Nexus Playground
Product & UX Specification

Version 1.1 · 18. November 2026 · Fachliche Baseline / Review-Fassung

Vollständige Konsolidierung der gemeinsam beschlossenen Funktionen, Interaktionen und Releases. Mit acht neu gezeichneten UI- und Architektur-Mockups, Workflows, Datenregeln und Review-Auftrag für Claude.

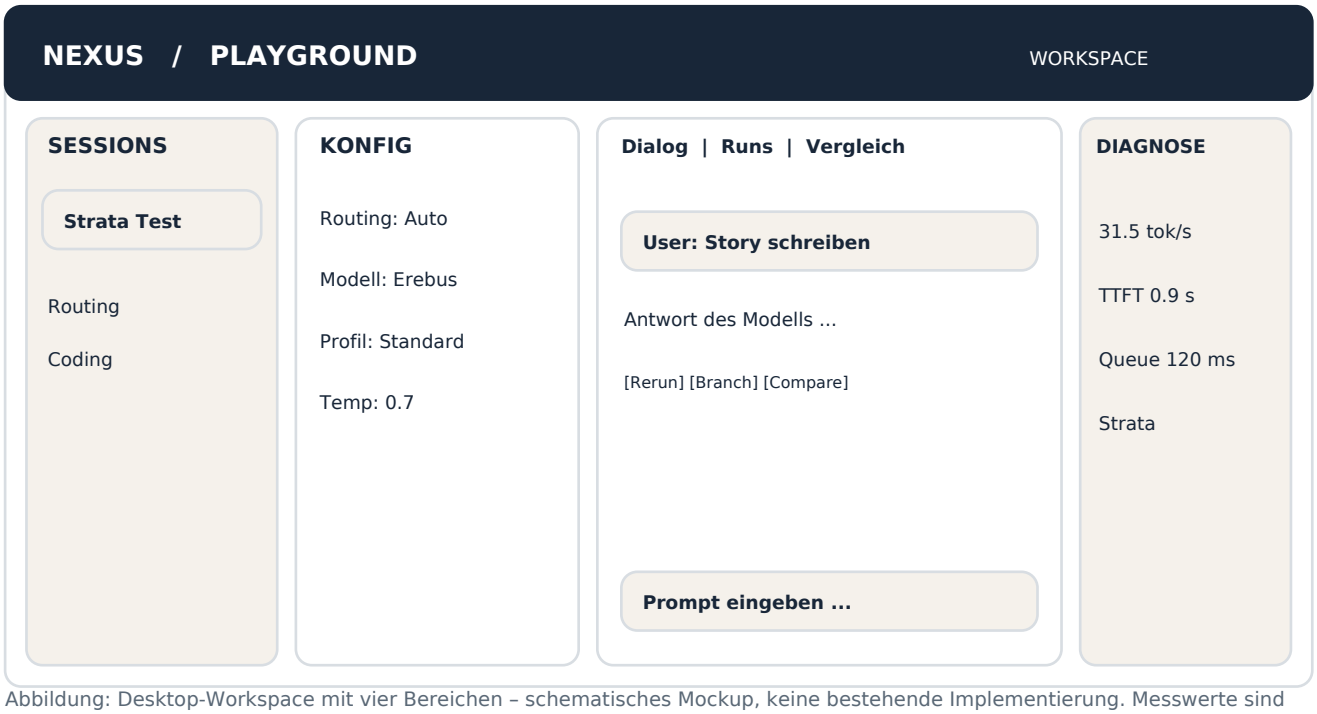


Abbildung: Desktop-Workspace mit vier Bereichen – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

Dokumentstatus: Fachliches Soll-Konzept. Keine Aussage über den tatsächlichen Implementierungsstand. Die technische Machbarkeit ist anhand des Nexus-Repositories zu verifizieren.

1. Leseschlüssel und Geltungsbereich

BESCHLOSSEN: gemeinsam gewählte Produktentscheidung. VORSCHLAG: konkretisierte Ausgestaltung, die beim Architekturreview bestätigt oder geändert werden kann. OFFEN: vor Implementierung zu klären.

Dieses Dokument erweitert das Fachkonzept v1.0. Die früheren interaktiven Chat-Vorschauen sind nicht als exportierbare Bilddateien vorhanden; ihre Inhalte wurden als konsistente statische Mockups neu gezeichnet. Mockups illustrieren Verhalten, nicht pixelgenaue Designvorgaben.

1.1 Produktziel und Abgrenzung

- Chat: alltagstaugliche Unterhaltung, Historie, Modellwahl, TTS; technische Diagnostik nicht im Vordergrund.
- Playground: gezielte Experimente, Kontextkontrolle, Modell-/Engine-Vergleiche, Diagnose, Bewertung und Dokumentation.
- Gemeinsame Basis: Nexus-Gateway, Modellregistrierung, Router, Scheduler, Engine-Lifecycle, Logging und Telemetrie. Keine zweite Inferenz-Infrastruktur.

1.2 Nicht-Ziele für die erste Umsetzung

- Kein frei andockbarer Window-Manager; Panels sind ein-/ausklappbar und größenveränderbar.
- Keine automatische Zeitplan-/Event-Ausführung von Testsuites; Start zunächst explizit per Benutzeraktion.
- Keine Garantie bitidentischer Antworten bei Wiederholung; stattdessen transparente Ausführungsbedingungen.
- Keine dauerhafte Speicherung ungebremster Rohtelemetrie und keine Umgehung des zentralen Schedulers.

2. Beschlussregister – 19 Entscheidungen

Nr. / Thema	Beschluss	Wichtige Konsequenz
01 Produkt	Chat und Playground getrennt	Zwei Oberflächen, gemeinsame Backend-Funktionen
02 Routing	Automatik plus unabhängige Overrides	Requested und actual config getrennt protokollieren
03 Organisation	Sessions als Hauptcontainer	Runs und Vergleiche in Session
04 Kontext	Dialog und isolierte Einzeltests	Exakten gesendeten Kontext sichern
05 Vergleiche	Historische und neue Multi-Modell-Tests	Ressourcenkonkurrenz dokumentieren
06 Diagnose	Erweiterte Metriken auf Abruf	Keine Volltrace-Pflicht im UI
07 Konfiguration	Globale Presets und Session-Einstellungen	Preset als Kopie laden
08 Prompts	Zentrale Bibliothek mit Versionen	Exakte Texte und Versionen speichern
09 Ausführung	Normal und kontrolliert	Keine unzulässige Ressourcenverdrängung
10 Persistenz	Vollständige unveränderliche Run-Historie	Annotationen separat änderbar

