SkillPilot Entwicklerhandbuch · Band VII

Entwickler-, Schema- und Editor-Handbuch

Dokumentationsrichtlinien, Landscape-JSON-Feldreferenz, Laufzeit-Fallbacks,

Autoreneditoren, Geschwisterknoten-Sortierung, Liquibase und SSE-Auto-Update

Geltungsbereich dieses Handbuchs

Dieser Band bündelt die dauerhaften Entwickler-, Schema-, Autorenwerkzeug- und Plattformreferenzen von SkillPilot: die Dokumentationsrichtlinien samt allen elf zweisprachigen Single-Source-Kompendien und dem `[ DE | EN ]`-Sprachumschalter, die Feldreferenz für `SkillLandscape`-JSONs, das minimale Projektionsschema mit Laufzeit-Fallbacks, die vier lokalen Vite-Autoreneditoren, die topologische Geschwisterknoten-Sortierung unter „Meine Lernziele“ sowie Datenbankmigrationen mit Liquibase und Echtzeit-Updates per Server-Sent Events (SSE).

Stand: 11. November 2026 · Lizenz: Apache-2.0 (Software-, Schema- und Entwicklerdokumentation; siehe LICENSING.md).

Datierte Migrations-Workboards und Rollout-Dossiers unter docs/dev/ bleiben eigenständige Arbeitsdokumente.

Inhaltsverzeichnis

1 Dokumentationsrichtlinien **3**
1.1 Dokumentrollen und Ablageregeln . . . 3
1.2 Zweisprachige Single-Source-Kompendien (`docs/compendia.json`) . . . 3
1.3 MkDocs-`[ DE | EN ]`-Schalter, Generierungshinweise, Vorlagen und Checks . . . 4

2 Landscape-JSON-Feldreferenz **5**
2.1 Top-Level-Felder einer `SkillLandscape` . . . 5
2.2 `LearningGoal`, `dimensionTags` und Provenienzfelder . . . 5

3 Minimales Projektionsschema, Laufzeit-Fallback und Mathe-Pilot **6**
3.1 Additive Schemafelder und Fallback-Reihenfolge . . . 6
3.2 Einstiegsscope-Auflösung und kanonischer Mathematik-Pilot . . . 6

4 Curriculum-Autoreneditoren **7**
4.1 Vier lokale Vite-Editoren und Routenarchitektur . . . 7
4.2 Drei-Spalten-Layout, blockierende Prüfungen vs. Warnungen und Phasenfolge . . . 7

5 Graph-Editor **8**
5.1 Fokussierter `requires`-Refactoring-Ablauf . . . 8

6 Flashcard-Editor **9**
6.1 Deck-JSON-Schema, Sprachpaarung und Dual-Write-Speicherung . . . 9

7 Knotensortierung unter „Meine Lernziele“ **10**
7.1 Harte `requires`-Topologiesortierung und sechsstufiger Tie-Breaker . . . 10

8 Datenbankmigrationen mit Liquibase **11**
8.1 Changelog-Struktur und vierstufiger Migrationsablauf . . . 11

9 Echtzeit-Auto-Update-Architektur (SSE) **12**
9.1 Backend-Event-Bus, SSE-Endpunkt und React-Reconnect-Hook . . . 12

1 Dokumentationsrichtlinien

Jede Dokumentationsdatei unter `docs/` besitzt genau eine primäre Rolle, damit dauerhafte Konzepte, QS-Regeln, generierte Statusberichte und datierte Dossiers sauber getrennt bleiben.

1.1 Dokumentrollen und Ablageregeln

| Rolle | Zweck | Typischer Ablageort | Source of Truth? | | — | — | — | — | | **Konzept** | Dauerhaftes Domänenmodell, Semantik und Architektur | `docs/concept/` | Ja | | **Runbook** | Wiederholbare operative Befehlsschritte | Nahe beim Prozessdokument | Ja | | **Prozess** | Review-Regeln, Statusbedeutungen und Entscheidungsgates | `docs/qa-ci/` oder `docs/dev/` | Ja | | **Generierter Status** | Berechneter Bericht, Queue oder Dashboard-Snapshot | `docs/qa-ci/status/` | Nein | | **Pilot oder Dossier** | Datierte Untersuchung, PoC oder einmaliges Audit | `docs/qa-ci/archive/` oder `docs/dev/` | Nein (außer benannt) | | **Implementierungsnotiz** | Entwicklerseitiges Schema-, Tool- oder Migrationsdetail | `docs/dev/` (`tmp/` für Experimente) | Teilweise | {cols="20%,34%,28%,18%"}

