktionen.

5.7.8 Dateien, Uploads und Anhänge

Dateien sind besonders sensibel, weil sie oft mehr enthalten als auf den ersten Blick sichtbar ist. Ephraim prüft deshalb Uploads auf mehrere Arten: Dateiname, Größe, erlaubte Endung, erkannter MIME-Typ, Magic Bytes, Struktur von Office-Containern und optional Virenscan. Nicht erlaubte oder widersprüchliche Dateien werden abgelehnt oder als Sicherheitsereignis protokolliert.

Nach erfolgreicher Prüfung werden persönliche Dateien verschlüsselt gespeichert. Im Chat erscheinen sie als Anhänge oder Vorschauelemente, nicht als öffentlich zugängliche Direktlinks. Projektdateien werden als projektbezogene Materialien behandelt und folgen den Projektberechtigungen.

5.7.9 Embeddings, RAG und Spark-Speichergrenze

Embeddings sind Zahlenvektoren, mit denen Ephraim passende Textabschnitte findet. Für Datei-RAG, Projektmaterial-RAG, Basiswissen-RAG und semantische Suche zerlegt Ephraim Quellen in Chunks. Für jeden Chunk wird über den lokalen SGLang-Embedding-Dienst `school-ui-embedding` ein Vektor

berechnet. Bei einer späteren Frage wird auch die Suchfrage in einen Vektor übersetzt. Ephraim vergleicht diese Vektoren auf dem Webserver und übernimmt nur die passendsten Abschnitte in den Kontext.

Datenschutzrechtlich ist wichtig: Für die Vektorerzeugung müssen der freigegebene Klartext-Chunk und die konkrete Suchfrage kurz im aktuellen Request an die interne Spark übergeben werden. Der Embedding-Dienst erzeugt daraus keine Chatantwort, sondern nur eine Zahlenliste. Die Spark speichert dabei weder Klartext-Chunks noch Suchfragen noch Embedding-Vektoren. Sie schreibt diese Inhalte auch nicht als Embedding-Log.

Dauerhaft gespeichert wird auf dem Webserver: persönliche Datei-Chunks und ihre Embeddings mit dem User-Vault verschlüsselt, Projektmaterial-Chunks mit dem jeweiligen Projektdatei-Schlüssel, zentrale Knowledge-Chunks mit dem Knowledge-Schlüssel und private Kalender-Chunks mit einem vaultgebundenen Kalender-Quellschlüssel. Die Suche läuft in der Webanwendung: Sie prüft Schlüsselzugriff und Berechtigungen, vergleicht Vektoren und übernimmt nur zulässige Treffer in den Kontext.

5.7.10 Visualisierungen und aktiver Inhalt

Visualisierungen sind sicherheitsrelevant, weil frei erzeugter Code im Browser gefährlich wäre. Ephraim lässt die KI deshalb nicht beliebiges JavaScript ausführen. Diagramme, Zeichnungen, chemische Strukturen und Plots werden aus begrenzten JSON-Beschreibungen erzeugt und validiert.

HTML-Code aus Antworten bleibt zunächst Code. Eine Vorschau läuft isoliert: ohne Zugriff auf App-Cookies, ohne App-Speicher und ohne Netzwerkzugriff für den generierten Inhalt. Das schützt die laufende Ephraim-Sitzung vor Inhalten, die zwar von der KI formuliert, aber nicht als vertrauenswürdige Anwendung behandelt werden dürfen.

5.7.11 Suche ohne Klartextindex

Die persönliche Chatsuche verzichtet auf einen dauerhaften Klartext-Volltextindex. Wenn gesucht wird, entschlüsselt Ephraim die relevanten Inhalte für diesen Request im laufenden Prozess und liefert Treffer zurück. Dadurch ist die Suche etwas bewusster konstruiert als bei Systemen, die alle Chattexte unverschlüsselt indizieren.

Das ist ein Datenschutzkompromiss zugunsten des Schutzes ruhender Daten: Suche bleibt möglich, aber die Datenbank enthält keinen allgemeinen durchsuchbaren Klartextbestand privater Chattexte.

5.7.12 Export: bewusste Klartext-Ausleitung

Der Datenexport ist absichtlich nutzergesteuert. Ein Export mit Dateien muss private Inhalte entschlüsseln und in eine herunterladbare Form bringen. Ohne Dateien entsteht eine JSON-Datei; mit Dateien erstellt Ephraim ein temporäres ZIP außerhalb des öffentlichen Webbereichs, streamt es an dich und löscht es anschließend wieder.

Nicht exportiert werden Passwörter, 2FA-Geheimnisse, Recovery-Codes, Tokens oder Vault-Schlüssel. Ein Export ist dennoch eine Klartextkopie. Nach dem Download hängt der Schutz dieser Datei von deinem Gerät, deinem Speicherort und deiner Weitergabe ab.

5.7.13 Löschung und Grenzen der Wiederherstellung

Das Chat-Aufbewahrungslimit unterstützt Datensparsamkeit auf Kontoebene. Es löscht keine Chats automatisch, sondern stoppt neue gespeicherte Chats, sobald alle konfigurierten Plätze belegt sind. Die Löschung bleibt eine sichtbare Nutzerentscheidung im Aufräummodal.

Bei der Self-Service-Kontolöschung fragt Ephraim vorab, ob du Daten exportieren möchtest oder bewusst ohne Export fortfährst. Danach verlangt das System Passwort, Bestätigung und Schutz gegen unbeabsichtigte Formulare. Die Löschung entfernt persönliche Daten, Chats, Dateien, Erinnerungen, private Kalender und zugehörige Bezüge.

Wichtig ist der Unterschied zwischen Löschen und Entschlüsseln: Ephraim muss private Inhalte nicht entschlüsseln, um sie zu entfernen. Datenbankeinträge, Dateiblöcke und Zuordnungen können gelöscht werden, auch wenn der Klartext nicht gelesen wird. Bei Passwort-Reset über Administration kann alter Vault-Inhalt bewusst verloren gehen, weil er mit dem alten Schlüssel nicht mehr sinnvoll entschlüsselbar ist.

5.7.14 Sitzungen, 2FA und Sicherheitsereignisse

Ephraim speichert Sitzungstokens nicht im Klartext, sondern als Hash. Geräte- und Zugriffshinweise werden gekürzt oder gehasht, damit eine Sitzungsübersicht möglich ist, ohne unnötig genaue Profile aufzubauen. Nutzerinnen und Nutzer können andere Sitzungen beenden.

Zwei-Faktor-Authentifizierung schützt Konten zusätzlich. Das TOTP-Geheimnis wird verschlüsselt gespeichert; Recovery-Codes werden nur gehasht abgelegt und nach Nutzung verbraucht. Sicherheitsereignisse wie verdächtige Uploads oder wiederholte Fehlversuche dienen dem Schutz des Systems, nicht der inhaltlichen Auswertung von Chats.

5.7.15 Was Administration sieht und was nicht

Administration ist für Betrieb notwendig: Konten anlegen, Rollen setzen, Klassen pflegen, Speichergrenzen konfigurieren, Uploads absichern, Basiswissen verwalten, Systemstatus prüfen. Daraus folgt aber nicht, dass Admins private Klartextchats lesen sollen. Viele Inhalte sind verschlüsselt oder nur in eng begrenzten technischen Abläufen als Klartext verfügbar.

Administrierbar	Nicht Zweck der Administration
Rollen, Klassen, Zugänge, 2FA-Pflicht, Speicherquoten	Private Lernverläufe einzelner Personen ohne Anlass lesen
Uploadregeln, Dateitypen, Sicherheitsereignisse	Private Dateien als allgemeine Wissensquelle durchsuchen
Zentrales Basiswissen und freigegebene Quellen	Private Kalender oder Erinnerungen anderer Nutzer übernehmen
Projektstatus und freigegebene Projektfunktionen	Normale private Chats über ein Projekt öffnen

5.7.16 Ehrliche Klartextmomente

Kein KI-System kann vollständig ohne Klartextverarbeitung funktionieren. Ephraim begrenzt diese Momente, verschweigt sie aber nicht:

- Im Browser siehst du deine Inhalte im Klartext, sobald du sie öffnest.
- Während einer KI-Anfrage liegen Prompt und freigegebener Kontext im Arbeitsspeicher des Webservers und der Spark vor.
- Während einer Embedding-Anfrage liegen freigegebene Text-Chunks und die konkrete Suchfrage kurz im Arbeitsspeicher des Webservers und des lokalen Embedding-Dienstes vor.
- Beim Export entstehen bewusst herunterladbare Klartextdateien oder ein temporäres ZIP.
- Beim Dateidownload und bei Vorschauen wird der Inhalt für die berechtigte Person entschlüsselt ausgeliefert.
- Bei TTS wird nur der bewusst ausgewählte Chattext für die konkrete Spracherzeugung an den internen Dienst übergeben.

Der Datenschutzgewinn liegt darin, dass diese Momente zweckgebunden, zeitlich begrenzt und nicht als dauerhafte Klartextbestände in Datenbank, Dateisystem oder externen Modellplattformen abgelegt werden.

Für Spark-TTS gilt zusätzlich: Der Dienst speichert im Normalbetrieb weder vorzulesenden Text noch erzeugtes Audio und schreibt keine TTS-Anfrage-, Access- oder Syntheselogs. Browser-Web-Speech wird dafür nicht genutzt; ohne Spark-TTS bleiben die Vorlesen-Buttons deaktiviert.

5.7.17 Verantwortung der Nutzerinnen und Nutzer

Technische Schutzmaßnahmen ersetzen nicht umsichtiges Verhalten. Wer vertrauliche Informationen in einen Chat schreibt, macht sie für die konkrete Antwortverarbeitung sichtbar. Wer einen Export herunterlädt, hat danach eine Klartextkopie. Wer Projektmaterial freigibt, teilt eine projektbezogene Kopie mit dem Projekt.

Darum gilt: Nur das eingeben, was für den schulischen Zweck nötig ist; persönliche Geheimnisse wie Passwörter, private Tokens oder medizinische Details nicht in Prompts schreiben; Exporte sorgfältig speichern; Projektmaterial vor der Freigabe prüfen.

Datenschutz in Ephraim ist eine Kombination aus lokaler Infrastruktur, Verschlüsselung, klaren Rollen und bewussten Nutzeraktionen. Die technische Kryptoarchitektur wird im Administratorhandbuch gesondert beschrieben; Begriffe wie Vault, CSRF oder Embedding stehen im Glossar. Dieses Handbuch ersetzt keine rechtlichen Verfahrensdokumente.

Quelle: web/manuals/datenschutz-tiefgehend/index.html

EINORDNUNG

6. Wie Ephraim rechtlich eingeordnet und freigegeben wird

> **Schulische KI ist nicht nur eine technische Entscheidung. Ephraim muss erklären, welche personenbezogenen Daten entstehen, wie Risiken begrenzt werden und welche Dokumente vor dem Produktivbetrieb nötig sind.**

Das rechtliche Kapitel ordnet Ephraim bewusst nicht als fertige Rechtsbehauptung ein. Es erklärt die Prüffelder: DSGVO-Grundsätze, schulischer Auftrag, lokale Verarbeitung, Transparenz, Datenschutz-Folgenabschätzung, Verfahrensverzeichnis und zielgruppengerechte Information.

Die konkreten Legal-Entwürfe gehören nicht in ein öffentliches Handbuch. Das Whitepaper zeigt stattdessen, wie das Projekt diese Aufgaben Schritt für Schritt abarbeitet und warum diese Dokumentation Teil der Architektur ist.

6.1 Rechtliche Herausforderungen

> Schulische KI ist rechtlich anspruchsvoll, weil sie nicht nur allgemeine Texte erzeugt. Sobald eine Plattform mit echten Aufgaben, Dateien, Projekten, Klassen, Lernwegen oder Rückmeldungen arbeitet, verarbeitet sie personenbezogene Daten im Verantwortungsbereich der Schule.

Stand: Juni 2026

6.1.1 Der Ausgangspunkt

Die einfache Frage lautet: Darf eine Schule künstliche Intelligenz im Unterricht einsetzen? Die bessere Frage lautet: Unter welchen Bedingungen darf eine Schule eine KI-Plattform einsetzen, die mit schulischem Kontext arbeitet und dabei Daten von Schülerinnen, Schülern, Lehrkräften und projektbezogenen Gästen verarbeitet?

Ephraim entsteht genau aus dieser zweiten Frage. Ein isoliertes Chatfenster für allgemeine Fragen ist rechtlich und technisch ein anderes Verfahren als eine Schulplattform mit Konten, Rollen, Projekten, Dateien, Basiswissen, Wiederherstellung, Monitoring und lokaler Inferenz. Die rechtliche Herausforderung ist deshalb nicht „KI ja oder nein“, sondern Zweck, Umfang, Transparenz, Schutzmaßnahmen und Kontrolle.

> **Kernsatz:** Schulische KI kann verantwortbar werden, wenn die Schule nicht verdrängt, dass personenbezogene Daten entstehen, sondern diese Verarbeitung begrenzt, erklärt, dokumentiert und technisch schützt.

6.1.2 Der Rechtsrahmen in wenigen Sätzen

Der zentrale Rahmen ist die Datenschutz-Grundverordnung. Besonders wichtig sind die Grundsätze aus Art. 5 DSGVO, die Rechtsgrundlagen aus Art. 6 DSGVO, besondere Kategorien personenbezogener Daten nach Art. 9 DSGVO, Informationspflichten nach Art. 13 und 14 DSGVO, Betroffenenrechte nach Art. 15 bis 22 DSGVO, Datenschutz durch Technikgestaltung nach Art. 25 DSGVO, Auftragsverarbeitung nach Art. 28 DSGVO, das Verzeichnis von Verarbeitungstätigkeiten nach Art. 30 DSGVO, Sicherheit der Verarbeitung nach Art. 32 DSGVO und die Datenschutz-Folgenabschätzung nach Art. 35 DSGVO.

Für eine Schule kommt der schulische Auftrag hinzu. In Baden-Württemberg ist die Verarbeitung schulischer Daten nicht frei erfunden, sondern an Schulrecht, Landesdatenschutzrecht und den Erziehungs- und Bildungsauftrag gebunden. Für den unterrichtlichen KI-Einsatz sind insbesondere § 1 SchG Baden-Württemberg in Verbindung mit § 4 LDSG beziehungsweise Art. 6 Abs. 1 lit. e DSGVO sowie § 115b Abs. 9 SchG Baden-Württemberg als Prüfrahmen relevant. Die Digitalunterrichtsverordnung, vor allem § 6 DUVO, konkretisiert den pädagogischen Rahmen automatisierter, anpassungsfähiger Verfahren. Die KI-Verordnung der Europäischen Union, die Verordnung (EU) 2024/1689 über künstliche Intelligenz,

ersetzt diesen Datenschutzrahmen nicht. Sie kommt als zusätzlicher KI-Rechtsrahmen hinzu. Welche konkreten Pflichten daraus folgen, hängt von Einordnung, Rolle und Einsatzkontext ab. Für Ephraim ist wichtig, dass Rollen, Risiken, Transparenz und menschliche Kontrolle von Anfang an mitgedacht werden.

Diese Normen sind im Handbuch keine Rechtsgutachten. Sie markieren Prüffelder, die eine Schule und ihr Datenschutzbeauftragter nachvollziehen müssen: Zweck, Erforderlichkeit, Datenkategorien, Empfänger, Löschung, technische Schutzmaßnahmen, Transparenz, Risikoabwägung und Freigabe.

6.1.3 Warum Schule fast immer personenbezogen wird

Personenbezogene Daten sind alle Informationen, die sich auf eine identifizierte oder identifizierbare Person beziehen. Im Schulalltag entsteht dieser Bezug sehr schnell. Ein Chat ist einem Konto zugeordnet. Eine Datei enthält eine eigene Lösung. Ein Projekt zeigt Teilnahme, Arbeitsstand und Abschluss. Eine Lehrkraft entscheidet, welche Materialien und Rückmeldungen in einem Unterrichtszusammenhang sinnvoll sind.

Situation	Rechtliche Frage	Typischer Prüfbezug
Persönlicher Chat	Welche Inhalte werden gespeichert und wer kann sie sehen?	Art. 5, 6, 25 und 32 DSGVO
Projektarbeit	Welche Status- und Fazitdaten sieht die Lehrkraft?	Transparenz, Zweckbindung, Datenminimierung
Dateien	Wie werden Uploads geprüft, gespeichert, ausgeliefert und gelöscht?	Art. 5 Abs. 1 lit. e und Art. 32 DSGVO
KI-Kontext	Welche Texte werden der KI für eine Antwort im Klartext gegeben?	Erforderlichkeit, Zugriffskontrolle, Klartextmoment
Externe Quellen	Welche Informationen verlassen die Schule und an wen?	Empfänger, Drittlandbezug, Auftragsverarbeitung

6.1.4 Warum externe KI-Dienste schwer zu prüfen sind

Externe KI-Dienste sind nicht automatisch unzulässig. Sie sind aber prüfintensiv. Eine Schule muss klären, wer Verantwortlicher ist, ob Auftragsverarbeitung vorliegt, welche Unterauftragnehmer beteiligt sind, wo die Verarbeitung stattfindet, welche Daten in Logs, Support, Training, Missbrauchserkennung oder Produktverbesserung auftauchen und wie Löschung, Auskunft und Widerspruch praktisch umgesetzt werden.

Dazu kommt das technische Grundproblem: Ein Sprachmodell kann nur antworten, wenn es die Frage und den benötigten Kontext kurz im Klartext verarbeitet. Transport- und Speicher-Verschlüsselung helfen sehr, beseitigen aber nicht den Klartextmoment bei der eigentlichen Inferenz. Bei einem externen Anbieter liegt dieser Moment in fremder Infrastruktur. Für eine Schule bedeutet das zusätzliche Verträge, Kontrollen, technische Zusicherungen und eine dauerhaft nachvollziehbare Konfiguration.

Für skeptische Leser ist diese Frage berechtigt: Wenn das System morgen anders konfiguriert wird, ein neuer Unterauftragnehmer auftaucht oder ein externer Dienst neue Produktlogik einführt, wer bemerkt das und wer verantwortet es? Ephraim wurde gebaut, um genau diese Abhängigkeit für die Kern-KI-Verarbeitung zu vermeiden.

6.1.5 Was KI zusätzlich schwierig macht

KI-Systeme wirken oft wie normale Software, verhalten sich aber anders. Eine Datenbank liefert gespeicherte Daten zurück. Ein Sprachmodell erzeugt eine Antwort aus Prompt, Kontext und Modellzustand. Dabei können Unsicherheiten, Halluzinationen, unangemessene Schlüsse oder zu selbstbewusst formulierte Antworten entstehen. Pädagogisch ist das beherrschbar, wenn KI-Ausgaben als Arbeitsmaterial behandelt werden. Rechtlich wird es problematisch, wenn daraus automatische Bewertungen, Sanktionen oder verdeckte Profile entstehen. Bei ausschließlich automatisierten Entscheidungen mit rechtlicher oder ähnlich erheblicher Wirkung greift Art. 22 DSGVO. KI-Systeme, die

bestimmungsgemäß Lernergebnisse bewerten, fallen außerdem unter den Bildungsbereich des Anhangs III der KI-Verordnung und sind nach Art. 6 Abs. 2 KI-Verordnung ab dem 2. August 2026 besonders streng zu prüfen.

Für Ephraim ist deshalb entscheidend: KI-Ausgaben ersetzen keine Lehrkraftentscheidung, keine Benotung und keine schulische Verantwortung. Projektmonitoring und Fazitfunktionen müssen vor Projektbeitritt transparent sein. Private Chats dürfen nicht zur verdeckten Beobachtung werden. Eine Plattform für Minderjährige muss erklären, welche Daten sie verarbeitet und welche Grenzen gelten.

6.1.6 Die berechtigten Fragen eines Skeptikers

Ein Datenschutzbeauftragter, ein Elternteil oder eine kritische Lehrkraft muss nicht zuerst an das Versprechen glauben. Die richtigen Fragen sind konkret:

- Welche Datenarten verarbeitet Ephraim und zu welchem Zweck?
- Welche Rechtsgrundlage trägt Konto, Chat, Projekt, Datei, Recovery, TTS und Basiswissen?
- Welche Inhalte sieht die Spark, welche Inhalte speichert sie nicht?
- Welche Daten sind verschlüsselt gespeichert und wann entstehen Klartextmomente?
- Welche Rollen dürfen welche Daten sehen?
- Welche Funktionen sind transparent, bevor Schülerinnen und Schüler sie nutzen?
- Welche externen Verbindungen gibt es und welche sind ausgeschlossen?
- Welche Dokumente, Freigaben und Prüfungen liegen vor Produktivbetrieb vor?

Diese Fragen sind nicht störend. Sie sind der Kern einer seriösen Einführung. Ephraim ist so dokumentiert, dass diese Fragen nicht mit Vertrauenswerbung beantwortet werden, sondern mit Architektur, Handbüchern, Rechtstexten, Tests, Audits und Betriebsvorgaben.

6.1.7 Was daraus für Ephraim folgt

Ephraim muss mehr leisten als eine gute Oberfläche. Das System muss rechtlich lesbar sein. Deshalb sind die Handbücher nicht nur Bedienungsanleitungen, sondern auch eine Brücke zwischen Technik, Pädagogik und Datenschutz.

Die nächsten beiden Artikel zeigen die konkrete Antwort: Ephraims rechtliche Antwort beschreibt die Architekturentscheidungen. Dokumente und Freigaben erklärt, welche Unterlagen und Informationswege vor dem Produktivbetrieb erforderlich sind.

Dieser Artikel erklärt die rechtlichen Prüffelder allgemeinverständlich. Er ersetzt keine Freigabeentscheidung der verantwortlichen Stelle und keine Prüfung durch den Datenschutzbeauftragten.

Quelle: web/manuals/rechtliche-herausforderungen/index.html

6.2 Rechtliche Antwort von Ephraim

> **Ephraim beantwortet rechtliche Herausforderungen nicht mit einem einzelnen Datenschutzversprechen. Die Antwort besteht aus Architektur: lokale KI-Verarbeitung, begrenzte Datenräume, Verschlüsselung, transparente Projektfunktionen, minimiertes Logging und klar dokumentierte Freigabegrenzen.**

Stand: Juni 2026

6.2.1 Nicht risikofrei, sondern kontrolliert

Kein ernsthaftes KI-System ist risikofrei. Ephraims Anspruch ist deshalb nicht, jede Datenschutzfrage durch Technik verschwinden zu lassen. Der Anspruch ist konkreter: Risiken werden sichtbar gemacht, technisch begrenzt, organisatorisch eingehegt und dokumentiert. Genau das entspricht dem risikobasierten Denken der DSGVO.

Der Unterschied zu einer bloßen Nutzungsvorschrift ist wichtig. Eine Regel wie „Gib keine personenbezogenen Daten ein“ verschiebt die Verantwortung auf einzelne Nutzerinnen und Nutzer. Ephraim baut Schutzgrenzen in das System: Rollen, Vault, Projekttrennung, lokale Inferenz, Loggrenzen, Datei-Prüfung, Datenexport, Löschung und transparente Hinweise.

6.2.2 Lokale Inferenz als rechtlich-praktischer Kern

Die eigentliche Antworterzeugung läuft auf der Ephraim Spark. Ephraim sendet KI-Anfragen nicht an öffentliche Modellanbieter oder externe KI-Proxys. Die Spark nutzt keinen freien Internetzugang, um Antworten zu erzeugen, und speichert KI-Anfragen nicht dauerhaft.

Rechtlich-praktisch ist das entscheidend, weil die unvermeidbaren Klartextmomente bei der KI-Verarbeitung in der schulischen Infrastruktur bleiben. Das entbindet die Schule nicht von Dokumentation und Schutzmaßnahmen. Es reduziert aber die Zahl externer Empfänger, Drittlandfragen, Unterauftragnehmer und Produktabhängigkeiten im KI-Kern.

Die technische Vertiefung steht in Inferenz auf der Spark und in der Systemarchitektur.

6.2.3 Datenräume statt Einheitsdatenbank

Ephraim behandelt nicht alle Inhalte gleich. Das System trennt private Konto- und Vault-Daten, Projektkontexte, zentrale Wissensquellen, Admin-Daten, Betriebsdaten und flüchtige KI-Verarbeitung. Diese Trennung ist keine optische Ordnung, sondern eine rechtliche Schutzidee: Zweckbindung und Zugriffskontrolle werden technisch abgebildet.

Datenraum	Schutzentscheidung	Rechtlicher Bezug
Persönlicher Vault	Private Chats, Dateien und abgeleitete Inhalte werden verschlüsselt gespeichert.	Art. 25 und 32 DSGVO
Projekt	Projektauftrag, Material, Teilnahme und Projektchat bleiben projektbezogen getrennt.	Zweckbindung, Datenminimierung
Projektcode-Gast	Gastzugänge sind projektgebunden und bilden keinen allgemeinen privaten Nutzerbereich.	Erforderlichkeit, Zugriffsbeschränkung
Basiswissen	Freigegebene Quellen werden kontrolliert vorbereitet; die Spark surft nicht frei.	Empfänger- und Zweckkontrolle
Betriebsdaten	Logs und Audits werden auf technische und sicherheitsbezogene Zwecke begrenzt.	Art. 5 Abs. 1 lit. e und Art. 32 DSGVO

6.2.4 Transparenz in Projekten

Projekte sind ein besonders sensibler Unterrichtsbereich, weil hier Lehrkraftauftrag, Projektmaterial, Schüleraktivität, Chatverlauf und Abschluss zusammenkommen. Ephraim trennt deshalb private Chats von Projektchats und gibt Lehrkräften keine Rohsicht auf private Schülerkommunikation.

Monitoring und Fazitfunktionen sind keine verdeckten Kontrollinstrumente. Sie müssen im Projekt sichtbar sein. Schülerinnen, Schüler und Projektcode-Gäste werden darauf hingewiesen, wenn persönliches Fazit oder Monitoring aktiv ist. Lehrkräfte erhalten reduzierte projektbezogene Informationen, nicht den vollständigen privaten Chat.

Die Bedien- und Datenschutzgrenzen stehen in Projekte: Datenschutz und Projekte: Abschluss und Fazit.

6.2.5 Klartextmomente ehrlich benennen

Verschlüsselung schützt gespeicherte Daten. Sie ersetzt aber nicht die Tatsache, dass eine KI für eine konkrete Antwort Frage und freigegebenen Kontext im Klartext verarbeiten muss. Ephraim beschreibt diesen Klartextmoment offen, statt ihn sprachlich zu verstecken.

Der Schutz liegt in der Begrenzung: Klartext entsteht im Browser der berechtigten Person, im aktuellen Server-Request und bei der lokalen Spark-Verarbeitung. Dauerhafte Ablage erfolgt wieder verschlüsselt oder als reduzierte technische Metadaten. Die Spark speichert keine Prompts, keine Modellantworten,

keine TTS-Texte und keine KI-Requestpayloads.

> **Wichtig:** Ein vollständig kompromittierter laufender Server ist durch Verschlüsselung allein nicht unsichtbar. Ephraims Schutz richtet sich deshalb zusätzlich auf Rollen, Härtung, Logging-Grenzen, lokale Netzgrenzen, Backups, Audits und reduzierte Persistenz.

6.2.6 Prüffrage und Ephraim-Antwort

Prüffrage	Ephraim-Antwort	Vertiefung
Wer sieht private Chats?	Im normalen Berechtigungsmodell sieht nur die berechtigte Person ihre privaten Chat- und Vault-Inhalte; Lehrkräfte und Admins erhalten keine Rohsicht auf fremde private Vault-Inhalte.	Vault
Wo läuft das Modell?	Auf der schuleigenen Spark; keine externe Modell-API für die Antworterzeugung.	Inferenz
Welche Logs entstehen?	Keine Webserver-Logs mit Anfrageinhalten und Tokens; keine KI-Payloadlogs auf der Spark; Admin- und Betriebslogs begrenzt.	Logging
Wie werden Dateien geschützt?	Uploadprüfung, Dateitypgrenzen, ClamAV im Produktivbetrieb, verschlüsselte Ablage und kontrollierte Auslieferung.	Dateien
Was passiert bei Passwortverlust?	Ohne vorbereitete Recovery kann ein Reset alte Vault-Inhalte unzugänglich machen; mit Recovery ist datenerhaltende Wiederherstellung möglich.	Passwörter
Was ist mit externen Quellen?	Externe Wissensquellen werden kontrolliert vom Webserver vorbereitet; die Spark erhält nur den freigegebenen Kontext.	Basiswissen

6.2.7 Was Ephraim nicht durch Technik allein löst

Einige Punkte bleiben organisatorisch. Die verantwortliche Stelle muss den Einsatz freigeben, die Rechtsgrundlage für den konkreten Betrieb festhalten, den Datenschutzbeauftragten einbeziehen, Eltern, Schülerinnen und Schüler informieren, Lehrkräfte einweisen und klare Nutzungsgrenzen beschließen.

Außerdem ist jede wesentliche Erweiterung neu zu prüfen: ein externes Modell, ein Vision-Modell für Bildinhalte, neue externe Quellen, automatisierte Leistungsbewertung, andere Speicherfristen oder erweiterte Monitoringfunktionen wären keine bloßen Schönheitskorrekturen. Sie verändern das Verfahren und müssen dokumentiert, bewertet und freigegeben werden.

6.2.8 Die kurze rechtliche Antwort

Ephraim löst die rechtliche Herausforderung nicht dadurch, dass keine Daten verarbeitet werden. Das wäre bei sinnvoller schulischer KI unrealistisch. Ephraim löst sie dadurch, dass die Verarbeitung lokal, zweckgebunden, rollenbasiert, verschlüsselt, transparent und dokumentierbar gestaltet wird.

Der nächste Schritt ist die formale Seite: Dokumente und Freigaben erklärt, welche Unterlagen daraus entstehen müssen.

Dieser Artikel beschreibt Ephraims technische und organisatorische Antwort. Die formale Freigabe bleibt Aufgabe der verantwortlichen Stelle in Abstimmung mit dem Datenschutzbeauftragten.

Quelle: web/manuals/rechtliche-antwort-von-ephraim/index.html

6.3 Dokumente und Freigaben

> Rechtliche Arbeit endet nicht bei guter Technik. Vor dem Produktivbetrieb braucht Ephraim eine nachvollziehbare Datenschutzakte: Verzeichnis der Verarbeitungstätigkeiten, Datenschutz-Folgenabschätzung, Einsatzregeln, Informationen für Betroffene und einen dokumentierten Freigabeprozess.

Stand: Juni 2026

6.3.1 Warum diese Dokumente nötig sind

Dokumente sind bei Datenschutz nicht Papierdekoration. Sie zwingen dazu, ein Verfahren so zu beschreiben, dass Außenstehende es prüfen können: Welche Daten werden verarbeitet? Für welchen Zweck? Wer ist verantwortlich? Wer erhält Zugriff? Wie lange wird gespeichert? Welche Schutzmaßnahmen gibt es? Welche Risiken bleiben?

Ephraim verarbeitet schulische Daten in einer neuen technischen Form. Deshalb reicht es nicht, nur den Code zu besitzen oder die Architektur intern verstanden zu haben. Die Schule muss zeigen können, dass sie die Verarbeitung überblickt, begrenzt, verantwortet und regelmäßig überprüft.

> **Prinzip:** Das Handbuch erklärt die Vorgehensweise. Die formalen Entwürfe und Freigabedokumente selbst gehören in die Datenschutzakte der Schule, nicht als Rohfassung in das öffentliche Handbuch.

6.3.2 Verzeichnis der Verarbeitungstätigkeiten

Art. 30 DSGVO verlangt ein Verzeichnis von Verarbeitungstätigkeiten. In der schulischen Praxis wird diese Dokumentationsarbeit häufig als Verfahrensverzeichnis oder Datenschutzakte geführt. Für Ephraim bedeutet das: Das Verfahren muss als Ganzes beschrieben werden, aber auch mit seinen wichtigen Teilverarbeitungen verständlich bleiben. Dazu gehören Konto, Rollen, Chat, Projekte, Dateien, Vault, Kontowiederherstellung, TTS, Basiswissen, Embeddings, Logging, Adminfunktionen und externe Quellen.

Bestandteil	Was geklärt werden muss
Verantwortliche Stelle	Schule, Vertretung, Kontakt und Datenschutzbeauftragter.
Zwecke	Unterricht, Projektarbeit, sichere Konten, Dateiablage, KI-Unterstützung, Betrieb und Sicherheit.
Betroffene Personen	Schülerinnen und Schüler, Lehrkräfte, Administratoren, Projektcode-Gäste und Dritte in Inhalten.
Datenkategorien	Kontodaten, Chatdaten, Dateien, Projektmetadaten, Fazit- und Rückmeldestatus, Schülerstellungnahmen, Wissensquellen, Betriebsdaten und minimierte Sicherheitsereignisse.
Empfänger	Interne Systemkomponenten, berechtigte Rollen, Spark, Mailserver und bewusst konfigurierte externe Quellen.
Löschung	Speicherfristen, Projektarchivierung, Chatlimit, Datenexport, Kontolöschung und technische Cleanup-Prozesse.
Schutzmaßnahmen	Vault, Verschlüsselung, Rollen, 2FA, Logging-Grenzen, redigierte Dateisicherheitsereignisse, Backup, Dateiprüfung und Netztrennung.

Das Verzeichnis ist die Landkarte des Verfahrens. Es muss nicht spannend sein, aber vollständig genug, damit ein Datenschutzbeauftragter die Datenflüsse und Zuständigkeiten nachvollziehen kann.

6.3.3 Datenschutz-Folgenabschätzung

Art. 35 DSGVO verlangt eine Datenschutz-Folgenabschätzung, wenn eine Verarbeitung voraussichtlich ein hohes Risiko für die Rechte und Freiheiten natürlicher Personen zur Folge hat. Schulische KI mit Minderjährigen, persönlichen Inhalten, Projektkontexten, Dateien, lokalen Modellen, Wiederherstellungsmechanismen und Monitoring-Optionen ist genau die Art von Verfahren, bei der eine ernsthafte DSFA naheliegt.

Eine DSFA beschreibt nicht nur Technik. Sie prüft Notwendigkeit und Verhältnismäßigkeit, betrachtet Risiken für Betroffene, benennt Schutzmaßnahmen und bewertet, welches Restrisiko nach diesen Maßnahmen bleibt. Ist ein Datenschutzbeauftragter benannt, muss nach Art. 35 Abs. 2 DSGVO seine Beratung eingeholt werden; die Hinweise gehören in die Freigabeakte.

DSFA-Frage	Bei Ephraim besonders wichtig
Was ist das Risiko?	Ungewollte Offenlegung, falsche Rollenrechte, zu viel Monitoring, unklare Löschung, unangemessene KI-Nutzung.
Wie wahrscheinlich ist es?	Bewertung anhand von Code, Rollenmodell, Testbetrieb, Audits, manueller Prüfung und Betriebserfahrung.
Wie schwer wäre die Folge?	Schülerinnen und Schüler sind besonders schutzbedürftig; private Inhalte und Lernstände brauchen hohe Sorgfalt.
Welche Maßnahme senkt das Risiko?	Lokale Spark, Vault, Verschlüsselung, keine Rohchat-Einsicht, keine KI-Payloadlogs, Transparenzhinweise, Schüleransicht individueller Lehrerfazits und freiwillige Stellungnahme.
Was bleibt offen?	Organisatorische Freigabe, Einsatzordnung, Restore-Nachweis, Information der Betroffenen und laufende Prüfung.

Für Projektfeedback ist in der DSFA besonders wichtig, dass Transparenz nicht nur als Hinweis vor dem Start erscheint. Wenn ein individuelles Lehrerfazit entsteht, kann die betroffene Person dasselbe Fazit sehen und freiwillig kommentieren. Die Handreichungen für Datenschutzbeauftragte und Lehrkräfte sollten diese Schutzmaßnahme ausdrücklich beschreiben, weil sie verdeckte oder einseitige Auswertung begrenzt.

6.3.4 Informationen für Eltern, Schülerinnen, Schüler und Lehrkräfte

Datenschutzrechtlich reicht eine interne Akte nicht. Betroffene Personen müssen verstehen, was mit ihren Daten geschieht. Art. 13 DSGVO betrifft Informationen bei der Erhebung bei der betroffenen Person; Art. 14 DSGVO betrifft Informationen, wenn Daten nicht direkt bei ihr erhoben werden. Je nach Erhebungssituation gehören dazu Verantwortlicher, Zwecke, Rechtsgrundlagen, Datenkategorien, Empfänger, Speicherfristen, Rechte und Beschwerdemöglichkeit.

Für Ephraim braucht diese Information mehrere Zielgruppen, weil nicht alle dieselbe Perspektive haben:

Zielgruppe	Was erklärt werden muss
Schülerinnen und Schüler	Was Ephraim tut, was die KI nicht kann, welche Daten privat bleiben, wann Projektmonitoring aktiv ist, wie individuelle Lehrerfazits angezeigt werden können und wie man vorsichtig arbeitet.
Eltern	Warum die Schule Ephraim nutzt, wo die KI läuft, welche Daten verarbeitet werden, welche Rechte bestehen und wer Ansprechpartner ist.
Lehrkräfte	Erlaubte Zwecke, verbotene Nutzung, Umgang mit sensiblen Daten, Projektregeln, keine automatisierte Benotung, keine Rohchatübernahme und Transparenzpflichten bei Lehrerfeedback.
Administratoren	Rollenverantwortung, 2FA, Logging-Grenzen, keine private Vault-Einsicht, Datenexport, Löschung, Updates und Auditpflichten.

Die Sprache muss dabei zur Zielgruppe passen. Schülerinnen und Schüler brauchen keine juristische Langfassung. Sie müssen verstehen, was privat ist, was im Projekt sichtbar wird, warum KI Fehler machen kann und an wen sie sich wenden können.

6.3.5 Einsatzordnung und pädagogische Grenzen

Eine Einsatzordnung beschreibt, wofür Ephraim genutzt wird und wofür nicht. Sie ist wichtig, weil Technik allein keine pädagogische Verantwortung ersetzt. KI-Ausgaben sind Arbeitsmaterial. Sie dürfen keine automatisierte Benotung, keine Sanktion und keine verdeckte Leistungsbewertung begründen. Diese Grenze ist auch rechtlich zentral: Art. 22 DSGVO schützt vor ausschließlich automatisierten erheblichen Entscheidungen; KI-Systeme zur Bewertung von Lernergebnissen sind ab dem 2. August 2026 nach Art. 6 Abs. 2 in Verbindung mit Anhang III Nr. 3 lit. b der KI-Verordnung besonders streng einzuordnen.

Die Nutzungsordnungen sind die zielgruppengerechte Ausprägung dieser Einsatzordnung im System. Sie sollen nicht erklären, wie Ephraim technisch funktioniert, sondern welche Regeln für Schülerinnen und Schüler, Projektcode-Gäste, Lehrkräfte oder Administration in ihrer jeweiligen Rolle gelten. Die konkreten Ordnungstexte gehören nicht als Rohfassung in dieses Handbuch, sondern in den Freigabeprozess der Schule.

Ephraim unterscheidet dabei vier Arten von Nutzungsordnungen. Diese Trennung ist wichtig, weil die Gruppen nicht dieselben Möglichkeiten, Pflichten und Risiken haben. Eine Schülerin braucht andere Regeln als eine Lehrkraft, und ein Projektcode-Gast braucht keine Ordnung für Funktionen, die er gar nicht nutzen kann.

Nutzungsordnung	Zielgruppe	Worum es vor allem geht
Schülerordnung	Reguläre Schülerkonten	Schulische Nutzung, eigene Chats, Projekte, Dateien und vorsichtiger Umgang mit privaten Angaben.
Gästeordnung	Projektcode-Gäste	Begrenzte Teilnahme an einem Projekt ohne vollständigen persönlichen Arbeitsbereich.
Lehrkräfteordnung	Lehrkräfte	Pädagogische Verantwortung, Projektsteuerung, Materialfreigabe und Grenzen bei sensiblen Daten.
Administrationsordnung	Admins und Super-Admins	Rollenverantwortung, 2FA, Statuskontrolle, Datenminimierung, Dateisicherheitsereignisse ohne sichtbare Dateinamen und sorgfältige Systemverwaltung.

Die Verwaltung im Admin-Dashboard ist bewusst differenziert. Sie trennt Entwurf, Aktivierung, Akzeptanzsteuerung und Statuskontrolle, damit neue Fassungen nicht unbemerkt oder unklar in den Betrieb gelangen.