Nr. / Thema	Beschluss	Wichtige Konsequenz
11 Navigation	Session-Sidebar + drei Tabs	Runs nicht in Sidebar verschachteln
12 Layout	Flexible Panels, Layout persistent	Tabwechsel überschreibt Benutzerlayout nicht
13 Testdialog	Hybrid mit Rerun, Branch und Compare	Originalverlauf bleibt erhalten
14 Vergleich	Antwortspalten plus Metrikmatrix	2–4 Spalten als UX-Vorschlag
15 Bewertung	Manuell und LLM-Judge	Bewertungen separat speichern
16 Export	Daten, Berichte, Testpakete	Import prüft Kompatibilität
17 Testsuites	Wiederverwendbar, Start per Klick	Definition und Ausführung trennen
18 Routing	Diagnose plus Simulation	Gleiche Entscheidungslogik
19 Releases	Inkrementell nutzbare Releases	R1 bis R5 mit Abnahmekriterien

3. Architektur und Integrationsprinzipien

Der Playground ist ein eigenständiges Fachmodul im Nexus-Frontend und nutzt Application Services für Session-Verwaltung, Testläufe, Vergleiche, Bewertungen, Testsuites und Exporte. Reale Inferenz läuft ausschließlich über bestehende Nexus-Core-Dienste.



Abbildung: Logische Schichten und Wiederverwendung vorhandener Nexus-Dienste – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

3.1 Fachliche Schnittstellen (Vorschlag)

- Test-API: Session-CRUD, Run-Erstellung, Statusabfrage/Streaming, Run-Details, Branching und Vergleiche.
- Konfigurations-API: Presets, Session-Konfiguration, Modell-/Engine-/Profil-Registry, validierte Overrides.
- Diagnose-API: Router-Kandidaten, Ausschlussgründe, Queue-/Ladezeiten, Ressourcen-Snapshots, Correlation-ID.
- Bewertungs-API: Bewertungsschema, manuelle Bewertung, LLM-Judge-Auftrag und Bewertungsverlauf.
- Suite-/Export-API: Suite-Definition, Suite-Ausführung, Ergebnisaggregation, Export und Import.

3.2 Integrationsinvarianten

- Jeder echte Run besitzt eine eindeutige Run-ID und eine durchgängige Correlation-ID zur Nexus-Anfrage.
- Requested Routing (Benutzervorgabe) und Actual Routing (wirkliche Auswahl) werden separat gespeichert.
- Fehlerhafte Override-Kombinationen werden vor Ausführung verständlich abgelehnt, nicht stillschweigend ersetzt.
- Ausführung und Modell-Lifecycle respektieren Scheduler, Prioritäten, Warteschlange und Ressourcenverwaltung.
- Routing-Simulation verwendet dieselben Entscheidungsregeln, jedoch einen ausdrücklich simulierten Zustand.

4. Informationsarchitektur und Workspace

4.1 Navigation

Links: einklappbare Session-Liste mit Suche und Session-Auswahl. Innerhalb der Session: Dialog, Testläufe, Vergleiche. Testläufe erscheinen in ihrem eigenen Tab, nicht als endloser Baum in der Sidebar.

4.2 Panels und Layoutpersistenz

Vier Bereiche: Session-Sidebar, Konfiguration, Hauptarbeitsfläche und Diagnose. Seitenbereiche lassen sich unabhängig schließen und in ihrer Breite verändern. Mindestbreiten und eine sinnvolle Standardanordnung verhindern unbenutzbare Layouts. Persönliche Layoutänderungen bleiben beim Tabwechsel erhalten.

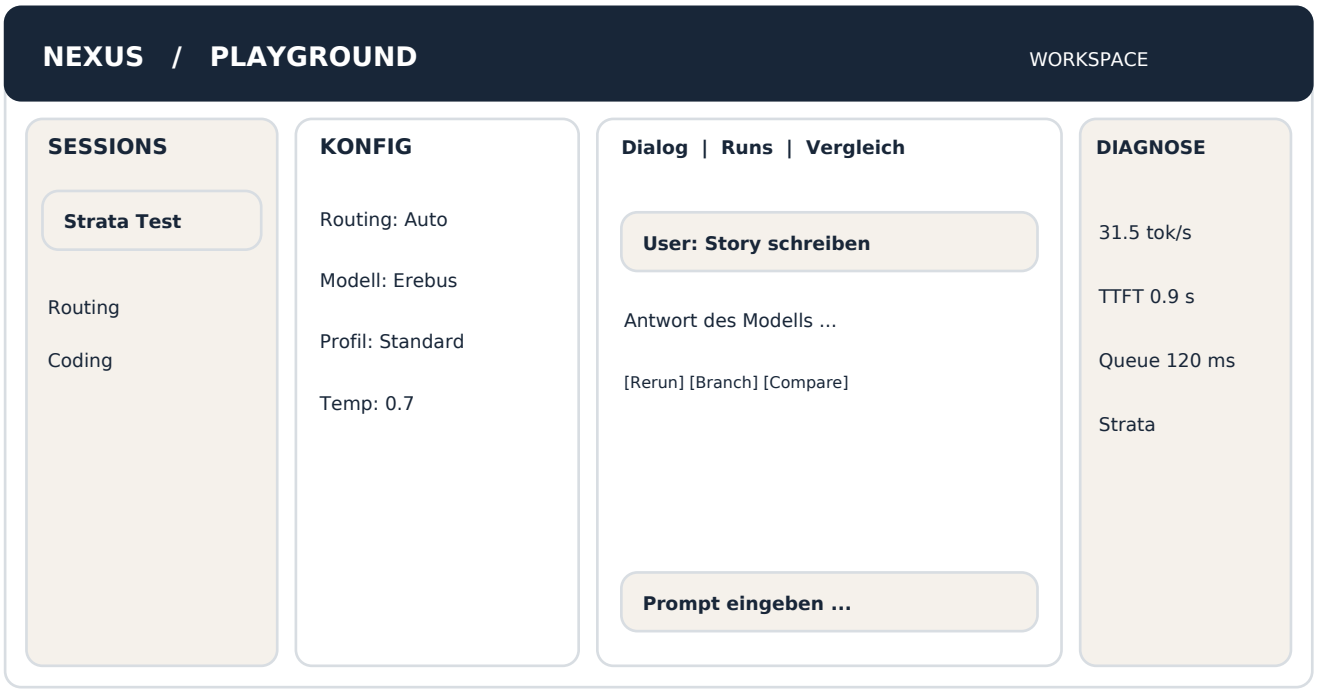


Abbildung: Panelaufteilung und sichtbare Arbeitsbereiche – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

4.3 Zustände und Feedback

- Deutlich erkennbare Zustände: bereit, in Queue, Modell lädt, läuft/streamt, abgeschlossen, abgebrochen, fehlgeschlagen.
- Beim Umschalten von Routing-Modi bleiben nicht anwendbare Overrides sichtbar oder werden begründet deaktiviert.
- Ein laufender Test darf beim Tabwechsel nicht unbemerkt verschwinden; Status bleibt übergreifend nachvollziehbar.