1.2 Zweisprachige Single-Source-Kompendien (`docs/compendia.json`)

Dauerhafte Referenzhandbücher werden als zweisprachige (`de`/`en`) Single-Source-Kompendien gepflegt und durch `scripts/doc_compendium/` sowie `build_compendia.py` in MkDocs-Webseiten und gesetzte LuaLaTeX-PDFs übersetzt. Sprachen werden niemals innerhalb einer Datei gemischt und generierte Webseiten nie manuell editiert; gepflegt werden ausschließlich `source/chapters/*.{de,en}.md` mit 1:1-AST-Blockparität, identischen `{#sec:...}`- / `anchors="..."`-Markern, `@ref{sec:...}`, kanonischen `repo:...`-Links und reinen ASCII-Codeblöcken (`<= 72` Zeichen/Zeile). Vertragsgepinnte Deploy-/Freeze-Dokumente, datierte Workboards und generierte Statusreports bleiben eigenständige Markdown-Dateien.

| Kompendium-ID | Kanonisches `source/` | Tracking-Politik | Generierte PDFs (`DE` / `EN`) | | — | — | — | — | | `skill-graph` | `docs/concept/skill-graph/source/` | `untracked` | `skillpilot-graph-definition.{pdf,en.pdf}` | | `skill-graph-architecture` | `docs/concept/skill-graph/architecture-source/` | `committed-web` | `skillpilot-skill-graph-architecture.{de,en}.pdf` | | `whitepaper` | `docs/whitepaper/source/` | `committed-web` (+ DE-PDF) | `whitepaper.{de,en}.pdf` | | `glossary` | `docs/concept/glossary/source/` | `committed-web` | `skillpilot-glossary.{de,en}.pdf` | | `didactic` | `docs/concept/didactic/source/` | `committed-web` | `skillpilot-didactics-and-planning.{de,en}.pdf` | | `runtime-workflows` | `docs/concept/runtime-workflows/source/` | `committed-web` | `skillpilot-coach-and-runtime-architecture.{de,en}.pdf` | | `production-pipelines` | `docs/production-pipelines/source/` | `committed-web` | `skillpilot-production-pipelines.{de,en}.pdf` | | `qa-ci` | `docs/qa-ci/source/` | `committed-web` | `skillpilot-curriculum-quality-handbook.{de,en}.pdf` | | `qa-ci-lanes` | `docs/qa-ci/lanes-source/` | `committed-web` | `skillpilot-qa-lanes-and-release-contracts.{de,en}.pdf` | | `security` | `docs/security/source/` | `committed-web` | `skillpilot-security-and-privacy.{de,en}.pdf` | | `dev-handbook` | `docs/dev/source/` | `committed-web` | `skillpilot-developer-and-editor-guide.{de,en}.pdf` | {cols="22%,32%,18%,28%"longtable=true}

```
python3 scripts/build_compendia.py --web-and-tex
python3 scripts/build_compendia.py --pdfs
python3 scripts/build_compendia.py --check-committed-web
python3 scripts/build_compendia.py --check-untracked-outputs
```