Admin-Funktion	Datenschutzbedeutung
Aktive Fassung je Zielgruppe anzeigen	Es ist nachvollziehbar, welche Ordnung gerade für welche Gruppe gilt.
Markdown-Fassung mit Titel und Version hochladen	Ordnungstexte bleiben versioniert und können vor Aktivierung geprüft werden.
Entwurf speichern oder Fassung aktivieren	Eine neue Ordnung kann vorbereitet werden, ohne sofort Nutzerinnen und Nutzer zu blockieren.
Akzeptanz erneut verlangen oder Hinweis-Modus wählen	Wesentliche Änderungen können blockierend akzeptiert werden; kleinere Änderungen können als dauerhafter Hinweis in Mein Ephraim erscheinen.
Akzeptanzstatus in Übersicht und Nutzerverwaltung prüfen	Offene Akzeptanzen bleiben sichtbar, ohne Chat-, Datei- oder Projektinhalte offenzulegen.

Ephraim kann solche Ordnungen als versionierte Markdown-Fassungen verwalten. Wenn für eine Zielgruppe keine aktive Fassung existiert, erscheint nichts und es wird auch keine Akzeptanz verlangt. Wenn eine aktive Fassung existiert, wird sie nach dem Login und vor der weiteren App-Nutzung vollständig angezeigt. Neue Fassungen können erneut blockierend akzeptiert werden müssen oder als nicht wegklickbarer Hinweis in Mein Ephraim erscheinen, bis die neue Fassung akzeptiert wurde.

Der Akzeptanznachweis bleibt datensparsam: gespeichert werden die betroffene Fassung, der Inhalts-Hash, der Zeitpunkt, die Zielgruppe, die UI-Sprache und bei Projektcode- Gästen optional der Projektbezug. IP-Adresse und User-Agent werden dafür nicht gespeichert. Die Akzeptanz ersetzt weder Datenschutzinformation noch pädagogische Einführung; sie dokumentiert nur, dass die für diese Zielgruppe aktive Ordnung akzeptiert wurde.

- Ephraim unterstützt Lernen, Projektarbeit, Erklärung, Materialarbeit und Reflexion.
- Lehrkräfte bleiben für Unterricht, Bewertung und pädagogische Entscheidungen verantwortlich.
- Besondere Kategorien personenbezogener Daten nach Art. 9 DSGVO werden nicht gezielt verarbeitet.
- Projektmonitoring wird nur genutzt, wenn es vor Projektbeitritt transparent ist.
- Externe Modelle, Vision-Funktionen oder neue Datenquellen benötigen eine neue Prüfung.

6.3.6 Schrittweise Freigabe statt großer Behauptung

Ephraims Vorgehen ist schrittweise. Zuerst wird das System technisch gebaut und dokumentiert. Dann werden Datenflüsse, Risiken und Schutzmaßnahmen beschrieben. Danach folgen Tests, Audits, manuelle Prüfung, Datenschutzbewertung, Korrekturen und erst dann ein begrenzter Pilotbetrieb.

Freigabe ist kein einmaliger Haken. Wenn sich Funktionen, Datenflüsse oder Risiken ändern, beginnt die passende Prüfschleife erneut.

6.3.7 Warum die Entwürfe nicht ins Handbuch gehören

Das Handbuch richtet sich an Interessierte. Es soll erklären, wie Ephraim gedacht ist. Formale Datenschutzdokumente sind dagegen Arbeits- und Freigabedokumente der verantwortlichen Stelle. Sie enthalten interne Bewertungen, offene Punkte, konkrete Betriebsannahmen, Zuständigkeiten und Prüfergebnisse. Auch konkrete Nutzungsordnungstexte gehören nicht als ungeprüfte Rohfassungen in eine öffentliche Dokumentationsseite.

Sinnvoll ist deshalb diese Trennung: Das Handbuch erklärt öffentlich die Logik und die Vorgehensweise. Die Datenschutzakte enthält die vollständigen formalen Unterlagen. Aus diesen Unterlagen entstehen wiederum verständliche Informationen für Eltern, Schülerinnen, Schüler und Lehrkräfte.

6.3.8 Was vor Produktivbetrieb abgeschlossen sein muss

Vor einem echten Produktivbetrieb müssen die Unterlagen nicht perfekt klingen, sondern tragfähig sein. Entscheidend ist, dass offene Risiken, Zuständigkeiten und Grenzen nicht im Nebel bleiben.

- Verzeichnis der Verarbeitungstätigkeiten ist aktuell und passt zur tatsächlichen V1-Konfiguration.
- Datenschutz-Folgenabschätzung bewertet die realen Funktionen und benennt Restrisiken.
- Datenschutzbeauftragter und Schulleitung kennen offene Punkte und Freigabegrenzen.
- Eltern, Schülerinnen, Schüler und Lehrkräfte erhalten zielgruppengerechte Informationen.
- Einsatzordnung und pädagogische Grenzen sind beschlossen und praktisch kommuniziert.
- Aktive Nutzungsordnungen sind, sofern vorgesehen, zielgruppengerecht freigegeben und vor Nutzung akzeptierbar.
- Testbetrieb, Audits, Backups, Restore-Verfahren und Betriebsmonitoring sind geprüft.
- Jede spätere wesentliche Änderung löst eine erneute Prüfung der betroffenen Dokumente aus.

Dieser Artikel beschreibt die rechtliche Vorgehensweise. Die konkreten Entwürfe bleiben interne Freigabedokumente der verantwortlichen Stelle.

Quelle: web/manuals/rechtliche-dokumente-und-freigaben/index.html

EINORDNUNG

7. Wie Ephraim im Schulbetrieb kontrollierbar bleibt

> **Ein schulisches KI-System braucht mehr als gute Antworten. Es braucht Statusanzeigen, Limits, Wartung, Datei-Sicherheit, Drittanbieter-Kontrolle und nachvollziehbare Betriebsdiagnose.**

Die Admin-Manuals erklären Ephraim aus Betriebssicht: Welche Dienste laufen? Was macht Cron? Wie werden Dateien geprüft? Welche Drittanbieter-Komponenten sind gebündelt? Wie lassen sich Benchmarks lesen?

Diese Betriebsebene ist entscheidend, weil Schulen keine Blackbox übernehmen sollen. Ephraim legt sichtbar offen, welche Komponenten beteiligt sind und welche Grenzen sie haben.

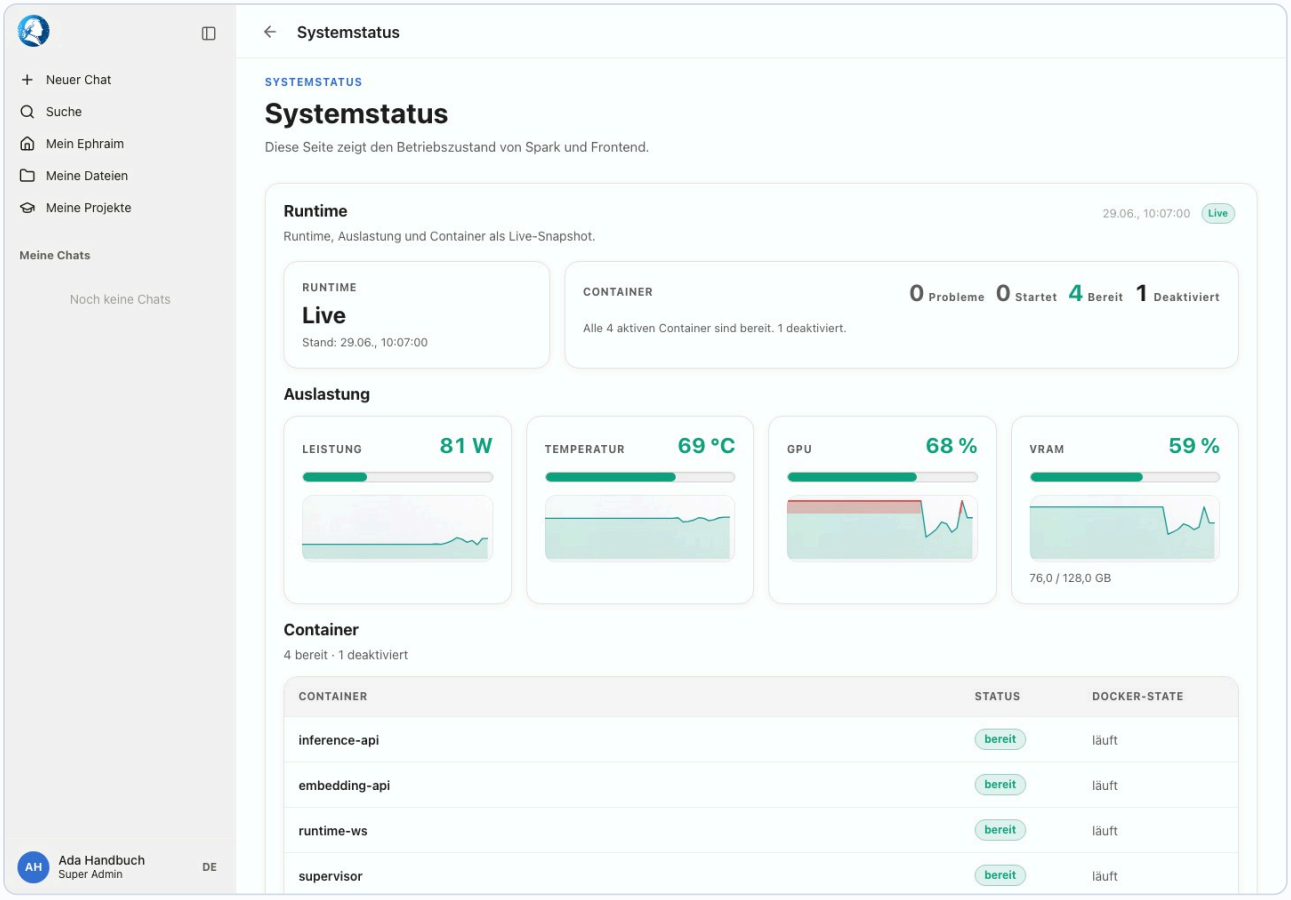
7.1 Status

> Die Statusseite ist der schnelle Betriebsüberblick für Ephraim: Spark-Erreichbarkeit, Runtime, Containerzustände, Auslastung und Hinweise auf Dienste, die aktiv, bereit, problematisch oder bewusst deaktiviert sind.

Stand: Mai 2026

7.1.1 Zweck der Statusseite

Die Statusseite beantwortet die Betriebsfrage: „Kann Ephraim gerade KI-Funktionen zuverlässig anbieten?“ Sie ist bewusst als Übersicht gebaut und nicht als Reparaturkonsole. Administratoren sehen auf einen Blick, ob Webserver, Spark-Dienste und Runtime-Container erwartungsgemäß erreichbar sind.



Die Statusseite ist der erste Diagnosepunkt: Sie zeigt technische Zustände, aber keine privaten Chatinhalte.

7.1.2 Welche Signale angezeigt werden

Signal	Bedeutung
Spark und Inference	Erreichbarkeit der lokalen KI-Inferenz und Antwortfähigkeit des Modells.
Runtime und Supervisor	Technische Betriebsdienste für Terminal, Logs, Status und administrative Aktionen.
Container	Zustand einzelner Dienste: laufend, bereit, problematisch oder deaktiviert.
Ressourcen	Auslastungs- und Gesundheitswerte wie Speicher, VRAM, Temperatur oder Stromdaten. Die Anzeige hängt davon ab, welche Werte die Runtime im Snapshot liefert.

7.1.3 Status richtig lesen

Ein rotes oder gelbes Signal bedeutet nicht automatisch, dass die Anwendung komplett ausgefallen ist. Manche Dienste sind optional oder nur für bestimmte Funktionen relevant. Ein deaktivierter Container ist kein Fehler, wenn er bewusst deaktiviert wurde. Problematisch wird es, wenn ein Dienst aktiv sein soll, aber nicht antwortet oder wiederholt fehlschlägt.

> **Praktisch:** Bei Nutzerbeschwerden zuerst prüfen, ob die Inference erreichbar ist. Bei Admin-Terminal-Problemen dagegen Runtime und Supervisor prüfen.

Fehlende Temperatur-, Strom- oder VRAM-Werte sind deshalb nicht automatisch ein Fehler der Statusseite. Entscheidend ist, ob der Runtime-Snapshot diese Werte für die aktuelle Spark-Konfiguration liefert.

7.1.4 Datenschutz und Sichtbarkeit

Die Statusseite zeigt technische Zustände, keine privaten Chatinhalte. Sie dient der Betriebsdiagnose und soll nicht zur Analyse einzelner Nutzerverläufe verwendet werden. Wenn Detaildiagnose nötig ist, erfolgt sie über Logs, Cronstatus oder das Supervisor-Terminal mit klarer Fragestellung.

7.1.5 Typische Reaktion auf Probleme

1. Prüfen, ob der Webserver selbst erreichbar ist und Anmeldungen funktionieren.
2. Auf der Statusseite Spark, Inference und Runtime vergleichen.
3. Bei Containerproblemen im Supervisor-Webterminal Logs und Status abfragen.
4. Bei wiederkehrenden Wartungsproblemen Cronstatus und letzte Läufe prüfen.
5. Erst danach Dienste gezielt neu starten oder Super-Admin-Aktionen ausführen.

Die Statusseite ist der Einstieg in die Diagnose. Reparaturen und Detailanalysen gehören in die dafür vorgesehenen Adminwerkzeuge, vor allem Logging, Supervisor-Webterminal und Cron.

Quelle: web/manuals/admin-status/index.html

7.2 Inferenz-Benchmarks

> Die Inferenz-Benchmarks zeigen, wie die lokale Spark im Instant-Modus unter synthetischer Schullast reagiert. Ein kurzer Thinking-Kontrolllauf ordnet ergänzend ein, warum Thinking ähnlich schnell startet, aber länger rechnet. Der Artikel erklärt die Messwerte, die Diagramme und die praktische Betriebsfolge: Ephraim bleibt stabil, bildet bei dichter Last aber eine Warteschlange.

Stand: Juni 2026

7.2.1 Kurzfassung für Admins

Die bisher ausgewerteten Instant-Benchmarks zeigen ein klares Bild: Ephraim verarbeitet die gemessene Last ohne technische Fehler. Bei realistischen Schulszenarien mit bis zu etwa 100 zufallsverteilten Anfragen in fünf Minuten wirkt das Verhalten schulisch gut brauchbar. Bei 200 zufallsverteilten Anfragen in fünf Minuten bleibt das System ebenfalls stabil, fühlt sich aber nicht mehr wie sofortige Antwort an.

Der ergänzende Thinking-Teillauf bestätigt die praktische Erwartung: Thinking beginnt ähnlich schnell sichtbar zu antworten wie Instant, braucht aber länger bis zur fertigen Antwort. Für die Kapazitätsplanung bleibt deshalb Instant die belastbare Hauptkurve; Thinking ist der bewusst langsamere Modus für Aufgaben, bei denen gründlichere Verarbeitung wichtiger ist als die kürzeste Gesamtdauer.

> **Kernaussage:** Die Spark bricht unter der gemessenen Last nicht weg. Bei sehr dichter Last verwirft Ephraim die Anfragen nicht, sondern arbeitet sie geordnet über eine Warteschlange ab. Dadurch steigen vor allem die Zeit bis zum ersten sichtbaren Antwortteil und die Gesamtlatenz.

Für den Schulbetrieb ist das ein wichtiges Signal: Die Frage lautet nicht nur, wie viele Antworten am Ende erfolgreich abgeschlossen werden. Entscheidend ist auch, ob Nutzerinnen schnell sehen, dass ihre Anfrage angenommen wurde und bearbeitet wird.

7.2.2 Was genau gemessen wird

Die Benchmarks messen ausschließlich die Inferenzstrecke der Spark. Frontend, Browser, Anmeldung, Session-Verwaltung, Chatoberfläche, Projektlogik, Datei-Retrieval und echte Unterrichtsabläufe sind nicht Teil dieser Messung. Die Ergebnisse beschreiben also die technische Antwortleistung des lokalen Modells unter synthetischer Last.

Verwendet wurden synthetische deutschsprachige Prompts aus typischen Schulkontexten: Erklärungen, kurze Wissensfragen, Strukturierungsaufträge und mathematisch-logische Aufgaben. Echte Nutzerfragen, echte Chatverläufe und Projektinhalte wurden nicht verwendet. Antworttexte wurden für die Veröffentlichung nicht gespeichert.

Merkmal	Gemessener Stand
Modus	Instant; ergänzender Thinking-Kontrolllauf
Reasoning-Stufe	Instant mit kurzer Antwortführung; Thinking mit zusätzlicher Denkzeit
Modellpfad	Lokaler Schul-Inferenzdienst
Prompt-Typ	Synthetische deutschsprachige Unterrichtsprompts
Inhaltsspeicherung	Keine veröffentlichten Antworttexte, keine echten Nutzerdaten

7.2.3 Kennzahlen richtig lesen

Ein Lasttest ist nur nützlich, wenn die Kennzahlen richtig gelesen werden. Eine niedrige Fehlerquote sagt: Die Anfragen wurden technisch verarbeitet. Sie sagt aber nicht, dass sich die Nutzung unter Last angenehm anfühlt.

Kennzahl	Bedeutung	Warum sie wichtig ist
p50	Median: Die Hälfte der Anfragen ist schneller, die andere Hälfte langsamer.	Beschreibt den typischen Fall.
p95	95 Prozent der Anfragen sind schneller als dieser Wert, 5 Prozent langsamer.	Zeigt die langsamen Fälle, die im Unterricht als störend wahrgenommen werden.
Zeit bis zum ersten Token	Zeit, bis die Antwort sichtbar zu beginnen scheint.	Bestimmt stark das Nutzergefühl: „Ephraim arbeitet“ oder „es passiert nichts“.
Gesamtlatenz	Zeit bis zum Abschluss der Antwort.	Zeigt, wann der vollständige Text vorliegt.
Durchsatz	Erfolgreiche Antworten pro Minute.	Ein Plateau zeigt, dass zusätzliche Last vor allem Wartezeit erzeugt.

> **First Token ist zentral:** Eine Antwort, die nach einer halben Sekunde sichtbar beginnt und dann 30 Sekunden schreibt, fühlt sich anders an als eine Antwort, bei der 30 Sekunden lang gar nichts sichtbar passiert.

7.2.4 Zwei Szenarien

Die vorhandenen Instant-Läufe bestehen aus zwei unterschiedlichen Lastbildern. Beide sind wichtig, weil Schulen sowohl harte Spitzen als auch verteilte Nutzung erleben.

Szenario	Was simuliert wird	Was daraus gelesen wird
Gleichzeitigkeit	1 bis 90 virtuelle Nutzer starten exakt gleichzeitig; jede Stufe wird dreimal wiederholt.	Wie die Spark auf harte Lastspitzen reagiert.
Fünf-Minuten-Szenario	50, 100 oder 200 Nutzer starten zufallsverteilt innerhalb von 300 Sekunden.	Wie sich eine natürlichere Unterrichtslast über mehrere Minuten anfühlt.

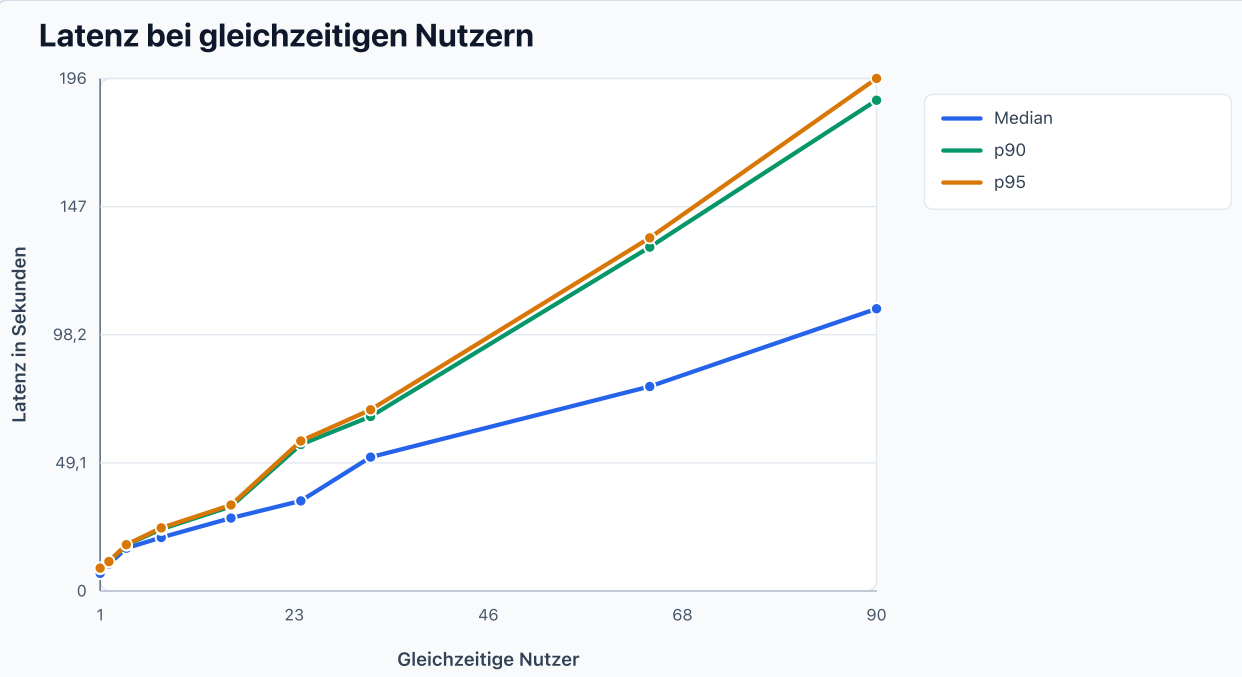
Das Fünf-Minuten-Szenario ist für Schulbetrieb besonders aussagekräftig. Auch wenn viele Personen mit Ephraim arbeiten, verteilen sich echte Fragen meist über Minuten: Einige lesen, andere überarbeiten, andere diskutieren, wieder andere stellen Folgefragen.

7.2.5 Ergebnis: Gleichzeitige Nutzer

Der Gleichzeitigkeitstest vom 03.06.2026 schloss 723 von 723 Messanfragen erfolgreich ab. Die Fehlerquote lag in allen Stufen bei null Prozent. Gleichzeitig zeigt der Lauf eine deutliche Sättigungskurve: Ab etwa 24 bis 32 gleichzeitig gestarteten Anfragen wird die interne Warteschlange sichtbar.

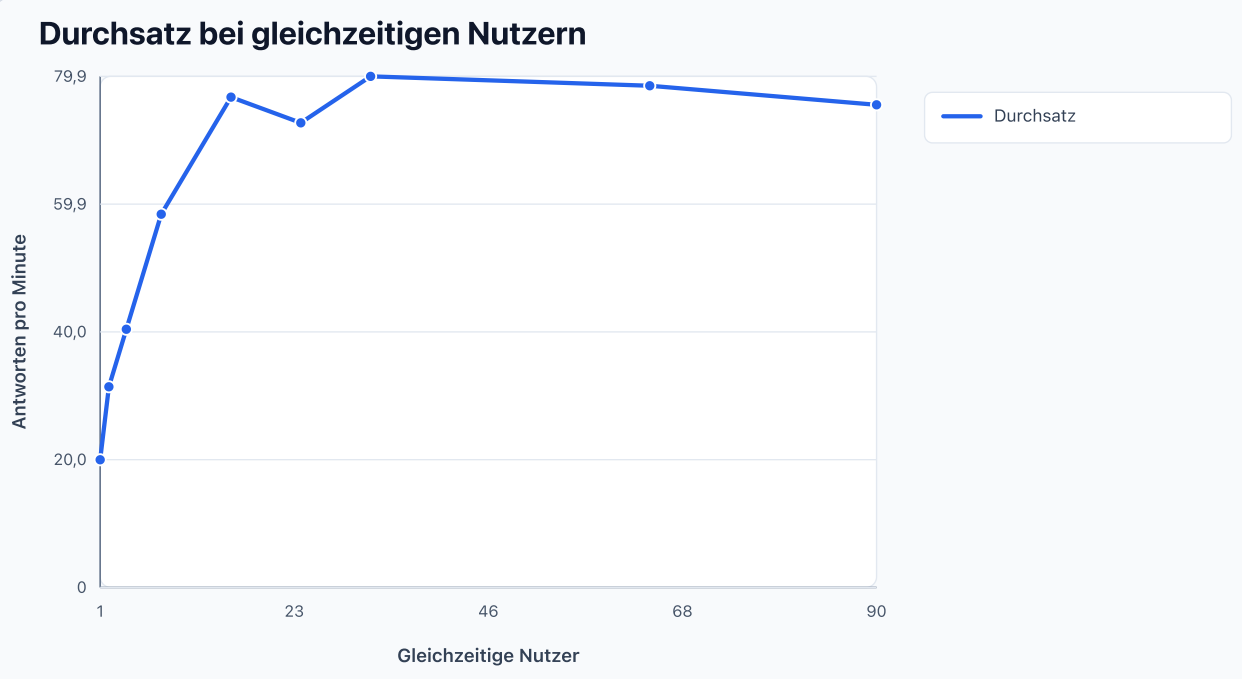
Gleichzeitige Nutzer	Erfolgreich	Fehlerquote	p50 Latenz	p95 Latenz	p50 erster Token	p95 erster Token	Durchsatz
1	3/3	0 %	6,8 s	8,8 s	0,2 s	0,2 s	20,0/min
2	6/6	0 %	10,3 s	11,3 s	0,3 s	0,3 s	31,4/min
4	12/12	0 %	16,4 s	17,7 s	0,3 s	0,3 s	40,4/min
8	24/24	0 %	20,5 s	24,2 s	0,3 s	0,3 s	58,4/min
16	48/48	0 %	27,9 s	32,9 s	0,3 s	13,1 s	76,7/min
24	72/72	0 %	34,5 s	57,5 s	0,4 s	30,6 s	72,7/min
32	96/96	0 %	51,3 s	69,4 s	22,3 s	47,6 s	79,9/min
64	192/192	0 %	78,3 s	135,3 s	50,0 s	108,2 s	78,5/min
90	270/270	0 %	108,1 s	196,3 s	73,7 s	163,2 s	75,5/min

Der Durchsatz steigt zuerst an und erreicht dann ein Plateau im Bereich von etwa 75 bis 80 Antworten pro Minute. Zusätzliche Gleichzeitigkeit erhöht den Durchsatz danach nicht mehr wesentlich, sondern verlängert vor allem die Wartezeit.



Die Latenzgrafik zeigt die Sättigung: Die Spark bleibt erreichbar, aber die langsameren Fälle wachsen bei höherer Parallelität deutlich an.

So liest man das Diagramm: Die waagerechte Achse zeigt die Zahl gleichzeitig gestarteter Nutzerinnen und Nutzer, die senkrechte Achse die Sekunden bis zur fertigen Antwort. p50 beschreibt den typischen Fall, p90 und p95 die langsameren Fälle. Wichtig ist der Knick ab etwa 24 bis 32 gleichzeitigen Starts: Ab dort wächst vor allem die Wartezeit in der Queue.



Der Durchsatz steigt zunächst und erreicht dann ein Plateau. Ab diesem Bereich bedeutet mehr gleichzeitige Last vor allem längere Warteschlangen.

So liest man das Diagramm: Die waagerechte Achse zeigt wieder die gleichzeitigen Starts, die senkrechte Achse die erfolgreich abgeschlossenen Antworten pro Minute. Eine steigende Linie bedeutet, dass zusätzliche Last noch in mehr abgeschlossene Antworten umgesetzt wird. Das Plateau bei etwa 75 bis 80 Antworten pro Minute zeigt die praktische Kapazitätsgrenze dieses Laufs.

Die Fehlerquote bleibt bei null Prozent. Das ist ein gutes Stabilitätssignal, ersetzt aber nicht die Bewertung von First Token und Gesamtlatenz.

So liest man das Diagramm: Jede Säule steht für eine Laststufe, die senkrechte Achse zeigt den Anteil fehlgeschlagener Anfragen. Null Prozent in allen Stufen bedeutet: Die Anfragen wurden technisch angenommen und abgeschlossen. Für das Nutzergefühl müssen trotzdem die Latenzgrafiken daneben gelesen werden.

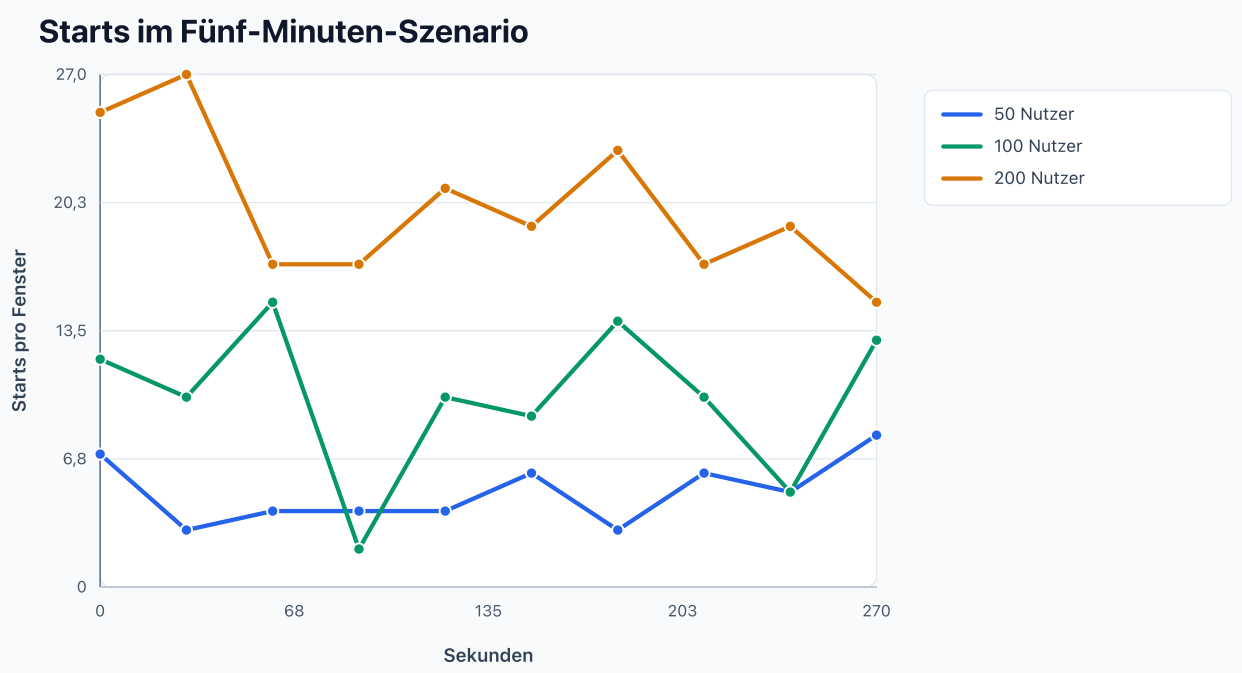
7.2.6 Ergebnis: Zufallsverteilte Anfragen

Das Fünf-Minuten-Szenario vom 04.06.2026 schloss 350 von 350 Messanfragen erfolgreich ab. Aus Schulsicht ist dieses Szenario besonders wichtig, weil Unterrichtslast selten exakt gleichzeitig entsteht.

Virtuelle Nutzer in 5 Minuten	Erfolgreich	Fehlerquote	Max. Parallelität	p50 Latenz	p95 Latenz	p50 erster Token	p95 erster Token	Durchsatz
50	50/50	0 %	6	15,1 s	23,0 s	0,3 s	0,4 s	9,5/min
100	100/100	0 %	17	27,3 s	37,0 s	0,4 s	0,5 s	18,6/min
200	200/200	0 %	71	94,0 s	144,3 s	62,9 s	114,0 s	27,4/min

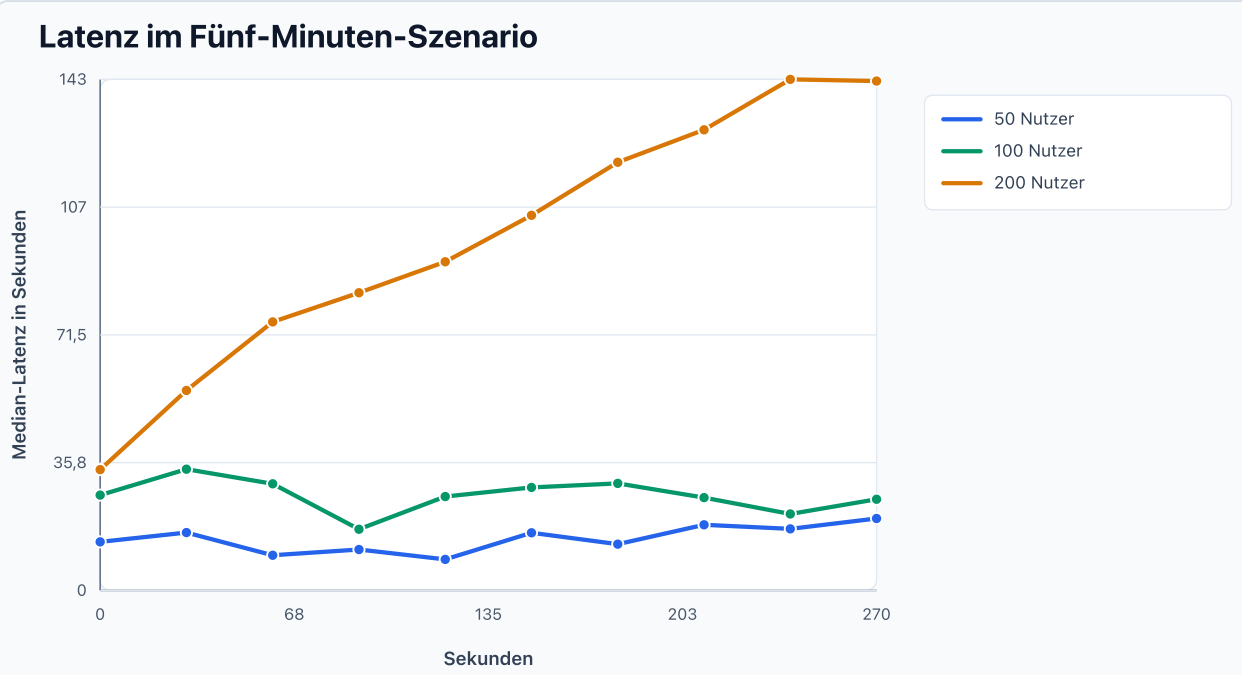
50 zufallsverteilte Anfragen in fünf Minuten sind in diesem Lauf unproblematisch. Auch 100 zufallsverteilte Anfragen sehen für reale schulische Nutzung gut aus: alle Anfragen waren erfolgreich, die p95-Gesamtlatenz lag bei 37 Sekunden und die Zeit bis zum ersten Token blieb niedrig.

200 zufallsverteilte Anfragen in fünf Minuten zeigen die Reserve und die Grenze zugleich: Ephraim bleibt stabil, aber die p50-Zeit bis zum ersten Token steigt auf 62,9 Sekunden. Der Benchmark-Runner selbst startete die Anfragen pünktlich; die Verzögerung entsteht nach Annahme der Anfragen in der Inferenz-Abarbeitung.



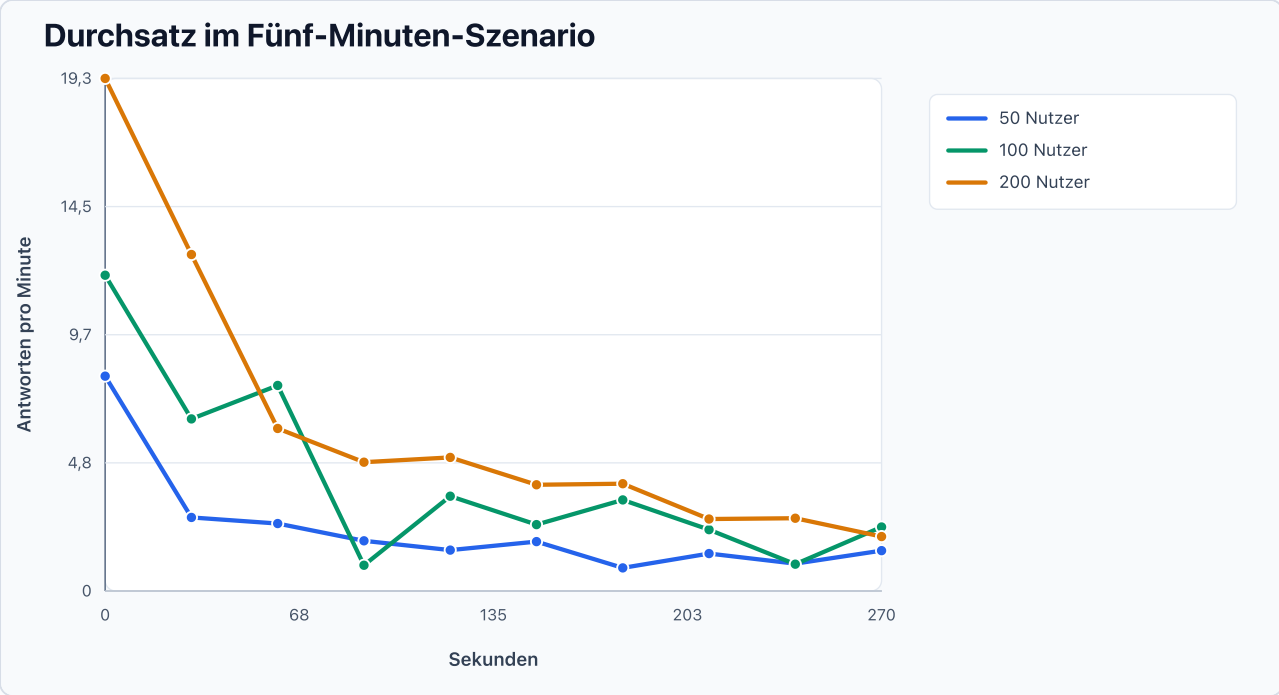
Die Startgrafik zeigt die zufallsverteilten Anfragen pro 30-Sekunden-Fenster. Die 200-Nutzer-Linie liegt erwartbar höher und erzeugt deutlich dichtere Last.

So liest man das Diagramm: Die waagerechte Achse zeigt die fünf Minuten in Zeitfenstern, die senkrechte Achse die neu gestarteten Anfragen je Fenster. Einzelne Spitzen sind bei Zufallsverteilung normal. Entscheidend ist, wie dicht mehrere Spitzen aufeinander folgen, weil genau dann die Warteschlange schnell wächst.



Die Latenzgrafik macht die Grenze sichtbar: Bei 50 und 100 Nutzern bleibt die Median-Latenz vergleichsweise stabil, bei 200 Nutzern steigt sie im Verlauf deutlich.

So liest man das Diagramm: Die waagerechte Achse zeigt den Verlauf des Messzeitraums, die senkrechte Achse die typische Gesamtdauer abgeschlossener Antworten. Eine stabile Linie bedeutet, dass die Spark ungefähr so schnell abarbeitet, wie neue Arbeit hinzukommt. Eine steigende Linie zeigt, dass die Queue wächst und spätere Anfragen länger warten müssen.



Die Durchsatzgrafik zeigt, wie viele Antworten pro Minute abgeschlossen werden. Dichtere Last erhöht den Durchsatz nicht unbegrenzt; ab der Kapazitätsgrenze wächst vor allem die Warteschlange.

So liest man das Diagramm: Die waagerechte Achse zeigt die Zeit, die senkrechte Achse die abgeschlossenen Antworten pro Minute. Die Kurve darf der Startgrafik nicht eins zu eins entsprechen: Starts erzeugen Arbeit, Durchsatz zeigt erledigte Arbeit. Wenn viele Starts auftreten, der Durchsatz aber nicht entsprechend steigt, wird die überschüssige Arbeit in der Queue gespeichert.

7.2.7 Warum die Warteschlange kein Fehler ist

Eine Warteschlange klingt zunächst negativ. Für ein Schulsystem ist geordnetes Abarbeiten aber besser als aggressive Fehler. Ohne Queue müsste ein überlasteter Inferenzdienst Anfragen ablehnen, Verbindungen abbrechen oder unzuverlässig reagieren. Mit Queue bleiben Anfragen erhalten und die Spark kann ihre Rechenkapazität kontinuierlich nutzen.

Die Kehrseite ist das Nutzergefühl: Wenn eine Anfrage angenommen wurde, aber lange kein erster Antwortteil erscheint, wirkt die Oberfläche schnell unklar. Deshalb sollte Ephraim bei hoher Last verständlich signalisieren, dass die Anfrage angenommen wurde und in Arbeit ist. Für Admins sind First-Token-Zeit, aktive Requests und künftig Queue-Signale die wichtigsten Frühwarnwerte.

> **Praktische Marke:** 100 zufallsverteilte Instant-Anfragen in fünf Minuten sind nach diesem Lauf ein guter Orientierungsbereich. 200 Anfragen in fünf Minuten zeigen: stabil, aber warteschlangenartig.