5. Hybrider Testdialog – Verhalten

BESCHLOSSEN: Ein chatähnlicher Dialog mit technischen Aktionen an jeder Antwort. Der Kontextmodus lässt sich zwischen Dialog und Isoliert wechseln. Die Bedienung bleibt lesbar; tiefe Diagnosen werden im Diagnosepanel gezeigt.

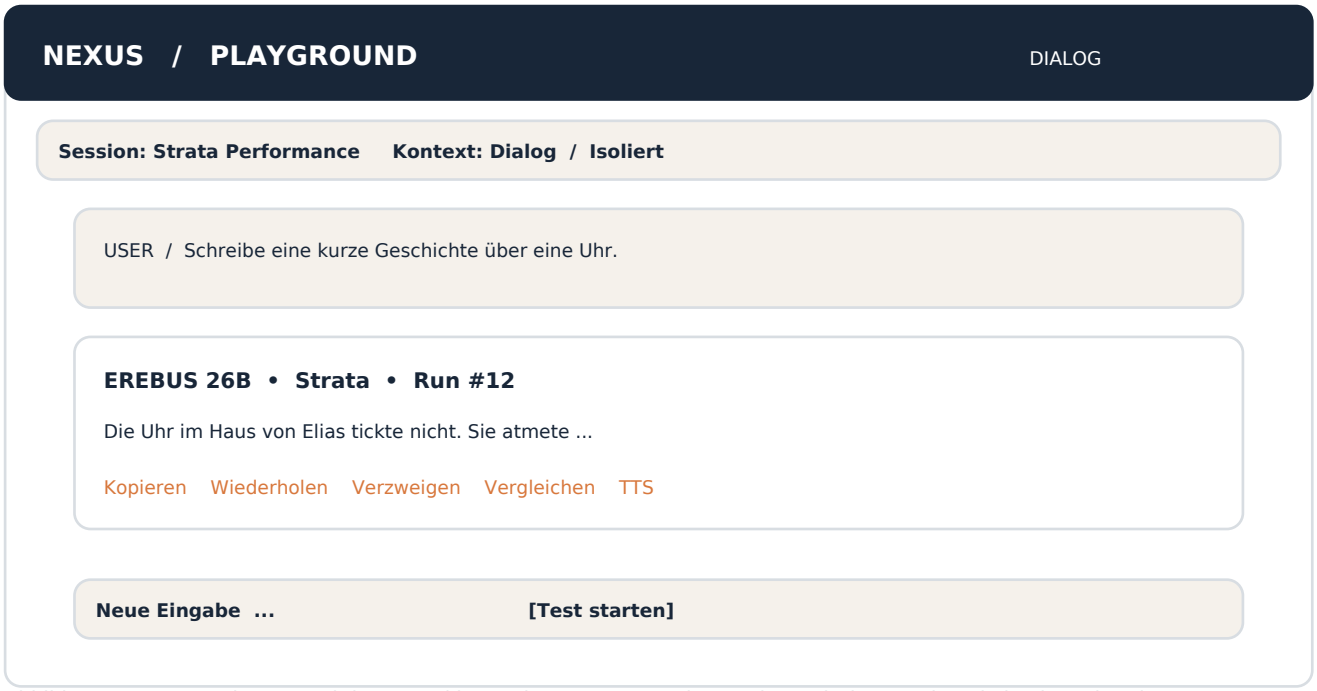


Abbildung: Kontextmodus, Run-Aktionen und kompakte Antwortanzeige – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

5.1 Aktionen pro Antwort

- Wiederholen: erneuter Run mit bewusst ausgewählter eingefrorener oder aktueller Konfiguration; das Ergebnis ist immer ein neuer Run.
- Verzweigen: neuen Dialogzweig ab einer früheren Nachricht beginnen; ursprünglicher Verlauf bleibt unverändert.
- Vergleichen: bestehenden Run zu einer Vergleichsauswahl hinzufügen; keine neue Inferenz erforderlich.
- Kopieren / TTS: optionale Antwortaktionen, soweit die vorhandenen Nexus-Dienste diese unterstützen.

5.2 Dialogkontext und Branching

Im Dialogmodus wird die tatsächliche Folge der an das Modell gesendeten Nachrichten festgehalten, einschließlich System-Prompt, Rollen und Reihenfolge. Im isolierten Modus wird keine frühere Unterhaltung implizit übertragen. Ein Branch referenziert einen Vorgängerpunkt und erzeugt neue Runs; alte Antworten werden nicht überschrieben.

5.3 Beispiel-Workflow: Branch mit anderem Modell

1. Nutzer öffnet Run #12. 2. Aktion „Verzweigen“ startet einen neuen Zweig ab der gewählten Nachricht. 3. Modell oder Profil wird geändert. 4. Nexus zeigt die wirksame Konfiguration vor Start. 5. Neuer Run erhält eigene ID und verknüpften Branch-Ursprung. 6. Beide Runs können verglichen werden.

6. Konfiguration, Presets und Prompt-Bibliothek

6.1 Konfigurationsebenen

- Globales Preset: wiederverwendbare Vorlage; beim Laden in eine Session wird eine Kopie erzeugt.
- Session-Konfiguration: aktuelle Arbeitswerte; Änderungen wirken nur auf zukünftige Runs.
- Run-Snapshot: tatsächlich übertragene Parameter, System-/User-Prompts, Kontext, Profil, Engine, Modell, Quantisierung und wirksame Overrides; nach Abschluss unveränderlich.

6.2 Routing-Overrides

Automatik ist Standard. Engine, Modell, Profil und weitere unterstützte Optionen dürfen unabhängig überschrieben werden. Der Backend-Validator prüft Verfügbarkeit, Kompatibilität und Berechtigung. Der UI-Zustand allein gilt nicht als Validierung.

6.3 Prompt Library

Zentrale Bibliothek für System- und User-Prompts, Metadaten, Tags und versionierte Inhalte. Gespeicherte Versionen werden nicht rückwirkend verändert. Jeder Run dokumentiert sowohl die Prompt-Version als auch den exakt gesendeten Text. Bibliothek kann später von anderen Nexus-Modulen genutzt werden.

7. Ausführungsmodi und Fairness

- Normal: über den gemeinsamen Nexus-Scheduler unter realen Betriebsbedingungen; Last und Queuezeiten werden protokolliert.
- Kontrolliert: Vorabprüfungen, Reservierung der durch Nexus verwalteten Ressourcen und definierte Ausführungsbedingungen; laufende fremde Jobs werden nicht willkürlich beendet.
- Externe GPU-/CPU-Last kann die Ergebnisse trotzdem beeinflussen. Kontrolliert bedeutet keine garantierte physische Isolation.
- Parallele Modellvergleiche können durch Ressourcenkonkurrenz verzerrt werden; sequentielle Vergleiche haben andere Umgebungsbedingungen. Ausführungsstrategie muss dokumentiert sein.