1.3 MkDocs-[DE | EN]-Schalter, Generierungshinweise, Vorlagen und Checks

- **Navigation & [DE | EN]-Sprachumschalter:** mkdocs.yml (site_url: https://enpasos.github.io/skillpilot/) paart Compendium (EN)- / Kompendium (DE)-Gruppen und registriert Einzelpaare unter standaloneBilingualPairs in docs/compendia.json. language-switcher.js + bilingual-map.js schalten aktive Sprachseiten um, während einsprachige Betriebs- und Arbeitsdokumente in beiden Modi sichtbar bleiben. Datierte Dossiers und docs/qa-ci/status/-Berichte bleiben über die Abschnitts-index.md erreichbar.
- **Generierungshinweise & Vorlagen:** Generierte Markdown-Dateien führen Generated by-, Regenerate with- und Source of truth-Köpfe gemäß generatedMarkdownNoticeRegistry.ts (npm run check:generated-doc-notices, npm run check:generated-status-registry). Dauerhafte QA-Lane-Dokumente folgen Purpose / Source Of Truth / Generated Artifacts / Run / Blocking Semantics / Review Ledger / Review Workflow / Expansion Criteria / Related Documents, Runbooks folgen Run The Audit / Inspect Results / Decide Or Fix An Item / Interpret Diagnostics / Verify Local Changes, und PoC-/Archivdateien tragen ISO-Daten im Dateinamen.
- **Terminologie, Links außerhalb von docs/ und CI-Gates:** Das Kernvokabular folgt dem Glossar und checkTerminology.ts (npm run check:terminology). Verweise aus docs/ auf andere Repositorypfade nutzen GitHub-URLs (https://github.com/enpasos/skillpilot/blob/main/... bzw. /tree/main/...), geprüft in .github/workflows/docs_checks.yml:

```
git diff --check
cd app
npm run check:terminology
npm run check:docs-links
npm run check:docs-indexes
```

2 Landscape-JSON-Feldreferenz

Dieses Kapitel dokumentiert den feldscharfen Authoring- und Laufzeitvertrag für `Skill Landscape`-JSON-Dateien gegen `docs/landscape-runtime.schema.json` und deren Abbildung auf `UiGoal` in `goalTypes.ts` (Knotensemantik in Knotentypen, Projektion in Sichtenprojektion).

2.1 Top-Level-Felder einer `SkillLandscape`

| Feld | Pflicht? | Zweck & Randbedingungen | | — | — | — | | `landscapeId` | Ja | Global eindeutige, stabile Kennung der Skill-Landschaft. | | `locale` | Ja | Primäre Autorensprache (`"de"` oder `ën"`). | | `subject` | Ja | Kanonischer Fachschlüssel für Routing, Gruppierung und Asset-Pfade. | | `title`, `description` | Ja | Titel und Kurzbeschreibung (`titleEn`, `descriptionEn` optional für englische Inline-Übersetzung). | | `frameworkId`, `filters` | Optional | Institutioneller Rahmenbezeichner und auswählbare Scope-Filter (z. B. `GK`, `LK`, Bundesland). | | `programUnits`, `goalPlacements`, `competencyCatalog` | Optional | Additive Projektions-Arrays (3). | | `goals` | Ja | Geordnetes Array von `LearningGoal`-Einträgen der `contains`- und `requires`-DAGs. | {cols="26%,14%,60%"}

2.2 `LearningGoal`, `dimensionTags` und Provenienzfelder

| Feldgruppe | Felder & Laufzeitbedeutung | | — | — | | **Identität & Texte** | `id` (stabile UUID, nie für neue Bedeutung wiederverwendet), optional landschaftsweit eindeutiges `shortKey`, `title` (`titleEn`), `description` (`descriptionEn`). | | **Graphrelationen** | `contains` (geordnete Kind-IDs; leer `[]` = atomares Ziel, nichtleer = Cluster-Ziel) und `requires` (Voraussetzungs-IDs; Zielzustand rein atomar-zu-atomar). | | **Klassifikation & Gewicht** | `weight` (Aufwands-/Fortschrittsgewicht), optional `core` (`boolean`), `courseLevel`, `tags` (`string[]`), `semanticKind` / `nodeKind` / `type`, `applicability` und `competencyRefs`. | | **Fachliche Nutzdaten** | `examples`, `extendedData` (inkl. `applicabilityOverrides` und Sortierhinweisen), `examData` (`taskContent`, `solutionContent`, `scoring`) sowie `experimentData`. | | **`dimensionTags`** | Strukturiertes Metadatenobjekt: `phase` (`E`, `Q1`–`Q4`, `SekI`, `GLOBAL`, Kompatibilitäts-/Badge-Feld), `demandLevel` (`AB1`–`AB3`), `processCompetencies`, `guidingIdeas`, `framework`, `area`, `topicCode`. | | **Provenienz & Medien** | `sourceRef` (amtliche Fundstelle) und `resourceLinks` (`type: "curriculum"` oder `"goal-visualization"`, keine neuen Drittanbieterlinks an kanonischen Zielen; siehe Quellen- & Ressourcenlinks). | {cols="26%,74%"}