7.2.8 Instant und Thinking einordnen

Thinking wurde nicht als vollständiger Lasttest zu Ende geführt. Das ist fachlich vertretbar, weil der Kontrolllauf bereits die entscheidende Beobachtung zeigte: Der Start der Antwort ist nicht das Problem. Die erste sichtbare Ausgabe kam fast genauso schnell wie bei Instant. Länger wird vor allem die Zeit bis zur vollständigen Antwort, weil Thinking mehr Rechenarbeit pro Anfrage investiert.

Laststufe	Modus	Erfolgreich	p50 Latenz	p50 erster Token	Durchsatz
1 gleichzeitiger Nutzer	Instant	3/3	6,8 s	0,2 s	20,0/min
1 gleichzeitiger Nutzer	Thinking	3/3	9,8 s	0,2 s	18,3/min
2 gleichzeitige Nutzer	Instant	6/6	10,3 s	0,3 s	31,4/min
2 gleichzeitige Nutzer	Thinking	6/6	13,8 s	0,3 s	25,1/min

Für Admins ist diese Unterscheidung wichtig: Thinking ist nicht „kaputt langsam“, sondern pro Anfrage rechenintensiver. Es ist deshalb plausibel, dass ein vollständiger Thinking-Lasttest bei hohen Nutzerzahlen vor allem längere Warteschlangen zeigen würde. Für den normalen Kapazitätsvergleich reicht Instant als Hauptmessung, solange Thinking bewusst als Qualitätsmodus mit längerer Fertigstellung kommuniziert wird.

> **Betriebslesart:** Instant beantwortet die Frage „Wie viel gleichzeitige Schullast trägt die Spark?“. Thinking beantwortet eher die Frage „Wie viel zusätzliche Denkzeit ist uns für anspruchsvollere Antworten wert?“.

7.2.9 Aktives Modell und Fähigkeitsgrenzen

Die Benchmarks gelten immer für das aktive Inferenzprofil. In Produktion ist das stabile Profil **gpt-oss** aktiv; es stellt GPT-OSS-120B über den lokalen OpenAI-kompatiblen Inferenzdienst bereit. Das Frontend spricht diesen Dienst über den stabilen Modellnamen `school-ui` an.

Der dokumentierte V1-Betrieb ist bewusst auf dieses eine lokale Textmodell begrenzt. Externe Modellanbieter und externe Modellproxys gehören nicht zur freigegebenen Produktionskonfiguration. Abweichungen vom dokumentierten GPT-OSS-120B-Pfad wären eine wesentliche Änderung und müssten fachlich, technisch und datenschutzseitig neu dokumentiert und freigegeben werden.

Admin-Schritt	Wofür er gedacht ist
`plans/spark-model-switch.sh status`	Zeigt den aktiven lokalen Inferenzdienst und relevante Inferenzcontainer.

Der produktive Modellpfad verarbeitet Text. Bilddateien können gespeichert, angezeigt, geprüft und über Metadaten in den Kontext eingeordnet werden; der aktuelle produktive Modellpfad analysiert ihren visuellen Inhalt nicht.

> **Aktueller Betriebsstand:** Version 1 nutzt GPT-OSS-120B als einziges dokumentiertes lokales Sprachmodell. Vision-Fähigkeiten sind aktuell nicht freigegeben.

Für Admins folgt daraus: Benchmarks, Statusseite, Logs, Antwortqualität, First-Token-Zeiten und Gesamtlatenz beziehen sich auf genau diesen V1-Modellpfad. Wenn mehr gleichzeitige Nutzung getragen werden soll, ist eine zweite Spark als replizierter zusätzlicher Inferenzknoten die naheliegende Erweiterung.

7.2.10 Was eine zweite Spark bedeuten würde

Die Benchmarks zeigen vor allem Warteschlangen bei dichter Last. Daraus folgt für die Skalierung: Wenn das Ziel mehr gleichzeitige Nutzerinnen und Nutzer ist, ist ein zweiter unabhängiger Inferenzknoten mit Lastverteilung der naheliegende erste Weg. Neue Anfragen würden dann auf Spark A oder Spark B verteilt; jede Spark hätte ihre eigene Queue und eigene Inferenzkapazität.

Für den gemessenen 200-Nutzer-Fall ist diese Denkweise plausibel: Statt 200 Anfragen auf eine Queue zu legen, würden grob 100 Anfragen pro Spark landen. Da 100 zufallsverteilte Anfragen auf einer Spark deutlich besser aussahen als 200, wäre eine spürbare Verkürzung der Warteschlange zu erwarten. Das ist keine Garantie für exakt doppelte Leistung, aber der direkteste Skalierungsansatz für unabhängige Schulfragen.

Der dokumentierte Ausbaupfad bleibt dabei derselbe Modellpfad auf zusätzlicher lokaler Kapazität. Für viele kleine unabhängige Schulfragen ist Replikation mit Lastverteilung verständlich und betrieblich robust.

NVLink ist dabei nicht einfach „mehr normales RAM“. Es kann GPUs mit hoher Bandbreite koppeln und dadurch verteilte Modell- oder Tensorparallelität ermöglichen, wenn der Inferenz-Stack das sauber unterstützt. Praktisch kann dadurch mehr nutzbarer GPU-Speicher für ein größeres Modell, längere

Kontexte oder größere KV-Caches entstehen. Es ist aber kein automatischer Schalter für doppelte Leistung: Wenn eine Anfrage beide GPUs gemeinsam nutzt, konkurriert sie nicht mehr mit einer zweiten unabhängigen Anfrage auf einer zweiten separaten Spark.

Für den hier gemessenen Schultyp mit vielen unabhängigen kurzen bis mittleren Anfragen wäre deshalb zuerst ein replizierter Betrieb mit Lastverteilung zu prüfen. Eine gekoppelte NVLink-Variante lohnt sich eher dann, wenn ein größeres Modell, sehr lange Kontexte oder spezielle Batch-Strategien im Vordergrund stehen. Beides kann sinnvoll sein, beantwortet aber unterschiedliche Fragen.

Ziel	Naheliegender Ansatz	Betriebliche Lesart
Mehr gleichzeitige Nutzer	Zwei replizierte Inferenzknoten mit Lastverteilung	Kürzere Queues, bessere Verteilung unabhängiger Anfragen.
Größere Modelle oder längere Kontexte	Gekoppelte verteilte Inferenz	Mehr Komplexität; Nutzen hängt stark von Modell, Framework und Parallelisierung ab.
Zwei Spark per NVLink gekoppelt	Gemeinsame Nutzung mehrerer GPUs, falls Software und Modell es unterstützen	Eher mehr nutzbarer GPU-Speicher und Modellspielraum; nicht automatisch doppelte Nutzerkapazität.

7.2.11 Grenzen der Benchmark-Aussage

Die Benchmarks messen keine Antwortqualität. Sie prüfen nicht, ob eine Erklärung fachlich korrekt, altersgerecht oder didaktisch hilfreich ist. Sie messen auch nicht, wie schnell der Browser rendert, ob ein Chatverlauf übersichtlich bleibt oder ob Projektmaterial passend ausgewählt wurde.

Ebenfalls nicht gemessen werden produktive Mischlasten wie Datei-Uploads, Basiswissen- Refresh, Adminaktionen, Vorlesen, Projektfazit und parallele Nutzerinteraktionen im Webfrontend. Die Zahlen sind Betriebsindikatoren, keine allgemeine Kapazitätsgarantie.

> **Nicht überinterpretieren:** Ein erfolgreicher Benchmarklauf zeigt einen konkreten Lauf mit konkretem Modus, Modellstand, Promptauswahl und Zielumgebung. Nach relevanten Konfigurationsänderungen müssen Benchmarks wiederholt werden.

7.2.12 Datenschutz und Artefakte

Die Benchmarks verwenden synthetische Prompts. In den Veröffentlichungsartefakten werden technische Messwerte wie Latenz, Erfolgsstatus, Prompt-ID, Modus und Durchsatz gespeichert. Echte Chatinhalte, echte Nutzerfragen, Session-Cookies, API-Schlüssel, IP-Adressen, Projekt-IDs und vollständige Modellantworten gehören nicht in die veröffentlichten Auswertungen.

Rohmesswerte bleiben technische Betriebsartefakte. Für Handbücher, Präsentationen und externe Einordnung sind aggregierte Tabellen, Zusammenfassungen und Diagramme gedacht. Auch diese sollten nur mit Kontext weitergegeben werden: Was wurde gemessen, welcher Modus lief, und welche Fragen beantwortet der Benchmark ausdrücklich nicht?

7.2.13 Empfehlungen für den Betrieb

1. Thinking als Qualitätsmodus mit längerer Fertigstellung einplanen; kurze Kontrollmessungen reichen, solange Instant die Kapazitätskurve liefert.
2. In der Oberfläche lange Wartephasen klar als „angenommen und in Arbeit“ kommunizieren.
3. Für reale Nutzung einen Richtwert formulieren, etwa 100 zufallsverteilte Instant-Anfragen in fünf Minuten als gut tragbaren Bereich.
4. Bei hoher Last Queue-Länge, aktive Requests und geschätzte Wartezeit als Admin-Signale priorisieren.
5. Nach relevanten Konfigurationsänderungen zuerst Smoke-Tests und Capability-Prüfung durchführen; danach Benchmarks, Antwortqualität und Latenzen neu bewerten.
6. Bei steigender Nutzung zuerst Lastverteilung auf replizierte Inferenzknoten prüfen; verteilte Inferenz für größere Modelle separat bewerten.

Dieser Artikel erklärt die ausgewerteten Instant-Inferenz-Benchmarks und den Thinking-Kontrolllauf für Administratoren. Für den laufenden Betrieb bleiben Status, Logging, Supervisor-Webterminal und Cron die angrenzenden Diagnoseartikel.

Quelle: web/manuals/admin-inferenz-benchmarks/index.html

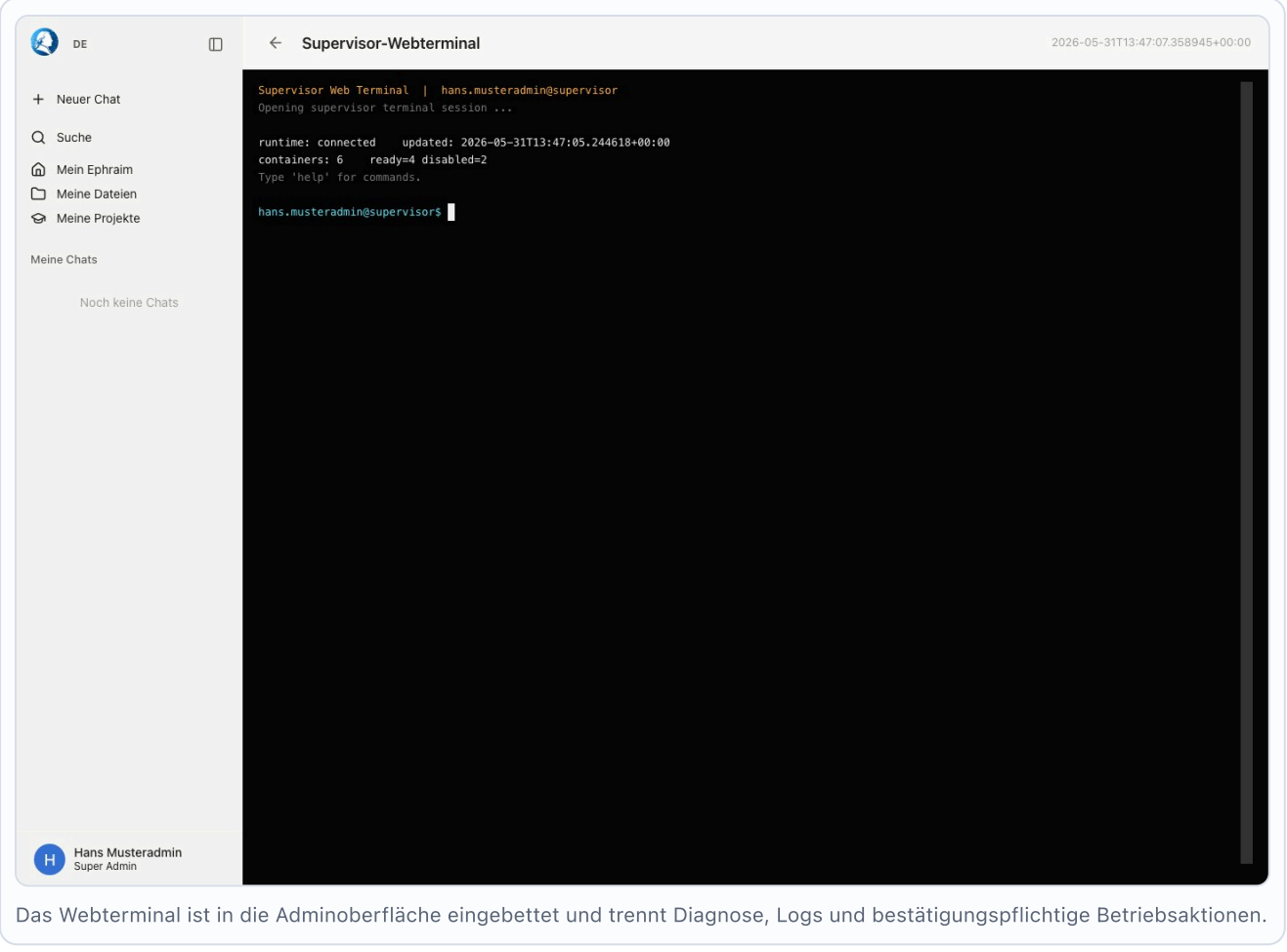
7.3 Supervisor-Webterminal

> Das Supervisor-Webterminal verbindet die Ephraim-Administration mit den Betriebsdiensten der Spark. Es zeigt Status, Logs und Runtime-Daten und stellt ausgewählte, abgesicherte Administrationsaktionen bereit.

Stand: Mai 2026

7.3.1 Rolle des Webterminals

Das Webterminal ist kein allgemeiner Shell-Zugang für beliebige Kommandos. Es ist eine administrierte Konsole mit vordefinierten Betriebsfunktionen. Administratoren können Dienste prüfen, Logs lesen, Containerstatus sehen und ausgewählte Aktionen auslösen, ohne direkt auf die Spark per Systemterminal wechseln zu müssen.



Das Webterminal ist in die Adminoberfläche eingebettet und trennt Diagnose, Logs und bestätigungspflichtige Betriebsaktionen.

7.3.2 Authentifizierung zur Runtime

Der Zugriff auf Runtime und Supervisor wird mit kurzlebigen signierten Tokens abgesichert. Ephraim erzeugt diese Tokens serverseitig für berechtigte Admins. Die Runtime prüft Signatur, Aussteller, Ziel und Ablaufzeit. Dadurch reicht eine normale Browsersitzung allein nicht aus, um dauerhaft Runtimezugriff zu erhalten.

> **Sicherheitsgrenze:** Runtime-Tokens sind kurzlebig und zweckgebunden. Sie sind nicht als dauerhaftes Passwort für die Spark gedacht.

7.3.3 Verfügbare Funktionen

Bereich	Typische Nutzung
Status	Runtime, Supervisor, Inference, TTS und Containerzustände prüfen.
Logs	Fehler und Betriebsereignisse lesen, ohne private Chatinhalte zu dumpen.
Statistiken	Ressourcen, Auslastung und Dienstzustände für Diagnose einordnen.
Aktionen	Gezielte Betriebsaktionen auslösen, wenn Berechtigung und Bestätigung passen.

7.3.4 Super-Admin-Aktionen

Einige Aktionen sind Super-Admins vorbehalten, etwa besonders tiefgreifende Updates, Audits oder Benutzeränderungen mit Systemwirkung. Solche Aktionen sind nicht nur durch Rolle, sondern auch durch Bestätigungsmechanismen und serverseitige Prüfungen geschützt. Das verhindert, dass ein versehentlicher Klick oder ein alter Tab unmittelbar kritische Betriebszustände verändert.

7.3.5 Umgang mit Logs

Logs sind Betriebsdaten, keine Datensammlung für Unterrichtsinhalte. Bei Fehlersuche suchen Administratoren nach Statuscodes, Dienstnamen, Job-IDs und Fehlerzeitpunkten. Private Prompts oder Dateien dürfen nicht als Debug-Ausgabe in Logs kopiert werden. Wenn ein Log unerwartet sensible Daten enthält, ist das ein Sicherheitsproblem.

Die Grundregeln für Production-Logging, Webservergrenzen, Spark-Logs und Admin-Problem-Mails sind im Artikel Logging zusammengefasst.

7.3.6 Betriebspraxis

Das Webterminal ist besonders hilfreich, wenn die Statusseite ein Problem zeigt, aber der betroffene Dienst noch nicht feststeht. Die Reihenfolge lautet: Status prüfen, betroffene Komponente identifizieren, Logs lesen, aktuelle Jobs oder Container prüfen, dann gezielt handeln.

Das Webterminal bündelt mächtige Funktionen. Der Betrieb nutzt dafür 2FA-geschützte Admin-Konten.

Quelle: web/manuals/admin-supervisor-webterminal/index.html

7.4 Cron

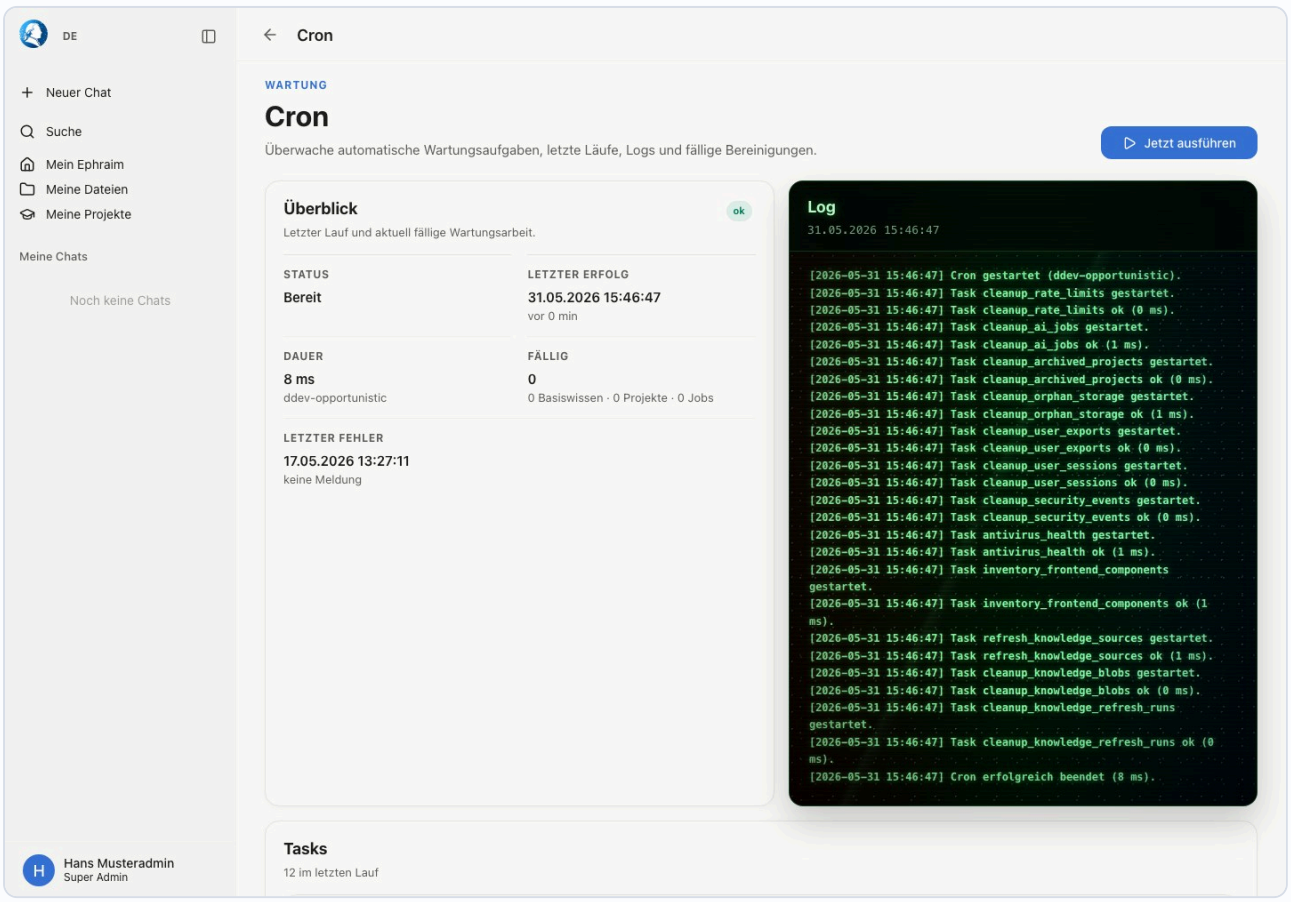
> **Cron hält Ephraim im Hintergrund sauber: Aufräumen, Sicherheitsprüfungen, Knowledge-Refresh, Jobpflege, Projektabläufe und technische Inventuren laufen nicht im normalen Seitenaufruf, sondern als geplante Wartungsaufgaben.**

Stand: Mai 2026

7.4.1 Warum Cron nötig ist

Cron ist die Wartungsschicht von Ephraim. Viele Aufgaben müssen regelmäßig laufen, dürfen aber normale Seitenaufrufe, Chats und Projektarbeit nicht ausbremsen: abgelaufene Jobs bereinigen, alte Sicherheitsdaten kürzen, archivierte Projekte endgültig löschen, zentrale Wissensquellen aktualisieren und Drittanbieter-Komponenten inventarisieren. Genau diese Arbeit bündelt Ephraim im Cron-Lauf.

Die Adminseite zeigt den letzten Lauf, offene Wartung, Logauszüge, Einzelergebnisse pro Task und die wichtigsten Umgebungsdaten. Berechtigte Admins können zusätzlich einen manuellen Lauf starten. Das ist ein Diagnose- und Wartungswerkzeug; es ersetzt im Produktivbetrieb keinen echten systemweiten Zeitplan.



Die Cronseite zeigt geplante Wartungsaufgaben, letzte Ergebnisse und manuelle Startmöglichkeiten für berechtigte Administratoren.

7.4.2 Die Adminseite richtig lesen

Oben steht der zusammengefasste Betriebszustand. **Bereit** bedeutet: Der letzte Lauf wurde ohne gemeldeten Taskfehler beendet und ist nicht veraltet. **Läuft gerade** bedeutet: Ein Cron-Lauf hat begonnen und noch kein Endergebnis geschrieben. **Fehler** bedeutet: Mindestens eine Aufgabe hat ein Fehlerergebnis geliefert. Ein veralteter erfolgreicher Lauf wird zurückhaltend markiert, weil er technisch zuletzt erfolgreich war, aber zu lange zurückliegt.

Bereich	Was Admins daraus lesen
Überblick	Status, letzter Erfolg, Dauer, Ausführungsart und fällige Wartung für Knowledge, Projekte und KI-Jobs.
Log	Gekürztes, adminlesbares Protokoll des letzten Laufs mit Start, Tasknamen, Erfolg oder Fehlermeldung.
Tasks	Ein Ergebnis pro Wartungsaufgabe: Status, Dauer und fachliche Kennzahlen wie gelöschte Einträge oder freigegebene Bytes.
Umgebung	Start-/Endzeit, HTTP-Tokenstatus, erlaubte IPs, DDEV-Erkennung und opportunistisches DDEV-Intervall.
Refreshes	Letzte Knowledge-Refresh-Läufe mit Quelle, Status, Startzeit, Trigger und erzeugten Chunks oder Ereignissen.

> **Wichtig:** Ein grüner Cron-Status sagt, dass die Wartungsaufgaben technisch erfolgreich liefen. Er sagt nicht, dass jede fachliche Quelle inhaltlich korrekt ist oder dass ein externes System immer aktuelle Daten geliefert hat.

7.4.3 Konkrete Aufgaben

Ephraim führt die Aufgaben nacheinander aus. Jede Aufgabe liefert ein eigenes `ok`-Flag, eine Dauer und zusätzliche Kennzahlen. Wenn eine Aufgabe fehlschlägt, laufen die übrigen Aufgaben weiter; am Ende wird der Gesamtlauf als Fehler markiert.

Aufgabe	Was konkret passiert	Warum das wichtig ist
Rate-Limits bereinigen	Einträge in `auth_rate_limits`, deren Zeitfenster älter als 24 Stunden ist, werden gelöscht.	Login- und Schutzmechanismen bleiben wirksam, ohne alte Zähler dauerhaft mitzuschleppen.
KI-Jobs pflegen	Abgelaufene Snapshots und Chunks werden gelöscht; offene abgelaufene Jobs werden auf Fehler gesetzt; alte terminale Jobs und hängen gebliebene Artefakte werden bereinigt.	Streaming, Wiederaufnahme und Artefakterzeugung hinterlassen keine dauerhaften technischen Altlasten.
Archivierte Projekte löschen	Projekte, die archiviert sind und deren Löschdatum erreicht ist, werden endgültig entfernt.	Die Retention-Regel für Projekte wird technisch durchgesetzt.
Verwaiste Dateien entfernen	Dateien im Nutzer- und Projektspeicher werden gegen die Datenbankreferenzen geprüft; nicht mehr referenzierte Binärdateien werden gelöscht.	Der Dateispeicher wächst nicht durch gelöschte oder verschobene Metadaten weiter.
Nutzerexporte aufräumen	Temporäre Exportdateien im System-Temp-Verzeichnis werden nach Ablauf entfernt.	Datenexporte bleiben kurzlebige Arbeitsdateien und werden nicht dauerhaft im Temp-Bereich vergessen.
Nutzersitzungen bereinigen	Abgelaufene Einträge aus der Sitzungsübersicht werden gelöscht.	Die Anzeige aktiver Sitzungen bleibt aussagekräftig und sammelt keine unnötigen Altzustände.
Sicherheitsereignisse kürzen	Alte Datei-Sicherheitsereignisse werden nach der vorgesehenen Aufbewahrungsfrist entfernt.	Uploadschutz bleibt nachvollziehbar, ohne dauerhaft alle alten Ereignisse aufzubewahren.
Antivirus prüfen	Wenn ClamAV aktiviert ist, prüft Ephraim die Verfügbarkeit des Scanners.	Admins erkennen, ob der Uploadschutz technisch erreichbar ist.
Drittanbieter inventarisieren	Lokale Komponenten werden erkannt; erlaubte externe Hinweise werden verarbeitet; Mismatch-Badges und Super-Admin-Hinweise werden aktualisiert.	Komponentenpflege bleibt kontrolliert, ohne Seitenaufrufe mit externen Prüfungen zu blockieren.
Basiswissen aktualisieren	Fällige globale Remote-Quellen und serverseitige Wetterquellen werden verarbeitet und für Retrieval vorbereitet; Embedding-Vektoren werden lokal berechnet.	Freigegebenes schulisches Wissen bleibt nutzbar, ohne dass die Spark selbst im Internet sucht oder Embedding-Inhalte dauerhaft speichert.
Knowledge-Blobs aufräumen	Verschlüsselte Basiswissen-Dateien im Knowledge-Speicher werden gegen die Datenbank abgeglichen.	Verwaiste Dateien aus zentralen Wissensquellen werden entfernt.
Refresh-Diagnosen kürzen	Alte Einträge aus den Knowledge-Refresh-Protokollen werden entfernt.	Die Diagnose bleibt übersichtlich und speichert nicht unbegrenzt alte Refreshdetails.

7.4.4 Ausführungsarten

Im Produktivbetrieb braucht Ephraim einen echten Zeitgeber: Crontab, systemd-Timer, Container-Cron oder einen gleichwertigen Betriebsmechanismus. Der bevorzugte Weg ist ein CLI-Aufruf von `web/cron.php`, weil dafür kein HTTP-Token notwendig ist und der Lauf nicht vom öffentlichen Webzugriff abhängt.

Der HTTP-Aufruf ist zusätzlich möglich, aber nur mit konfiguriertem `CRON_SECRET_TOKEN` und erlaubter Client-IP. Das Token kann als Header `X-Cron-Token` oder als Query-Parameter übergeben werden. Fehlt das Token, antwortet der Endpunkt nicht mit einem Default, sondern mit einem Konfigurationsfehler. Eine falsche IP wird abgewiesen.

Ausführungsart	Anzeige in Ephraim	Einordnung
CLI	`cli`	Empfohlener Produktionsweg über Server- oder Container-Zeitplan.
HTTP	`http`	Nur mit Token und IP-Whitelist verwenden; geeignet für bewusst abgesicherte externe Auslöser.
Adminseite	`admin`	Manueller Lauf aus dem Adminbereich mit Login und CSRF-Schutz.
Admin-Terminal	`admin-terminal`	Manueller Lauf über das Supervisor-Webterminal und die Admin-Terminal-API.
DDEV-Entwicklung	`ddev-opportunistic`	Lokale Entwicklungshilfe: normale HTML-GET-Aufrufe können nach der Antwort einen fälligen Lauf anstoßen.

> **Produktionsregel:** Die DDEV-Automatik ist eine Entwicklungshilfe. Für den realen Schulbetrieb wird Cron explizit als Betriebsaufgabe eingerichtet und überwacht.

7.4.5 Was Ephraim speichert

Cron schreibt keine lange Rohhistorie in die Manualoberfläche. Die Adminseite liest Systemsettings mit dem aktuellen Monitorzustand. Dadurch sieht man schnell, ob Cron läuft, wann der letzte Erfolg war und welche Tasks im letzten Lauf auffällig waren.

Feld	Bedeutung
`cron_last_status`	`running`, `ok` oder `error`.
Start, Ende, Erfolg, Fehler	Zeitpunkte für letzten Start, letztes Ende, letzten erfolgreichen und letzten fehlerhaften Lauf.
Dauer und Ausführungsart	Laufzeit in Millisekunden und Quelle des Laufs, etwa `cli`, `http` oder `admin`.
`cron_last_summary`	JSON-Ergebnis aller Tasks mit Status, Dauer und Kennzahlen.
`cron_last_log`	Gekürztes Log des letzten Laufs, begrenzt für die Adminanzeige.

Ein erfolgreicher Lauf gilt im Panel als veraltet, wenn der letzte Erfolg älter als `CRON_MONITOR_MAX_AGE_MINUTES` ist. Der Standardwert beträgt 15 Minuten. Das ist kein Datenverlust, sondern ein Überwachungssignal: Cron hat zu lange keinen frischen Erfolg geschrieben.

7.4.6 Manuelle Läufe

Manuelle Cronläufe sind nützlich nach Konfigurationsänderungen, beim Testen neuer Quellen oder nach Störungen. Sie ersetzen keinen sauber eingerichteten Zeitplan. Beim manuellen Start über die Adminseite gelten Login, Adminrolle und CSRF-Schutz. Im Supervisor-Terminal stehen dafür die Cron-Befehle bereit; technisch ruft das Terminal dieselbe `CronRunner`-Logik auf.

Ein manueller Lauf startet alle Wartungsaufgaben, nicht nur eine einzelne Zeile aus der Tabelle. Deshalb sollte man ihn nicht „zum Spaß“ mehrfach hintereinander auslösen. Wenn nur eine Drittanbieter-Prüfung gewünscht ist, wird diese Prüfung in der Komponentenverwaltung angefordert; Cron verarbeitet sie dann im nächsten Lauf.

7.4.7 Fehlerverhalten

Cron ist fehlertolerant aufgebaut: Eine fehlgeschlagene Task stoppt nicht den gesamten Lauf. Ephraim protokolliert die fehlgeschlagene Aufgabe, meldet den Fehler im Gesamtergebnis und führt die übrigen Tasks weiter aus. Dadurch bleibt zum Beispiel eine Speicherbereinigung möglich, auch wenn eine Knowledge-Quelle gerade nicht erreichbar ist.

Beobachtung	Konkrete Bedeutung	Nächster Schritt
Status `running` bleibt stehen	Der letzte Lauf hat begonnen, aber kein Ende geschrieben oder läuft noch.	Log, Serverprozess und Zeitpunkt prüfen; danach erneut kontrollieren, ob ein neuer Lauf den Status überschrieben hat.
Gesamtstatus `error`	Mindestens eine Task hat `ok=false` geliefert.	Tasktabelle und Log lesen; zuerst die konkrete Task, nicht den ganzen Cron verdächtigen.
Letzter Erfolg ist veraltet	Cron lief zu lange nicht erfolgreich.	Zeitplan, Container, CLI-Pfad oder HTTP-Token/IP prüfen.
HTTP-Aufruf liefert 401, 403 oder 503	Token fehlt/falsch, IP nicht erlaubt oder Token nicht konfiguriert.	`CRON_SECRET_TOKEN` und `CRON_ALLOWED_IPS` in der Betriebsumgebung prüfen.
Knowledge-Refresh schlägt fehl	Eine freigegebene Quelle konnte nicht aktualisiert oder verarbeitet werden.	Refresh-Tabelle lesen und danach den Artikel Admin: Basiswissen prüfen.

7.4.8 Datenschutz und Schlüsselgrenzen

Cron läuft ohne aktive Nutzersitzung. Deshalb besitzt Cron keinen User-DEK und öffnet keine persönlichen Vaults. Persönliche Dateien, private RAG-Inhalte, private Kalender, Erinnerungen und persönliche Chat-Zusammenfassungen werden nicht im Hintergrund massenhaft entschlüsselt oder neu erzeugt.

Zentrales Basiswissen ist anders eingeordnet: Es ist freigegebener schulweiter Kontext und wird serverseitig verschlüsselt verwaltet. Diese Quellen darf Cron kontrolliert aktualisieren. Auch Wetterabrufe laufen serverseitig, damit keine Nutzer-IP direkt an den Wetterdienst geht. Die Spark erhält daraus später nur ausgewählten Kontext für eine konkrete Anfrage; sie erhält durch Cron keinen freien Internetzugang.

Die Schlüsselgrenzen sind im Artikel Kryptoarchitektur ausführlich erklärt. Für Datenschutzfragen zu Basiswissen und Wikipedia siehe Basiswissen und Wikipedia.

7.4.9 Sicherheit des Cron-Zugangs

Der öffentliche Cron-Endpunkt ist kein normaler Adminbereich. Er ist bewusst schlicht, damit ein Serverzeitplan ihn aufrufen kann. Deshalb hat er eigene Schutzregeln: kein Defaulttoken, timing-sicherer Tokenvergleich und IP-Whitelist. Der sicherste Weg bleibt der lokale CLI-Aufruf, weil dabei kein HTTP-Token über das Netz verwendet wird.

Die Adminseite und das Admin-Terminal verwenden dagegen die normale Anmeldung, Rollenprüfung und CSRF-Schutz. Normale Nutzer sehen diese Funktionen nicht. Fehlerdetails werden in der Adminanzeige zusammengefasst, damit interne Exceptiondetails nicht an ungeeignete Stellen gelangen.

7.4.10 Praktische Diagnose

1. Im Adminbereich zuerst Status, letzten Erfolg und Ausführungsart prüfen.
2. Wenn der letzte Erfolg veraltet ist, den echten Zeitgeber prüfen: crontab, systemd-Timer, Container-Cron oder HTTP-Aufrufer.
3. Bei Fehlerstatus die Tasktabelle lesen und nur die fehlgeschlagene Aufgabe untersuchen.
4. Bei Knowledge-Problemen Refresh-Tabelle und Admin-Basiswissen prüfen.
5. Bei Komponentenproblemen den Komponentenartikel lesen und kontrollieren, ob eine manuelle Prüfung nur angefordert, aber noch nicht verarbeitet wurde.
6. Bei Uploadschutzproblemen ClamAV-Konfiguration und Datei-Sicherheitsseite prüfen.
7. Nach einer Korrektur einen manuellen Lauf starten und kontrollieren, ob Status, Log und Taskergebnis plausibel sind.

Cron ist kein Reparaturknopf für jede Störung. Er zeigt aber oft den besten Einstieg: Welche Wartung läuft, welche zuletzt fehlgeschlagen ist und ob Ephraim gerade technische Altlasten aufräumen kann.

Cron ist Betriebshygiene. Wenn Cron dauerhaft ausfällt, wachsen Folgeprobleme oft schleichend. Verwandte Adminbereiche sind Status, Logging, Basiswissen, Datei-Sicherheit und Drittanbieter-Komponenten.

Quelle: web/manuals/admin-cron/index.html

7.5 Datenbank

> **Wie Ephraim in Produktion MySQL als dauerhafte Datenbank nutzt und Redis/Valkey nur als maschineninternen Speed-Layer für laufende Streams und Worker-Wakeups einsetzt.**

Stand: Juni 2026

7.5.1 Kurzfassung

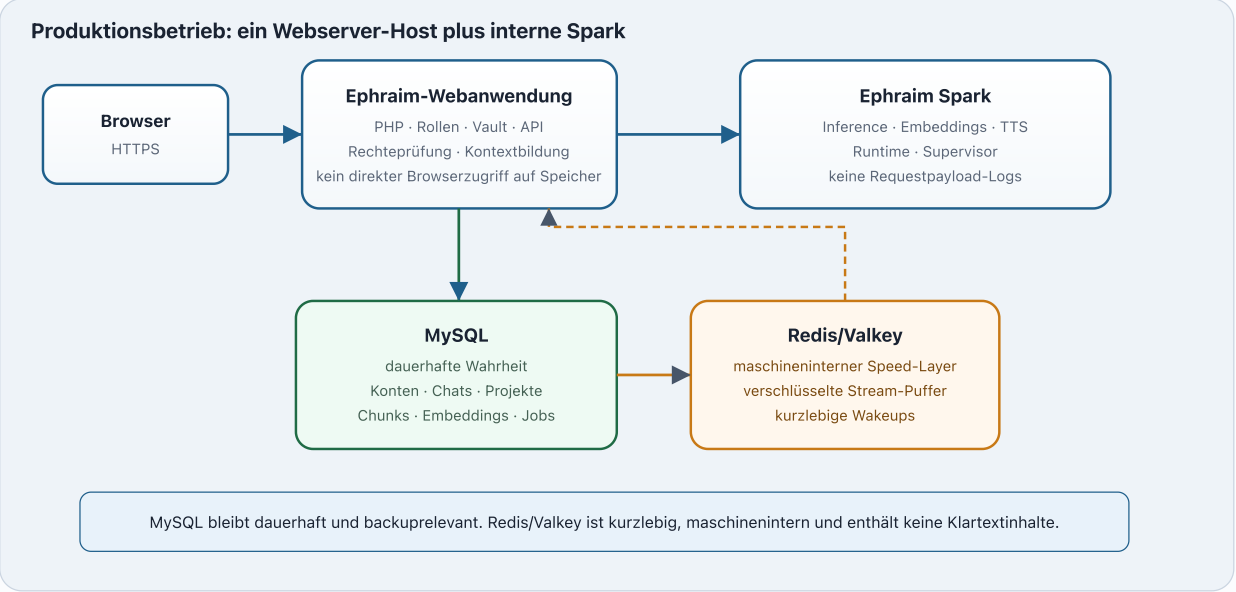
In Produktion ist **MySQL** die zentrale Datenbank von Ephraim. MySQL ist die dauerhafte Wahrheit: Konten, Rollen, Projekte, verschlüsselte Chats, Jobzustände, Knowledge-Metadaten, verschlüsselte Chunks, verschlüsselte Embeddings, Einstellungen und Auditdaten werden dort verwaltet.

Redis/Valkey ist davon getrennt. Es läuft auf demselben Host wie die Webanwendung und ist nur über ein maschineninternes Server-/Containernetz erreichbar. Ephraim verwendet Redis/Valkey nicht als zweite Nutzerdatenbank, nicht als Session-Speicher, nicht als Vault-Speicher und nicht als RAG-

Vektordatenbank.

> **Merksatz:** MySQL hält den Zustand. Redis/Valkey beschleunigt laufende Arbeit. Wenn Redis/Valkey ausfällt, darf das nicht bedeuten, dass private Daten verloren sind oder Klartextdaten offengelegt werden.

7.5.2 Zusammenspiel der Komponenten



MySQL und Redis/Valkey gehören beide zum Webserver-Betrieb, aber nicht zur gleichen Vertrauens- und Persistenzebene. Die Spark erzeugt Antworten, Vektoren und Audio; sie ist nicht die Datenbank.

7.5.3 MySQL: dauerhafte Wahrheit

MySQL ist der Zustandsspeicher von Ephraim. Wenn ein Konto angelegt wird, ein Chat gespeichert wird, ein Projekt entsteht, eine Datei Metadaten bekommt oder ein Knowledge-Chunk indexiert wird, ist MySQL der verbindliche Ort für diese Information. Viele Inhalte liegen dort nicht als Klartext, sondern verschlüsselt oder als Hash, HMAC, Fingerprint, Status oder Metadatum.