8. Vergleichsansicht

BESCHLOSSEN: kombinierte Darstellung mit Antwortspalten und technischer Metrikmatrix. Der Nutzer kann historische Runs auswählen oder neue Vergleiche starten. Antworten bleiben direkt nebeneinander sichtbar; Detaildiagnosen sind optional.

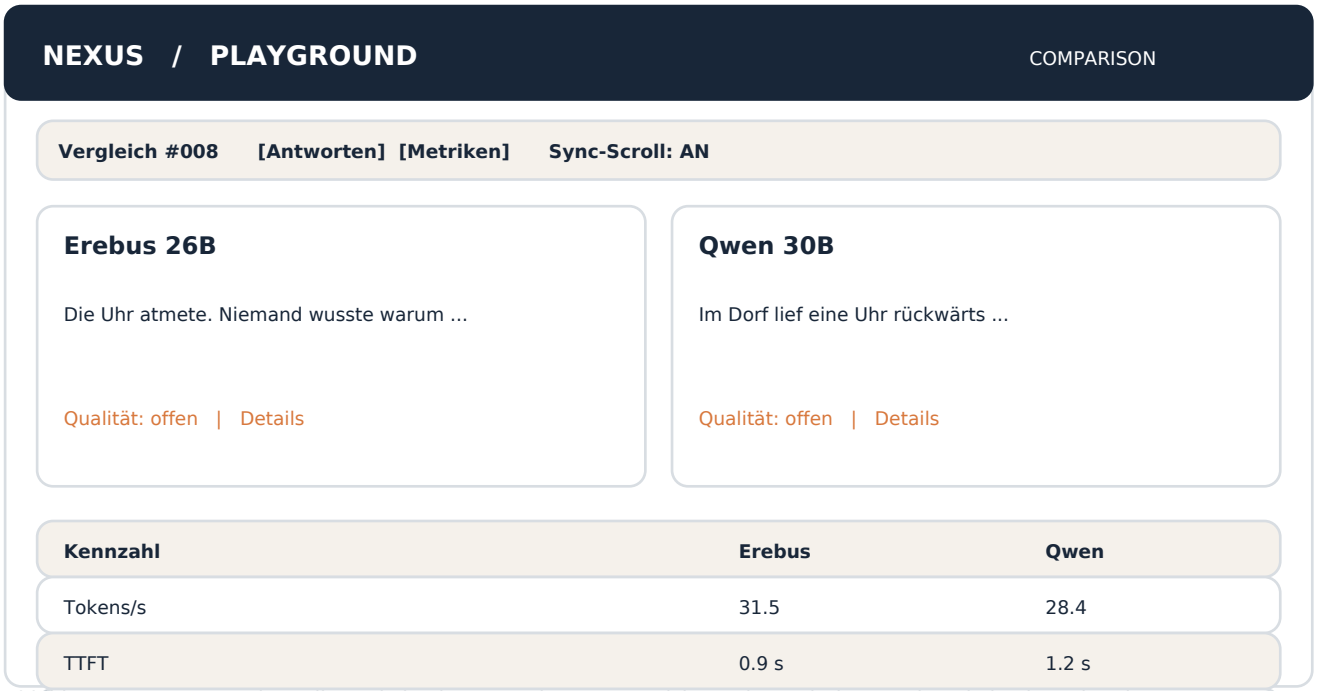


Abbildung: Antworten und Metrikmatrix in einer gemeinsamen Ansicht – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

8.1 Interaktion (Vorschlag)

- Zwei bis vier Antwortspalten bei ausreichender Breite; bei kleineren Displays horizontal navigierbar.
- Synchrones oder unabhängiges Scrollen; lange Antworten dürfen nicht die gesamte Seite unkontrolliert verlängern.
- Vergleich speichert Run-Referenzen und optional Bewertungen; Antworten werden nicht dupliziert.
- Abweichende Engine-, Modell-, Profil-, Quantisierungs- und Ausführungsbedingungen deutlich kennzeichnen.
- Qualität und Performance getrennt darstellen: keine automatische Gesamtsieger-Kür allein aus Tokens/s.

8.2 Vergleichsmetrik und Interpretation

Mindestens: TTFT, Ausgabe-Tokens/s, Dauer, Input-/Output-Tokens, Queuezeit, Modellladezeit (sofern vorhanden), Engine, Modell, Profil, Ressourcen-Snapshots, Fehlermeldungen. Nicht verfügbare Messwerte werden als „nicht erhoben“ gekennzeichnet, niemals mit Null gleichgesetzt.

9. Qualitätsbewertung

BESCHLOSSEN: manuelle und LLM-basierte Bewertungen werden unabhängig gespeichert. Bewertungsschemata sind wiederverwendbar und können je Aufgabentyp variieren (z. B. Coding, Text, Kreativität).