Zur Laufzeit überführen Frontend-Loader `LearningGoal`-Objekte in `UiGoal` (`goalTypes.ts`) und werten Scope-Filter, Frontier-Verfügbarkeit, SRS-Decks und Mastery-Aggregation aus, ohne die stabilen JSON-Identitäten zu verändern.

3 Minimales Projektionsschema, Laufzeit-Fallback und Mathe-Pilot

Um bestehende Landschaften ohne Bruch nicht migrierter Curricula in das Schichtenmodell (Allgemeines Zielsystem) zu überführen, ergänzt SkillPilot vier optionale additive JSON-Strukturen mit deterministischer Laufzeit-Fallback-Reihenfolge.

3.1 Additive Schemafelder und Fallback-Reihenfolge

```
{
  "programUnits: [
    { ïd: ßek2-e", "kind: "phase", "label: Ë-Phase", örder: 10 }
  ],
  "goalPlacements: [
    {
      "goalId: «uuid>",
      ünitId: ßek2-e",
      "relation: "primary",
      "context: { "jurisdiction: "DE-HE", ßtage: SSekII" }
    }
  ],
  "competencyCatalog: [
    { ïd: "K1", "label: "Mathematisch argumentieren", "dimension: "process" }
  ]
}
```

Ziele tragen ergänzend optional `competencyRefs: ["K1", ...]`. Die Laufzeitauswertung folgt festen, rückwärtskompatiblen Regeln:

1. **Composition View zuerst:** Passt eine geprüfte `.view.json` auf den aufgelösten Scope, wird der Default-Baum aus Composition View plus kanonischem `contains` kompiliert.
2. **Explizite Placement-Projektion als Zweites:** Liegen `programUnits` und passende `primary-goalPlacements` vor, wird der Programmbaum direkt daraus projiziert.
3. **Klassischer `contains-` / `phase-Fallback als Drittes:** Landschaften ohne View- oder Placement-Daten rendern unverändert ihre `contains`-Hierarchie und `dimensionTags.phase`-Badges.
4. **Unveränderte Mastery & Frontier:** Mastery-Speicherung und `requires`-Frontier arbeiten ausschließlich auf kanonischen atomaren `LearningGoal`-IDs, niemals auf `programUnits` oder View-Strukturknoten.

3.2 Einstiegsscope-Auflösung und kanonischer Mathematik-Pilot

Die Scope-Auflösung durchläuft fünf deterministische Schritte: (1) Basis-Curriculum wählen, (2) Bundesland / Dauer / Stufe / Kursprofil (`Level 2`) binden, (3) passende `.view.json` bzw. Placement-Kontext auflösen, (4) benötigte atomare Voraussetzungen $L = \text{Closure}(S)$ auf dem unbeschnittenen kanonischen Graphen schließen und (5) optionale `Level 3`-Fokusfilter anwenden. `curricula/DE/Gymnasium/canonical/DE_DEU_S_GYM_CANONICAL_MATHEMATIK.de.json` dient als Referenzpilot für das Zusammenspiel aus `programUnits`, `goalPlacements`, `competency Catalog` und bundeslandspezifischen `.view.json`-Dateien.

4 Curriculum-Autoreneditoren

SkillPilot trennt lokale Autorenwerkzeuge im Vite-Entwicklungsserver strikt nach semantischer Schicht, statt kanonische Graphpflege, Lernkartenredaktion und Scope-Views in einer monolithischen Oberfläche zu vermischen.