Datenbereich	Typische Speicherung in MySQL	Grenze
Konten und Rollen	Benutzername, Rolle, Klasse, Passwort-Hash, 2FA-Status, Session-Version	Kein Klartextpasswort; Tokens werden nicht roh gespeichert.
Chats und Jobs	verschlüsselte Nachrichten, Jobstatus, verschlüsselte Payloads, Stream-Fallbacks	Private Inhalte bleiben an Vault oder Scope gebunden.
Dateien	Metadaten, Status, verschlüsselter Anzeigename, Speicherreferenz, Hashes	Dateiinhalt liegt verschlüsselt im Storage, nicht öffentlich im Webroot.
RAG und Basiswissen	verschlüsselte Text-Chunks, verschlüsselte Embedding-Vektoren, Quellstatus	Retrieval, Rechteprüfung und Scoreberechnung laufen in der Webanwendung.
Admin und Betrieb	Systemeinstellungen, Cronstatus, Komponentenfreigaben, Auditmetadaten	Adminrechte öffnen keine fremden Vault-Klartexte.

7.5.4 Redis/Valkey: kurzlebiger Speed-Layer

Redis/Valkey wird nur dort gebraucht, wo Latenz zählt: bei laufenden Antwortstreams und beim Wecken von Workern. Ein Stream ist eine Antwort, die Stück für Stück beim Browser ankommt. Redis/Valkey kann diese laufenden Stücke schneller bereitstellen, als wenn jeder kurze Zwischenstand ausschließlich über MySQL gelesen würde.

Datenschutzentscheidend ist, was dort nicht liegt. Redis/Valkey speichert keine Klartextchats, keine Dateien, keine Vault-Schlüssel, keine Passwörter, keine Nutzerprofile, keine Sessions, keine RAG-Klartexte und keine Embedding-Vektoren. Stream-Chunks und Reconnect-Snapshots sind verschlüsselt. Wakeup-Signale enthalten nur technische Angaben wie Job-ID, Job-Typ und Zeitpunkt.

Redis/Valkey-Inhalt	Lebensdauer	Einordnung
Verschlüsselte Stream-Chunks	kurzlebig während laufender Antwortstreams	Reconnect und flüssige Ausgabe; kein lesbarer Chattext.
Verschlüsselte Reconnect-Snapshots	kurzlebig	Hilft nach Reload oder Verbindungsunterbrechung.
Worker-Wakeup-Signale	typischerweise bis zu einer Stunde	Technische Metadaten, damit Worker nicht träge pollen müssen.

7.5.5 Ausfall und Fallback

Redis/Valkey ist nicht die Quelle der Wahrheit. Ephraim schreibt Streamdaten weiterhin in den dauerhaften MySQL-Pfad beziehungsweise kann den Datenbankpuffer als Fallback verwenden. Ein Redis/Valkey-Problem ist deshalb in erster Linie ein Betriebs- und Performanceproblem: Streams können langsamer werden oder der Fallback muss greifen. Es ist kein Verlust privater Inhalte.

Für Administratoren heißt das: MySQL-Verfügbarkeit, Backup und Restore sind systemkritisch. Redis/Valkey-Verfügbarkeit ist wichtig für Reaktionsgeschwindigkeit und Worker-Latenz, aber nicht für die dauerhafte Integrität privater Daten.

7.5.6 Betriebsregeln für Production

- MySQL ist die produktive Datenbank und Teil des Backup- und Restore-Konzepts.
- Redis/Valkey läuft auf demselben Host wie die Webanwendung.
- Redis/Valkey ist nur maschinenintern erreichbar; es gibt keinen öffentlichen Port.
- Redis/Valkey wird nicht an externe Dienste, Monitoringanbieter oder Managed-Redis ausgelagert.
- Redis/Valkey-Persistenz bleibt für diesen Zweck deaktiviert.
- Redis/Valkey-Datenverzeichnisse werden nicht als dauerhafte Ephraim-Datenquelle gesichert.
- Diagnosebefehle, die alle Redis-Kommandos mitschneiden, gehören nicht in den Regelbetrieb.
- Redis/Valkey wird nicht für Sessions, Vaults, Dateien, Nutzerprofile, RAG-Klartexte oder dauerhafte Queues zweckentfremdet.

> **Neubewertung nötig:** Wenn Redis/Valkey über Hostgrenzen hinweg betrieben, persistent gesichert, repliziert, extern erreichbar oder für neue Datenarten verwendet wird, ist das keine kleine technische Änderung. Dann müssen Datenschutz, Betrieb und Dokumentation neu bewertet werden.

7.5.7 Admin-Checkliste

Prüfpunkt	Erwartung
MySQL erreichbar	App, Login, Chat, Projekte, Dateien und Adminbereich funktionieren.
MySQL-Backup	Datenbankdump und Restore-Test sind dokumentiert.
Redis/Valkey erreichbar	Streams und Worker reagieren schnell; bei Ausfall greift der Datenbankpfad.
Redis/Valkey nicht extern	Keine öffentliche Erreichbarkeit, kein externer Dienst, kein Managed-Redis.
Redis/Valkey nicht persistent	Keine dauerhafte Redis-Datenquelle, keine normalen Redis-Datenbackups.
Datenschutzgrenze	Keine Klartextinhalte, Vault-Schlüssel, Dateien, Sessions oder RAG-Klartexte in Redis/Valkey.

Dieser Artikel beschreibt den produktiven Datenbankbetrieb von Ephraim in Version 1. Für Wartung siehe Cron, für Dateien Dateien, für Logging Logging und für die Gesamtarchitektur Systemarchitektur.

Quelle: web/manuals/admin-datenbank/index.html

7.6 Logging

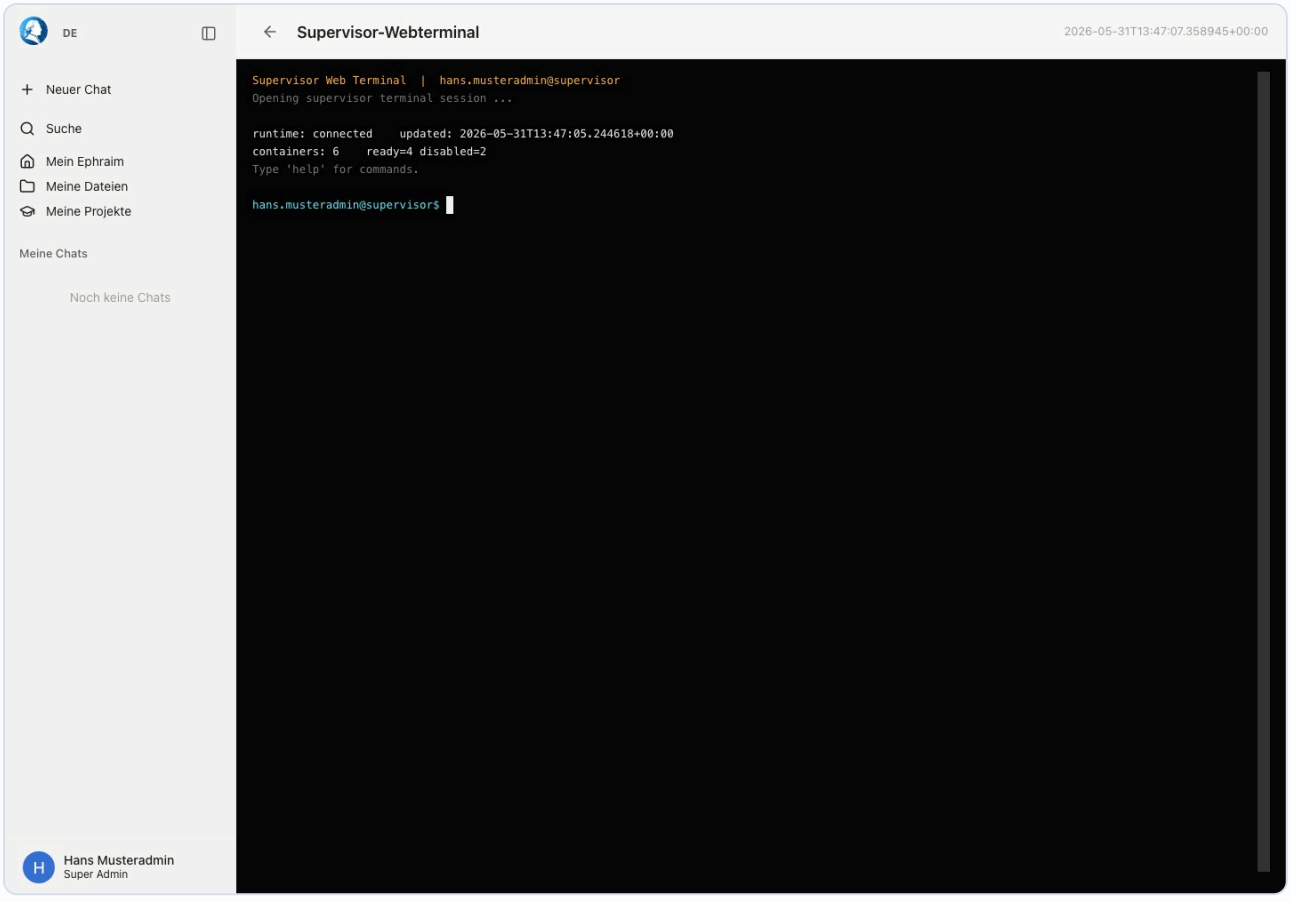
> Logging in Ephraim ist für den Produktionsbetrieb gedacht: Störungen erkennen, Ursachen eingrenzen und administrative Eingriffe nachvollziehbar machen. Es ist ausdrücklich nicht dafür gedacht, Prompts, Antworten, Dateien, Tokens oder private Arbeitsverläufe mitzuschreiben.

Stand: Mai 2026

7.6.1 Produktionsprinzip

In Production gilt ein anderes Logging-Modell als in der lokalen Entwicklung. Ephraim zeigt interne PHP-Fehler nicht im Browser an, schreibt keine normale PHP-Fehlerlogdatei als Hauptsignal und meldet kritische App-Probleme über eine begrenzte Admin-Problem-Mail. Die Diagnose beginnt deshalb nicht mit einem großen Rohlog, sondern mit Statusseite, Problem-Mail, Cronmonitor, Runtime-Logs und Super-Admin-Auditberichten.

> **Leitregel:** Logs enthalten technische Betriebsereignisse. Private Inhalte gehören in verschlüsselte Nutzer- oder Projektdaten, nicht in Protokolle.



Das Supervisor-Webterminal ist der gezielte Einstieg in Runtime- und Containerlogs. Es ersetzt keine allgemeine Rohdatensammlung.

7.6.2 Webserver und Apache

Der produktive Schul-Webserver wird so betrieben, dass Apache kein Diagnosewerkzeug für fachliche HTTP-Anfragen ist. Es gibt kein Apache-Logging von Request-Bodies: keine Formularinhalte, keine Chatnachrichten, keine Datei- oder JSON-Payloads und keine sonstigen Anfrageninhalte. Solche Inhalte werden nicht als Webserverlog benötigt und dürfen dort nicht erscheinen.

Tokens gehören ebenfalls nicht in Apache-Logs. Dazu zählen Session-Cookies, CSRF-Tokens, Reset- und Setzlinks, Cron-Tokens, Bearer-Tokens, Runtime-JWTs, OAuth-Codes und API-Schlüssel. Deshalb ist in Production kein Apache-Accesslog mit Requestzeilen, Query-Strings, Headern oder Cookies als fachliche Diagnosequelle vorgesehen.

> **Konsequenz:** Fehleranalysen verlangen keine „vollständige Anfrage“ aus Apache. Technische Hinweise kommen aus den dafür vorgesehenen Ephraim-Signalen: redigierte Problem-Mails, Statusseite, Cronmonitor, Runtime-Logs und Auditberichte.

7.6.3 PHP-App und Problem-Mails

Die PHP-Anwendung behandelt Production-Fehler bewusst zurückhaltend. Interne Details werden nicht an Nutzer ausgegeben. Gefangene Fehlerpfade und globale Fatal- oder Exception-Handler laufen in die Admin-Problem-Mail-Pipeline. Diese Mails sind rate-limitiert, nach Schweregrad filterbar und redigieren offensichtliche Secrets.

Element	Production-Verhalten
Browserausgabe	Keine internen PHP-Fehlerdetails für Nutzer.
PHP-Fehlerlog	In Production abgeschaltet; nicht als primäre Betriebsquelle vorgesehen.
Admin-Problem-Mail	Redigierter Bericht mit Kontext, Zeit, Schweregrad, Request-Zusammenfassung, IP, User-Agent, Details, Metadaten und gekürztem Trace.
Rate-Limit	Gleiche Problemfingerprints werden begrenzt, damit ein Fehler nicht die Mailbox flutet.
Redaktion	Tokens, Passwörter, API-Schlüssel, Authorization-Header, lange Hexwerte, Base64-ähnliche Secrets und JWT-ähnliche Werte werden bestmöglich maskiert.

Die Redaktion ist eine Schutzschicht, keine Einladung zum sorglosen Loggen. Wer eine Fehlermeldung formuliert, schreibt keine Passwörter, DEKs, private Schlüssel, vollständige Prompts, Dateiinhalte oder langen Rohlogs hinein.

7.6.4 Spark und KI-Dienste

Die Spark ist die lokale KI-Maschine. Sie verarbeitet Prompts, Kontext, Embeddings, TTS-Anfragen und Runtime-Kommandos für die konkrete Aufgabe. Sie loggt aber keine fachlichen KI- oder TTS-Anfragen: keine KI-Requestpayloads, keine Promptinhalte, keine Modellantworten, keine TTS-Texte, keine TTS-Audiodaten, keine Roh-Tokens, keine Auth-Tokens, keine DEKs und keine Secrets.

Für Retrieval und Vektorerzeugung ruft die Webanwendung den lokalen SGLang-Embedding-Dienst **school-ui-embedding** über den konfigurierten `/embeddings`-Endpunkt auf. Das betrifft Datei-RAG, Projektmaterial-RAG und Basiswissen-RAG. Chunk-Auswahl, Rechteprüfung, Entschlüsselung, Scoreberechnung und Kontextpaket entstehen in Ephraim selbst und werden ebenfalls nicht als Inhaltslog geführt. Die Spark speichert Embedding-Klartexte, Suchfragen und Vektor-Payloads im Normalbetrieb nicht dauerhaft.

Dienst	Was für den Betrieb sichtbar sein darf	Was nicht geloggt wird
Inference	Technische Metriken wie Anfragezählung, Latenz und Tokenzählungen als Zahlenwerte.	Prompts, Kontexttexte, Modellantworten und Roh-Tokens.
Embedding	Technische Verfügbarkeit und Fehlerzustände des konfigurierten Embedding-Endpunkts.	Klartext der zu vektorisierenden Textausschnitte, Suchfragen und Vektor-Payloads.
Retrieval in der Webanwendung	Technische Kennzahlen wie Trefferstatus, Größenklasse oder Fehlertyp.	Chunktexte, private Kalenderinhalte, Dateiinhalte und erzeugte Kontextpakete als Log.
Runtime und Supervisor	Admin-Aktionen, Jobstatus, Exitcodes, Dienstnamen, Zeitpunkte und technische Meldungen.	Private Chatprompts, Modellantworten, DEKs und Auth-Secrets.
TTS	Bereitschaft, Start- und Crashdiagnose des lokalen Sprachdienstes.	Vorzulesender Text, Audiodaten, Access-Logs und Synthese-Metadaten im Normalbetrieb.

7.6.5 Runtime- und Containerlogs

Das Supervisor-Webterminal zeigt für bekannte Runtime-Container einen aktuellen Logauszug oder verfolgt einen Logstream. Die typischen Befehle entsprechen `docker logs <container>` und `docker logs -f <container>`, laufen aber über die abgesicherte Runtime-Verbindung und kurzlebige Admin-Tokens.

Diese Logs beantworten Betriebsfragen: Startet ein Dienst? Ist ein Modell bereit? Gibt es Timeouts, Updatefehler, Ressourcenprobleme oder nicht erreichbare interne Dienste? Sie beantworten nicht die Frage, was ein Schüler in einem privaten Chat geschrieben hat.

> **Praxisregel:** Beim Kopieren von Logauszügen für Fehlersuche nur den relevanten technischen Ausschnitt übernehmen. Lange Dumps, Tokens, private Texte und vollständige Stacktraces werden gekürzt oder redigiert.

7.6.6 Cron-Logs und Wartung

Cron besitzt eine eigene adminlesbare Logspur für den letzten Wartungslauf. Sie zeigt Start, Ende, betroffene Aufgaben, Fehler und Kennzahlen. Zusätzlich speichert Ephraim eine Zusammenfassung pro Task. Fehlschläge einzelner Tasks werden als Problem gemeldet, ohne den gesamten Lauf sofort abzubrechen.

Cron-Logging ist bewusst kein langfristiges Vollarchiv. Es soll beantworten, ob Wartung läuft, ob sie zuletzt erfolgreich war und welche Aufgabe zuletzt auffällig war. Für Details zu Ausführungsarten, Token/IP-Schutz und Datenschutzgrenzen siehe Cron.

7.6.7 Audit, Sicherheitsereignisse und Grenzen

Neben Logs gibt es Audit- und Sicherheitsereignisse. Datei-Sicherheitsereignisse dokumentieren blockierte oder auffällige Uploads mit technischer Begründung. Super-Admin- Auditberichte verdichten Runtime- und Sitzungsdaten zu Markdown-Berichten. Sie dienen der Systemprüfung und nicht dem Öffnen privater Chats.

Roh-Exports bleiben ein Super-Admin-Werkzeug. KI-Auditberichte werden vor der Auswertung und Speicherung zusätzlich bereinigt: Bearer-Tokens, benannte Secret-Felder, lange Tokenstrings, URLs und private Schlüssel werden maskiert. Trotzdem gilt: Ein Auditbericht ist ein sensitives Betriebsdokument. Er gehört nicht in öffentliche Tickets, nicht in normale Chatverläufe und nicht in ungeschützte Cloudordner.

7.6.8 Was nicht in Logs gehört

Ephraim folgt bei Logging einem Minimalprinzip. Wenn eine Information nicht nötig ist, wird sie nicht protokolliert. Wenn eine Information für Diagnose nötig ist, wird sie so weit reduziert, dass sie den Betrieb erklärt, aber private Inhalte schützt.

Nicht loggen	Stattdessen verwenden
Passwörter, Resetlinks, Setzlinks, CSRF-Tokens, Session-Cookies, Bearer-Tokens, OAuth-Codes, Runtime-JWTs.	Tokenart, Zeitpunkt, Endpunkt und redigierte Fehlerklasse.
User-DEKs, Server-Schlüssel, private Schlüssel, API-Keys und vollständige Connection-Strings.	Konfigurationsbereich, Dienstname und Hinweis „Secret fehlt“, „Secret ungültig“ oder „Signaturprüfung fehlgeschlagen“.
Prompts, Chatantworten, private Dateien, Projektdateien, Kalenderinhalte, Erinnerungen und private Wissensquellen.	Interne ID, Dateityp, Größenklasse, Job-ID, Chunkanzahl oder Status.
Lange Rohlogs, vollständige Stacktraces in Tickets, komplette Audit-Exports.	Kleiner relevanter Ausschnitt, Fehlerfingerprint, Zeitfenster und betroffener Dienst.

7.6.9 Ablauf bei einer Produktionsstörung

1. Statusseite prüfen: Webserver, Spark, Inference, Runtime und Containerzustände einordnen.
2. Prüfen, ob eine Admin-Problem-Mail zum Zeitpunkt der Störung eingegangen ist.
3. Bei Wartungsproblemen Cronstatus, letzten Erfolg, Tasktabelle und letztes Cron-Log lesen.

4. Bei Spark-Problemen gezielt Runtime- oder Containerlogs für den betroffenen Dienst öffnen.
5. Bei Sicherheits- oder Updatefragen Super-Admin-Auditberichte und Datei-Sicherheitsereignisse heranziehen.
6. Nur die kleinste nötige technische Information weitergeben; Tokens, private Inhalte und Rohdumps redigieren.
7. Nach der Korrektur erneut Status, Cron und betroffenen Dienst prüfen.

7.6.10 Abgrenzung

Logging ersetzt keine Datensicherung, kein Patchmanagement, keine Zugriffsprüfung und keine pädagogische Bewertung. Logs zeigen technische Ereignisse. Sie erklären nicht automatisch fachliche Ursachen und sie berechtigen nicht dazu, private Lernarbeit zu lesen.

Die konkrete Aufbewahrung von Spark-SSH-Adminlogs, Supervisor-Logs, Containerlogs und Auditberichten ist Teil des organisatorischen Betriebskonzepts. Ephraim liefert dafür reduzierte technische Spuren und Redaktionsmechanismen; die Schule legt Zugriff, Retention und Verantwortlichkeiten im Produktionsbetrieb verbindlich fest.

Logging in Ephraim ist datensparsame Betriebsdiagnose. Verwandte Artikel sind Status, Supervisor-Webterminal, Cron, Auditberichte und Datei-Sicherheit.

Quelle: web/manuals/admin-logging/index.html

7.7 Dateien im Betrieb

> **Dateien sind in Ephraim kein statischer Webordner. Sie werden geprüft, verschlüsselt abgelegt, über kontrollierte Endpunkte ausgeliefert und in Produktion zusätzlich durch ClamAV gescannt.**

Stand: Juni 2026

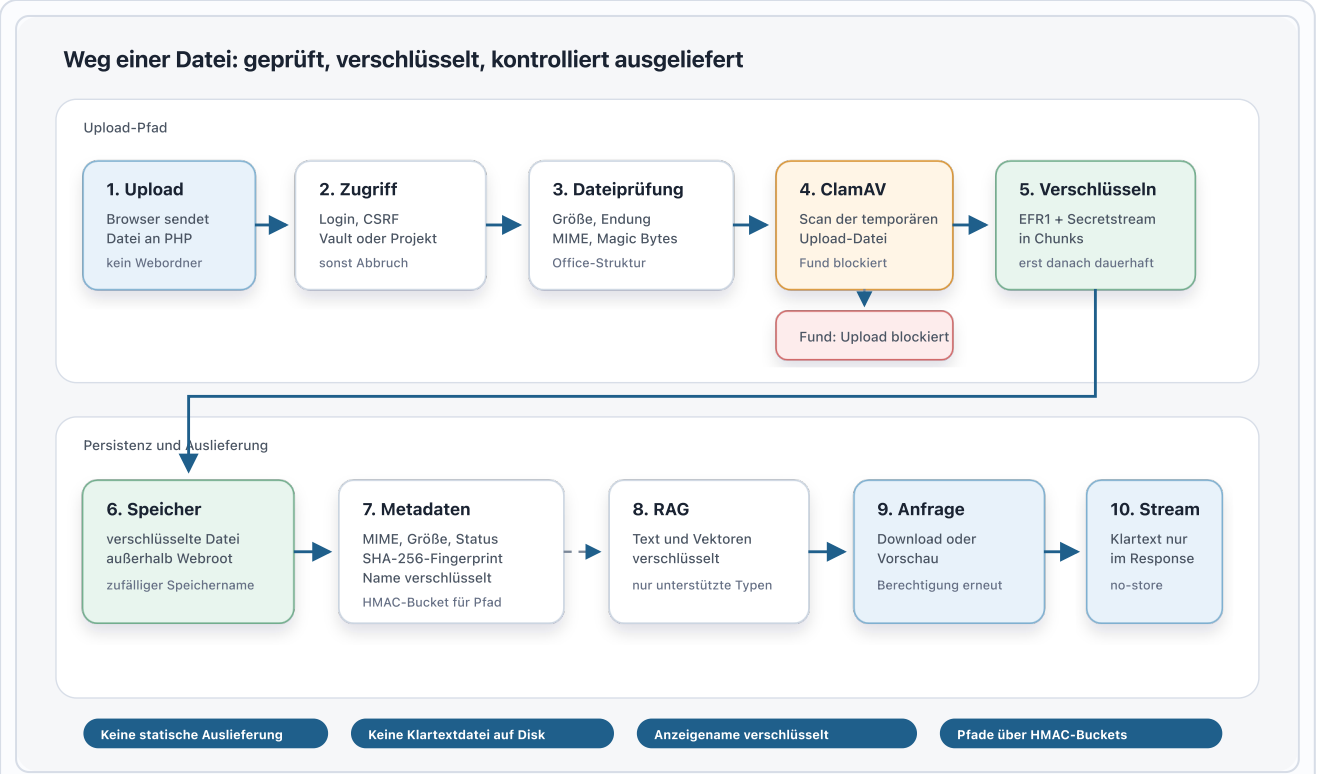
7.7.1 Kurzfassung

Ephraim speichert hochgeladene Dateien außerhalb des Webroots. Der Webserver liefert diese Dateien nicht direkt aus. Jede Anzeige, Vorschau und jeder Download läuft durch PHP-Endpunkte, die Anmeldung, Berechtigung, Vault-Zustand, Dateieigentum und Projektzugang erneut prüfen.

Uploads durchlaufen Dateitypprüfungen und einen ClamAV-Scan. Danach werden Inhalte verschlüsselt gespeichert; Anzeigenamen werden verschlüsselt und Speicherverzeichnisse über HMAC-Hashing aus Nutzer- oder Projekt-IDs abgeleitet. Auf dem Datenträger stehen dadurch keine Klartextdateien, keine sprechenden Dateinamen und keine direkt lesbaren Nutzer- oder Projektpfade.

Für Administratoren sind drei Fragen entscheidend: Welche Dateiarten dürfen überhaupt hochgeladen werden? Reicht Speicherplatz und Quota? Und meldet die Datei-Sicherheitsseite auffällige Uploads oder Scannerprobleme?

Einfach gesagt: Eine hochgeladene Datei wird nicht wie ein Bild in einem öffentlichen Website-Ordner abgelegt. Ephraim nimmt sie entgegen, prüft sie, verschließt sie kryptografisch und legt nur die verschlossene Fassung ab. Wenn eine berechtigte Person die Datei später öffnet, wird sie nur für diese konkrete Antwort wieder entschlüsselt.



Der Weg einer Datei führt immer durch PHP: erst prüfen und scannen, dann verschlüsseln und speichern, später nur nach erneuter Berechtigungsprüfung entschlüsseln und streamen.

7.7.2 Drei Dateiklassen

Betrieblich sehen Nutzerinnen einfach „Dateien“. Technisch unterscheidet Ephraim drei Klassen, weil Datenschutz, Freigabe und Schlüsselverwaltung unterschiedlich sind.

Dateiklasse	Speicherort	Schlüssel	Zugriff
Persönliche Dateien	`STORAGE_ROOT/users`	User-Vault-DEK des Besitzers	Nur der Besitzer mit entsperrtem Vault
Projektmaterialien	`STORAGE_ROOT/projects`	Eigener Projektdateischlüssel, serverseitig geschützt	Projektbesitzer, Admins und berechtigte Projektteilnehmer
Zentrale Basiswissen-Dateien	`STORAGE_ROOT/knowledge`	Serverseitiger Knowledge-Schlüssel	Kontrolliert über Basiswissen-Quellen und Sichtbarkeit

Diese Trennung ist gewollt. Eine persönliche Datei wird nicht automatisch zu Projektmaterial. Wenn eine Lehrkraft eine private Datei in ein Projekt übernimmt, entsteht eine bewusste Projektkopie mit eigenem Schlüssel. Das private Original bleibt privat.

7.7.3 Speicherung auf dem Server

Der Standard-Speicher liegt neben dem Webroot im Verzeichnis `storage`. In Produktion wird der Speicherort über `STORAGE_ROOT` festgelegt und muss außerhalb des auslieferbaren Webbereichs liegen. Pfade enthalten keine Klarnamen und keine direkt ablesbaren Benutzer- oder Projekt-IDs.

Ebene	Was gespeichert wird	Warum das wichtig ist
Dateisystem	Verschlüsselte `EFR1`-Dateien mit zufälligem Speichernamen	Ein Dateisystemfund ist kein Klartextfund.
Datenbank	Metadaten wie MIME-Typ, Größe, Status, SHA-256-Fingerprint und Speichername	Die Anwendung kann Dateien verwalten, ohne sie öffentlich abzulegen.
Anzeigename	Verschlüsselt beim jeweiligen Dateiobjekt	Dateinamen verraten oft Inhalte und werden deshalb geschützt.
RAG-Chunks	Extrahierter Text und Embeddings verschlüsselt in Chunk-Tabellen	Auch Suchkontext bleibt an den passenden Schlüssel gebunden.

Das Dateiinnere wird streamend mit XChaCha20-Poly1305 verarbeitet. Eine Datei besteht aus dem Marker `EFR1`, einem Secretstream-Header und verschlüsselten Chunks. Der letzte Chunk trägt ein Abschluss-Tag. Fehlt dieses Abschluss-Tag, gilt die Datei beim Entschlüsseln als unvollständig.

Präzise formuliert: Der sichtbare Anzeigename wird verschlüsselt, der technische Speicherdateiname wird zufällig erzeugt, und das Nutzer- oder Projektverzeichnis wird per HMAC aus der jeweiligen ID abgeleitet. So bleibt auch die Verzeichnisstruktur für Außenstehende unlesbar.

Der SHA-256-Fingerprint ist keine Kopie des Inhalts und nicht zurückrechenbar. Er ist aber ein stabiler technischer Fingerabdruck: dieselbe Klartextdatei erzeugt denselben Wert. Deshalb gehört er zu den Metadaten und nicht zu den öffentlich sichtbaren Informationen.

7.7.4 Upload-Pipeline

Uploads werden nicht erst nach dem Speichern geprüft. Die Prüfungen laufen vor der dauerhaften Persistenz. Erst wenn die Datei Größe, Name, Typ, Struktur und den Virenscan passiert hat, wird sie verschlüsselt abgelegt.

Schritt	Prüfung	Fehlerwirkung
1	Anmeldung, CSRF-Token und bei persönlichen Dateien entsperrter Vault	Request wird abgewiesen; persönliche Dateien bleiben ohne Vault unzugänglich.
2	PHP-Uploadstatus und maximale Dateigröße	Zu große oder fehlerhaft übertragene Dateien werden abgelehnt.
3	Dateiname, gefährliche Namen, Endung und systemweite Allowlist	Riskante Namen und nicht erlaubte Typen werden protokolliert und blockiert.
4	MIME-Erkennung, Magic Bytes, HEIC/HEIF-Header und Office-/iWork-Struktur	Inhalt muss zur behaupteten Datei passen.
5	ClamAV-Scan im Produktionsbetrieb	Malware-Fund blockiert den Upload und erzeugt ein kritisches Sicherheitsereignis.
6	Quota und physischer Speicherplatz	Nutzer-, System- und Speichergrenzen verhindern Überfüllung.
7	Verschlüsselung, Datenbankeintrag und optionale RAG-Indexierung	Erst jetzt wird die Datei dauerhaft nutzbar.

7.7.5 ClamAV in Produktion

In Produktion ist ClamAV Teil der Upload-Pipeline. Der Webprozess übergibt die temporäre Upload-Datei an den Scanner, bevor Ephraim sie verschlüsselt und persistiert. Das System unterscheidet vier Ergebnisse: sauber, infiziert, Scannerfehler und nicht verfügbar. Nur ein Malware-Fund ist ein fachlicher Upload-Blocker. Scannerfehler und fehlende Erreichbarkeit sind Betriebszustände, die Admins über Datei-Sicherheit und Cron im Blick behalten.

Scan-Ergebnis	Betriebliche Bedeutung
Sauber	Die Datei läuft weiter durch Quota, Verschlüsselung und Speicherung.
Infiziert	Die temporäre Datei wird verworfen, der Upload endet mit Fehler und ein kritisches Ereignis wird protokolliert.
Scannerfehler	Der Upload wird nicht durch einen temporären Scannerfehler blockiert; Ephraim protokolliert den Fehler und Admins prüfen den Scannerzustand.
Nicht verfügbar	Der Zustand ist ein Betriebsfehler. Die Adminseite und Cron zeigen, ob ClamAV aktiviert und erreichbar ist.

> **Wichtig:** ClamAV ersetzt nicht die anderen Prüfungen. Ephraim prüft Dateinamen, Typen, MIME-Werte, Magic Bytes und Containerstrukturen zusätzlich, weil ein Virenscanner bekannte Schadsoftware erkennt, aber keine vollständige Dateipolitik ist.

7.7.6 Auslieferung und Vorschau

Downloads und Vorschauen werden dynamisch erzeugt. Die Anwendung lädt den verschlüsselten Datenträgerinhalt, prüft erneut die Berechtigung, entschlüsselt nur für diesen Request und streamt den Klartext direkt in die HTTP-Antwort. Es gibt keinen öffentlichen Link auf die gespeicherte Datei.

Funktion	Auslieferung	Schutz
Persönlicher Download	Als Attachment an den eingeloggten Besitzer	Login, Vault-DEK, Eigentümerprüfung
Persönliche Vorschau	PDF inline oder Bild-Thumbnail	Same-Origin-Framing, `no-store`, `nosniff`
Projekt-Download	Als Attachment an berechtigte Projektakteure	Projektzugang, Projektstatus, Schlüsselentsiegelung
Projekt-Vorschau	PDF inline oder Bild-Thumbnail	Same-Origin-Framing, `no-store`, `nosniff`

Bildvorschauen sind keine persistierten Klartextdateien. Ephraim entschlüsselt das Bild für den aktuellen Request, erzeugt ein verkleinertes JPEG ohne Metadaten und sendet es sofort aus. Für PDF-Vorschauen wird die geschützte Quelle inline ausgeliefert.

7.7.7 Quotas und Speicherpflege

Die Datei-Quota zählt persönliche Dateien und aktive Projektmaterialien zusammen. RAG-Chunks werden nicht ein zweites Mal als Nutzerquota gezählt. Für Admins ist wichtig: Uploadgröße, Nutzerquota, systemweite Datei-Quota und freier physischer Speicher wirken gemeinsam.

Grenze	Wo sie wirkt
Maximale Uploadgröße	Kleinster Wert aus Admin-Einstellung, Konfiguration, PHP-Limits und App-Hard-Cap.
Nutzerquota	User-Override, Rollen-Default oder System-Default.
Systemweite Datei-Quota	Schützt den Gesamtspeicher vor Überbelegung.
Physischer Speicher	Ephraim blockiert Uploads, wenn der verfügbare Speicher unter die Reserve fällt.

Der Cron-Task `cleanup_orphan_storage` entfernt verwaiste Binärdateien unter `STORAGE_ROOT/users` und `STORAGE_ROOT/projects`, wenn sie nicht mehr in den Datei-Tabellen referenziert sind. `cleanup_security_events` entfernt alte Datei-Sicherheitsereignisse nach 90 Tagen.

7.7.8 Adminaufgaben

1. Datei-Sicherheit regelmäßig prüfen: erlaubte Typen, ClamAV-Status und letzte Sicherheitsereignisse.
2. Speicherquoten beobachten: Systemlimit, Rollenlimits, Nutzer-Overrides und freien physischen Speicher.
3. Cron überwachen: verwaiste Speicherdateien und alte Sicherheitsereignisse müssen regelmäßig bereinigt werden.
4. Bei ClamAV-Fehlern nicht nur Uploads testen, sondern den Scannerzustand und die Sicherheitsereignisse prüfen.
5. Projektmaterialien als bewusste Freigabe behandeln: Eine private Datei bleibt privat, bis sie aktiv ins Projekt kopiert oder dort hochgeladen wird.

Angrenzende Artikel: Datei-Sicherheit, Speicherquoten, Cron, Logging und Kryptoarchitektur.

Quelle: web/manuals/admin-dateien/index.html

7.8 Drittanbieter-Komponenten

> Ephraim nutzt ausgewählte Drittanbieter-Komponenten lokal gebündelt. Sie rendern Markdown, Formeln, Code, Diagramme, Terminalansichten, E-Mails, PDFs und serverseitige Vorlagen. Die Adminfunktion macht diese Abhängigkeiten sichtbar und trennt streng zwischen Hinweis, technischer Prüfung, Freigabe, Installation und Rollback.

Stand: Mai 2026

7.8.1 Warum Ephraim Drittanbieter-Komponenten gesondert verwaltet

Drittanbieter-Komponenten sind Programmteile, die nicht von Ephraim selbst stammen, aber im Produkt mit ausgeliefert werden. Sie laufen im Sicherheitskontext der Anwendung: JavaScript- und CSS-Komponenten beeinflussen die Darstellung im Browser, PHP-Komponenten erzeugen E-Mails, PDFs oder serverseitige Vorlagen. Ein Fehler in einer solchen Komponente berührt deshalb nicht nur Komfort, sondern auch Sicherheit, Datenschutz, Exportqualität und Betriebssicherheit.

Ephraim behandelt diese Komponenten als kontrollierte Lieferkette. Ein externer Versionshinweis ist keine Installationsfreigabe. Ein Download ist kein geprüfter Zielstand. Eine technische Prüfung ist noch keine Entscheidung. Erst ein freigegebener, erneut geprüfter und vorbereiteter Paketstand wird installiert.

> **Kernsatz:** Die Komponentenverwaltung ersetzt automatische Paketupdates durch einen nachvollziehbaren Freigabeprozess mit Rollen, Prüfsummen, Staging, Backup und Rollback.

7.8.2 Welche Komponenten zum Inventar gehören

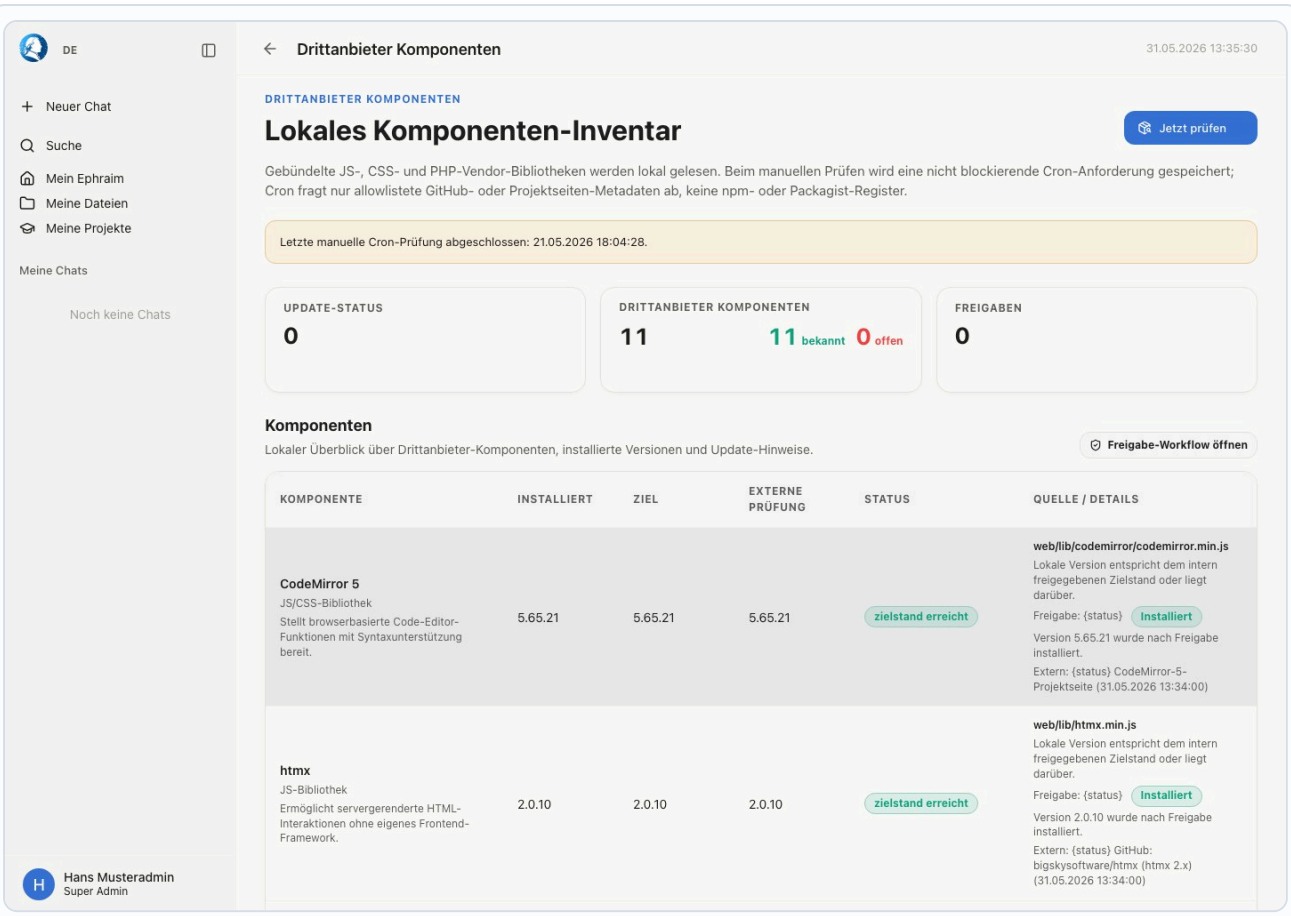
Das Inventar umfasst lokal gebündelte Browser- und PHP-Komponenten des Frontends. Nicht dazu gehören Betriebssystempakete, Datenbankserver, PHP selbst, Webserver, Container, die Spark-Dienste, externe Inferenzsysteme oder Browser der Nutzerinnen und Nutzer. Diese Grenze ist wichtig: Die Adminseite beantwortet die Frage, welche in Ephraim ausgelieferten Bibliotheken gepflegt werden.