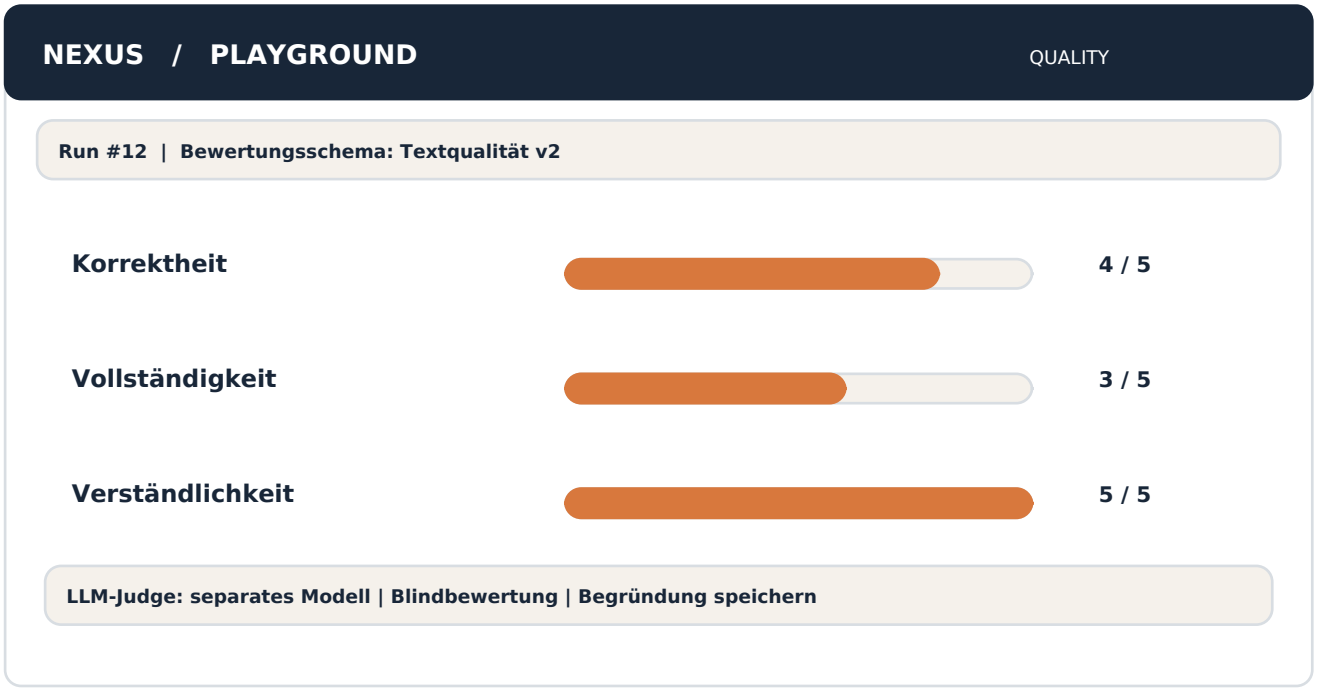


Abbildung: Manuelle Kriterien und separater LLM-Judge – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

9.1 Bewertungsschema

- Schema enthält Version, Kriterien, Skalen, Gewichtungen (optional) und Bewertungsanweisungen.
- Manuelle Bewertungen dokumentieren Urheber, Zeitpunkt, Schema-Version, Einzelwerte und optionale Kommentare.
- LLM-Judge dokumentiert Judge-Modell, Prompt-Version, Referenzantwort (falls vorhanden), Einzelwerte und Begründung.
- Blindbewertung ohne Modellnamen wird als Option vorgesehen, um erkennbare Identitätsverzerrungen zu reduzieren.
- Bewertungen überschreiben weder Runs noch frühere Bewertungen; erneute Bewertung erzeugt einen neuen Datensatz.

9.2 Grenzen automatischer Bewertung

LLM-Judges können falsch, inkonsistent oder voreingenommen urteilen. Eine Bewertung ist ein gesondertes Urteil, keine objektive Ground Truth. Bei sicherheits- oder korrektheitskritischen Benchmarks sind Referenztests oder menschliche Prüfung vorzuziehen.

10. Diagnose und Routing-Simulation

Erweiterte Diagnose ist runbezogen und bei Bedarf sichtbar. Routing-Analyse zeigt geprüfte Kandidaten, Auswahlgründe und Ausschlussgründe. Simulation verändert ausschließlich angenommene Zustände, startet keine Inferenz und reserviert keine realen Ressourcen.

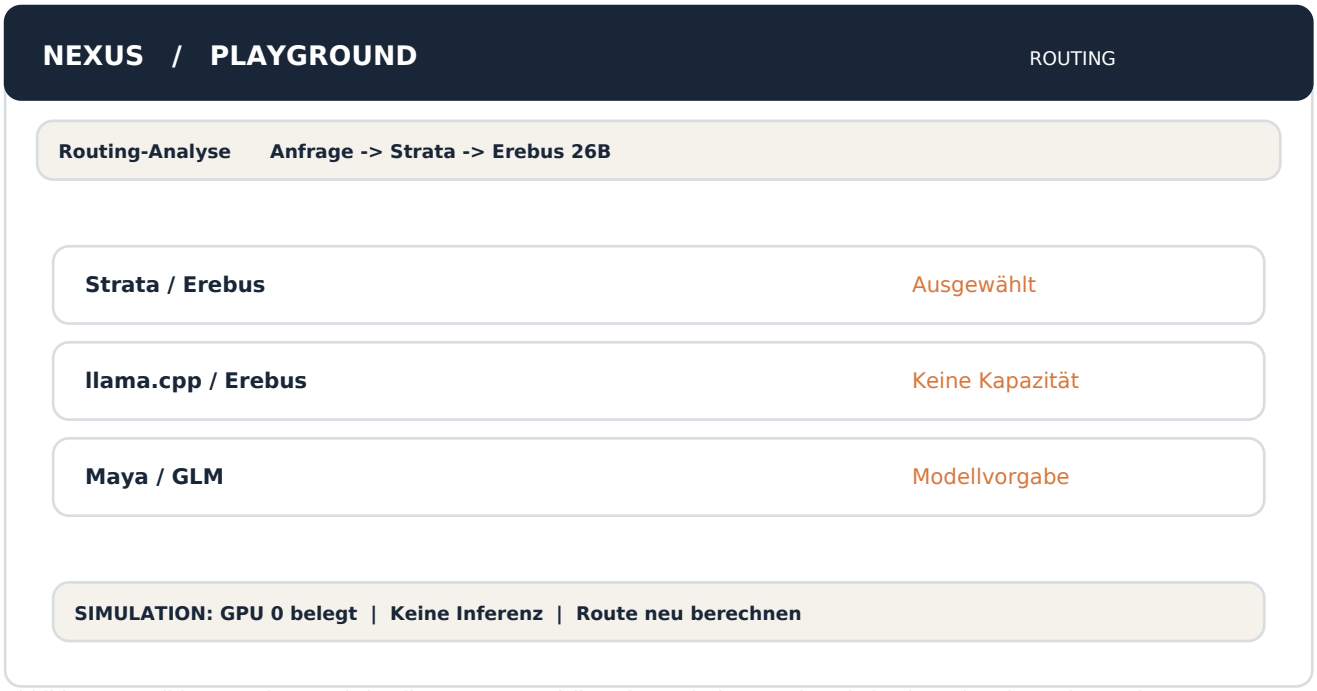


Abbildung: Kandidatenanalyse und simulierter GPU-Ausfall – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

10.1 Diagnosefelder

- Run-ID, Correlation-ID, Requested/Actual Routing, Engine- und Modellversion, Fehlercodes und verständliche Fehlertexte.
- Zeitpunkte und Dauern: Annahme, Queue, Modell-Lifecycle, TTFT, Streaming, Abschluss.
- Ressourcen: GPU-/CPU-Auslastung, VRAM/RAM, Gerätezuordnung und Lastindikatoren, soweit zuverlässig verfügbar.
- Messquelle, Messzeitpunkt und fehlende Werte kenntlich machen; keine falsche Präzision suggerieren.

10.2 Simulations-Workflow

1. Ausgangsrequest und Routing-Regeln auswählen. 2. Ressourcen-/Engine-Zustände überschreiben (z. B. GPU 0 belegt, Engine offline, VIP-Priorität). 3. Router mit simuliertem Zustand auswerten. 4. Kandidaten und Entscheidung anzeigen. 5. Simulation optional als Analyseartefakt sichern. Keine echte Inferenz.