4.1 Vier lokale Vite-Editoren und Routenarchitektur

| Editor | Frontend-Route & Dev-Middleware | Verantwortete Artefaktschicht | | — | — | — | | **Flashcard-Editor** (6) | `/flashcard-editor` · `/__deck-editor/*` | `*_deck*.json`-Lernkartendecks (`curricula/...` + `app/public/data/`). | | **Graph-Editor** (5) | `/graph-editor` · `/__graph-editor/*` | Fokussiertes `requires`-Refactoring und Cluster-zu-Atomar-Auflösung in Landscape-JSONs. | | **Canonical-Cluster-Editor** | `/canonical-cluster-editor` · `/__canonical-cluster-editor/*` | Kanonische `contains`-Clusterhierarchie, Kindreihenfolge und Clustermetadaten unter `curricula/DE/Gymnasium/canonical/`. | | **Composition-View-Editor** | `/composition-view-editor` · `/__composition-view-editor/*` | Scope-spezifische `*.view.json`-Oberbäume (`structure`-Knoten + reine Cluster-`canonicalSubtree`-Verweise). | {cols="26%,34%,40%"}

Sämtliche `/__*`-Schreibendpunkte existieren **ausschließlich** in der lokalen Vite-Dev-Middleware (`vite.config.ts`) und sind niemals Teil produktiver Builds.

4.2 Drei-Spalten-Layout, blockierende Prüfungen vs. Warnungen und Phasenfolge

Beide V1-Strukturediteren nutzen ein Drei-Spalten-Layout (links Baumnavigation, mittig Knoten-/Referenz-Inspektor, rechts kompilierte Live-Vorschau samt Diagnose):

- **Blockierende Speicherprüfungen (Schreibabbruch):** Ungültiges JSON, doppelte IDs, `contains`-Zyklen, unaufgelöste `goalId`-Verweise, direkt in `.view.json` eingebettete atomare Ziele oder überlappende `canonicalSubtree`-Expansionen gegen das Einmalvorkommen (Sichtenprojektion).
- **Nicht-blockierende Warnungen (Speichern mit Diagnosehinweis):** Noch nicht eingebundene kanonische Cluster im Entwurf, `unplacedInScope`-Hinweise oder ausstehende Folge-Checks (`npm run validate:graph`, `validate:views`).
- **Vierstufige Umsetzung:** (1) Schreibgeschützter Baum- und Diagnose-Inspektor → (2) Speicherpfad `Canonical Cluster Editor` → (3) `.view.json`-Bearbeitung im `Composition View Editor` → (4) länderübergreifender Abdeckungsvergleich (bewusste Nicht-Ziele: Cloud-Mehrbenutzerbetrieb, automatisches KI-Umschreiben oder Bearbeitung von Lernständen).

5 Graph-Editor

Der lokale Graph-Editor (`/graph-editor`, implementiert in `GraphEditorView.tsx` über die Vite-Dev-Middleware `/__graph-editor/{list,load,save}`) ist eine fokussierte Refactoring-Werkbank für `requires`-Voraussetzungen in `SkillLandscape`-JSON-Dateien.

5.1 Fokussierter `requires`-Refactoring-Ablauf

Statt eines freien Zeichenbretts unterstützt der Editor gezielt die kanonische `requires`-Bereinigung ($R \subseteq A \times A$; siehe Graph-Definition):

- **Getrennte Voraussetzungsansicht:** Teilt die `requires`-Einträge des gewählten Ziels in **atomare Voraussetzungen**, **geerbte Cluster-Voraussetzungen** und **unaufgelöste/externe IDs**, sodass abzulösende Clusterkanten sofort sichtbar sind.
- **Geführte Cluster-zu-Atomar-Auflösung:** Zeigt die atomaren Nachfahren eines referenzierten Clusters, damit exakt benötigte Atomziele übernommen werden können, bevor die Clusterkante entfernt und transitiv reduziert wird.
- **In-Place-Speicherung (2-Space-JSON):** `/__graph-editor/save` schreibt formatiertes UTF-8-JSON mit zwei Leerzeichen Einrückung direkt in die Quelldatei unter `curricula/` zurück (ohne Spiegelkopie nach `app/public/data/`).

```
cd app
npm run dev
# Nach dem Speichern unter http://localhost:5173/graph-editor:
npm run validate:graph
```

6 Flashcard-Editor

Der lokale Flashcard-Editor (`/flashcard-editor`, implementiert in `FlashcardEditorView.tsx` und `FlashcardFlipCard.tsx` über die Vite-Dev-Middleware `/__deck-editor/{list,load,save}`) dient der Pflege und Live-Vorschau von Spaced-Repetition-Decks (`*_deck*.json`) inklusive KaTeX-Mathematik und Markdown.