Bereich	Komponenten	Aufgabe in Ephraim
Chat-Rendering	Marked, KaTeX, PrismJS	Markdown, mathematische Formeln und Codehervorhebung werden im Browser dargestellt.
Editoren und Interaktion	CodeMirror 5, htmx, xterm.js	Editorflächen, servergerenderte Interaktionen und Terminalansichten werden bereitgestellt.
Visualisierungen	JSXGraph, SmilesDrawer	Interaktive mathematische Plots und chemische Strukturformeln werden gerendert.
Serverseitige Ausgabe	Twig, PHPMailer, TCPDF	Vorlagen, E-Mails und PDF-Dokumente werden erzeugt.

Die Liste ist bewusst klein und namentlich bekannt. Ephraim durchsucht keine beliebigen Ordner nach unbekannten Bibliotheken und übernimmt keine Komponenten aus Nutzereingaben. Jede inventarisierte Komponente hat einen festen Namen, eine feste fachliche Aufgabe und eine definierte Methode zur lokalen Versionserkennung.

7.8.3 Was die Übersicht zeigt

Die Adminübersicht ist die tägliche Kontrollfläche. Sie liest den lokalen Stand, zeigt intern freigegebene Zielstände und stellt den zuletzt gecachten externen Hinweis daneben. Dadurch sehen Admins in einer Tabelle, ob Ephraim eine Komponente erkennt, ob eine Quelle neuere Informationen gemeldet hat und ob bereits eine interne Freigabe existiert.



Die Übersicht trennt installierten Stand, Zielstand, externe Prüfung, Status und Freigabezustand. Aktionen führen von hier in den Freigabe-Workflow.

Spalte	Bedeutung
Komponente	Name, Kategorie und kurze fachliche Beschreibung der Bibliothek.
Installiert	Version, die lokal aus den gebündelten Dateien erkannt wurde.
Ziel	Version, die intern freigegeben wurde. Ohne Freigabe bleibt das Feld leer.
Externe Prüfung	Zuletzt gespeicherter Versionshinweis aus einer erlaubten Projektquelle.
Status	Einordnung wie erkannt, aktuell, neuer Stand gefunden, Zielstand freigegeben oder Zielstand erreicht.
Quelle / Details	Herkunft des Hinweises, letzter Prüfzeitpunkt, Freigabestatus und Detailmeldungen.

7.8.4 Warum Ephraim keine Paketregister automatisch nutzt

Die Komponentenpflege fragt keine allgemeinen Paketregister ab. Ephraim nutzt für diesen Prozess kein npm, kein Packagist, keine CDN-Autoupdates, keine frei eingebbaren Download-URLs, keine privaten Repositories und keine Zugangsdaten zu externen Quellen. Diese Entscheidung reduziert die Angriffsfläche: Admins tragen keinen beliebigen Link in die Anwendung ein und installieren daraus keinen Code.

Externe Informationen sind Hinweise. Sie stammen aus fest erlaubten öffentlichen GitHub-Projekten oder ausdrücklich erlaubten Projektseiten. Die Anwendung übernimmt daraus zunächst nur Metadaten wie einen gemeldeten Versionsstand. Der normale Adminseitenaufruf wartet nicht auf eine externe Abfrage. Eine manuelle Prüfung speichert eine Anforderung; der Wartungslauf verarbeitet diese Anforderung im Hintergrund und aktualisiert den gecachten Stand.

> **Abgrenzung zur KI-Nutzung:** Diese externen Komponentenprüfungen gehören zur Admin-Wartung des Websystems. Sie sind kein Internetzugriff des Spark-Modells und verändern nicht die lokale Inferenz: Das Modell surft weiterhin nicht im Web.

7.8.5 Rollen und Zuständigkeiten

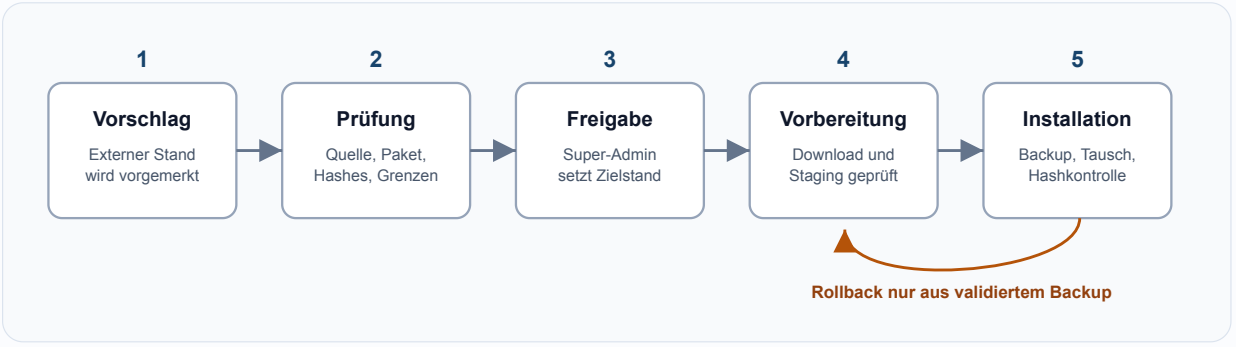
Der Workflow trennt Lesen, Vorschlagen, Entscheiden und Installieren. Dadurch entstehen klare Zuständigkeiten und keine einzelne Oberfläche führt unbemerkt ein vollständiges Update aus.

Rolle	Berechtigung im Komponentenbereich	Nicht erlaubt
Admin	Inventar lesen, manuelle Prüfung anfordern, extern gefundenen Stand als Vorschlag markieren.	Technisch prüfen, Zielstand freigeben, installieren, zurückrollen.
Super-Admin	Vorschläge technisch prüfen, Zielstände freigeben, Vorschläge ablehnen, Pakete vorbereiten, installieren und zurückrollen.	Beliebige Quellen hinzufügen oder Prüfregeln in der Oberfläche umgehen.
Wartungslauf	Inventar regelmäßig aktualisieren, manuelle Prüfanfragen verarbeiten, Dashboard-Badge berechnen, Super-Admins informieren.	Zielstände freigeben, Updates installieren oder Rollbacks auslösen.

Die gleichen fachlichen Grenzen gelten auch für terminalgestützte Adminarbeit und berechtigte Automationsclients. Der technische Zugang nutzt denselben Komponentenworkflow; er ist keine Abkürzung um Rollen, Bestätigungen oder Prüfbedingungen herum.

7.8.6 Der Freigabe-Workflow im Überblick

Der Freigabe-Workflow besteht aus fünf getrennten Zuständen. Jeder Schritt erzeugt eine eigene fachliche Aussage: Ein Vorschlag sagt, welcher externe Stand betrachtet wird. Die technische Prüfung sagt, ob genau dieser Stand aus einer erlaubten Quelle reproduzierbar und mit Hashwerten belegbar ist. Die Freigabe sagt, dass ein Super-Admin diesen geprüften Stand als Ephraim-Zielstand akzeptiert. Vorbereitung und Installation prüfen das Paket erneut, bevor Dateien ersetzt werden.



Der Ablauf ist absichtlich unterbrochen: Jede Stufe hat eigene Voraussetzungen, eigene Protokolle und eigene Abbruchpunkte.

Schritt	Konkrete Bedeutung	Ergebnis
Vorschlagen	Ein Admin markiert den gecachten externen Stand als prüfenswert.	Ein Vorschlag mit Komponente, Version, Akteur und optionaler Begründung liegt vor.
Technisch prüfen	Ein Super-Admin startet die Prüfung gegen eine erlaubte Paketquelle.	Archiv-Hash, Paket-Hash, Datei-Hashes, Manifest und Prüfergebnis werden gespeichert.
Zielstand freigeben	Ein Super-Admin akzeptiert genau den geprüften Stand als internen Zielstand.	Die Übersicht zeigt einen freigegebenen Zielstand; noch ist nichts installiert.
Vorbereiten	Das freigegebene Artefakt wird erneut geladen, gegen die gespeicherten Hashes geprüft und gestaged.	Ein vorbereiteter Paketstand liegt außerhalb der produktiven Dateien bereit.
Installieren	Vor dem Dateitausch wird ein Backup erstellt; danach prüft Ephraim die installierten Dateien gegen das Manifest.	Der Paketstand ist installiert oder der Vorgang bricht mit Rollback ab.

7.8.7 Erlaubte Paketquellen und Paketmodi

Eine installierbare Paketquelle entsteht nicht durch einen Klick in der Adminoberfläche. Sie wird als Wartungsentscheidung hinterlegt und ist für Ephraim nur lesbar. Darin steht, welches öffentliche Projekt, welcher Tag- oder Versionsmechanismus, welcher Host, welcher Zielbereich, welche Einstiegspunkte und

welcher Paketmodus für eine konkrete Komponente zulässig sind. Hashes stehen dort nicht fest vorgegeben; Ephraim berechnet sie während der technischen Prüfung selbst.

Wenn für eine Komponente nur eine Hinweisquelle existiert, bleibt sie im Workflow lesbar. Die UI zeigt dann Wartungsbedarf oder blockierte Installation. Das ist kein Fehlerzustand, sondern eine Schutzregel: Ein Hinweis auf eine neue Version reicht nicht aus, um ausführbaren Code in Ephraim zu übernehmen.

Paketmodus	Was geprüft wird	Typische Verwendung
Einzeldatei	Eine oder mehrere ausdrücklich erlaubte Archivdateien ergeben eine lokale Zieldatei; Einstiegspunkte und Hashes werden gespeichert.	Kleine Browser-Bundles, Stylesheets oder zusammengesetzte JavaScript-Pakete.
Verzeichnisersatz	Ein ganzer Komponentenbereich wird aus einem geprüften Release-Paket ersetzt; Dateitypen, Zahl der Dateien, Größe und Einstiegspunkte sind begrenzt.	Komponenten mit mehreren Dateien, etwa CSS, JavaScript, Fonts und Lizenz.
PHP-Vendor	PHP-Dateien, Manifest, Hashes, Syntaxprüfung und komponentenspezifischer Smoke-Test werden geprüft.	Serverseitige Bibliotheken für E-Mail, PDF oder Templates.

7.8.8 Was bei der technischen Prüfung passiert

Die technische Prüfung macht aus einem Vorschlag ein reproduzierbares Paket. Ephraim prüft den Komponentenschlüssel, die Version, den Quelltyp, den Anbieter, den Zielbereich und den Paketmodus. Danach lädt die Anwendung das erlaubte öffentliche Artefakt über HTTPS, begrenzt Zeit, Dateigröße und Umleitungen und verwirft Quellen, die nicht zur erlaubten Hostliste passen.

Anschließend wird das Archiv normalisiert. Ephraim akzeptiert keine Pfade, die aus dem Zielbereich ausbrechen, keine Symlink-Tricks, keine unerwarteten Einstiegspunkte und keine Pakete oberhalb der definierten Größen- und Dateigrenzen. Aus dem Archiv entstehen Datei-Hashes, ein Paket-Hash und ein Archiv-Hash. Diese Werte werden gespeichert und später erneut verglichen.

Prüfung	Konkrete Schutzwirkung
Version und Major-Track	Ein freigegebener Track bleibt eingehalten; ein inkompatibler Hauptversionssprung wird blockiert.
Host und Quelle	Nur öffentliche, erlaubte Projektquellen werden gelesen; freie URLs und private Quellen bleiben ausgeschlossen.
Archivstruktur	Pfade, Verzeichniswurzel, Dateitypen, Dateianzahl und Paketgröße müssen zur Komponente passen.
Einstiegspunkte	Die Dateien, die Ephraim tatsächlich lädt oder aufruft, müssen im Paket vorhanden sein.
SHA-256-Hashes	Archiv, Paket und einzelne Dateien werden später unverändert wiedererkannt.
PHP-Smoke-Tests	Serverseitige Bibliotheken müssen laden und eine minimale fachliche Funktion ausführen.

Die PHP-Smoke-Tests sind absichtlich klein und ohne externe Nebenwirkungen: PHPMailer wird geladen und instanziiert, ohne E-Mail zu versenden. TCPDF erzeugt eine minimale PDF-Ausgabe im Speicher. Twig lädt lokal, liest die Version und rendert ein minimales Template. Zusätzlich werden die PHP-Dateien syntaktisch geprüft.

7.8.9 Freigabe, Vorbereitung und Installation

Die Freigabe speichert keinen abstrakten Begriff wie „neueste Version“, sondern eine konkrete Version mit Quelle, Tag oder Referenz, Paketdaten, Hashes und Akteur. Damit bleibt später nachvollziehbar, welcher Stand gemeint war. Nach der Freigabe beginnt noch keine Installation. Die Vorbereitung lädt das Artefakt erneut und vergleicht es mit den gespeicherten Hashes. Weicht der Download ab, stoppt der Vorgang.

Vor der Installation prüft Ephraim die Produktionsbedingungen: Direkte Updates müssen aktiviert sein, Staging- und Backupbereiche müssen beschreibbar und außerhalb öffentlich ausgelieferter Bereiche liegen, benötigte PHP-Erweiterungen müssen verfügbar sein und bei PHP-Vendor muss ein echter PHP-CLI-Syntaxcheck laufen. Fehlt eine Bedingung, zeigt die Oberfläche den Blocker und führt keinen Dateitausch aus.

Beim Installieren erstellt Ephraim zuerst ein Backup des betroffenen Komponentenbereichs. Danach werden nur die im Manifest erlaubten Dateien ersetzt. Direkt nach dem Tausch prüft Ephraim die aktivierten Dateien gegen die freigegebenen Datei-Hashes. Erst danach wird das Paket als installiert markiert und das Inventar aktualisiert.

7.8.10 Rollback und Fehlerverhalten

Rollback ist kein improvisiertes Zurückkopieren. Ephraim nutzt ein selbst erzeugtes und validiertes Backup-Manifest. Dieses Manifest beschreibt, welcher Komponentenbereich vor der Installation gesichert wurde und welche Dateien wiederhergestellt werden. Ohne gültiges Backup bleibt Rollback gesperrt.

Schlägt eine Installation fehl, versucht Ephraim automatisch, aus dem gerade erzeugten Backup wiederherzustellen. Das Ereignis wird als fehlgeschlagene Installation mit Rollback protokolliert. Schlägt auch das Rollback fehl, meldet die Anwendung diesen Zustand ausdrücklich, weil dann ein manueller Betriebseingriff erforderlich ist.

> **Betriebsregel:** Nach einem Rollback zählt nicht die frühere Freigabe, sondern der tatsächlich lokal erkannte Zustand. Die Übersicht inventarisiert erneut und zeigt dadurch, ob Zielstand und lokale Dateien wieder auseinanderfallen.

7.8.11 Wartungslauf, Badge und Super-Admin-Mails

Der Wartungslauf inventarisiert die Komponenten regelmäßig und verarbeitet manuelle Prüfanfragen. Als Mismatch gelten erkannte Update-Hinweise, freigegebene aber lokal noch nicht erreichte Zielstände und Erkennungsfehler. Die Anzahl erscheint als Badge im Admin-Dashboard, damit offene Komponentenpflege nicht in der Unterseite verborgen bleibt.

Bei Mismatches informiert Ephraim Super-Admins per E-Mail, sofern E-Mail-Adressen hinterlegt sind. Die Benachrichtigung nutzt einen Fingerprint der Mismatch-Liste. Wiederholte Mails entstehen erst, wenn sich der relevante Zustand ändert. Dadurch erhalten Super-Admins Hinweise auf neue Arbeit, ohne bei unverändertem Stand dauerhaft gleiche Meldungen zu bekommen.

7.8.12 Nachvollziehbarkeit und Audit-Spur

Der Komponentenworkflow speichert Vorschläge, technische Prüfungen, Zielstandsfreigaben, Paketdaten, Jobs, Backup-Manifeste und relevante Ereignisse. Zu den gespeicherten Informationen gehören Akteur, Zeitpunkt, Komponente, Version, Status, Quelle, Prüfergebnis und Hashdaten. Diese Audit-Spur beantwortet später drei zentrale Fragen: Wer hat welchen Stand vorgeschlagen? Wer hat ihn freigegeben? Welches Paket wurde installiert oder zurückgerollt?

Die Protokollierung ist nicht nur für Fehlersuche nützlich. Sie macht den Umgang mit Drittanbieter-Code gegenüber Schulleitung, Datenschutzbeauftragten und technischen Prüfern erklärbar. Die Protokolle zeigen, dass Komponenten nicht zufällig aus dem Internet geladen wurden, sondern einem mehrstufigen internen Prozess folgten.

7.8.13 Entscheidungshilfe für Admins und Super-Admins

Komponentenpflege verlangt keine tägliche Aktivität, aber klare Entscheidungen, wenn die Oberfläche einen neuen Stand meldet. Admins und Super-Admins verwenden die folgenden Regeln:

Situation	Richtige Handlung
Externer Stand gefunden, keine offene Freigabe	Als Vorschlag markieren und den Grund benennen, etwa Sicherheitsupdate, Bugfix oder Wartungsabgleich.
Technische Prüfung meldet fehlende erlaubte Paketquelle	Wartung der erlaubten Quelle veranlassen; keinen freien Download als Ersatz verwenden.
Technische Prüfung schlägt fehl	Prüfmeldung auswerten, Quelle oder Version korrigieren lassen und danach erneut prüfen oder ablehnen.
Neuer Major-Track außerhalb der erlaubten Grenze	Nicht freigeben; zuerst Kompatibilität in Ephraim bewerten und den erlaubten Track bewusst ändern lassen.
Zielstand freigegeben, Installation blockiert	Preflight-Blocker beheben; ohne grüne Produktionsbedingungen keine Installation starten.
Installation erfolgreich	Betroffene Oberfläche prüfen: Chat-Rendering, PDF, E-Mail, Terminal, Editor oder Visualisierung je nach Komponente.
Nach Installation treten fachliche Fehler auf	Rollback ausführen, Inventar prüfen und die Ursache anhand der protokollierten Paketdaten untersuchen.

7.8.14 Bewusste Grenzen des Systems

Die Komponentenverwaltung ist kein allgemeiner Software-Updater. Sie aktualisiert keine Serverpakete, keine Container, keine Datenbank, keine Spark-Dienste und keine Betriebssystembibliotheken. Sie löst auch keine Lizenzbewertung und keine fachliche Kompatibilitätsprüfung automatisch. Sie liefert die technische Grundlage, damit ein geprüfter Komponentenstand sicherer und nachvollziehbarer übernommen wird.

Ebenso ersetzt eine grüne technische Prüfung keine fachliche Abnahme. Nach Updates an Rendering-Komponenten werden Chatnachrichten, Mathematik, Codeblöcke, Visualisierungen und Exportpfade geprüft. Nach Updates an PHP-Komponenten werden E-Mail-Versand, PDF-Erzeugung oder Template-Ausgaben passend zur Komponente geprüft. Die Adminfunktion kontrolliert die Lieferkette; die Betriebspraxis kontrolliert die konkrete Wirkung in Ephraim.

Dieses Handbuch beschreibt das Management lokal gebündelter Drittanbieter-Komponenten in Ephraim. Für Entscheidungen gilt der in der Adminoberfläche sichtbare aktuelle Zustand. Das technische Gesamtbild steht in der Systemarchitektur, die Entstehungsgründe stehen im Artikel Entwicklungsprozess.

Quelle: web/manuals/admin-drittanbieter-komponenten/index.html

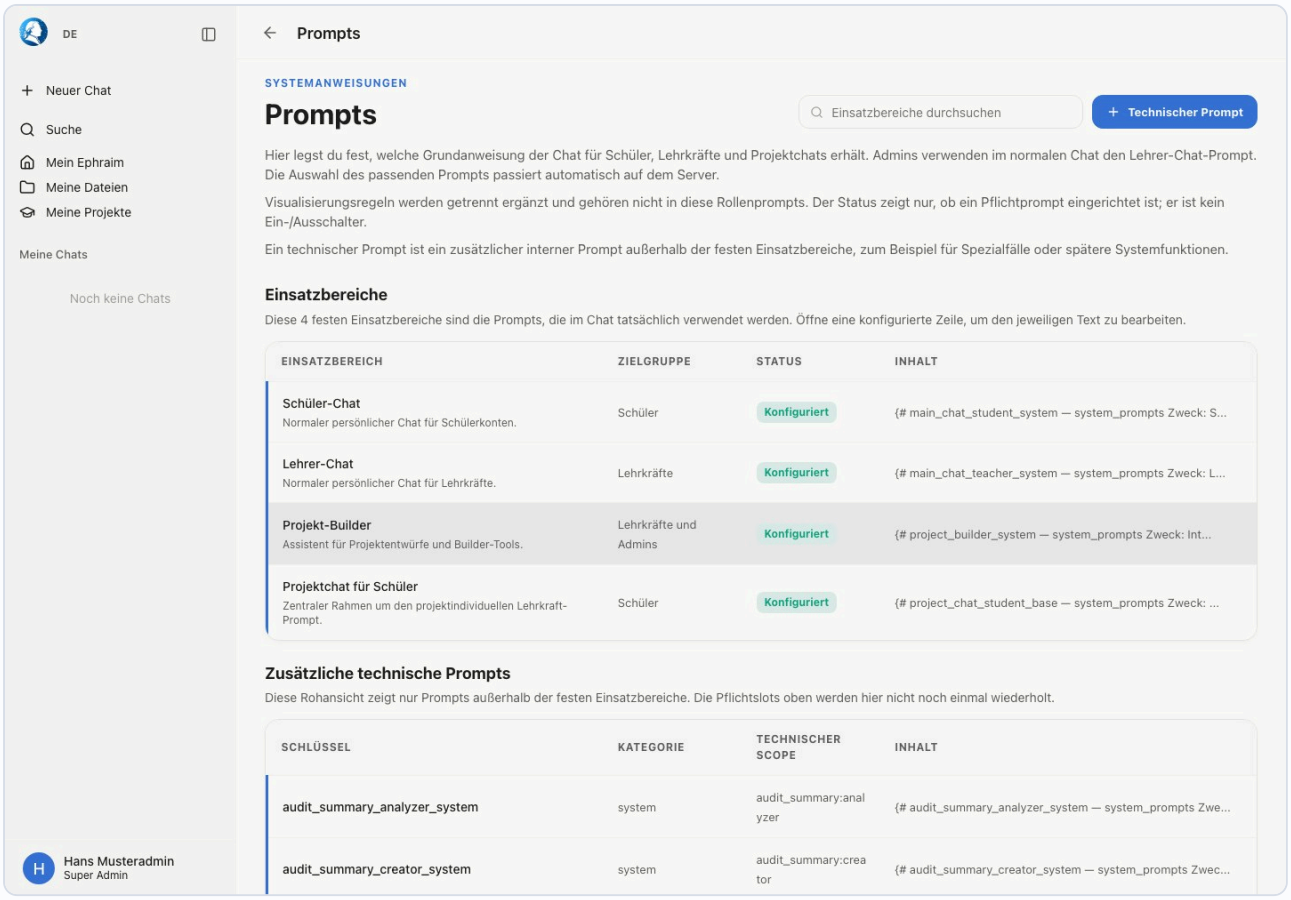
7.9 Prompts

> **Prompts steuern den Rahmen, in dem Ephraim antwortet: Rollenprompts für Chat und Projekte, technische Prompts für Reparaturen und Audits sowie Systemvorgaben, die nicht mit Nutzertext verwechselt werden dürfen.**

Stand: Juni 2026

7.9.1 Aufgabe der Promptverwaltung

Die Promptverwaltung ist kein Textbausteinarchiv, sondern ein Steuerzentrum für KI-Verhalten. Sie legt fest, welche Haltung, Grenzen, Ausgabeformate und Sicherheitsregeln Ephraim in bestimmten Kontexten verwendet. Ein guter Prompt ist klar, knapp genug für wiederholte Nutzung und kompatibel mit den technischen Fähigkeiten der Anwendung.



Die Promptverwaltung zeigt die gepflegten Promptbereiche und macht sichtbar, welcher Kontext durch welche Vorgaben gesteuert wird.

7.9.2 Prompts und Fähigkeiten nicht verwechseln

Prompts sind der Grundrahmen eines Chatkontexts. Sie sind von Anfang an im Systemkontext und sollen Rolle, Haltung, Grenzen und allgemeine Regeln festlegen. Deshalb müssen sie kurz genug bleiben, um in vielen Anfragen wiederverwendet werden zu können.

Fähigkeiten arbeiten anders: Ephraim nennt dem Modell zunächst nur Name und Kurzbeschreibung. Die vollständige Anleitung wird erst geladen, wenn das Modell diese Fähigkeit wirklich braucht. Dadurch bleiben lange Spezialregeln aus dem normalen Prompt heraus.

Wenn du änderst …	Dann betrifft das …
einen Prompt	den Grundton und die allgemeinen Regeln eines Chat- oder Projektkontexts.
eine Fähigkeit	ein Spezialverhalten, sobald Ephraim diese Fähigkeit im konkreten Fall lädt.

7.9.3 Rollen- und Kontextslots

Slot	Wofür er wirkt
Schülerchat	Normale persönliche Chats von Schülerinnen und Schülern.
Lehrkraftchat	Normale persönliche Chats von Lehrkräften und administrativen Rollen.
Projektbuilder	Unterstützung beim Entwerfen von Projektaufträgen und Materialien.
Projektchat	Antwortverhalten innerhalb freigegebener Projekte.

7.9.4 Technische Prompts

Einige Prompts sind technische Verträge. Sie beschreiben zum Beispiel, welche JSON-Formate für Reparaturen fehlerhafter Visualisierungsblöcke gelten oder wie Auditberichte zusammenzufassen sind. Solche Prompts dürfen nicht nur stilistisch betrachtet werden; sie müssen zur Validierung im System

passen.

Die ausführlichen Visualisierungsregeln für neue Antworten liegen dagegen als Fähigkeit vor. So kann Ephraim genaue Regeln für Diagramme, Skizzen, chemische Strukturformeln und Plots bereitstellen, ohne jeden Chat dauerhaft mit diesen Detailanweisungen zu belasten.

7.9.5 Rechte und Schutz

Administratoren können bestehende Slots pflegen. Besonders systemnahe technische Prompts sind stärker geschützt; Erstellen oder Löschen bestimmter systemischer Einträge ist Super-Admins vorbehalten. Das verhindert, dass zentrale KI-Verträge versehentlich beschädigt werden.

7.9.6 Was Prompts nicht leisten

Ein Prompt gibt keine Datenrechte. Wenn ein Nutzer keinen Zugriff auf eine Datei, ein Projekt oder eine Quelle hat, kann ein Prompt diesen Zugriff nicht herbeischreiben. Dasselbe gilt für Fähigkeiten. Umgekehrt ersetzen Prompts und Fähigkeiten auch keine technische Validierung: Visualisierungs-JSON, Uploads und Rechte werden serverseitig geprüft.

7.9.7 Änderungen verantwortungsvoll testen

Promptänderungen beeinflussen viele Antworten. Nach Änderungen werden typische Chats, Projektfälle, Visualisierungen und mehrsprachige Antworten geprüft. Besonders riskant sind unklare Rollenformulierungen, zu breite Freigaben, widersprüchliche Ausgabeformate oder Texte, die das Modell zu nicht vorhandenen Funktionen auffordern.

Prompts formen den Grundrahmen. Fähigkeiten liefern Spezialanweisungen auf Abruf. Berechtigungen, Verschlüsselung und Datenflüsse bleiben technische Regeln des Systems.

Quelle: web/manuals/admin-prompts/index.html

7.10 Fähigkeiten

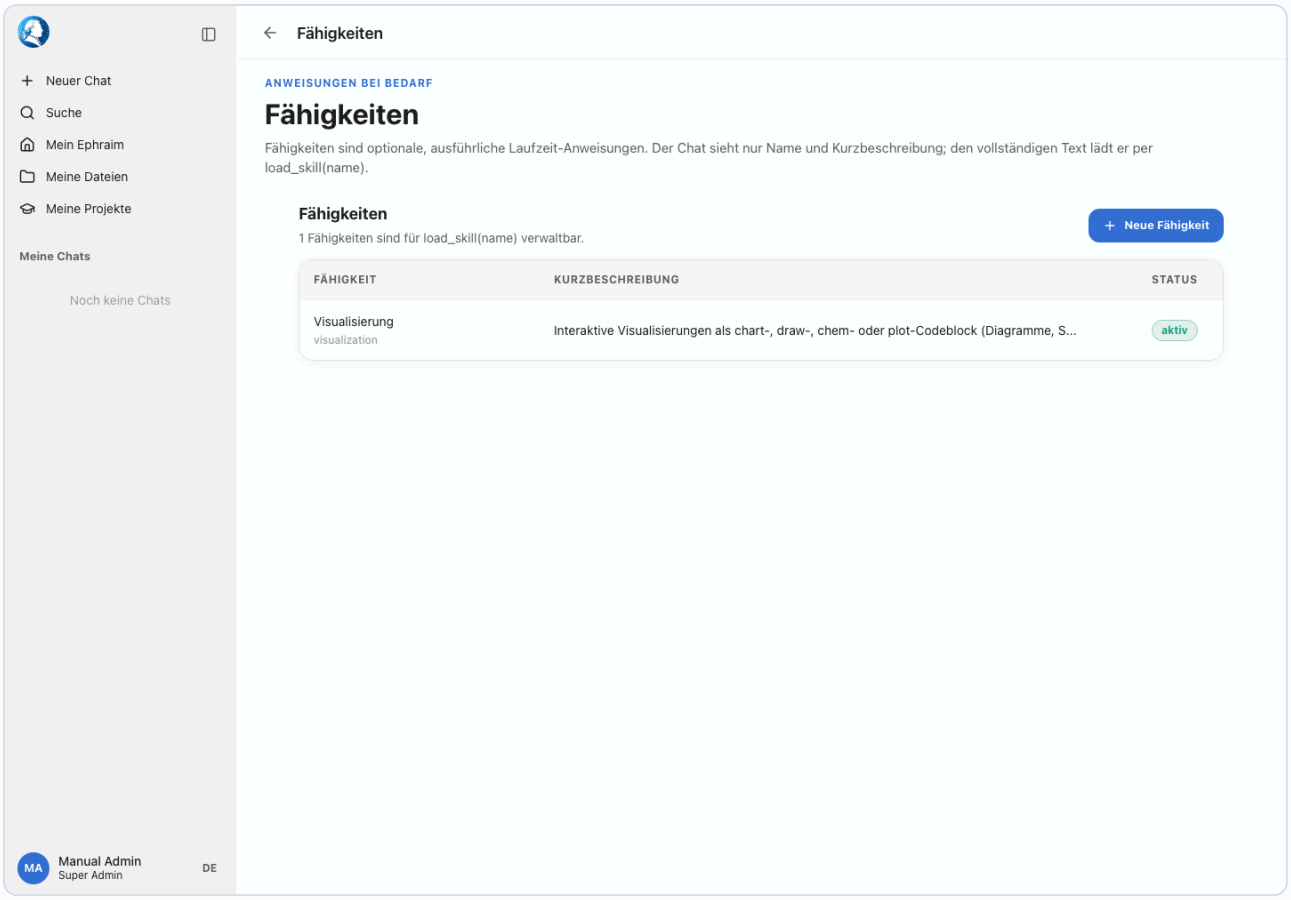
> **Fähigkeiten sind ausführliche KI-Anleitungen für Spezialfälle. Ephraim hält sie aus dem normalen Prompt heraus: Im Chat kennt das Modell zunächst nur Name und Kurzbeschreibung. Die vollständige Anleitung wird erst geladen, wenn sie für eine Antwort gebraucht wird.**

Stand: Juni 2026

7.10.1 Was eine Fähigkeit ist

Eine Fähigkeit beschreibt Spezialverhalten. Sie kann ausführlicher sein als ein normaler Prompt, weil sie nicht immer mitgesendet wird. Ephraim nennt dem Modell im Chat nur, welche Fähigkeiten verfügbar sind und wofür sie gedacht sind. Erst wenn die Antwort diese Fähigkeit braucht, lädt das Modell die vollständige Anleitung über das interne Werkzeug `load_skill`.

> **Kernpunkt:** Eine Fähigkeit ist kein neuer Datenzugriff. Sie ist eine Detailanweisung für ein bestimmtes Antwortverhalten. Rechte, Quellen und Werkzeuge bleiben technische Regeln des Systems.



Die Fähigkeiten-Verwaltung zeigt Name, Kurzbeschreibung und Aktivstatus. Der vollständige Inhalt wird erst beim Bearbeiten oder im Chat bei Bedarf geladen.

7.10.2 Unterschied zu Prompts

Bereich	Wirkung	Typische Änderung
Prompt	Grundrahmen eines Chatkontexts: Rolle, Haltung, Grenzen und allgemeine Regeln.	Ton, Sicherheitsrahmen oder Verhalten eines ganzen Kontextes ändern.
Fähigkeit	Spezialanleitung auf Abruf, zum Beispiel für ein bestimmtes Ausgabeformat.	Detailregeln für einen Spezialfall präzisieren, ohne jeden Chatprompt zu vergrößern.

Prompts sind der Grundrahmen und sollten stabil bleiben. Fähigkeiten sind dagegen Detailwissen für den Moment, in dem es gebraucht wird. Dadurch muss Ephraim lange Formatregeln nicht in jeder normalen Antwort mitführen.

7.10.3 Beispiel: Visualisierung

Die wichtigste aktuelle Fähigkeit ist die Visualisierung. Sie beschreibt, wie Ephraim sichere Blöcke für Diagramme, Zeichnungen, chemische Strukturformeln und mathematische Plots erzeugt. Der normale Chat bekommt nur den Hinweis, dass diese Fähigkeit existiert. Bei einer passenden Anfrage lädt Ephraim die ausführlichen Regeln nach.

Format	Wofür es gedacht ist
`chart`	Balken, Linien, Flächen, Kreise und Streudiagramme aus strukturierten Daten.
`draw`	Abläufe, Mindmaps, Zeitstrahlen, Matrizen und einfache strukturierte Skizzen.
`chem`	Chemische 2D-Strukturformeln aus geprüften SMILES-Angaben.
`plot`	Mathematische Graphen, Punkte, Parameterkurven und einfache 3D-Flächen.

7.10.4 Verwaltung

Die Fähigkeiten-Verwaltung zeigt Name, Kurzbeschreibung und Aktivstatus. Der Name ist der interne Aufrufname für `load_skill`. Die Kurzbeschreibung ist der knappe Hinweis, den das Modell im normalen Systemkontext sieht. Der vollständige Anleitungstext bleibt bis zum Abruf der Fähigkeit verborgen.

Nur Super-Admins bearbeiten, löschen oder neu anlegen Fähigkeiten. Das ist bewusst restriktiv, weil eine Fähigkeit zwar kein eigenes Recht vergibt, aber Spezialantworten stark prägen kann. Eine deaktivierte Fähigkeit wird dem Modell nicht mehr angeboten.

7.10.5 Grenzen

Fähigkeiten ersetzen keine serverseitige Prüfung. Wenn eine Fähigkeit ein Ausgabeformat beschreibt, prüft Ephraim das Ergebnis trotzdem im Browser oder auf dem Server, bevor es als Visualisierung angezeigt wird. Wenn eine Fähigkeit eine Quelle erwähnt, entsteht daraus kein Zugriff auf diese Quelle. Zugriff folgt aus Rolle, Kontext, Freigabe und technischer Tool-Erlaubnis, nicht aus dem Text der Fähigkeit.

Prompts setzen den Grundrahmen. Fähigkeiten liefern lange Spezialanweisungen erst dann, wenn Ephraim sie für eine konkrete Antwort braucht.

Quelle: web/manuals/admin-skills/index.html

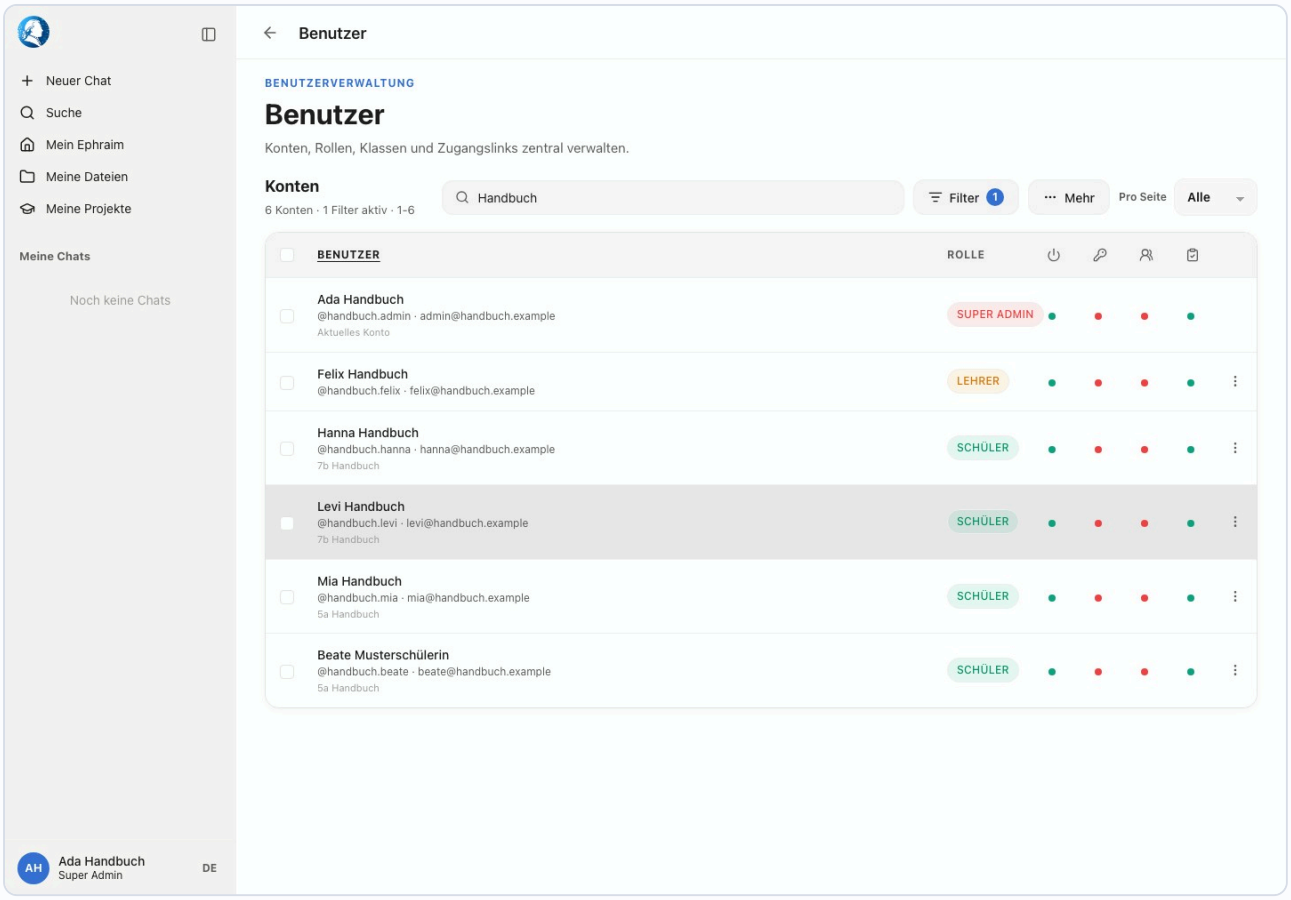
7.11 Benutzer

> **Die Benutzerverwaltung steuert Konten, Rollen, Klassenbezug, Zugangslinks, CSV-Importe, Passwort-Resets und 2FA-Pflichten. Sie ist der zentrale Ort für Identität und Berechtigung in Ephraim.**

Stand: Mai 2026

7.11.1 Rollenmodell

Ephraim unterscheidet Schüler, Lehrkräfte, Administratoren und Super-Admins. Diese Rollen bestimmen, welche Bereiche sichtbar sind und welche Aktionen erlaubt sind. Rollen sind deshalb keine bloßen Anzeigenamen, sondern Sicherheitsentscheidungen.



Die Benutzerverwaltung verbindet Identität, Rolle, Klasse, Sicherheitsstatus und administrative Aktionen in einer Arbeitsansicht.

Rolle	Typischer Zugriff
Schüler	Persönlicher Chat, Dateien, freigegebene Projekte und eigene Einstellungen.
Lehrkraft	Eigene Chats, Projektverwaltung und unterrichtsbezogene Arbeitsräume.
Admin	Verwaltung von Konten, Klassen, Prompts, Speicher und Sicherheitsfunktionen.
Super-Admin	Zusätzliche System- und Runtimefunktionen mit besonders hohem Risiko.