10.3 Wichtige technische Bedingung

Routing-Simulation darf keine zweite Entscheidungslogik implementieren. Der produktive Router muss einen prüfbaren Entscheidungsweg bereitstellen, der mit einem expliziten Zustands-Snapshot aufgerufen werden kann. Ob die vorhandene Architektur dies erlaubt, ist offen und vor R5 zu prüfen.

11. Testsuites und Ausführung

BESCHLOSSEN: wiederverwendbare Testsuites mit manuellem Start. Sie können isolierte Prompts, Dialogabläufe und Vergleiche kombinieren, mehrfach wiederholen und Ergebnisse aggregieren.

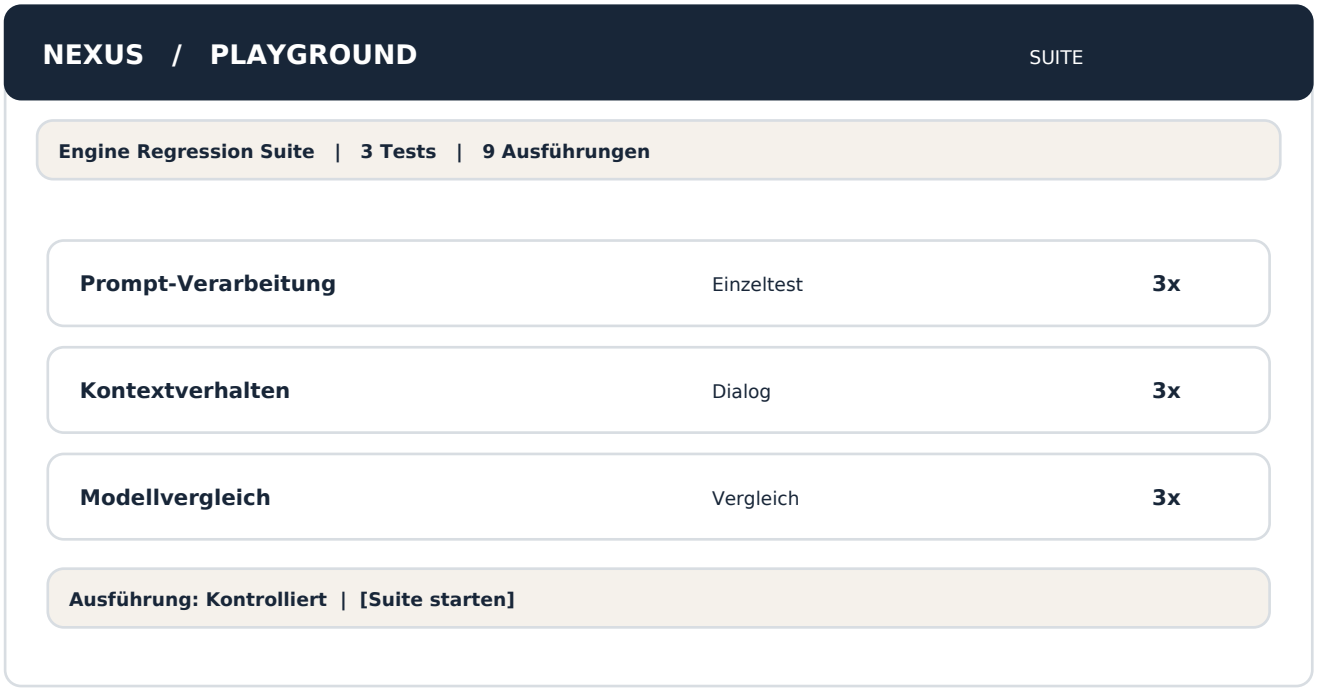


Abbildung: Wiederverwendbare Regression-Suite mit drei Testfällen – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

11.1 Suite-Definition versus Suite-Execution

- Suite-Definition: versionierte Testfälle, Prompts, Konfigurationen, Wiederholungszahl, Reihenfolge, Bewertungsregeln und Ausführungsmodus.
- Suite-Execution: Zeitpunkt, Startkontext, erzeugte Run-IDs, Ausführungsstatus, Fehler und aggregierte Ergebnisse.
- Wiederholungen sind eigenständige Runs; keine Vermischung von Durchschnittswerten und Rohdaten.
- Ereignis- und Zeitplantrigger gehören ausdrücklich nicht zum beschlossenen aktuellen Umfang.

12. Export, Berichte und importierbare Testpakete

BESCHLOSSEN: JSON, CSV und Markdown für Einzel-/Vergleichsdaten; strukturierte Berichte; versionierte portable Testpakete für spätere Importe und erneute Ausführung.

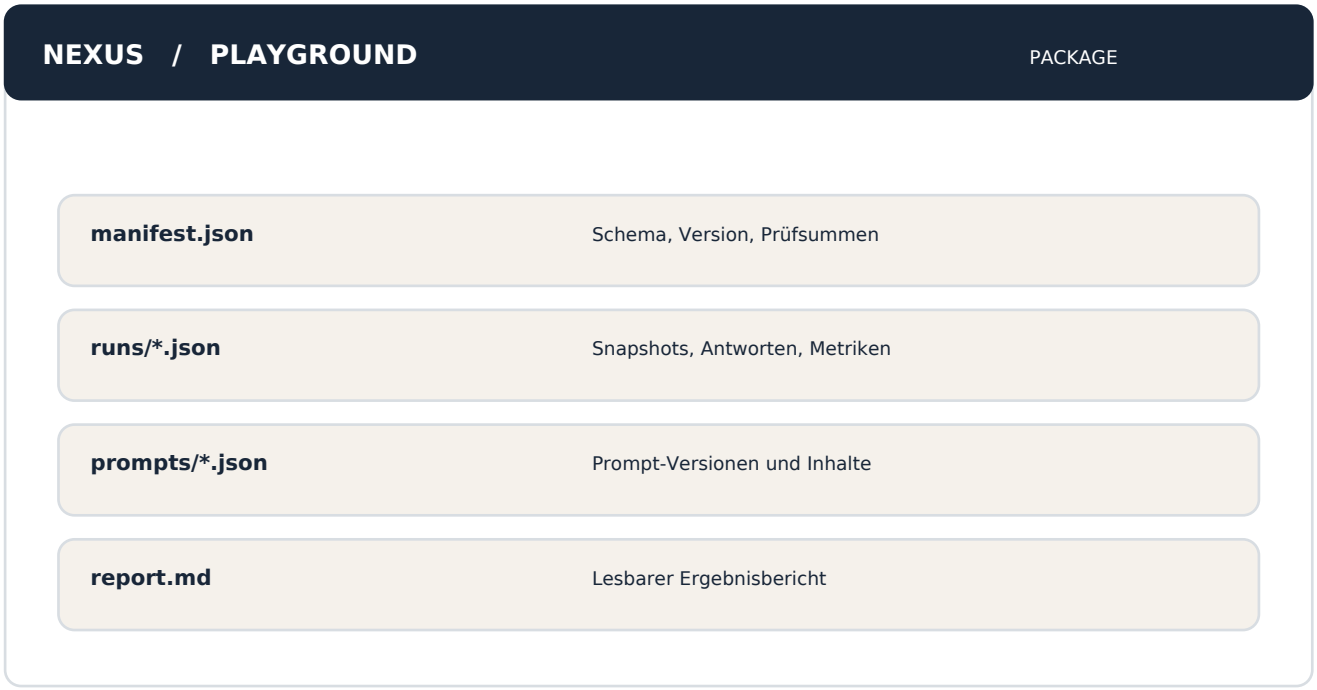


Abbildung: Beispielhafte Struktur eines Nexus-Testpakets – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