6.1 Deck-JSON-Schema, Sprachpaarung und Dual-Write-Speicherung

```
{
  "deckId: "physics_sek2_q1_fields_deck",
  "title: "Q1 Elektrisches und magnetisches Feld",
  "cards: [
    {
      id: "q1-field-01",
      "front: "Definiere die elektrische Feldstaerke $E$.",
      "back: "$E = \\frac{F}{q}$ in $\\mathrm{N/C}$ bzw. $\\mathrm{V/m}$.",
      "category: "Grundbegriffe",
      "tags: ["GK", "LK"]
    }
  ]
}
```

Redaktionsregeln:

- **Pflichtfelder:** Jedes Deck führt `deckId`, `title` und `cards[]`; jede Karte besitzt eine deckweit eindeutige `id`, `front`, `back`, `category` sowie optional `tags` (`string[]`). Karten-IDs müssen stabil bleiben, damit bestehende SM-2- und `verifiedRecall`-Lernstände (Knotentypen) erhalten bleiben.
- **DE-/EN-Deck-Paarung:** Optionale englische Begleitdecks (`*_deck.en.json`) nutzen identische Karten-`id`s, sodass ein Sprachwechsel keinen Abrufstatus verliert.
- **Dual-Write-Speicherung (Repository-Modus):** `/__deck-editor/save` aktualisiert synchron die Quelldatei unter `curricula/...` und die Laufzeitkopie unter `app/public/data/<basename>.json` (im Standalone-Paketmodus werden Decks über `CurriculumResourceResolver` aus `data/cards/` aufgelöst).

7 Knotensortierung unter „Meine Lernziele“

Im lernendenseitigen Kompetenzbaum (`CompetenceTree.tsx`) sortiert `sortSiblingGoalsBalanced` in `goalSorter.ts` Geschwisterknoten unterhalb eines Clusters deterministisch so, dass Voraussetzungen vor abhängigen Zielen erscheinen und didaktische Klammern gewahrt bleiben.

7.1 Harte `requires`-Topologiesortierung und sechsstufiger Tie-Breaker

Für Geschwisterknoten eines Elternclusters (zyklenfrei bezüglich `requires`) führt `sortSiblingGoalsBalanced` eine Kahn-Topologiesortierung über **direkte `requires`-Kanten zwischen Geschwistern** aus und entscheidet Gleichstände in der Bereit-Menge streng sechsstufig:

| Stufe | Kriterium | Exakte Entscheidungsregel | | — | — | — | | **1. Manuelle Ordnung** | Explizite Autorenvorgabe | Numerisches `extendedData.{treeOrder,sortOrder,displayOrder,order,position}` oder Tag `order:<n>`. | | **2. Didaktischer Bucket** | Pädagogische Klammerung | `why` (Orientierung/Motivation) < `tutorial` / Fachkompetenzen < `flashcards` (Lernkarten) < `exercises` / Prüfungen. | | **3. Phasenfolge** | Curriculare Progression | `SekI` / 5–10 < `E` < `Q1` < `Q2` < `Q3` < `Q4` < `Abitur` / `GLOBAL`. | | **4. Nachfahren-Support** | Weiche Teilbaum-`requires` | `softScore(v)` aus Nachfahren-Voraussetzungen, geschützt durch `minAbsoluteSupport = 2` und `minRelativeSupport = 0.25`, damit einzelne Randkanten große Themenblöcke nicht unerwartet umsortieren. | | **5. Lokalisierter Titel** | Lesbare Rückfallordnung | Sprachsensitives `localeCompare` (`de-DE`, `numeric: true`). | | **6. Kanonische ID** | Endgültige Deterministik | Lexikographischer Vergleich von `goal.id` für 100 % reproduzierbare Reihenfolge. | {cols=”20%,24%,56%”}

Während die reine Inhaltsbaum-Inspektion (Sichtenprojektion) die explizite `contains`- bzw. `treeOrder`-Reihenfolge abbildet, steuert `sortSiblingGoalsBalanced` die didaktische Lernansicht überall dort, wo kein manuelles `treeOrder` gesetzt ist.