7.11.2 Konten anlegen

Konten können einzeln oder per CSV angelegt werden. Statt Klartextpasswörter dauerhaft zu speichern, arbeitet Ephraim mit initialen Zugangsschritten und Einmallinks. Nach der Einrichtung verfügt das Konto über eigene Anmeldedaten und, bei normalem Konto, einen persönlichen Vault.

Für große Lerngruppen ist der CSV-Import der zuverlässigere Weg, weil Namen, Rollen und Klassen strukturiert übernommen werden. Nach dem Import werden Rolle, Schreibweise und Klassenzuordnung in Stichproben geprüft.

7.11.3 Zugangslinks, PDFs und E-Mail

Die Benutzerverwaltung kann Zugangsinformationen als Liste, PDF oder per E-Mail bereitstellen. Sensible Initialinformationen werden nur im vorgesehenen Moment angezeigt oder versendet. Danach ist der Link kein allgemein wiederanzeigbares Passwortarchiv.

7.11.4 Passwort-Reset und Vault-Folgen

Ein normaler Passwortwechsel durch die Nutzerin oder den Nutzer verpackt den persönlichen Vault-Schlüssel neu und erhält private Inhalte. Ein administrativer Reset ist anders: Er deaktiviert das alte Passwort. Wenn die alte User-DEK nicht mehr mit dem alten Passwort entsperrt werden kann, werden alte private Vault-Inhalte beim Einlösen des Resetwegs nicht erhalten.

> **Administrativer Hinweis:** Ein Reset ist ein Sicherheitswerkzeug, kein komfortabler Passwortwechsel. Bei planbaren Änderungen verwenden Nutzer den normalen Passwortwechsel.

7.11.5 2FA verwalten

Administratoren können Zwei-Faktor-Authentifizierung verpflichtend machen oder zurücksetzen. Das ist besonders für Admin- und Lehrkraftkonten relevant. 2FA-Geheimnisse werden nicht im Klartext angezeigt; ein Reset entfernt den alten Faktor und zwingt zur Neueinrichtung.

7.11.6 Schutz gegen Selbstabschaffung

Ephraim schützt zentrale Rollen vor unbeabsichtigtem Verlust. Der letzte Super-Admin kann sich nicht einfach selbst löschen oder in eine unbrauchbare Verwaltungslage bringen. Solche Schutzregeln sind wichtig, weil Wiederherstellung ohne funktionsfähigen Adminzugang deutlich riskanter wäre.

Benutzerverwaltung ist Sicherheitsverwaltung. Rollen- und Resetänderungen erfolgen nachvollziehbar.

Quelle: web/manuals/admin-benutzer/index.html

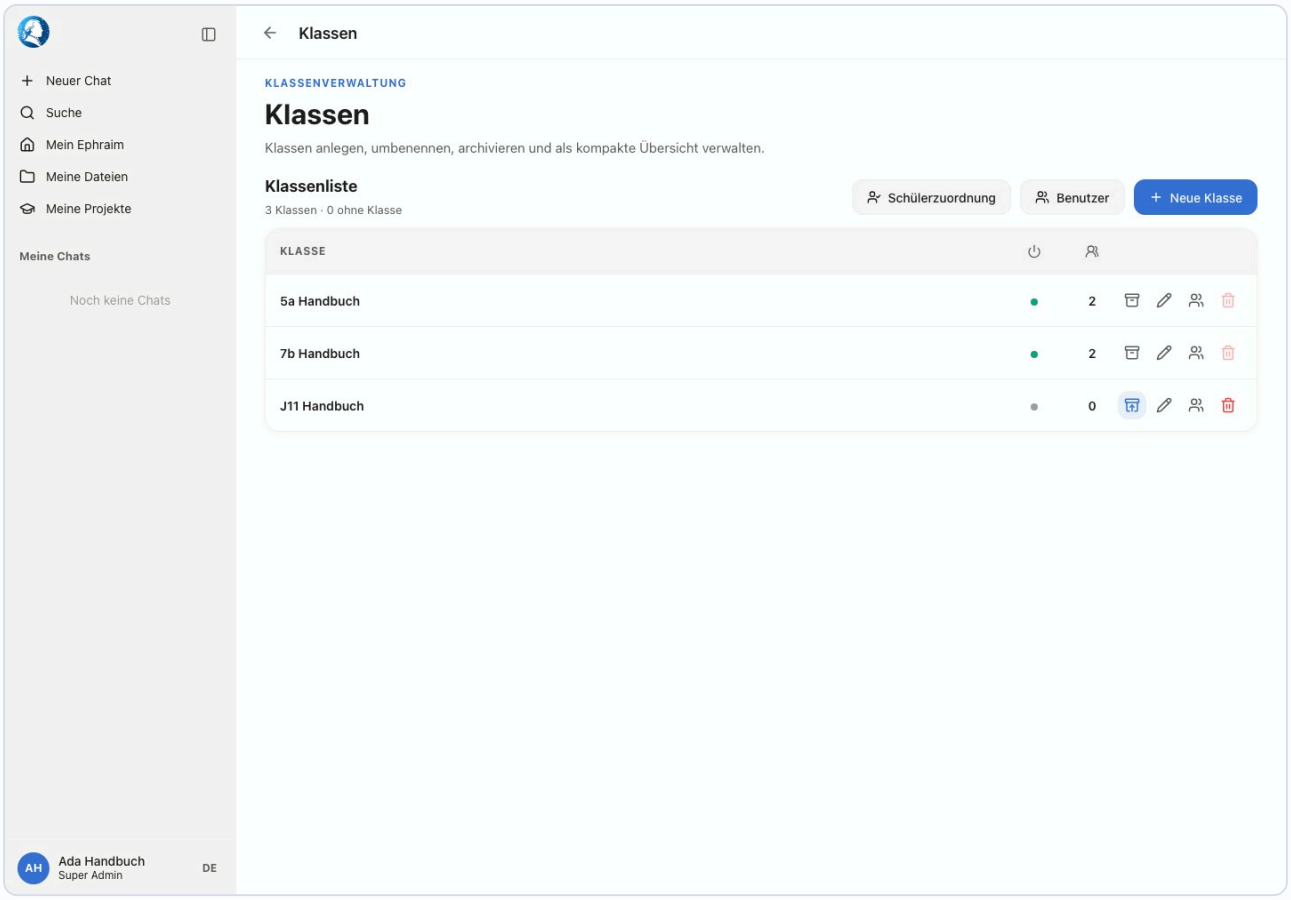
7.12 Klassen

> Die Klassenverwaltung ordnet Schülerinnen und Schüler organisatorisch. Sie unterstützt Standardklassen, Archivierung, Wiederherstellung und die Zuordnung von Konten zu Klassen.

Stand: Mai 2026

7.12.1 Zweck der Klassenverwaltung

Klassen erleichtern Import, Filterung, Projektfreigaben und administrative Übersicht. Sie sind eine organisatorische Zuordnung für Schülerkonten. Lehrkräfte und Admins werden nicht über Klassen verwaltet, weil ihre Berechtigungen rollenbasiert sind.



Die Klassenverwaltung zeigt Klassen, Schülerzahlen und Aktionen für Zuordnung, Archivierung und Wiederherstellung.

7.12.2 Standardklassen

Ephraim kann typische Klassen wie Unter- und Mittelstufenklassen sowie Jahrgangsstufen bereitstellen. Diese Standardstruktur beschleunigt den Schuljahresstart. Administratoren können Klassen ergänzen, umbenennen, archivieren oder wiederherstellen, sofern keine Schutzregel dagegen spricht.

7.12.3 Schüler zuordnen

Die wichtigste Alltagsfunktion ist die Zuordnung von Schülerkonten zu Klassen. Das kann beim CSV-Import, über Filter in der Benutzerverwaltung oder über Sammelaktionen geschehen. Eine saubere Zuordnung erleichtert später Projektfreigaben und administrative Nacharbeit.

7.12.4 Archivieren, Wiederherstellen, Löschen

Archivieren blendet eine Klasse aus aktiven Arbeitslisten aus, erhält aber historische Zuordnungen. Wiederherstellen macht sie wieder aktiv. Löschen ist enger begrenzt und ist für leere oder irrtümlich angelegte Klassen vorgesehen.

7.12.5 Arbeitsbereich ohne Klasse

Konten ohne Klassenzuordnung gehören in die regelmäßige Prüfung. Sie entstehen durch Importfehler, Umzüge, neue Zugänge oder bewusst klassenlose Situationen. Die Verwaltung hilft, diese Konten zu finden und gezielt einzuordnen.

Klassen sind Organisationsdaten. Sie bestimmen nicht allein, was ein Nutzer darf; das bleibt rollen- und projektabhängig.

Quelle: web/manuals/admin-klassen/index.html

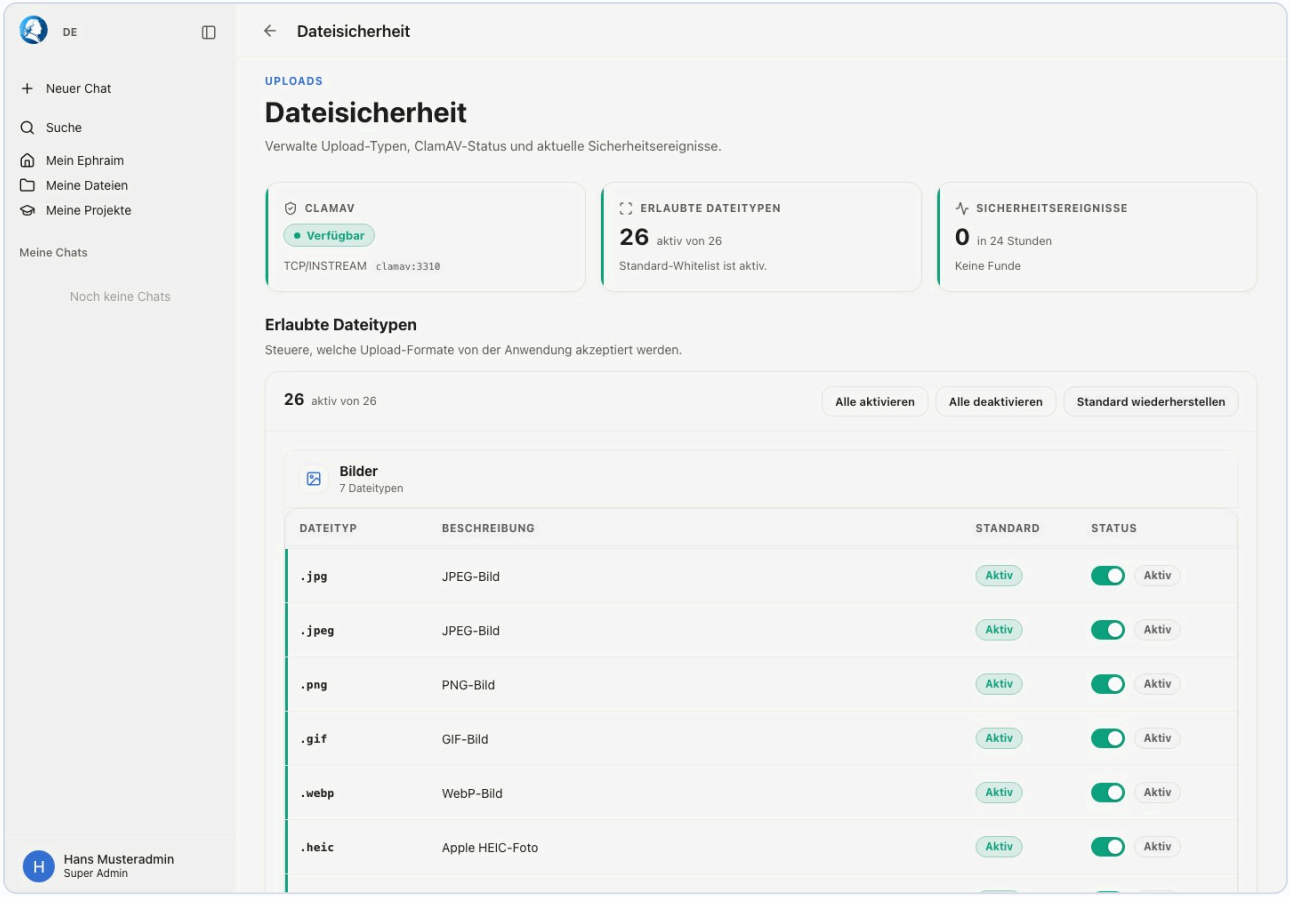
7.13 Datei-Sicherheit

> Datei-Sicherheit regelt, welche Uploads Ephraim akzeptiert, wie Dateitypen geprüft werden, wann ClamAV eingebunden ist und wie Administratoren riskante Formate sperren.

Stand: Mai 2026

7.13.1 Ziel der Uploadprüfung

Uploads sind ein direkter Eingangskanal in das System. Ephraim akzeptiert deshalb nicht blind alles, was ein Browser sendet. Die Prüfung schützt Nutzer, Webserver, Projektbereiche und die spätere Verarbeitung durch Textextraktion oder Vorschau.



Die Datei-Sicherheitsseite macht Uploadregeln, Scanstatus und verwaltete Dateitypen für den Betrieb sichtbar.

7.13.2 Mehrstufige Prüfungen

Prüfung	Warum sie wichtig ist
Dateiname und Endung	Offensichtlich riskante oder irreführende Namen früh erkennen.
Größe	Speicherquoten und Verarbeitungsgrenzen schützen.
MIME-Typ und Magic Bytes	Erkennen, ob der Inhalt zur behaupteten Datei passt.
Office-Struktur	Containerformate prüfen, statt nur auf die Endung zu vertrauen.
Virenscan	Falls ClamAV verfügbar ist, bekannte Malware erkennen.

7.13.3 Adminsteuerung

Die Adminseite zeigt erlaubte Dateitypen, deaktivierte Typen und den Zustand des Scanpfads. Administratoren können Formate sperren, wenn sie im schulischen Betrieb nicht benötigt werden oder ein erhöhtes Risiko darstellen. Systemseitig notwendige Grenzen bleiben trotzdem maßgeblich.

7.13.4 ClamAV

ClamAV ist optional eingebunden. Wenn der Dienst verfügbar ist, erweitert er die Prüfung um Signaturerkennung. Wenn er nicht verfügbar ist, ersetzen die anderen Prüfungen keinen Virenscanner vollständig; der Status bleibt deshalb sichtbar und wird im Betrieb ernst genommen.

7.13.5 Sicherheitsereignisse

Abgelehnte oder auffällige Uploads können als Sicherheitsereignisse protokolliert werden. Diese Ereignisse helfen, Fehlkonfigurationen, Missbrauch oder riskante Dateitypen zu erkennen. Sie sind technische Schutzdaten, keine inhaltliche Bewertung von Schülerarbeiten.

Datei-Sicherheit ist Verteidigung in Schichten. Keine Einzelprüfung ist allein ausreichend.

Quelle: web/manuals/admin-datei-sicherheit/index.html

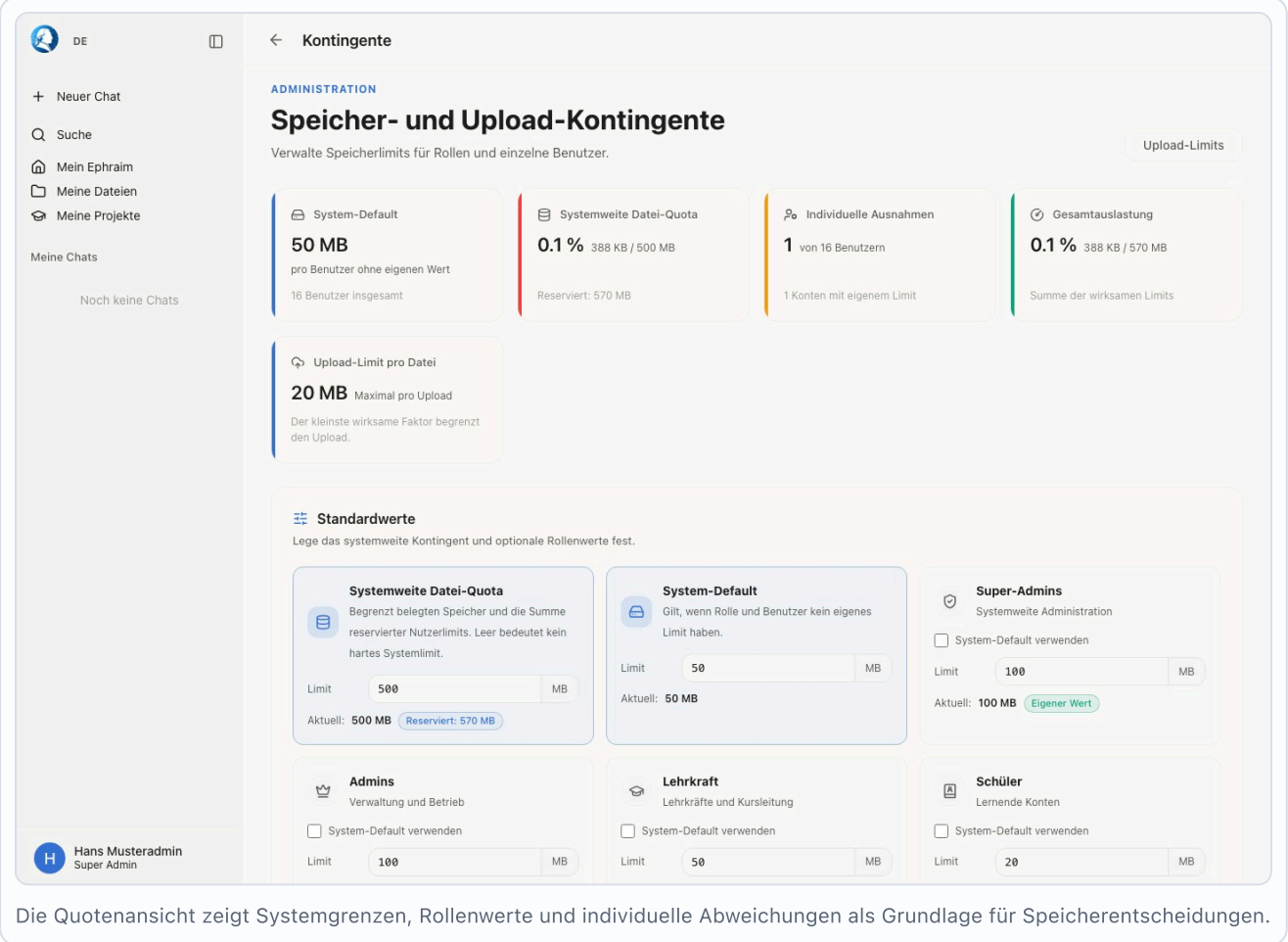
7.14 Speicherquoten

> Speicherquoten begrenzen persönlichen und projektbezogenen Dateispeicher. Sie schützen den Webserver vor unkontrolliertem Wachstum und machen Speichernutzung administrierbar.

Stand: Mai 2026

7.14.1 Quotenhierarchie

Ephraim wertet Quoten hierarchisch aus. Eine individuelle Benutzerquote kann die Rollenvorgabe übersteuern. Wenn keine individuelle Quote gesetzt ist, gilt die Rolle. Wenn auch dort keine spezielle Regel greift, gilt der Systemstandard.



Die Quotenansicht zeigt Systemgrenzen, Rollenwerte und individuelle Abweichungen als Grundlage für Speicherentscheidungen.

1. Benutzer-Override

2. Rollenstandard
3. Systemstandard

7.14.2 Was gezählt wird

Gezählt werden persönliche Dateien und aktive projektbezogene Dateien nach den Systemregeln. Temporäre Exportdateien sollen nicht dauerhaft Quote verbrauchen, werden aber während ihrer Erstellung technisch Speicher benötigen. Uploadgrößenlimit und Gesamtspeicherquote sind unterschiedliche Grenzen.

7.14.3 Uploadgröße und Gesamtspeicher

Die maximale Uploadgröße verhindert einzelne zu große Dateien. Die Gesamtspeicherquote begrenzt dagegen den langfristig belegten Speicher einer Person oder Rolle. Eine Datei kann also wegen Größe abgelehnt werden, obwohl noch Gesamtquote frei wäre, oder wegen Gesamtquote, obwohl sie einzeln klein genug ist.

7.14.4 Adminfunktionen

Die Adminseite zeigt Kennzahlen, Standards und individuelle Abweichungen. Administratoren können Rollenlimits anpassen und für einzelne Konten Overrides setzen, etwa bei Projekten mit großen Materialien oder bei Konten, die bewusst weniger Speicher erhalten sollen.

7.14.5 Betriebspraxis

Quoten brauchen einen nutzbaren Arbeitsbereich und eine harte Obergrenze. Zu enge Quoten blockieren normale Unterrichtsarbeit; zu weite Quoten erlauben einem einzelnen Konto, den Server zu füllen. Der robuste Betrieb nutzt eine klare Rollenbasis plus wenige begründete Einzel-Overrides.

Speicherquoten schützen Verfügbarkeit. Sie sind kein Ersatz für Uploadprüfung oder regelmäßige Bereinigung.

Quelle: web/manuals/admin-speicherquoten/index.html

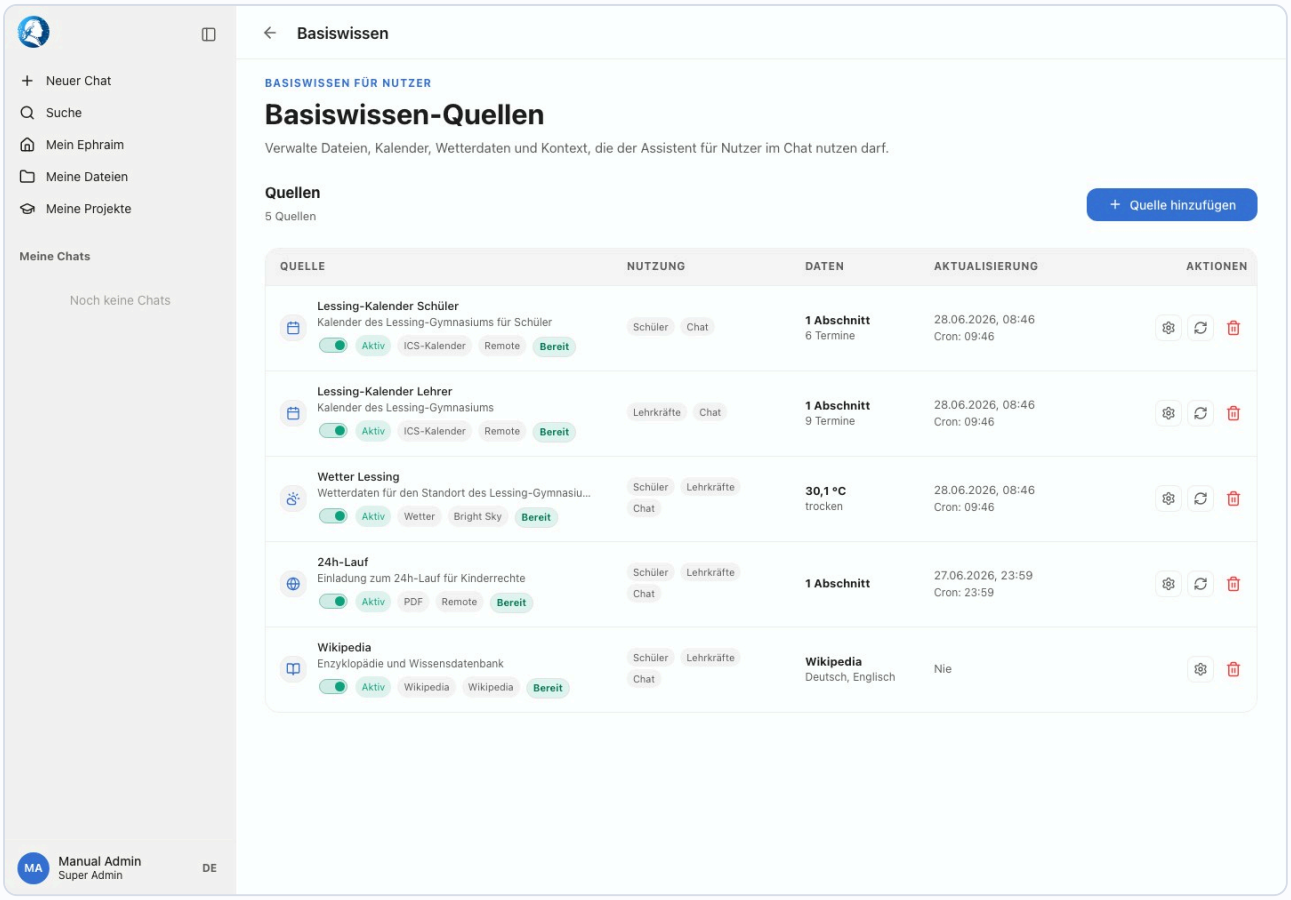
7.15 Basiswissen

> **Basiswissen macht freigegebene Quellen für Chats und Projekte nutzbar. Administratoren verwalten Quellen, Sichtbarkeit, Aktualisierung und Retrieval, ohne der Spark freien Internetzugriff zu geben.**

Stand: Juni 2026

7.15.1 Was Basiswissen ist

Basiswissen ist kuratierter Kontext, den Ephraim bei passenden Anfragen ergänzt. Es besteht aus zentralen Texten, Dateien, freigegebenen URLs, Kalenderquellen, Wetterdaten oder Wikipedia-Auszügen. Die Quelle wird nicht direkt von der KI besucht: Der Webserver ruft freigegebene Quellen kontrolliert ab, prüft sie und stellt nur passende Ausschnitte bereit.

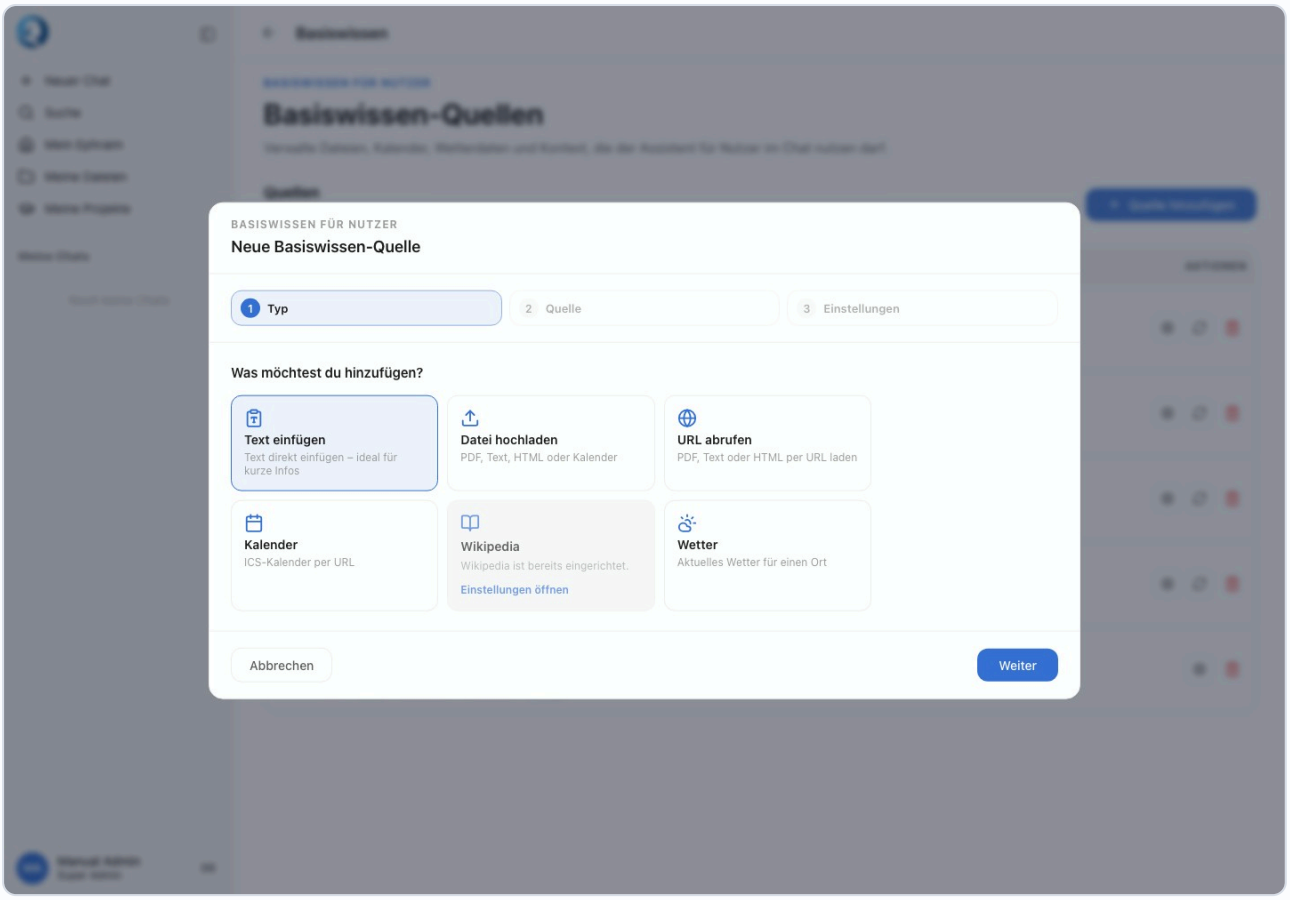


Die Basiswissen-Seite zeigt zentrale Quellen, Nutzung, Datenstatus, Aktualisierung und Aktionen. Freigegebener Kontext wird hier verwaltet, nicht durch freie Websuche der Spark.

7.15.2 Quellen anlegen

Neue Quellen werden über einen geführten Assistenten angelegt. Die Schritte sind bewusst getrennt: zuerst Art der Quelle, dann Datei, URL oder Providerangaben, danach Kontext, Sichtbarkeit und Aktualisierung, zuletzt eine Prüfung der Angaben. Gespeichert wird erst am Ende.

Der Assistent zeigt nur Felder, die zur gewählten Quelle passen. Ein Datei-Upload braucht keine Cron-Einstellung, eine Wikipedia-Quelle keine Upload-Datei und eine Wetterquelle Standortdaten statt Textinhalt. Dadurch landen irrelevante Einstellungen nicht versehentlich in der Quelle.



Der Assistent führt zuerst durch die Art der Quelle. Danach zeigt Ephraim nur die Felder, die zu dieser Quelle passen.

7.15.3 Quellentypen

Quelle	Was sie liefert	Besonderheit
Text	Schulische Informationen, Unterrichtsmaterial oder statische Hinweise.	Direkt eingefügter Text wird wie eine Basiswissen-Quelle verarbeitet.
Datei	PDF, Text, HTML oder ICS aus einem Upload.	Die Datei wird geprüft, begrenzt, verarbeitet und verschlüsselt gespeichert.
URL	PDF, Text oder HTML von einer freigegebenen Adresse.	Der Webserver ruft die Quelle kontrolliert ab; die KI surft nicht selbst.
ICS-Kalender	Termine, Stundenpläne oder Ereignisse als strukturierter Kontext.	Das Kalenderfenster legt fest, wie weit zurück und voraus indexiert wird.
Wetter	Standortbezogene Wetterdaten für schulische oder organisatorische Kontexte.	Bright Sky wird serverseitig abgefragt; frisches Wetter kann auch in der Begrüßung erscheinen.
Wikipedia	Kurze Auszüge aus deutscher oder englischer Wikipedia bei passenden Allgemeinwissensfragen.	Wikipedia ist ein dynamischer Provider, keine Sammlung fest hinterlegter Artikel.

7.15.4 Sichtbarkeit und Nutzung

Jede zentrale Quelle hat zwei Ebenen von Freigaben. Die Sichtbarkeit legt fest, ob Schülerinnen und Schüler, Lehrkräfte oder beide Gruppen die Quelle nutzen dürfen. Die Nutzung legt fest, ob die Quelle im Chat und zusätzlich im Projektbuilder verfügbar ist.

Diese Trennung ist wichtig: Ein interner Lehrkrafthinweis gehört nicht automatisch in Schülerantworten, und nicht jede Chatquelle ist sinnvoll für den Entwurf neuer Projekte. Der Aktiv-Schalter ist davon nochmals getrennt. Eine technisch bereite Quelle kann administrativ deaktiviert bleiben.

7.15.5 Status und Aktualisierung

Die Quellenliste unterscheidet fachliche Freigabe und technischen Zustand. Eine Quelle kann bereit, deaktiviert, im Refresh, fehlgeschlagen oder noch Entwurf sein. Remote-, Kalender- und Wetterquellen können automatisch aktualisiert werden; Uploads und eingefügte Texte bleiben statische Quellen, bis sie bewusst geändert werden.

Wenn bei einer bestehenden Kalenderquelle das Kalenderfenster geändert wird, verarbeitet Ephraim die Quelle neu. Erst nach erfolgreicher Verarbeitung gelten die neuen Werte. Bei Fehlern bleiben die bisherigen Kalenderdaten nutzbar.

7.15.6 Kein Internet für die Spark

Basiswissen ändert nicht das Grundprinzip der lokalen Inferenz. Die Spark führt das Modell aus und berechnet Antworten. Wenn eine URL, Wikipedia oder Wetterdaten benötigt werden, ruft der Webserver diese Quellen nach den konfigurierten Regeln ab. Die Spark erhält nur den relevanten Kontext, nicht die Freiheit, selbst im Internet zu suchen.

Für die Vektorerzeugung ruft Ephraim den lokalen SGLang-Embedding-Dienst `school-ui-embedding` auf. Die Spark berechnet aus freigegebenen Klartext-Chunks Zahlenvektoren, speichert diese Klartexte und Vektoren aber nicht. Persistiert wird nur die verschlüsselte Knowledge-Ablage auf dem Webserver.

7.15.7 Zentral vs. privat

Zentrales Basiswissen ist administrativ verwaltet. Private Kalenderabos einzelner Nutzer sind davon getrennt und gehören zur Personalisierung. Diese Trennung verhindert, dass ein privater Stundenplan plötzlich zur zentralen Wissensquelle wird.

Private Kalenderquellen werden nur für das jeweilige Konto im normalen Chat und in der persönlichen Startseitenbegrüßung genutzt. Sie werden nicht im Projektbuilder und nicht für fremde Konten verwendet. Zentrale Kalenderquellen bleiben dagegen Teil des administrativ freigegebenen Basiswissens.

Basiswissen ist kuratierter Kontext. Es ist keine freie Websuche der KI.

Quelle: web/manuals/admin-basiswissen/index.html

7.16 Ephraim MCP-Server

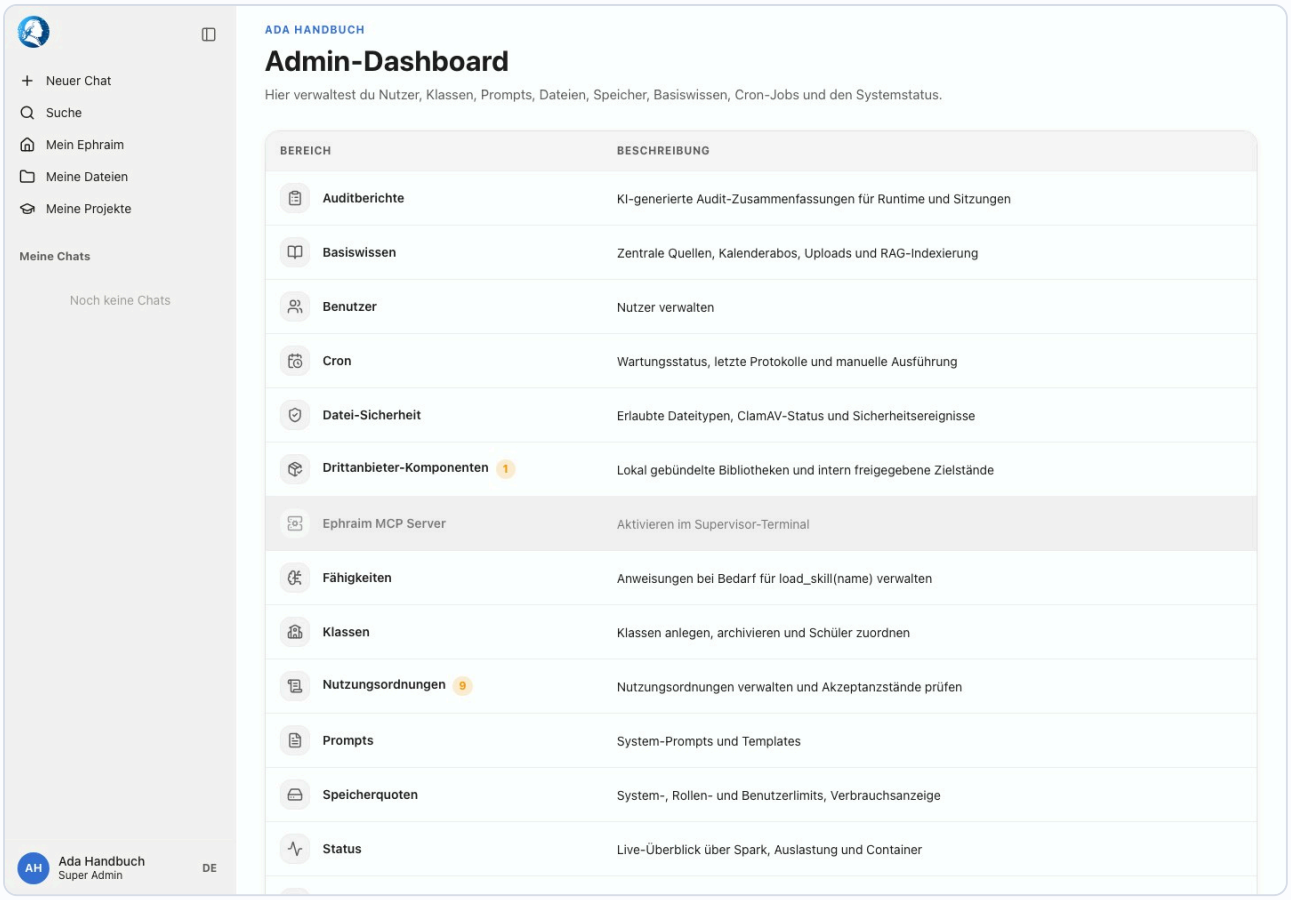
> **Der Ephraim MCP-Server stellt ausgewählte Adminwerkzeuge für externe Coding- und Assistenzclients bereit. Er ist standardmäßig deaktiviert und nur für bewusst autorisierte Admins gedacht.**

Stand: Mai 2026

7.16.1 Zweck

MCP erlaubt Werkzeugzugriffe aus spezialisierten Clients. Für Ephraim bedeutet das: Ein berechtigter Admin kann bestimmte Verwaltungs-, Runtime- oder Auditfunktionen in einem externen Werkzeug verwenden, ohne direkt in der Weboberfläche zu klicken. Weil diese Fähigkeit mächtig ist, ist sie im Standardbetrieb abgeschaltet.

Sinnvoll ist MCP nur für bewusst administrierte Arbeitsabläufe: technische Diagnose, wiederholbare Adminauswertungen, kontrollierte Auditabfragen oder dokumentierte Coding-Assistenz. Für normale Nutzerarbeit, Unterrichtschats oder spontane Komfortaktionen ist MCP der falsche Weg.



Im Admin-Dashboard bleibt der MCP-Eintrag sichtbar, ist aber deaktiviert, solange der Server nicht bewusst über die Betriebswerkzeuge freigeschaltet wurde.

7.16.2 OAuth mit PKCE

Der Zugriff läuft über OAuth mit PKCE. Clients erhalten keine statischen Passwörter. Tokens werden serverseitig nur gehasht gespeichert und laufen nach begrenzter Zeit ab. Ein Client muss explizit autorisiert werden und erhält nur die Scopes, die freigegeben sind.

7.16.3 Scopes

Scope-Gruppe	Bedeutung
Admin lesen/schreiben	Verwaltungsdaten ansehen oder ändern, je nach Freigabe.
Runtime lesen	Status und technische Betriebsdaten abfragen.
Supervisor Update	Besonders kontrollierte Betriebsaktionen.
Audit lesen/schreiben	Auditberichte abfragen oder verwalten.
Super-Admin schreiben	Hochprivilegierte Änderungen, nur mit passender Rolle.

7.16.4 Sicherheitsgrenzen

Der MCP-Server ist kein beliebiger Remote-Shell-Zugang. Werkzeuge sind definierte Funktionen mit Rollenprüfung, Scopeprüfung, CSRF- und Bestätigungslogik, wo nötig. Mutierende Aktionen können eine explizite Bestätigung verlangen. Das schützt vor versehentlichen Änderungen durch Assistenten oder automatisierte Clients.