12.1 Mindestanforderungen an das Austauschformat

- Versioniertes Manifest mit Schema-Version, Erstellungszeit, Herkunft, enthaltenen Objekten und Prüfsummen.
- Runs enthalten unveränderte Eingaben, Konfigurations-Snapshots, Routing, Antwort, Metriken und Ausführungsbedingungen.
- Prompt-Versionen und Bewertungsschemata müssen entweder enthalten oder eindeutig referenzierbar sein.
- Import prüft Datenformat, Integrität, zulässige Pfade, Kompatibilität und vorhandene Modell-/Engine-Abhängigkeiten.
- Fehlende Abhängigkeiten werden ausgewiesen; es gibt keine stille Substitution. Importierte Ergebnisse bleiben als historische Daten nutzbar.

12.2 Reproduzierbarkeit realistisch definieren

Wiederholung bedeutet: dieselbe dokumentierte Eingabe und bestmöglich dieselbe technische Konfiguration erneut verwenden. Gleiche Antwortbytes sind wegen nichtdeterministischer Inferenz, Hardware- und Versionsunterschieden nicht garantiert. Abweichungen werden sichtbar gemacht.

13. Fachliches Datenmodell und Lebenszyklus

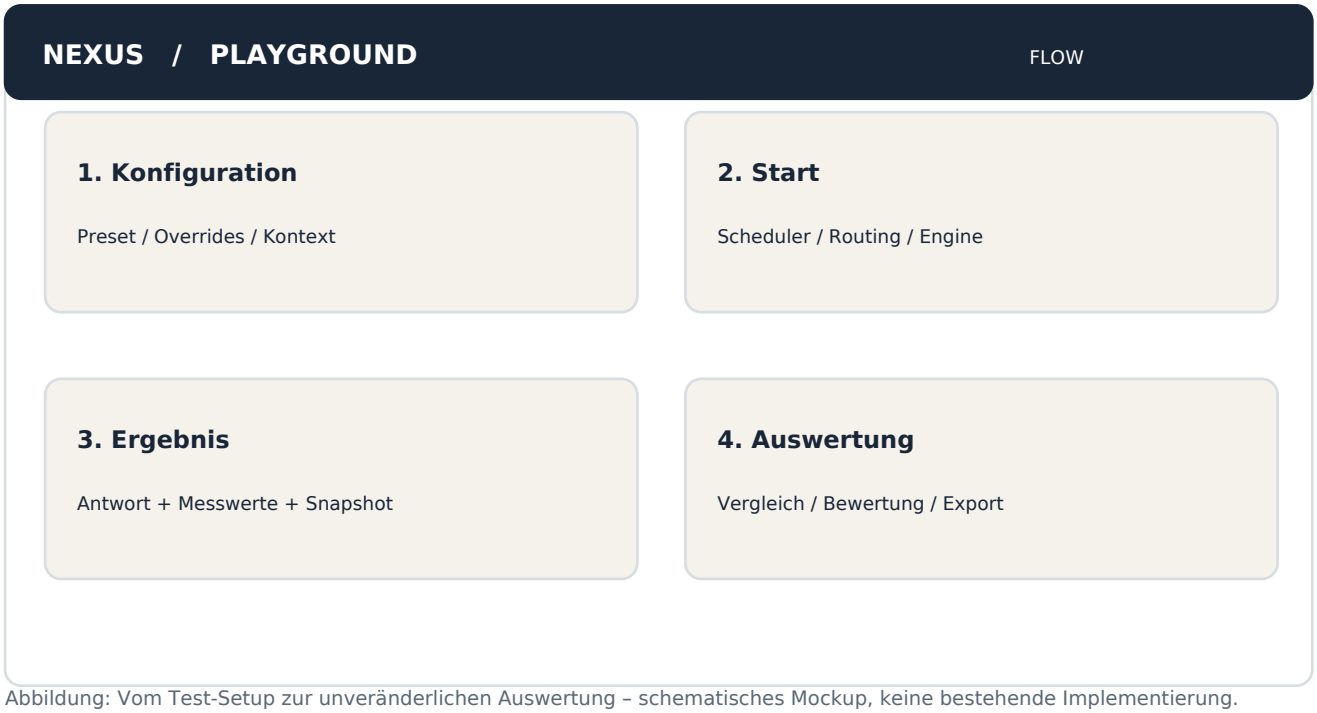


Abbildung: Vom Test-Setup zur unveränderlichen Auswertung – schematisches Mockup, keine bestehende Implementierung. Messwerte sind fiktiv.

Objekt	Wesentliche Inhalte / Regeln
TestSession	Metadaten, aktuelle Konfiguration, Referenzen auf Runs und Vergleiche
TestRun	Run-ID, Status, Request/Response, exakter Kontext, Snapshot, Routing, Metriken, Fehler
DialogBranch	Elternpunkt / Vorgänger-Run, eigene Fortsetzung; Original bleibt unverändert
Comparison	Referenzen auf Runs, Anzeigeoptionen, Notizen, ggf. Bewertungen
Prompt / PromptVersion	Bibliotheksmetadaten, immutable Versionen, exakter Inhalt
Preset	Globale Vorlage; beim Laden Kopie in Session
Evaluation / Schema	Schema-Version, Kriterien, manuelle oder LLM-Werte, Begründung
TestSuite / Execution	Wiederverwendbare Definition und konkrete Ausführung mit Runs
RoutingSimulation	Eingangsrequest, simulierte Zustände, Router-Ergebnis, Zeitpunkt
ExportPackage	Manifest, Schema-Version, Prüfsummen, referenzierte Artefakte

13.1 Run-Lebenszyklus (Vorschlag)

CREATED → QUEUED → PREPARING → RUNNING/STREAMING → COMPLETED; alternative Endzustände FAILED oder CANCELLED. Nicht jeder Engine-Adapter muss alle Zwischenzustände liefern; fehlende Telemetrie wird explizit markiert. Abgeschlossene Runs werden nicht inhaltlich editiert.