8 Datenbankmigrationen mit Liquibase

Das Spring-Boot-Backend erzwingt `spring.jpa.hibernate.ddl-auto=validate`: Hibernate prüft beim Start lediglich die Übereinstimmung der JPA-Entities mit der Datenbank, während **Liquibase** allein für jede Schemaanlage und -migration zuständig ist.

8.1 Changelog-Struktur und vierstufiger Migrationsablauf

- **Master-Changelog:** `backend/src/main/resources/db/changelog/db.changelog-master.yaml` bindet geordnete Migrationsdateien unter `backend/src/main/resources/db/changelog/changes/` ein (beginnend mit `001-initial-schema.yaml` samt `preConditions` und `onFail: MARK_RAN`, damit bestehende Datenbanken sauber in `DATABASECHANGELOG` übernommen werden).
- **Vier Schritte bei Schemaänderungen:**
  1. JPA-`@Entity`-Klasse unter `backend/src/main/java/com/skillpilot/backend/domain/` anpassen.
  2. Neue nummerierte Datei `NNN-kurzbeschreibung.yaml` unter `db/changelog/changes/` mit eindeutigem `(id, author)`-Changeset und expliziten Spaltentypen/Constraints anlegen.
  3. Datei per `include` am Ende von `db.changelog-master.yaml` registrieren.
  4. Backend bzw. Tests starten (`./gradlew bootRun` oder `./gradlew test`), sodass Liquibase die Migration ausführt und Hibernate `validate` die exakte Entity-Schema-Deckung bestätigt.
- **Unveränderlichkeitsregel:** Bereits eingecheckte und ausgeführte Changesets dürfen **niemals** nachträglich geändert werden (Prüfsummenfehler beim Start); Korrekturen erfolgen stets über ein neues Folge-Changeset.

9 Echtzeit-Auto-Update-Architektur (SSE)

Sobald ein externer KI-Coach (über MCP oder REST) Mastery speichert oder das aktive Lernziel wechselt, aktualisiert sich die Cockpit-WebGUI ohne manuelles Neuladen in Echtzeit über **Server-Sent Events (`SSE`)** (Client-State-Synchronisation).

9.1 Backend-Event-Bus, SSE-Endpunkt und React-Reconnect-Hook

```
LearnerService-Schreibzugriff -> LearnerStateChangedEvent
  -> SseService (SseEmitter-Map + ~25s-Heartbeat)
  -> GET /api/ui/updates/{skillpilotId} (text/event-stream)
  -> useLearnerUpdates(skillpilotId) -> refreshMastery() / State-Reload
```

- **Spring-Events:** Zustandsänderungen in `LearnerService.java` feuern ein `LearnerStateChangedEvent` (`MASTERY_UPDATE`, `GOAL_UPDATE` / `ACTIVE_GOAL_UPDATE`).
- **SSE-Controller & Emitter-Verwaltung:** `UpdateController` stellt `GET /api/ui/updates/{skillpilotId}` (`text/event-stream`) bereit; `SseService` verwaltet die `SseEmitter`-Instanzen je SkillPilot-ID, räumt geschlossene/abgelaufene Verbindungen auf und sendet alle `~25s` einen `heartbeat`-Event, damit Reverse-Proxys (Nginx/Cloudflare) ruhige Streams nicht trennen.
- **Frontend-Hook (`useLearnerUpdates`):** Öffnet eine Browser-`EventSource`, stößt bei eingehenden Events `refreshMastery()` sowie das Nachladen des aktiven Ziels an und verbindet sich nach Netzwerkabbrüchen mit exponentiellem Backoff neu.
- **Skalierungsgrenze:** Die In-Memory-NIO-Verteilung über `SseEmitter` bedient Einzelknoten-Instanzen ressourcenschonend; bei horizontaler Mehrknoten-Skalierung wird ein Pub/Sub-Relay (z. B. Redis Pub/Sub oder PostgreSQL `LISTEN/NOTIFY`) zwischen den Replikaten ergänzt.