Das Hauptrisiko liegt nicht im Protokollnamen, sondern in der Kombination aus Adminrechten und automatisierten Clients. Deshalb gelten knappe Scopes, kurze Tokenlaufzeiten, Widerruf nicht mehr benötigter Clients und eine saubere Auditspur.

7.16.5 Nachvollziehbarkeit

MCP-Aktionen müssen nachvollziehbar bleiben. Bei sicherheitsrelevanten Werkzeugen ist Auditierung wichtiger als Komfort. Administratoren prüfen Clients regelmäßig, widerrufen nicht mehr benötigte Tokens und vergeben Scopes knapp.

MCP ist ein Adminwerkzeug für bewusst autorisierte Clients, nicht der normale Weg für Nutzerinnen und Nutzer.

Quelle: web/manuals/admin-mcp-server/index.html

7.17 Session-Management

> Dieser Artikel erklärt, wie Ephraim angemeldete Sitzungen begrenzt, verlängert, rotiert und beendet. Er richtet sich an Admins und technische Betreuung, die einschätzen müssen, wo Vault-Schlüssel während einer Sitzung liegen, wann eine Anmeldung abläuft und welche Rolle Cron dabei wirklich spielt.

Stand: Juni 2026

7.17.1 Zielbild

> **Kurzfassung:** Ephraim verlängert Sitzungen nur nach echter Nutzeraktivität. Eine Sitzung läuft nach 45 Minuten Inaktivität ab, spätestens aber nach 12 Stunden. Die native Browser-Sitzung wird regelmäßig rotiert und serverseitig nur als HMAC gebunden, nicht im Klartext gespeichert.

Eine Sitzung ist in Ephraim mehr als ein Login-Cookie. Während der Anmeldung hängen daran Rollen, CSRF-Schutz, Vault-Entsperrung, Zwei-Faktor-Zustände, Aktivitätsfristen und die Frage, ob ein alter Browser-Tab noch vertrauenswürdig ist. Deshalb trennt Ephraim mehrere Ebenen: das Browser-Cookie, die verschlüsselte PHP-Session, die Sitzungsübersicht, den aktuell gültigen nativen Session-ID-Hash und die tatsächliche Aktivität im Browser.

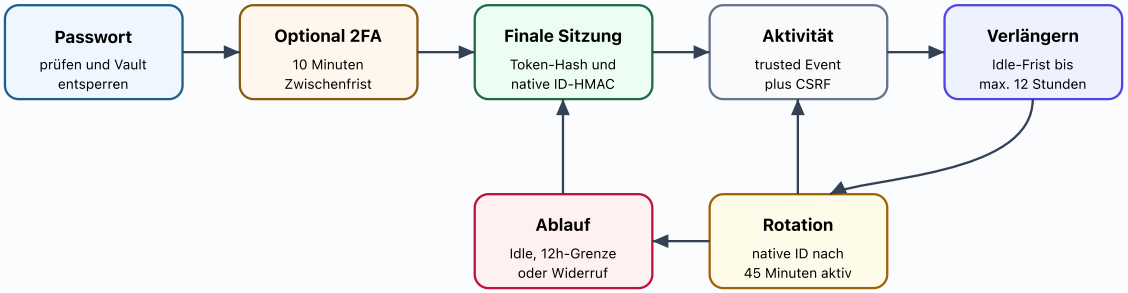
Für Admins ist die wichtigste Konsequenz: Ablauf und Sicherheit hängen nicht vom Gefühl „die Seite ist noch offen“ ab. Eine offene Seite reicht nicht. Ephraim verlangt eine gültige serverseitige Sitzung, eine passende native Session-ID und eine bestätigte Aktivitätsverlängerung innerhalb der Idle-Frist.

7.17.2 Begriffe

Begriff	Bedeutung in Ephraim	Admin-Relevanz
Native PHP-Session-ID	Der technische Bezeichner, den der Browser als Cookie mitschickt.	Wird regelmäßig erneuert und nicht im Klartext in der Datenbank gespeichert.
Logische Sitzung	Der serverseitig registrierte Anmeldezustand eines Nutzers.	Bleibt über eine native Session-ID-Rotation hinweg dieselbe fachliche Anmeldung.
Sessiontoken-Hash	Ein zufälliger Prüfwert für die Sitzungsübersicht.	Erlaubt Widerruf und Anzeige aktiver Sitzungen, ohne den Token selbst zu speichern.
Native Session-ID-HMAC	Serverseitiger HMAC der aktuell gültigen nativen PHP-Session-ID.	Bindet die logische Sitzung an genau die aktuelle Browser-Sitzung.
Idle-Timeout	Ablauf nach 45 Minuten ohne bestätigte Aktivität.	Schützt offene Geräte, vergessene Tabs und verlassene Arbeitsplätze.
Absolute Obergrenze	Spätester Ablauf nach 12 Stunden seit Sitzungsanlage.	Verhindert unbegrenzt gleitende Dauersitzungen.
Aktivitäts-Heartbeat	Ein CSRF-geschützter Hintergrundaufruf nach echter Nutzeraktivität.	Nur dieser Aufruf verlängert die Idle-Frist und löst bei Bedarf Rotation aus.

7.17.3 Lebenszyklus einer Anmeldung

Der Lebenszyklus beginnt mit Passwortprüfung, Vault-Entsperrung und optionaler Zwei-Faktor-Prüfung. Nach erfolgreicher Anmeldung erhält der Browser eine frische native Session-ID. Ephraim registriert die logische Sitzung serverseitig, speichert Hashwerte und setzt die erste Idle-Frist. Danach verlängert nicht jeder Seitenaufruf automatisch die Sitzung; Verlängerung passiert über den Aktivitäts-Heartbeat.



Prüfung und Verlängerung sind getrennt: normale geschützte Requests validieren nur, der Heartbeat verlängert.

Die Sitzung wird nach Anmeldung registriert, über echte Aktivität verlängert und regelmäßig an eine neue native Session-ID gebunden. Ablauf ist eine Prüfentscheidung, kein Cron-Ereignis.

7.17.4 Zeitregeln

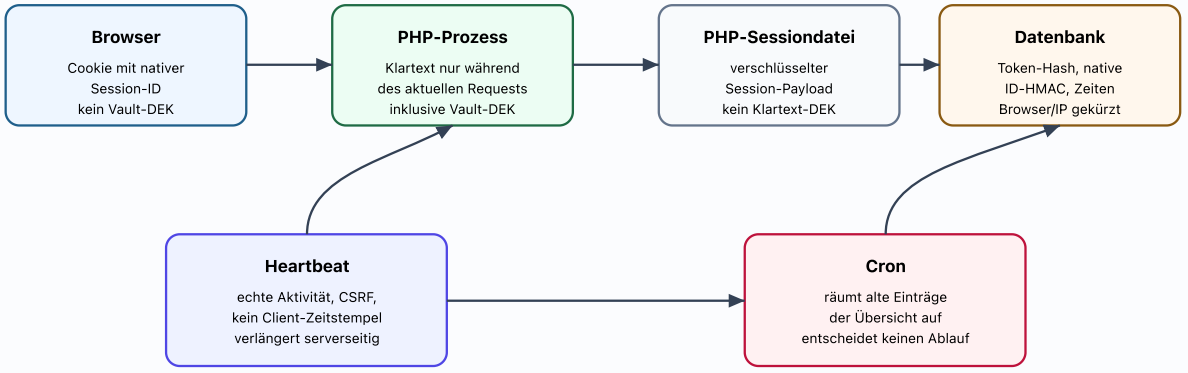
Ephraim nutzt mehrere Fristen, weil jede Frist ein anderes Risiko adressiert. Die Idle-Frist schützt verlassene Geräte. Die absolute Grenze verhindert unbegrenztes Weiterschieben. Die Rotationsfrist begrenzt die Lebensdauer einer nativen Session-ID. Die 2FA-Zwischenfrist verhindert, dass eine halb abgeschlossene Anmeldung mit bereits entsperrtem Vault-Material lange offen bleibt.

Frist	Dauer	Startpunkt	Was passiert beim Erreichen?
Inaktivität	45 Minuten	Letzte bestätigte Nutzeraktivität	Die Sitzung wird bei der nächsten Prüfung abgewiesen; der Nutzer muss sich neu anmelden.
Absolute Obergrenze	12 Stunden	Anlage der logischen Sitzung	Keine weitere Verlängerung; auch aktive Nutzer müssen danach neu anmelden.
Native Session-ID-Rotation	45 Minuten aktiver Nutzung	Letzte native ID-Erneuerung	Der Browser bekommt eine neue native Session-ID; der serverseitige HMAC wird aktualisiert.
2FA-Zwischenzustand	10 Minuten	Erfolgreiche Passwortprüfung bei aktivierter 2FA	Der Zwischenzustand wird verworfen, einschließlich temporärem Vault-Material.

> **Wichtig für Supportfälle:** „Ich war doch noch auf der Seite“ ist nicht dasselbe wie „die Sitzung wurde verlängert“. Verlängert wird erst, wenn der Browser eine echte Aktivität meldet, der Tab sichtbar ist, CSRF passt und die serverseitige Sitzung noch gültig ist.

7.17.5 Was wo gespeichert ist

Die häufigste Admin-Frage betrifft den Vault-Schlüssel: Er darf nicht unverschlüsselt auf der Serverplatte landen. Ephraim trennt deshalb zwischen Browser-Cookie, verschlüsselter Sessiondatei, Datenbank-Metadaten und kurzlebigem RAM während eines Requests.



Alte verschlüsselte Sessiondateien können bis zur Aufräumfrist existieren, werden aber ohne gültige Zeit- und HMAC-Prüfung nicht akzeptiert.

Klartext existiert nur kurz im aktuellen Serverprozess. Persistente Sessiondaten liegen verschlüsselt oder als Hash/HMAC vor; Cron bereinigt später nur Altlasten.

Ort	Typischer Inhalt	Vault-Schlüssel im Klartext?
Browser-Cookie	Native Session-ID, geschützt durch HttpOnly und SameSite.	Nein.
PHP-Prozess während eines Requests	Aus der Session entschlüsselte Nutzer- und Vault-Daten für die aktuelle Verarbeitung.	Ja, kurzzeitig im RAM, solange der Request verarbeitet wird.
PHP-Session-Speicher	Verschlüsselter Session-Payload, darunter Rollen, CSRF und bei entsperrtem Vault auch verschlüsseltes Vault-Material.	Nein, nicht als Klartextdatei.
Sitzungsübersicht	Token-Hash, native Session-ID-HMAC, Zeitpunkte, gekürzte IP-/Browserhinweise.	Nein.
Cron-Aufräumung	Löscht abgelaufene, widerrufene oder veraltete Übersichtseinträge.	Nein; Cron entscheidet nicht über die aktuelle Gültigkeit.

7.17.6 Sicherheitsfragen klar beantwortet

Die folgenden Antworten sind bewusst zweigeteilt. Die Kurzantwort beschreibt, was für Nutzerinnen, Nutzer und Datenschutzgespräche wichtig ist. Die technische Einordnung beschreibt, welche Prüfung oder welcher Speicherort dahintersteht.

Frage	Kurzantwort für Laien	Technische Einordnung
Gibt es während der Sitzung eine Datei mit dem Vault-Schlüssel im Klartext?	Nein. Auf der Serverplatte soll keine Klartextdatei mit dem persönlichen Vault-Schlüssel liegen.	Die PHP-Sessiondatei enthält einen verschlüsselten Session-Payload. Der persönliche DEK wird beim Verarbeiten einer Anfrage im PHP-Prozess entschlüsselt, aber nicht als Klartextdatei persistiert.
Was bedeutet „nur im RAM“ genau?	Der Schlüssel wird während einer konkreten Aktion benötigt, zum Beispiel um eine private Datei oder einen Chat zu entschlüsseln. Danach soll er nicht dauerhaft gespeichert bleiben.	Während eines Requests liegt der entschlüsselte Sessioninhalt im Arbeitsspeicher des PHP-Prozesses. Das schützt nicht gegen einen vollständig kompromittierten Server oder einen Angreifer mit Live-Zugriff auf den Prozessspeicher, reduziert aber das Risiko bei Datenbank- oder Plattenabzügen.
Ist die Datei nach Ende der Sitzung sofort weg?	Beim Logout oder Ablauf wird die Anmeldung beendet. Eine alte technische Sessiondatei kann noch bis zur normalen Aufräumung existieren, ist aber verschlüsselt und nicht mehr als gültige Anmeldung nutzbar.	Die lokale PHP-Session wird verworfen beziehungsweise zerstört. Zusätzlich prüfen Requests Ablaufzeit, Widerruf, Session-Version und native Session-ID-HMAC. Eine alte Datei allein reicht nicht, um die Sitzung wiederzubeleben.
Beginnen die 45 Minuten bei jeder Aktivität neu?	Ja, aber nur bei bestätigter echter Nutzung. Eine nur offene Seite oder reine Hintergrundaktivität reicht nicht.	Der Aktivitäts-Heartbeat akzeptiert nur vertrauenswürdige Nutzerereignisse in sichtbaren Tabs mit gültigem CSRF-Token. Der Server berechnet die neue Ablaufzeit selbst und begrenzt sie auf die absolute 12-Stunden-Grenze.
Warum gibt es zusätzlich eine 12-Stunden-Grenze?	Damit eine Sitzung nicht über Tage immer weiter offen bleiben kann, selbst wenn regelmäßig etwas geklickt wird.	Die Idle-Frist ist gleitend, die absolute Obergrenze nicht. Nach 12 Stunden seit Anlage der logischen Sitzung wird keine Verlängerung mehr akzeptiert.
Regelt Cron den Ablauf?	Nein. Cron räumt nur später auf. Ungültig wird eine Sitzung sofort, sobald der Server sie beim nächsten Zugriff prüft.	Ablauf, Widerruf, Session-Version und native HMAC-Bindung werden synchron in der Auth-Prüfung ausgewertet. Cron entfernt abgelaufene Übersichtseinträge und unterstützt damit Hygiene, nicht die Echtzeitentscheidung.

> **Grenze des Schutzmodells:** Diese Architektur schützt insbesondere gegen Klartextfunde auf Platte, Datenbankkopien und liegengebliebene Sessiondateien. Sie ist kein Schutz gegen einen Angreifer, der den laufenden Webserver, den PHP-Prozess oder privilegierten Arbeitsspeicherzugriff bereits kontrolliert.

7.17.7 Aktivitätsverlängerung

Der sichtbare Ablauftimer unter dem Profilmenü ist ein Hinweis, nicht die Autorität. Die Autorität liegt immer auf dem Server. Der Browser meldet Aktivität nur nach echten Nutzerereignissen wie Klick, Tastatur, Eingabe, Touch, Scroll oder Formularaktion. Reine Mausbewegungen werden nicht verwendet, weil sie zu viele Scheinsignale erzeugen und Sitzungen unnötig offen halten könnten.

Der Heartbeat läuft nicht blind im Hintergrund. Ist der Tab nicht sichtbar, läuft gerade ein anderer getrackter Request, ein Upload oder ein Chat-Stream, wartet der Timer. Dadurch wird eine Session-ID-Rotation nicht mitten in parallele Abläufe gedrückt. Der Heartbeat sendet außerdem keine eigene Uhrzeit, der Server berechnet Ablauf und absolute Grenze selbst.

> **Admin-Merksatz:** Der Timer zeigt an, wann die Sitzung aus Sicht der letzten Serverantwort endet. Verlängert wird erst, wenn der Server einen gültigen, CSRF-geschützten Aktivitätsaufruf akzeptiert.

7.17.8 Rotation und alte Tabs

Nach 45 Minuten aktiver Nutzung erneuert Ephraim die native Session-ID. Die fachliche Sitzung bleibt dieselbe, aber die alte native ID passt danach nicht mehr zum gespeicherten HMAC. Ein alter Tab, der noch mit der alten ID arbeitet, kann die logische Sitzung nicht weiter verlängern. Er verliert seine eigene lokale Anmeldung, widerruft aber nicht automatisch die frische Sitzung eines anderen Tabs.

Diese Trennung ist absichtlich. Ohne sie könnte ein verspäteter Request aus einem alten Tab eine neu rotierte Sitzung versehentlich beschädigen. Ephraim behandelt solche Fälle deshalb als lokale Ablaufentscheidung: Der alte Zugriff wird abgewiesen, die aktuelle gültige Sitzung bleibt über den neuen nativen HMAC gebunden.

Situation	Ergebnis	Warum
Aktiver Tab sendet rechtzeitig Heartbeat	Idle-Frist wird verlängert.	Aktivität, CSRF, Token und native ID passen zusammen.
Rotation ist fällig	Native ID wird erneuert, HMAC wird aktualisiert.	Die Lebensdauer eines technischen Session-Bezeichners bleibt begrenzt.
Alter Tab nutzt alte native ID	Request wird abgewiesen; lokale Sitzung endet.	Der gespeicherte HMAC passt nur noch zur neuen nativen ID.
12-Stunden-Grenze erreicht	Keine Verlängerung mehr möglich.	Die absolute Sitzungsdauer ist bewusst nicht gleitend.

7.17.9 Admin-Aktionen und ihre Folgen

Admins beeinflussen Sitzungen indirekt über Konto- und Sicherheitsaktionen. Besonders wichtig sind Passwort- und Resetvorgänge: Wenn ein Passwort geändert oder zurückgesetzt wird, steigt die Sicherheitsversion des Kontos. Bestehende Sitzungen mit alter Version werden danach nicht mehr akzeptiert. Die aktuelle legitime Sitzung kann auf die neue Version gehoben werden; andere Sitzungen laufen aus oder werden widerrufen.

Aktion	Folge für Sitzungen	Supporthinweis
Normales Logout	Die aktuelle logische Sitzung wird widerrufen und die lokale PHP-Session zerstört.	Das ist ein bewusster Sitzungsabschluss.
Idle- oder absolute Ablaufprüfung	Die lokale Session wird verworfen; der Übersichtseintrag muss nicht sofort gelöscht sein.	Ein späterer Cron-Lauf räumt alte Übersichtseinträge auf.
Passwortwechsel	Andere aktive Sitzungen werden beendet; der Vault-DEK wird mit dem neuen Passwort neu verpackt.	Bei geplantem Wechsel ist dies der bevorzugte Weg gegenüber Admin-Reset.
Admin-Reset	Alte Passwort- und Vault-Zugriffe werden ungültig; neue Anmeldung muss den Resetfluss abschließen.	Das ist ein Eingriff in den Konto-Sicherheitszustand und kein komfortabler Passwortwechsel.
2FA erforderlich oder neu eingerichtet	Zwischenzustände sind kurzlebig; finale Sitzung entsteht erst nach erfolgreicher Prüfung.	Abgelaufene 2FA-Seiten verlangen eine neue Anmeldung.
Andere Sitzungen beenden	Der ausgewählte logische Sitzungseintrag wird widerrufen.	Die eigene aktuelle Sitzung kann nicht auf diesem Weg versehentlich beendet werden.

7.17.10 Rolle von Cron

Cron ist für Session-Gültigkeit nicht der Schiedsrichter. Eine Sitzung wird abgewiesen, sobald ein Request oder Heartbeat serverseitig feststellt, dass Idle-Frist, absolute Grenze, Widerruf, Sicherheitsversion oder native ID-HMAC nicht mehr passen. Dafür muss kein Cron-Lauf vorher stattgefunden haben.

Cron räumt später auf: abgelaufene Einträge aus der Sitzungsübersicht, länger widerrufene Einträge und veraltete inaktive Zustände verschwinden aus der Datenbank. Die verschlüsselten PHP-Sessiondateien selbst folgen zusätzlich den normalen PHP-Aufräumregeln. Solange eine alte verschlüsselte Datei noch existiert, ist sie nicht automatisch eine gültige Anmeldung. Gültig wird sie nur, wenn alle serverseitigen Prüfungen bestehen.

> **Betriebsregel:** Cron ist wichtig für Hygiene und Übersicht. Die eigentliche Sicherheitsentscheidung passiert synchron bei der nächsten Nutzung der Sitzung.

7.17.11 Diagnose im Supportfall

Bei Sitzungsfragen hilft eine klare Trennung zwischen „abgelaufen“, „widerrufen“, „rotiert“ und „lokal verloren“. Für die meisten Supportfälle reicht es, die Symptome gegen die folgenden Muster zu halten.

Symptom	Wahrscheinliche Ursache	Sinnvolle Admin-Frage
Nutzer wird nach Pause ausgeloggt	45 Minuten Inaktivität oder sichtbarer Tab ohne bestätigten Heartbeat.	War der Nutzer wirklich aktiv oder nur die Seite geöffnet?
Nutzer wird trotz Arbeit nach einem halben Tag ausgeloggt	12-Stunden-Obergrenze erreicht.	Begann die Anmeldung am Morgen oder in einem alten Tab?
2FA-Seite akzeptiert Code nicht mehr	Der 10-Minuten-Zwischenzustand ist abgelaufen.	Wurde nach Passwortprüfung zu lange gewartet?
Ein alter Tab verliert die Anmeldung, ein neuer bleibt aktiv	Native Session-ID wurde rotiert; der alte Tab hat den alten Bezeichner.	Waren mehrere Tabs lange parallel offen?
Sitzungsübersicht zeigt alte Einträge noch kurz	Gültigkeit ist bereits beendet, Aufräumung folgt später.	Ist Cron gelaufen und liegt der Eintrag außerhalb der Aufräumfrist?

7.17.12 Betriebs-Checkliste

- Produktiv immer HTTPS verwenden, damit Session-Cookies nur verschlüsselt transportiert werden.
- Serverseitige At-Rest-Schlüssel sicher konfigurieren und nicht in Repositories oder Handbücher kopieren.
- Cron als Wartungsaufgabe einrichten, aber Session-Ablauf nicht von Cron abhängig planen.
- Admins und Support informieren: 45 Minuten Inaktivität und 12 Stunden absolute Grenze sind erwartetes Verhalten.
- Bei Passwort- oder Resetfällen erklären, dass andere Sitzungen bewusst ungültig werden.
- Bei verlorenen Geräten andere Sitzungen beenden und bei Bedarf Passwortwechsel oder Reset veranlassen.
- Bei Sicherheitsfragen zwischen verschlüsselter Sessiondatei, RAM während eines Requests und Datenbank-Hashwerten unterscheiden.

Dieses Handbuch beschreibt das Session-Management von Ephraim mit Stand Juni 2026. Maßgeblich für Betrieb und Sicherheitsbewertung ist der aktuelle Serverstand der installierten Ephraim-Instanz.

Quelle: web/manuals/admin-session-management/index.html

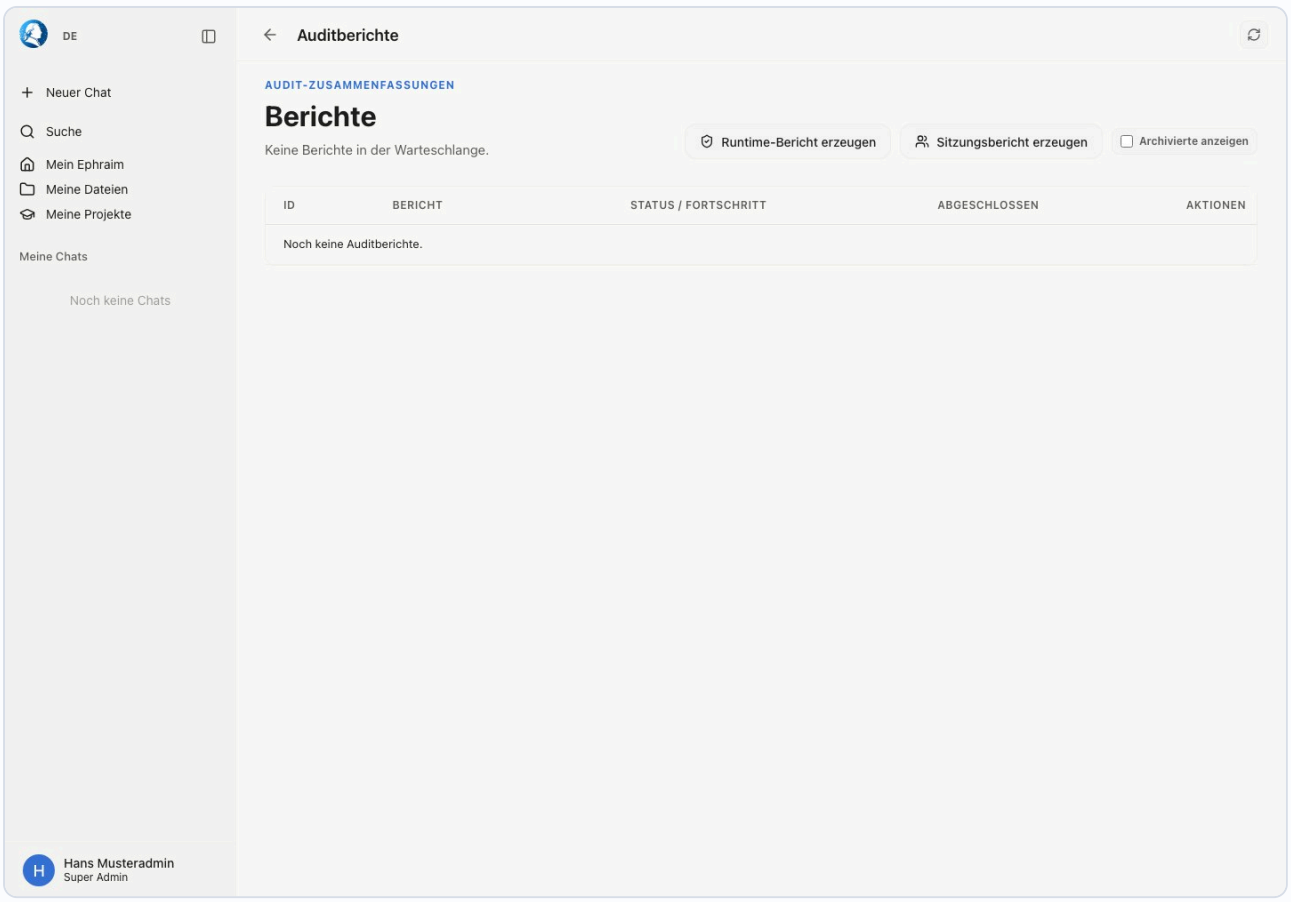
7.18 Auditberichte

> **Auditberichte fassen ausgewählte Betriebsdaten mit KI-Unterstützung zusammen. Sie sind Super-Admins vorbehalten und dienen der Systemprüfung, nicht dem Lesen privater Chats.**

Stand: Mai 2026

7.18.1 Zweck

Auditberichte helfen, Runtime- und Sitzungszustände strukturiert zu überblicken. Statt Rohdaten manuell zu durchsuchen, erzeugt Ephraim eine zusammenfassende Markdown-Auswertung. Der Fokus liegt auf Betrieb, Risiken, Auffälligkeiten und nächsten Prüfschritten.



Auditberichte werden als Super-Admin-Funktion verwaltet. Die Oberfläche trennt Berichtserstellung, Status und spätere Verwaltung.

7.18.2 Zugriff und Berechtigung

Die Funktion ist Super-Admins vorbehalten. Das ist angemessen, weil Auditberichte aggregierte technische Daten und Betriebsrisiken enthalten können. Normale Adminaufgaben wie Klassen oder Benutzerverwaltung brauchen keinen Zugriff auf diese Berichte.

7.18.3 Preflight vor dem Bericht

Vor dem Start prüft Ephraim Umfang, ungefähre Dauer, Datenarten und mögliche Risiken. Dieser Preflight hilft, große Berichte bewusst einzuplanen. Berichte können sofort oder zeitversetzt, etwa nachts, eingeplant werden.

7.18.4 Verarbeitung

Der Worker holt die freigegebenen Auditdaten serverseitig, entfernt oder maskiert Geheimnisse, zerlegt große Datenmengen in verarbeitbare Stücke und lässt die Spark daraus Zusammenfassungen erzeugen. Das Ergebnis wird als Bericht abgelegt, nicht als Rohdump in der Oberfläche angezeigt.

7.18.5 Verwalten von Berichten

Die Adminseite kann Berichte listen, herunterladen, archivieren, abbrechen oder löschen. Abgebrochene oder fehlerhafte Berichte werden mit Jobstatus und Logs eingeordnet. Ein Bericht ist ein Prüfwerkzeug und ersetzt keine eigene Bewertung durch den Super-Admin.

7.18.6 Datenschutzgrenze

Auditberichte sind nicht dafür gedacht, private Chatverläufe zu öffnen. Sie arbeiten mit technischen Exporten, Redaktionen und Zusammenfassungen. Wenn ein Bericht Hinweise auf ein Problem liefert, erfolgt die weitere Untersuchung mit minimalem Datenzugriff.

Damit unterscheiden sie sich klar von den Nutzer-Zusammenfassungen in Mein Ephraim und vom Chatkopf im persönlichen Chat. Diese Nutzer-Synopsen verdichten private Chats für das jeweilige Konto; Auditberichte verdichten Betriebs- und Sicherheitsdaten für Super-Admins.

Auditberichte verdichten technische Sichtbarkeit. Sie sind kein Ersatz für Rollen-, Log- und Datenschutzdisziplin.

Quelle: web/manuals/admin-auditberichte/index.html

EINORDNUNG

8. Code-Sicherheit

> **Code-Sicherheit ist bei Ephraim kein einzelnes Versprechen, sondern ein laufender Prüfprozess: interne Gegenprüfungen, dokumentierte Befunde, Fixes, Re-Verify und die ausdrückliche Einladung zu externen Reviews gehören zusammen.**

Das Kapitel erklärt, warum ein schulisches KI-System mit Rollen, Dateien, Kryptografie, Projekten und lokaler Inferenz nicht allein durch Selbsterklärung vertrauenswürdig wird und warum externe Reviews gewünscht, aber aktuell noch nicht vorhanden sind.

Es beschreibt außerdem, wie Ephraim den Einstieg für mögliche externe Reviewer nachvollziehbar macht: mit getrennten Reviewspuren, Freigabegrenzen, Befundregister, Verify und Re-Verify.

8.1 Problem: Warum externe Reviews nötig sind

> Ephraim möchte ausdrücklich externe Security-Reviews der Code- und Architektursicherheit gewinnen. Aktuell liegen solche externen Reviews noch nicht vor. Wer Architektur, Webanwendung oder Kryptografie unabhängig prüfen möchte, darf sich gerne melden. Die Plattform verarbeitet schulische Daten, Rollenrechte, Dateien, Projektkontexte und lokale KI-Anfragen. Deshalb braucht Vertrauen mehr als gute Absichten, gute Dokumentation und interne Prüfung.

Stand: Juli 2026

8.1.1 Was mit Code-Sicherheit gemeint ist

Dieses Thema meint nicht die persönlichen Sicherheitseinstellungen eines Kontos, nicht die Datei-Sicherheitsseite im Adminbereich und nicht einzelne Betriebsfunktionen. Diese Bereiche haben eigene Handbuchartikel, weil sie Bedienung, Rollen und Betrieb erklären.

Hier geht es um die Sicherheit des Systems selbst. Ist die Architektur tragfähig? Sind Vertrauensgrenzen richtig gezogen? Halten Rollen- und Objektprüfungen? Gibt es Angriffswege über Webendpunkte, Sessions, CSRF, XSS, Uploads oder Exporte? Werden Schlüssel, Tokens, Signaturen, Zufall und Recovery richtig eingesetzt?

Ephraim unterscheidet dabei zwischen Security-Reviews und breiteren Audits. Security-Reviews prüfen gezielt Code-, Architektur- und Kryptografierisiken. Audits können zusätzlich Qualität, Bedienbarkeit, Betrieb oder Wartbarkeit betrachten. Beides ist wertvoll, aber nur ein Security-Review kann eine konkrete Sicherheitsannahme entlasten oder ein Sicherheitsfinding begründen.

> **Abgrenzung:** Die vorhandenen Sicherheitsartikel erklären nutzbare Schutzfunktionen. Dieser Artikel erklärt, warum der Code, die Architektur und die kryptografischen Mechanismen von außen geprüft werden sollen. Breitere Audits ergänzen diese Arbeit, ersetzen sie aber nicht.

8.1.2 Warum Selbsterklärung nicht reicht

Ephraim ist bewusst lokal gebaut. KI-Anfragen gehen nicht an öffentliche Modellanbieter, Vault-Inhalte werden verschlüsselt gespeichert und Rollen begrenzen Zugriffe. Das sind wichtige Schutzentscheidungen. Sie beweisen aber nicht automatisch, dass jede konkrete Codekante hält.

Sicherheitsprobleme entstehen oft dort, wo ein System im Alltag kleinteilig wird: ein Spezialpfad beim Passwortwechsel, ein Exportendpunkt, ein Projektbeitritt, ein Upload, ein Rollenwechsel, eine Fehlermeldung oder ein Recovery-Flow. Solche Stellen kann das eigene Team übersehen, gerade weil es die beabsichtigte Architektur gut kennt.

8.1.3 Warum Nähe blinde Flecken erzeugt

Nähe zur Schule ist eine Stärke von Ephraim. Die Anforderungen kommen aus echter pädagogischer Praxis und nicht aus einem abstrakten Produktkonzept. Dieselbe Nähe kann aber blinde Flecken erzeugen. Wer ein System gebaut hat, liest Abläufe häufig so, wie sie gemeint waren, nicht so, wie sie missbraucht werden könnten.

Externe Reviewer bringen eine andere Haltung mit. Sie fragen weniger, ob ein Ablauf sinnvoll gedacht ist. Sie fragen, ob ein Nutzer ihn umgehen kann, ob ein Token zu viel darf, ob eine Rolle einen fremden Gegenstand erreicht, ob ein Browserpayload gerendert wird oder ob ein Schlüssel für zu viele Zwecke verwendet wird.

8.1.4 Welche Grenzen wir offen benennen

Ephraim ist weit für ein schulisches Projekt, aber genau diese Breite ist auch ein Risiko. Chat, Projekte, Dateien, Basiswissen, TTS, Adminfunktionen, Kryptografie, Rollen und lokale KI greifen ineinander. Kein kleines Team kann aus eigener Nähe heraus behaupten, alle Wechselwirkungen vollständig zu sehen.

Grenze	Warum sie relevant ist
Eigenes Team	Wer das System gebaut hat, kennt die Absicht. Externe Reviewer sehen eher, wie dieselbe Funktion missbraucht werden kann.
KI-gestützte Entwicklung	KI kann Code, Tests und Audits beschleunigen. Sie kann aber auch plausibel klingende Sicherheitserzählungen erzeugen.
Lokale Tests	Lokale DDEV- und Browserprüfungen sind wertvoll, ersetzen aber keine spätere Betriebsprüfung unter realen Zuständigkeiten.
Dokumentation	Handbücher schaffen Transparenz. Sie dürfen aber nicht schöner klingen als der geprüfte Stand tatsächlich ist.
Restunsicherheit	Auch nach behobenen Befunden bleiben neue Kombinationen, neue Funktionen, neue Konfigurationen und menschliche Fehler möglich.

Diese Grenzen sind kein Argument gegen Ephraim. Sie sind der Grund, warum externe Reviews gesucht werden und warum die eigene Review-Struktur nicht als Zertifikat, sondern als Einladung zur Gegenprüfung verstanden wird.

8.1.5 Welche externen Reviews wir suchen

Für Ephraim sind drei Reviewarten besonders wichtig. Sie sind vorbereitet, aber aktuell noch nicht extern übernommen oder abgeschlossen. Zusammen würden sie dieselbe Plattform aus unterschiedlichen Richtungen betrachten und ein belastbareres Bild ergeben als ein pauschales Sicherheitsurteil.

Daneben kann Ephraim auch breitere Audits sammeln, etwa zu Systemzustand, Wartbarkeit, UI/UX oder Betrieb. Solche Audits helfen beim Gesamtbild, dürfen aber nicht als Sicherheitsfreigabe gelesen werden. Für die Code-Sicherheit bleiben die folgenden Reviewarten der harte Maßstab.

Reviewart	Worauf der Blick fällt	Warum das für Schulen zählt
Architektur- und Threat-Model-Review	Komponenten, Rollen, Datenflüsse, Vertrauensgrenzen, Schutzannahmen und Missbrauchsszenarien.	Schulische KI braucht erklärbare Grenzen, nicht nur eine funktionierende Oberfläche.
Webanwendungs-Pentest	Anmeldung, 2FA, Sessions, CSRF, XSS, Uploads, Projektcodes, Exporte, Adminfunktionen und APIs.	Viele Risiken entstehen nicht im Modell, sondern an Formularen, Rechten, Endpunkten und Bedienwegen.
Kryptografisches Implementierungsreview	Schlüssel, Tokens, Signaturen, Zufall, Recovery, Vault, Zwecktrennung und Fehlerverhalten.	Verschlüsselung hilft nur, wenn Schlüsselwege, Token-Grenzen und Wiederherstellung sauber gebaut sind.

> **Einladung:** Externe Reviews sind ausdrücklich erwünscht. Wer einen unabhängigen Blick beitragen möchte, kann sich gerne melden. Bis dahin benennt Ephraim diese Reviews als Ziel, nicht als vorhandenen Nachweis.

8.1.6 Was ein externer Gegenblick leisten soll

Externe Reviews sollen Ephraim nicht einfach bestätigen. Sie sollen Annahmen angreifen, Lücken finden, falsche Sicherheit benennen und Prioritäten korrigieren. Ein guter Review kann auch Entwarnung geben, aber nur dort, wo ein konkreter Pfad wirklich nachvollziehbar geprüft wurde.

Entscheidend ist der Nachweis: Was wurde geprüft? Welcher Systemstand galt? Welche Rolle, welcher Datenfluss oder welcher Token war im Fokus? Welche Belege liegen vor? Welche Befunde bleiben offen? Genau diese Fragen führen zum zweiten Artikel: dem Lösungsansatz.

Dieser Artikel beschreibt das Problem hinter den gewünschten externen Reviews. Der Nachbarartikel zum Lösungsansatz erklärt, welche Review- und Audit-Struktur Ephraim dafür bereits vorbereitet hat.

Quelle: web/manuals/code-sicherheit-problem/index.html

8.2 Lösungsansatz: Review-System, Audits und Verify

> **Ephraim bittet um externe Reviews und möchte sie ausdrücklich gewinnen. Damit ein unabhängiger Blick nicht bei null anfangen muss, hat das Projekt eine Struktur vorbereitet, in der Code- und Architektursicherheit streng prüfbar bleiben und breitere Audits nachvollziehbar danebenstehen: getrennte Reviewspuren, dokumentierte Freigaben, Befundregister, Fixlogs, Verify-Läufe, Reviewer-Packs und Seiten für Menschen.**

Stand: Juli 2026

8.2.1 Welche Vorarbeit bereits geleistet wurde

Wer ein externes Review übernehmen möchte, trifft nicht auf eine leere Bitte um Hilfe. Ephraim hat bereits eine Review-Struktur aufgebaut, damit externe Reviewer schnell sehen, welche Fragen das Projekt selbst gestellt hat und wo Gegenprüfung besonders wertvoll ist.

Die Security-Review-Spuren bleiben dabei der strengere Kern: Dort zählen Scope, Belege, Finding, Verify und Re-Verify. Breitere Audits nutzen dieselbe nachvollziehbare Dokumentationslogik, betrachten aber auch Systemzustand, Qualität, Bedienbarkeit oder Betrieb. Sie dürfen deshalb nicht automatisch als Sicherheitsfreigabe gelesen werden.

Baustein	Wozu er dient
Reviewspuren	Architektur, Threat Model, Webanwendung und Kryptografie werden getrennt geprüft, damit Befunde nicht vermischt werden.
Freigabegrenzen	Aktive Tests sind lokal, defensiv und mit synthetischen Daten vorgesehen. Produktion und echte Nutzerdaten bleiben ausgeschlossen.
Befundregister	Findings haben Status, Schweregrad, Herkunft, Entscheidung und Nachweis statt nur lose Notizen.
Verify-Spur	Jedes Review-Ergebnis wird nachvollziehbar, weil Fund, Entwarnung, Verify, Fix und Re-Verify getrennt dokumentiert werden.
Seiten für Menschen	Für externe Reviewer gibt es eine Übersicht zu Security-Reviews und eine zweite Übersicht für breitere Audits, historische Stände und Folgeaufgaben.