13.2 Datenschutz und Sicherheit

Prompts, Antworten und importierte Pakete können vertrauliche Inhalte enthalten. Rollen-/Zugriffskontrolle, Exportberechtigungen, Importlimits, Pfadvalidierung und Redaction von Secrets in Logs müssen anhand des vorhandenen Nexus-Sicherheitsmodells geprüft werden.

14. Release-Roadmap und Abnahmekriterien

R1 – Playground Core

Umfang: Sessions, flexibler Workspace, Hybrid-Dialog (Grundfunktionen), Auto-/Override-Routing, persistierte Runs, Basisdiagnose.

Abnahme: Session anlegen; isolierten und Dialog-Run starten; Modell/Profil ändern; vorherigen Run mit tatsächlicher Konfiguration und Antwort öffnen; Fehler nachvollziehen.

Abhängigkeiten: Session-/Run-API, Streaming, Snapshot-Schema, Router-Integration.

R2 – Experimentation & Comparison

Umfang: Globale Presets, versionierte Prompt-Bibliothek, Rerun mit Konfigurationswahl, Branching, gespeicherte Vergleiche, Antworten + Metriken.

Abnahme: Prompt mit zwei Modellen ausführen; Ergebnisse nebeneinander vergleichen; Branch erzeugen ohne Original zu ändern; Preset-Änderung verändert alte Runs nicht.

Abhängigkeiten: R1-Persistenz, stabile Kontextabbildung, Prompt-Versionierung.

R3 – Diagnostics & Quality

Umfang: Erweiterte Metriken, Routing-Kandidaten und Ausschlüsse, Bewertungsschemata, manuelle Bewertung, LLM-Judge.

Abnahme: Routingentscheidung und Ausschlussgründe nachvollziehen; manuell und per Judge bewerten; zwei Bewertungen bleiben getrennt; fehlende Messwerte werden korrekt markiert.

Abhängigkeiten: Nexus-Telemetrie, korrelierte Logs, Judge über regulären Scheduler.

R4 – Benchmarking & Automation

Umfang: Kontrollierter Modus, Testsuites, Wiederholungen, Aggregation, JSON/CSV/Markdown, Berichte, Paket-Import/-Export.

Abnahme: Suite mehrfach ausführen; Runs und Aggregate prüfen; Export/Import validieren; fehlende Modelle melden; kontrollierte Ausführung beendet keine fremden Jobs.

Abhängigkeiten: Scheduler-Reservierung, Paketformat, Validierung und Migrationen.

R5 – Routing Simulation

Umfang: Simulierte Engine-/Ressourcenzustände, Kandidatenvergleich, VIP-/Override-Szenarien, speicherbare Simulationen.

Abnahme: GPU-Ausfall und Engine-Offline simulieren; Entscheidung erklären; keine Inferenz und keine Ressourcenreservierung auslösen.

Abhängigkeiten: Router-Entscheidungsfunktion mit injizierbarem Zustands-Snapshot.

14.1 Release-Gates

- Jedes Release hat nachvollziehbare Abnahmetests, Unit-/Integrationstests, CI-Prüfungen, Migrationen und Dokumentation.
- Schnittstellen zu bestehenden Nexus-Modulen werden vor Änderung geprüft und möglichst rückwärtskompatibel erweitert.

- Keine Terminbindung im Konzept; Reihenfolge folgt fachlichem Nutzen und technischen Abhängigkeiten.

15. Teststrategie und Qualitätsanforderungen

- Unit-Tests: Konfigurationsauflösung, Snapshot-Erstellung, Vergleichsberechnung, Schema-/Importvalidierung, Branch-Kontext.
- Integration: Gateway/Router/Scheduler/Engine-Adapter; echte und fehlgeschlagene Runs; Streaming und Abbruch.
- Regression: Preset-Änderungen mutieren keine alten Runs; Judge-Bewertung verändert keine Modellantwort; Branch bleibt getrennt.
- E2E: Session → Test → Rerun → Compare → Bewertung → Export; unterschiedliche Browserbreiten und persistierte Panels.
- Betrieb: strukturierte Logs, Fehlercodes, Trace-/Correlation-ID, ressourcenschonende Speicherung und migrationsfähige Datenbank.

16. Offene Architekturfragen für den Coding-Agenten

- Welche Module und APIs für Chat, Sessions, Scheduler, Router, Engine-Lifecycle und Telemetrie existieren bereits?
- Wie wird derzeit der effektive Modell-/Engine-/Profilzustand einer Anfrage ermittelt und geloggt?
- Welche Datenbank, Migrationstechnik und Aufbewahrungsregeln sind vorhanden?
- Welche Frontend-Technologien, UI-Komponenten und Design-Tokens nutzt Nexus?
- Welche Telemetrie ist pro Engine zuverlässig verfügbar; welche Werte fehlen?
- Wie lassen sich kontrollierte Ressourcenreservierung und Routing-Simulation ohne Router-Duplikat integrieren?
- Welche bestehenden Authentifizierungs-, Autorisierungs- und Secret-Redaction-Regeln gelten?
- Welche Schnittstellen müssen für die OpenAI-Kompatibilität unverändert bleiben?
- Welche technischen Risiken verschieben Teile von R1 oder erfordern eine vorgelagerte Grundlage?

17. Konkreter Review-Auftrag an Claude

Rolle: Kritischer Solution Architect / Senior Engineer. Auftrag: Dieses Fachkonzept gegen das aktuelle Nexus-Repository prüfen. Zunächst ausschließlich lesen und analysieren; keine Dateien ändern, keine Abhängigkeiten installieren, keine Dienste starten oder stoppen.

Erwartete Ergebnisse: (1) Ist-Architektur und wiederverwendbare Komponenten; (2) Widersprüche, Risiken und technische Lücken mit Schweregrad; (3) Vorschläge für modulare Integration; (4) überprüfte Daten- und API-Verträge; (5) realistische, abhängige R1-Arbeitspakete; (6) Tests und Abnahmekriterien; (7) explizite Rückfragen statt stiller Annahmen.

Wichtige Leitplanke: Fachliche Beschlüsse nicht stillschweigend umdeuten. Technische Gegenargumente sind ausdrücklich erwünscht, aber mit Begründung und Alternative zu dokumentieren. Kein Code vor Freigabe.

18. Änderungslog und Anhang

Version	Änderung
v1.0	Kompaktes Fachkonzept mit Release-Übersicht und schematischen Abbildungen.
v1.1	Vertiefte UX-/Funktionsspezifikation, acht neu gezeichnete Mockups, Workflows, Datenregeln, Abnahmekriterien, Review-Auftrag.

Legende: Sämtliche Screens und Metriken in diesem Dokument sind illustrative Darstellungen. Modellnamen, Zeiten und Antworten in den Mockups dienen nur der Veranschaulichung.