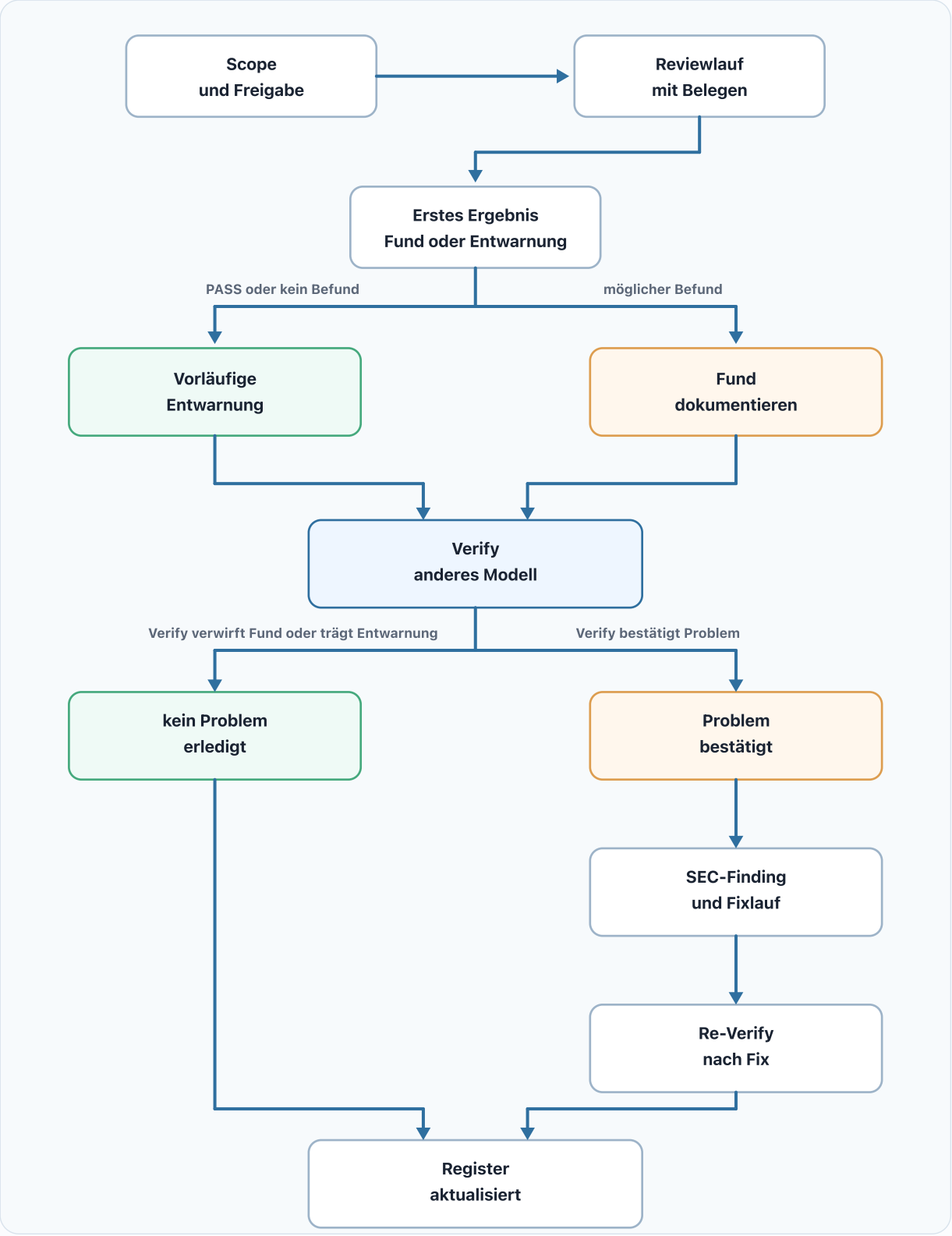
8.2.2 Welche Review- und Auditspuren vorbereitet sind

Ephraim trennt die Prüfung bewusst in mehrere Spuren. Dadurch kann ein Architekturproblem anders dokumentiert werden als ein XSS-Risiko, ein Kryptografieproblem oder eine offene Annahme im Threat Model. Zusätzlich gibt es breitere Auditspuren, damit Qualitäts-, UI-, Betriebs- und Systemfragen nicht in Sicherheitsfindings hineingedrückt werden.

Spur	Typische Fragen
Architektur	Welche Komponenten sprechen miteinander? Wo liegen Rollen- und Objektgrenzen? Welche Annahmen müssen im Code bestätigt werden?
Threat Model	Welche Assets sind besonders schutzbedürftig? Welche Angreiferrollen sind realistisch? Welche Abuse-Cases müssen getestet werden?
Webanwendung	Halten Authentifizierung, Autorisierung, CSRF-Schutz, Rendering, Uploads, Exporte und Adminwege auch in negativen Tests?
Kryptografie	Sind Token, Schlüssel, Recovery-Codes, Signaturen und Vault-Flows zweckgebunden, zufallsstark und nachvollziehbar?
Verify	Trägt eine Entwarnung? Ist ein Fund wirklich ein Problem? Wurde ein Fix danach unabhängig re-verifiziert?
Systemaudit	Ist der Systemzustand nachvollziehbar, sind zentrale Pfade dokumentiert und bleiben bekannte Altstände sichtbar?
Qualität und Wartbarkeit	Wo entstehen technische Schulden, fragile Kopplungen, unklare Verantwortlichkeiten oder fehlende Tests?
UI/UX-Audit	Sind Bedienwege verständlich, barrierearm, konsistent und für die jeweiligen Rollen nachvollziehbar?
Betriebsaudit	Sind Logging, Backup, Wiederherstellung, Updates, Monitoring und Zuständigkeiten belastbar beschrieben?

8.2.3 Architektur des Review-Systems

Ein Review beginnt mit einem abgegrenzten Prüffeld und einer dokumentierten Freigabe für aktive Tests. Danach entstehen Bericht, menschliche Zusammenfassung und Belege. Das erste Ergebnis ist ein Fund oder eine vorläufige Entwarnung. Beides geht anschließend in Verify: Ein unabhängiger Prüfer liest Scope, Methode, Belege und Ergebnis gegen. Erst wenn Verify ein Problem bestätigt, bleibt oder entsteht ein `SEC-*`-Finding. Nach dem Fix folgt Re-Verify.



Verify prüft Fund oder Entwarnung. Re-Verify prüft nur den Fix.

Die Grafik folgt der neuen Prüfkette: Fund oder Entwarnung bleiben vorläufig, bis Verify sie unabhängig bestätigt oder verwirft. Nur ein bestätigtes Problem führt zum `SEC-*`-Finding, Fixlauf und Re-Verify.

8.2.4 Das Multi-Modell-Konzept

Ephraims Review-System setzt nicht darauf, dass ein einzelnes KI-Modell Sicherheit abschließend beurteilen kann. Verschiedene Agenten und Modelle lesen Code, Berichte und Belege aus unterschiedlichen Blickwinkeln. Ein Modell strukturiert vielleicht die Architekturfrage gut, ein anderes findet eher Webpfade, ein weiteres formuliert bessere Negativtests oder erkennt Widersprüche in der Dokumentation.

Der Nutzen liegt nicht in einem Modellwettbewerb. Der Nutzen liegt in Reibung: mehrere Durchläufe, verschiedene Formulierungen, andere Schwerpunkte und eine dokumentierte Spur, die Menschen anschließend prüfen können. KI kann hier sehr hilfreich sein, weil sie große Mengen Code, Tests und Dokumentation schnell ordnet. Verbindlich wird ein Ergebnis aber erst durch Verify, Bewertung, Umsetzung und Re-Verify nach einem Fix.

> **Multi-Modell heißt:** Ephraim nutzt KI nicht nur zum Bauen, sondern auch zum Gegenlesen. Die Verantwortung bleibt trotzdem menschlich und organisatorisch bei den Personen, die den Einsatz freigeben und betreiben.

8.2.5 Was der Lösungsansatz nicht behauptet

Die vorbereitete Review-Struktur ist ein Fortschritt, aber sie ist kein Sicherheitssiegel. Ein Register, ein Diagramm, Seiten für Menschen und mehrere KI-gestützte Läufe beweisen nicht automatisch, dass kein Fehler mehr im System steckt. Sie machen nur besser sichtbar, was geprüft, behoben, verworfen oder noch offen ist.

Behauptet wird nicht	Stattdessen gilt
Alle Risiken sind erledigt.	Bekannte Befunde werden nachverfolgt. Neue Reviews können neue Befunde erzeugen.
KI kann unabhängige Prüfung ersetzen.	KI hilft beim Lesen, Sortieren und Testen. Entscheidungen brauchen menschliche Bewertung.
Lokale Inferenz löst jedes Sicherheitsproblem.	Lokale KI reduziert Datenabflussrisiken. Weblogik, Rollen, Tokens und Betrieb müssen trotzdem geprüft werden.
Ein internes Review ist ein externes Audit.	Interne Reviews schaffen Vorarbeit. Externe Reviewer sollen diese Vorarbeit ausdrücklich angreifen dürfen.
Ein breites Audit ist eine Sicherheitsfreigabe.	Audits können Qualität, Betrieb oder Bedienbarkeit prüfen. Sicherheitsentlastung braucht weiterhin Scope, Belege und Verify.

8.2.6 Schutzgrenzen bei aktiven Tests

Sicherheitsreviews dürfen nicht selbst zum Risiko werden. Aktive Tests laufen deshalb nur gegen freigegebene lokale Ziele und mit synthetischen Daten. Produktivsysteme, echte Konten, echte Schülerdaten und externe Dienste sind ohne gesonderte Freigabe ausgeschlossen. Wenn Datenbankzustände verändert werden müssen, geschieht das temporär, mit Rücknahme oder in einem dokumentierten Klon.

Grenze	Konsequenz für Reviews
Keine echten Nutzerdaten	Tests arbeiten mit fiktiven Konten, harmlosen Dateien und redigierten Artefakten.
Keine Produktionstests ohne Freigabe	Normale Reviews bleiben in der lokalen Entwicklungs- und Testumgebung.
Keine destruktiven Angriffe	Rate-Limits, Uploads, XSS und CSRF werden defensiv und mit harmlosen Markern geprüft.
Keine Geheimnisse in Artefakten	Tokens, Cookies, Passwörter, Mailinhalte und vollständige Adressen werden nicht veröffentlicht.

8.2.7 Vom Befund zur Korrektur

Ein Sicherheitsbefund ist kein Scheitern. Er ist der Punkt, an dem ein Risiko überprüfbar wird. Ephraim unterscheidet deshalb zwischen offenen Befunden, verworfenen Befunden, akzeptierten Restrisiken, behobenen Befunden und nachgeprüften Fixes.

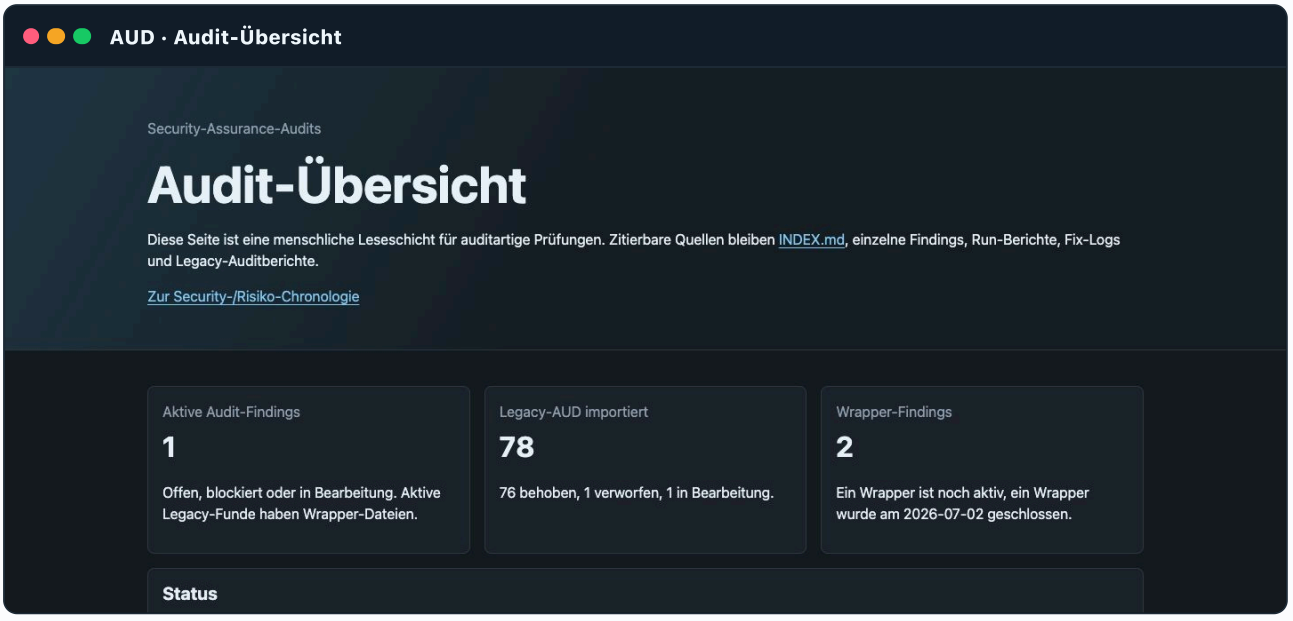
In der bisherigen Reviewarbeit wurden unter anderem Session-Grenzen, CSRF-Muster, Recovery-Codes, Token-Zwecktrennung, Objektberechtigungen und Projektbeitritte untersucht. Mehrere Befunde wurden behoben und re-verifiziert. Genau dieses Muster soll weitergeführt werden: finden, verifizieren, beheben, re-verifizieren, erklären.

8.2.8 Seiten für externe Reviewer und Audits

Die Seite für Security-Reviews bündelt Chronologie, Kennzahlen, Ablauf und Quellenhinweise. Sie ersetzt keine Prüfberichte, ist aber ein überzeugender Startpunkt für Menschen, die ein externes Security-Review übernehmen möchten: Die technische Struktur steht, die Reviewarten sind getrennt, die Sicherheitsgrenzen sind definiert und bisherige Befunde sind nachvollziehbar.

Die folgenden Momentaufnahmen zeigen die aktuellen menschlichen HTML-Übersichten vollständig.

SEC-Ansicht: Die menschliche Security-Review-Seite zeigt Kennzahlen, Ablauf, Chronologie, Finding-Stand und offene Agentenaufgaben als zusammenhängende Review-Spur.



AUD-Ansicht: Die auditartige Leseschicht zeigt Legacy-Abdeckung, Konzeptgruppen, offene Audit-Findings und den Abstand zum strengeren Security-Review-Begriff.

Daneben gibt es eine Audit-Übersicht für breitere Prüfungen. Dort können historische Auditstände, System-, Qualitäts-, UI-/UX- und Betriebsprüfungen sowie offene Folgeaufgaben sichtbar werden, ohne den strengeren Security-Review-Begriff zu verwässern.

Für Schulleitung, Datenschutzbeauftragte, Eltern und Lehrkräfte ist die Botschaft ähnlich wichtig. Ephraim setzt nicht nur auf lokale KI und Verschlüsselung. Ephraim behandelt Code-Sicherheit als laufenden Prozess, der dokumentiert, korrigiert und von außen überprüfbar sein muss.

Die fachlichen Nachbarartikel sind Problem, Systemarchitektur, Datenschutz tiefgehend, Kryptoarchitektur, Datei-Sicherheit und Session-Management.

Dieser Artikel beschreibt den Stand von Ephraims Review- und Audit-Prozess im Juli 2026. Maßgeblich bleiben die jeweils aktuellen Reviewartefakte, Auditstände, Befundregister, Fix-Nachweise und Freigabeentscheidungen.

Quelle: web/manuals/code-sicherheit-loesungsansatz/index.html

EINORDNUNG

9. Glossar

> **Ephraim verwendet Begriffe aus Pädagogik, Kryptografie, KI-Inferenz, Webbetrieb und schulischer Organisation. Das Glossar sammelt diese Begriffe an einer Stelle.**

Das Glossar ist bewusst als eigenes Kapitel geführt. Es erklärt wiederkehrende Fachwörter ohne den Lesefluss der inhaltlichen Kapitel zu unterbrechen.

Wer einzelne Begriffe nachschlagen möchte, findet hier die kurze Einordnung und kann anschließend in die ausführlichen Kapitel zurückspringen.

> Dieses Glossar erklärt wiederkehrende und erklärungsbedürftige Begriffe aus den Handbüchern. Es ist kein Informatiklexikon, sondern eine kurze Orientierung: Was meint Ephraim mit Vault, Nonce, Inferenz, Embedding, TTS, MCP oder Projektcode?

Stand: Juni 2026

2FA
: Zwei-Faktor-Authentifizierung. Zusätzlich zum Passwort wird ein zeitbasierter Code verlangt.

Admin-Audit
: Eine Betriebs- und Sicherheitsauswertung für Super-Admins. Sie ist nicht dasselbe wie eine private Chat-Zusammenfassung.

AEAD
: Authenticated Encryption with Associated Data. Verschlüsselung, die nicht nur unlesbar macht, sondern auch Manipulationen erkennt.

Apache
: Webserver-Software. In Ephraims Produktivbetrieb ist wichtig, dass der Webserver keine Anfrageinhalte, Prompts, Passwörter oder Tokens protokolliert.

API
: Programmierschnittstelle zwischen Systemteilen. In Ephraim nutzt der Browser APIs der Webanwendung; die Webanwendung ruft wiederum interne Spark-APIs auf.

Individuelles Lehrerfazit je Schüler
: Verdichtete pädagogische Rückmeldung zur Arbeit einer einzelnen Teilnahme, wenn diese Option vor Projektstart aktiviert wurde. Es ist kein Chatprotokoll.

Argon2id
: Verfahren zur Schlüsselableitung aus einem Passwort. Es ist absichtlich rechen- und speicheraufwändig, damit Passwortangriffe teurer werden.

At-Rest-Verschlüsselung
: Verschlüsselung ruhender Daten, also gespeicherter Daten in Datenbank oder Dateisystem. Sie schützt die Ablage, ersetzt aber nicht den kurzen Klartextmoment bei Anzeige, Export oder KI-Verarbeitung.

Audio-Blob
: Temporäres Audioobjekt im Browser. Bei der Sprachausgabe entsteht daraus der Player; es ist keine dauerhaft gespeicherte Audiodatei auf dem Server.

Auftragsverarbeitung

Datenschutzrechtlicher Begriff für Dienstleister, die personenbezogene Daten im Auftrag einer verantwortlichen Stelle verarbeiten. Ephraim reduziert solche Abhängigkeiten im KI-Kern durch lokale Spark-Verarbeitung.

Antwortstatistik

Optionale Anzeige unter KI-Antworten. Sie kann technische Werte wie Generierungsdauer und eine grobe Energieabschätzung zeigen. Diese Werte helfen bei Transparenz und Betriebseinordnung, bewerten aber nicht die fachliche Qualität der Antwort.

Backend

Serverseitiger Teil eines Systems. In Ephraim prüft das Backend Anmeldungen, Rechte, Vault-Zugriffe, Exporte, Dateiabrufe und Anfragen an Spark-Dienste.

Base64

Textdarstellung für Binärdaten. Sie macht Daten transportierbar, aber nicht geheim; eine Base64-Zeichenfolge ist keine Verschlüsselung.

Basiswissen / RAG

Freigegebene Quellen werden vor der Inferenz gesucht und als Kontext eingefügt. Das Modell bekommt den vorbereiteten Ausschnitt, nicht freien Internetzugriff.

bcrypt

Verfahren zum Speichern von Passwortprüfwerten. Ephraim speichert damit nicht das Passwort selbst, sondern einen absichtlich schwer angreifbaren Hash.

Bearer-Token

Ein Token, das bei Server-zu-Server-Anfragen im HTTP-Header mitgegeben wird. Wer es besitzt, kann die damit erlaubte Aktion ausführen.

Benchmark

Messlauf, der ein System unter definierten Bedingungen belastet. Benchmarks zeigen Kapazität und Grenzen, ersetzen aber keine Beobachtung im echten Schulbetrieb.

Browser-Cookie

Kleiner Wert, den der Browser bei Anfragen mitschickt. Ephraim nutzt Cookies für die technische Sitzung, nicht als Werbe- oder Trackingprofil.

Busfaktor

Risiko, dass zu viel Wissen an einzelnen Personen hängt. Dokumentation, Tests, nachvollziehbarer Git-Verlauf und KI-gestützte Analyse senken dieses Risiko, ersetzen aber keine Verantwortungsübergabe. Ephraim beschreibt dazu ein eigenes Kontinuitätsmodell.

Cache-Control: no-store

HTTP-Anweisung an Browser und Zwischenstationen, eine Antwort nicht aufzubewahren. Ephraim nutzt sie bei sensiblen Antworten wie Downloads oder TTS-Audio.

Chunk

Kleiner Abschnitt eines größeren Inhalts. Ephraim zerlegt Dateien, Kalenderdaten und Wissensquellen in Chunks, damit Suche, Verschlüsselung und Verarbeitung kontrollierbar bleiben.

Ciphertext

Verschlüsselter Text oder verschlüsselte Daten. Ciphertext kann gespeichert werden, ohne den ursprünglichen Inhalt direkt lesbar zu machen.

ClamAV

Virenscanner, den Ephraim in Produktion in die Upload-Prüfung einbindet. Er ergänzt Dateityp- und Strukturprüfungen, ersetzt sie aber nicht.

CLI

Command Line Interface: Bedienung über eine Kommandozeile. Für Administratoren ist das relevant, wenn Cron oder Wartung bewusst ohne Browseroberfläche ausgeführt werden.

Cron

Regelmäßige Wartung im Hintergrund. Cron räumt alte technische Zustände auf, prüft Dienste, aktualisiert freigegebenes Wissen und läuft ohne entsperrte Nutzer-Vaults.

CSRF

Schutz gegen ungewollte Aktionen aus fremden Webseiten. Schreibende Anfragen brauchen ein Sicherheitstoken aus der eigenen Sitzung.

CSS

Stylesheet-Sprache für Layout, Farben und Abstände im Browser. CSS bestimmt Darstellung, führt aber keine Anwendungslogik aus.

Datenschutz-Folgenabschätzung (DSFA)

Systematische Datenschutzprüfung für Verfahren mit voraussichtlich erhöhtem Risiko. Sie beschreibt Zweck, Notwendigkeit, Risiken, Schutzmaßnahmen und Restrisiken.

DDEV

Lokale Entwicklungsumgebung für Webprojekte. Ephraim nutzt DDEV zum Testen und Entwickeln, nicht als Produktionsbetrieb.

DEK

Data Encryption Key: Datenschlüssel für die eigentlichen Inhalte. In Ephraim verschlüsseln DEKs zum Beispiel Vault-Inhalte oder dateibezogene Daten.

Denkprozess

Aktivitätsansicht im Chat, die bei Thinking-Antworten verständliche Arbeitsschritte und Werkzeugereignisse zeigen kann. Sie hilft beim Nachvollziehen, ist aber kein vollständiges internes Modellprotokoll.

Drittanbieter-Komponenten

Mitgelieferte Bibliotheken, die nicht von Ephraim selbst geschrieben wurden. Ephraim verwaltet sie kontrolliert, lokal und mit Versionsnachweis.

DSGVO

Datenschutz-Grundverordnung der Europäischen Union. Sie regelt Grundsätze, Rechtsgrundlagen, Informationspflichten, Betroffenenrechte und Sicherheit personenbezogener Daten.

DSL

Domain Specific Language: eine kleine, zweckgebundene Beschreibungssprache. Ephraim nutzt solche begrenzten Formate für Visualisierungen statt freiem ausführbarem Code.

E2E-Test

End-to-End-Test. Ein vollständiger Ablauf wird wie aus Nutzersicht durchgespielt, zum Beispiel Projekt anlegen, beitreten, chatten, abschließen, Fazit erzeugen und Admin-/Lehrkraftsicht prüfen.

Ed25519

Verfahren für digitale Signaturen. Ephraim nutzt es bei kurzlebigen Runtime-JWTs, damit Dienste prüfen können, ob ein Token vom berechtigten Frontend stammt.

EFR1

Ephraims internes verschlüsseltes Dateiformat. Es kennzeichnet Dateien, die in verschlüsselten Blöcken gespeichert und beim Download kontrolliert entschlüsselt werden.

Embedding

Ein Zahlenvektor für einen Textabschnitt. Er hilft, inhaltlich passende Ausschnitte zu finden, auch wenn nicht dieselben Wörter verwendet werden. In Ephraim berechnet `school-ui-embedding` diesen Vektor lokal auf der Spark, speichert Klartext und Vektor dort aber nicht dauerhaft.

Feature Freeze

Zäsur vor einer stabilen Version. Für Version 1 bedeutet sie: keine normale Feature-Erweiterung, sondern Testen, Fehlerbehebung, Härtung und Dokumentationsabgleich. Notwendige Betriebs-, Datenschutz- oder Ausfallkorrekturen können trotzdem dazugehören.

Fähigkeit / Skill

Ausführliche Spezialanweisung, die Ephraim dem Modell nicht dauerhaft mitsendet. Das Modell sieht zunächst nur Name und Kurzbeschreibung und lädt den vollständigen Inhalt erst bei Bedarf mit `load_skill`.

Frontend

Der Teil einer Anwendung, den Nutzer im Browser sehen und bedienen. In Ephraim zeigt das Frontend Chats, Projekte, Einstellungen, Adminseiten und Handbücher.

Gastzugang

Eine Sitzung, die nur zu einem Projekt gehört. Sie hat keine normalen Kontoeinstellungen, keinen privaten Dateispeicher und keine allgemeinen Chatrechte.

Git

Versionsverwaltung für Code. Der Git-Verlauf macht nachvollziehbar, wann welche Änderung am Projekt entstanden ist.

GPT-OSS-120B / OpenAI OSS

Das aktuell verwendete lokale Sprachmodellprofil in Ephraim. Es läuft als Inferenzdienst auf der Spark und wird nicht als Cloud-API eines externen Chatdienstes genutzt.

GPU

Spezialisierter Rechenchip für viele parallele Berechnungen. Sprachmodelle und Embeddings profitieren davon, weil sehr viele Zahlenoperationen gleichzeitig laufen.

Hash

Ein berechneter Fingerabdruck von Daten. Ein guter Hash ist praktisch nicht zurückrechenbar, kann aber Gleichheit oder Veränderung erkennbar machen.

HEIC / HEIF

Bildformate, die auf vielen Smartphones vorkommen. Ephraim prüft solche Dateien nicht nur nach Endung, sondern auch nach erkennbarer Struktur.

HMAC

Ein Hash mit geheimem Schlüssel. Ephraim nutzt HMAC unter anderem, damit technische Speicherpfade nicht direkt aus Nutzer- oder Projekt-IDs ablesbar sind.

HTML

Auszeichnungssprache für Webseiten und formatierte Browserdokumente. Im Chat-Export ist HTML ein Format für lesbare, strukturierte Ablage ohne aktive Skripte.

HTTP / HTTPS

HTTP ist das Grundprotokoll für Webanfragen. HTTPS ist die verschlüsselte Variante und im Produktivbetrieb der relevante Weg für Browser, Webserver und interne Spark-Verbindungen.

ICS

Kalenderdateiformat. Private Kalenderquellen liefern häufig ICS-Daten; Ephraim behandelt deren URLs, Rohdaten und daraus entstehende Termine als vaultgebundene Inhalte.

Inferenz

Die Berechnung einer Antwort mit einem bereits trainierten Modell. Die einzelne Anfrage trainiert das Modell nicht neu.

IP-Adresse

Technische Adresse eines Geräts im Netzwerk. Webserver sehen IP-Adressen bei Verbindungen; Ephraim beschränkt deren Nutzung auf Betrieb, Sicherheit und Fehleranalyse.

JavaScript

Programmiersprache im Browser. Ephraim erlaubt bei Visualisierungen keine freie JavaScript-Ausführung aus KI-Antworten, sondern nutzt geprüfte lokale Renderer.

JSON

Textformat für strukturierte Daten. Ephraim nutzt JSON intern und bei sicheren Visualisierungsbeschreibungen; JSON ist nicht automatisch ausführbarer Code.

JWT

JSON Web Token: kompakter Berechtigungsnachweis mit eingebauten Angaben wie Ablaufzeit, Aussteller und Ziel. Ephraim nutzt kurzlebige JWTs für bestimmte Runtime-Zugriffe.

KEK

Key Encryption Key: Schlüssel, der einen anderen Schlüssel verpackt. Beim Passwortwechsel wird die User-DEK neu mit einer neuen KEK verpackt.

Kontextfenster

Die Textmenge, die das Modell gleichzeitig berücksichtigen kann. Ephraim muss deshalb auswählen, welche Quellen und Chatteile relevant sind.

Kontowiederherstellung

Datenerhaltender Weg für ein vergessenes Passwort. Ein vorbereiteter Wiederherstellungskontakt stimmt live zu, danach wird derselbe Vault-Schlüssel mit einem neuen Passwort verpackt.

Latenz

Wartezeit bis eine Reaktion sichtbar wird. Bei KI-Systemen meint sie zum Beispiel die Zeit bis zur ersten Antwort oder bis eine Anfrage vollständig beantwortet ist.

LDSG

Landesdatenschutzgesetz. Für Schulen in Baden-Württemberg ist es neben DSGVO und Schulrecht ein wichtiger Bezugspunkt für öffentliche Stellen.

LLM / Sprachmodell

Large Language Model. Ein Sprachmodell berechnet aus Prompt und Kontext wahrscheinliche Fortsetzungen; es besitzt keinen eigenen freien Internetzugang.

Logging

Datensparsame Betriebsprotokollierung. Ephraim unterscheidet technische Metriken, Admin-Aktionen, Cron-Ergebnisse und Problem-Mails; private Chatinhalte und Tokens gehören nicht in Logs.

load_skill

Interner Werkzeugaufruf, mit dem das Modell den vollständigen Inhalt einer aktivierten Fähigkeit lädt. Dadurch bleiben lange Spezialregeln aus dem normalen Prompt heraus.

Magic Bytes

Typische Anfangsbytes einer Datei, an denen sich ein Dateiformat erkennen lässt. Ephraim nutzt sie, weil Dateiendungen allein leicht täuschen können.

Markdown

Einfache Textschreibweise für Überschriften, Listen, Tabellen, Code und Links. Im Export ist Markdown gut für weiterbearbeitbaren Strukturtext.

MCP

Model Context Protocol. Ephraim kann ausgewählte Adminwerkzeuge für bewusst autorisierte Coding-Clients bereitstellen.

MIME-Typ

Technische Angabe, welcher Dateityp übertragen oder gespeichert wird, zum Beispiel PDF, Bild oder Text. Ephraim prüft MIME-Typen zusätzlich zu Dateiendung und Magic Bytes.

Monitoring

Reduzierte Projektlage-Daten wie Start, Aktivität, Abschluss, Nachrichtenimpulse, Fazitstatus, Lehrerfeedbackstatus und Transparenzstatus. Monitoring öffnet keine Rohchats.

MySQL

Produktive Datenbank von Ephraim. MySQL hält Konten, Einstellungen, verschlüsselte Chatdaten, Metadaten und dauerhafte Betriebszustände.

Nonce

Einmalwert in der Verschlüsselung. Eine Nonce ist kein Passwort und kein geheimer Schlüssel; sie sorgt dafür, dass gleiche Inhalte nicht immer gleich aussehen und Manipulationen zuverlässig erkannt werden.

OAuth

Verfahren, mit dem ein externer Client begrenzt autorisiert wird, ohne das Passwort des Nutzers zu erhalten.

Office / iWork

Dateifamilien für Textdokumente, Tabellen und Präsentationen. Ephraim prüft bei solchen Dateien auch Containerstruktur und erkannte Dateimerkmale.

Onboarding

Freiwillige Ersteinrichtung eines normalen Kontos. Sie führt zu Personalisierung, Kalender und Recovery-Hinweis, speichert aber nur den Einrichtungsstatus als eigenen Dialogstatus.

OpenAI-kompatibel

Technische Schnittstellenform, die wie eine bekannte OpenAI-API angesprochen werden kann. In Ephraim bedeutet das nicht automatisch externe OpenAI-Nutzung; die freigegebene Konfiguration nutzt lokale Spark-Dienste.

p50 / p95

Statistische Messpunkte in Benchmarks. p50 ist der mittlere typische Wert, p95 zeigt, wie langsam die langsamsten fünf Prozent der Fälle etwa werden.

PDF

Dokumentformat mit stabiler Seitenform. In Ephraim ist PDF für Weitergabe, Druck und fertige Ablage gedacht; ein PDF-Export ist eine bewusste Klartext-Ausleitung.

PHP

Serversprache der Ephraim-Webanwendung. PHP verarbeitet Anmeldungen, Berechtigungen, Datenbankzugriffe, Dateiabrufe, Exporte und interne API-Aufrufe.

PKCE

Zusatzschutz bei OAuth-Flüssen. Er bindet eine Autorisierung an einen vorher erzeugten Prüfwert, damit abgefangene Codes nicht einfach wiederverwendet werden können.

Projektcode

Ein Code, mit dem Teilnehmende projektgebunden beitreten können, ohne ein normales Konto zu verwenden.

Projektmaterial

Von der Lehrkraft freigegebenes Material, das in eine projektbezogene Kopie überführt und für den Projektchat vorbereitet wird.

Prompt

Text, der dem Modell als Aufgabe, Frage, Regel oder Kontext mitgegeben wird. Prompts steuern den Grundrahmen, ersetzen aber keine technischen Berechtigungen und sind nicht dasselbe wie bei Bedarf geladene Fähigkeiten.

Proxy

Vermittler zwischen Browser und internem Dienst. Ein Proxy kann Anfragen prüfen, begrenzen und Secrets serverseitig halten, statt sie an den Browser zu geben.

RAM

Arbeitsspeicher eines Rechners. Inhalte im RAM dienen der laufenden Verarbeitung und sind nicht dasselbe wie dauerhaft gespeicherte Dateien auf der Platte.

Rate-Limit

Begrenzung, wie oft eine Aktion in einem Zeitraum ausgeführt werden darf. Ephraim nutzt Rate-Limits, um Dienste vor Missbrauch, Überlastung und Fehlbedienung zu schützen.

Recovery-Code

Einmalcode für die Kontoabsicherung, zum Beispiel als Ersatzweg bei Zwei-Faktor-Problemen. Recovery-Codes sind Zugangsdaten und müssen wie Passwörter geschützt werden.

Redis / Valkey

Schneller, kurzlebiger Speicher für laufende technische Zustände. In Ephraim ist Redis/Valkey kein dauerhafter Speicher für private Inhalte und nicht von außen erreichbar.

Retrieval

Auswahl passender Quellenabschnitte vor der Antwort. Ephraim nutzt dafür Rechteprüfung, Kontextregeln und Embedding-Vektoren in der Webanwendung. Die Vektoren kommen vom lokalen SGLang-Embedding-Dienst; die Suche selbst läuft in der Webanwendung und ist keine freie Websuche.

Runtime

Betriebsschicht für technische Statusdaten und kontrollierte Aktionen rund um die Spark-Dienste.

Salt

Zufälliger Zusatzwert bei Passwort- oder Schlüsselableitungen. Ein Salt ist nicht geheim, verhindert aber, dass gleiche Passwörter automatisch gleiche abgeleitete Werte erzeugen.

SchG

Schulgesetz. Für Ephraim ist es ein rechtlicher Bezugspunkt, weil schulische Datenverarbeitung nicht nur technisch, sondern auch aus dem Bildungsauftrag heraus begründet werden muss.

ServerCrypto

Ephraims serverseitiger Verschlüsselungsbaustein für bestimmte technische Daten. Er schützt zum Beispiel Session- oder Betriebsdaten mit Server-Schlüsseln und Zweckbindung.

Session / Sitzung

Angemeldeter Arbeitszustand zwischen Browser und Server. In Ephraim ist die Sitzung wichtig, weil in ihr Rechte, CSRF-Schutz und der entsperrte Vault zusammenkommen.

SGLang

Lokaler Diensttyp für KI- und Embedding-Inferenz. Ephraim nutzt `school-ui-embedding` als lokalen SGLang-Embedding-Dienst auf der Spark.

SHA-256

Hashverfahren mit 256-Bit-Ausgabe. Ephraim nutzt solche Fingerabdrücke, um Dateien oder Artefakte eindeutig wiederzuerkennen, ohne daraus den Inhalt zurückzurechnen.

SMILES

Textschreibweise für chemische Strukturen. Ephraim nutzt sie, um lokal eine 2D-Strukturformel zu zeichnen.

Smoke-Test

Kleiner, schneller Grundtest. Er prüft nicht jede Einzelheit, sondern ob ein wichtiger Ablauf oder eine Komponente grundsätzlich lädt und eine minimale fachliche Funktion ausführt.

Spark

Die lokale KI-Hardware von Ephraim. Auf ihr laufen Inferenz, Embeddings und Spark-TTS; sie ist kein öffentlicher Webdienst und speichert keine fachlichen KI-Anfragen dauerhaft.

SSE

Server-Sent Events. Technik, mit der ein Server fortlaufend kleine Ereignisse an den Browser senden kann, zum Beispiel Antwortstücke während eines Chatstreams.

Supervisor

Admin-Dienst für Jobstatus, geplante Updates, Protokolle und besonders kontrollierte Betriebsaktionen.

SVG

Vektorformat für Grafiken im Browser. Ephraim nutzt SVG für Diagramme und Visualisierungen, setzt Inhalte aber kontrolliert, damit daraus kein freier HTML- oder JavaScript-Code wird.

TDDDG

Telekommunikation-Digitale-Dienste-Datenschutz-Gesetz. Es ist unter anderem relevant, wenn Informationen auf Endgeräten gespeichert oder ausgelesen werden.

Technische Schuld

Folgekosten von Abkürzungen im Code, in Tests, Architektur oder Dokumentation. KI kann solche Schuld schneller erzeugen, aber auch früher sichtbar machen und beim Aufräumen helfen.

TeX

Satzsystem für technisch und mathematisch anspruchsvolle Dokumente. Im Chat-Export ist TeX vor allem für Formeln und spätere Weiterverarbeitung gedacht.

Token (Berechtigung)

Ein zeitlich oder fachlich begrenzter Berechtigungsnachweis, zum Beispiel für interne Dienste oder OAuth-Clients.

Token (KI)

Kleine Texteinheit, mit der das Modell rechnet. Ein Token kann ein Wort, ein Wortteil, ein Satzzeichen oder ein Leerzeichen mit Wortanfang sein.

TOTP

Time-based One-Time Password. Das ist der zeitabhängige sechsstellige Code aus einer Authenticator-App für Zwei-Faktor-Anmeldung.

TTS

Text-to-Speech: Ephraim wandelt ausgewählten Chattext über den internen Spark-Dienst in gesprochene Sprache um.

URL

Adresse einer Ressource im Web oder in einem Dienst. Eine private Kalender-URL ist sensibel, weil sie häufig direkten Zugriff auf Kalenderdaten ermöglicht.

User-DEK

Der Datenschlüssel, mit dem Vault-Inhalte entschlüsselt werden. Er wird aus der Sitzung genutzt und bleibt nicht als Klartext in der Datenbank.

Vault

Der persönliche verschlüsselte Datenbereich eines Kontos. Darin liegen private Chats, Dateien, Erinnerungen, private Kalender und abgeleitete Zusammenfassungen.

Vektor

Eine geordnete Liste von Zahlen. Bei Embeddings beschreibt sie die Bedeutung eines Textes für die semantische Suche.

Verfahrensverzeichnis

Dokumentation einer Verarbeitungstätigkeit nach Datenschutzrecht. Es beschreibt unter anderem Zweck, Datenarten, Rollen, Empfänger, Schutzmaßnahmen und Löschlogik.

VRAM

Speicher auf der Grafikkarte. Große Sprachmodelle brauchen viel VRAM, weil Modellgewichte und laufende Berechnungen dort Platz finden müssen.

WAV

Audioformat. Ephraim nutzt WAV für Spark-TTS-Antworten; der Proxy hält das WAV nur im RAM und schreibt keine Audiodatei auf die Serverplatte.

Waveform

Grafische Darstellung einer Audiospur. Ephraim berechnet die Waveform lokal im Browser aus dem temporären Audio-Blob und speichert sie nicht.

Web-Speech

Browserfunktion für Sprachausgabe. Ephraim nutzt sie nicht als Fallback, weil die Sprachausgabe kontrolliert über Spark-TTS laufen soll.

Wiederherstellungskontakt

Vertrauensperson mit eigenem Ephraim-Konto. Sie bestätigt im Ernstfall live die Wiederherstellung, erhält aber keinen Zugriff auf fremde Chats, Dateien, Passwörter oder Vault-Schlüssel im Klartext.

XChaCha20-Poly1305

Modernes Verschlüsselungsverfahren mit Manipulationsschutz. Ephraim nutzt es unter anderem für verschlüsselte Daten und Dateistreams.

ZIP

Archivformat, das mehrere Dateien in einer Datei bündelt. Ephraim nutzt ZIP bei Datenexporten nur temporär und löscht das Archiv nach dem Download.

Das Glossar erklärt Begriffe für das Handbuch. Technische Details stehen in den jeweiligen Artikeln.

Quelle: web/manuals/begriffe-in-ephraim/index.html