SkillPilot Produktionshandbuch · Band V

Produktionspipelines für Skill-Graphen, Pakete und Prüfungen

Wiederholbare Blaupausen für Skill-Graph-Erzeugung, sichere Paketprovisionierung,

Fachexport-ZIPs, MEM/FWU-Core-Roundtrips und qualitätsgesicherte Prüfungsaufgaben

Geltungsbereich dieses Handbuchs

Dieser Band bündelt alle wiederholbaren Produktions- und Exportpipelines von SkillPilot: die achtstufige Erzeugung neuer `SkillLandscape`-JSON-Graphen, die sichere Quarantäne-, CAS-Aktivierungs- und Rollback-Provisionierung von `full-standalone-v1`-Paketen, das reproduzierbare Fachexport-Paket, den Core-first-MEM/FWU-Ontologie-Roundtrip (`DPK-008c/008d`) sowie die allgemeinen und Sek-I-spezifischen Produktionsworkflows für Prüfungsaufgaben (`examData`).

Stand: 11. November 2026 · Lizenz: Apache-2.0 (technische Produktions- und Exportdokumentation; siehe LICENSING.md).

Temporäre Arbeitsartefakte unter tmp/ und Referenzpools für Prüfungsaufgaben werden niemals in Git eingecheckt.

Inhaltsverzeichnis

1 Skill-Graph-Produktionspipeline **3**
1.1 Eingaben und Schritte 0–2: Extraktion, Gliederung und JSON-Erzeugung 3
1.2 Schritte 3–7: Relationen, Referenzimplementierungen, Validierung und Registrierung 3

2 Curriculum-Pakete sicher provisionieren **5**
2.1 Sechsstufiges Sicherheitsmodell und Store-Layout 5
2.2 Bedienfolge, Runtime-Konfiguration und Selbsttest 5
2.3 Bewusste Grenzen 6

3 SkillPilot-Fachexport-Paket **7**
3.1 Bau, unabhängige Validierung, Link-Audit und CI-Release-Gate 7
3.2 Paketstruktur, Quellen-/Merkkarten-Evidenz, Lizenzkategorien und neue Fächer 7

4 MEM/FWU-Core-Roundtrip-Pipeline **9**
4.1 Zwei koexistierende Lanes (`DPK-007a/008c/008d` vs. Slim-Pilot) 9
4.2 Core-First-Abbildungsregeln und Bild-Sidecars 9
4.3 Produktions- und Pilot-Kommandos 10

5 Klausurbeispiel-Produktionspipeline **12**
5.1 Verzeichnisstruktur und Blueprint-Schema 12
5.2 Zehnstufiger Workflow (Schritt 0 bis Schritt 9) 12

6 Produktionspipeline für Sek-I-Mathematikprüfungen **14**
6.1 Quellenhygiene und Arbeitspaket-Layout 14
6.2 Zehnstufiger Gate-Workflow und Releaseregel 14

1 Skill-Graph-Produktionspipeline

Dieses Kapitel definiert die allgemeine, wiederholbare und prüfbare Blaupause, mit der aus einer beliebigen amtlichen oder institutionellen Curriculumquelle ein sauberer `Skill Landscape`-JSON-Graph unter `curricula/<...>/json/<file>.json` (bzw. `curricula/DE/Gymnasium/canonical/`) entsteht (für Prüfungsaufgaben siehe 5).

1.1 Eingaben und Schritte 0–2: Extraktion, Gliederung und JSON-Erzeugung

Als Eingaben dienen (1) ein normalisierter Rohtext-Auszug mit Themen, Kompetenzen, Voraussetzungen, Bewertungsumfang und Quelllink, (2) eine menschlich kuratierte Gliederung in Wurzel, Cluster und messbare Atomziele sowie optional (3) benachbarte Landschaften für konsistente Namens- und ID-Konventionen:

- **Schritt 0 (Quellenextraktion):** Amtliche Vorlage in sauberen Klartext (kein HTML) mit eindeutiger URL-/Seitenreferenz überführen.
- **Schritt 1 (Strukturgliederung):** Wurzelknoten anlegen, Phasen/Themenfelder als Cluster gliedern und Kompetenzen in messbare **atomare Lernziele** (1–3 Aufgaben, Formulierung „Die lernende Person kann …“) schneiden. Für lernendenseitige Routen **Motivationsanker** sowie **phasenlokale Autonomiecluster** mit Klausurcharakter (`Übungen E-Phase`, `Übungen Q1`–`Q4` sowie ggf. `Übungen Prozesskompetenzen`) vorsehen und globale Abitur-Zweige als zusätzliche Prüfungsschicht davon getrennt halten.
- **Schritt 2 (JSON-Kodierung):**
  - **Stabile IDs:** UUID v5 im Projektnamensraum oder deterministische Abbildung aus `(landscapeId, shortKey)` (`shortKey` optional, aber innerhalb von `landscapeId` eindeutig; dieselbe `(goalId, shortKey)`-Kombination darf über Sprachdateien hinweg wiederholt werden, niemals jedoch ein `shortKey` für verschiedene `goalId`s).
  - **Pflichtfelder:** Top-Level `landscapeId`, `title`, `description`, `locale`, `subject`, `goals`; Ziel-Level `id`, `title`, `description`, `weight`, `requires`, `contains`, `dimensionTags` (mindestens `phase`; bevorzugt zusätzlich `area`, `topicCode`, `demandLevel`, `processCompetencies`, `guidingIdeas`; optional `tags`, `core`, `examples`, `sourceRef`, `resourceLinks`).
  - **Lokalisierung, Assets & Links:** Mehrsprachige Landschaften nutzen getrennte Locale-JSONs mit bedeutungsäquivalenten Texten; lokale Bilder liegen neben dem JSON mit relativen Markdown-Pfaden. `sourceRef` trägt die kanonische Herkunftsreferenz, `resourceLinks` ausschließlich zielspezifische Lernressourcen (Überblicksseiten auf Wurzel/Clustern, spezifische Übungs-/Visualisierungslinks auf Atomzielen).

1.2 Schritte 3–7: Relationen, Referenzimplementierungen, Validierung und Registrierung

- **Schritt 3 (`contains`-Hierarchie):** Zyklenfreien DAG (Cluster → Teilziele) mit flacher, fachlich sprechender Gliederung aufbauen.
- **Schritt 4 (`requires`-Sequenzierung):** Kanonisch **atomar-zu-atomar** ohne transitive Redundanz modellieren (Graph-Definition, Graph-Validierungsregeln); geerbte Cluster-`requires` gelten nur als Übergangsmechanismus. Jedes lernendenseitige Atomziel soll auf mindestens einer lehrbaren Route vom Motivationsanker zu einem lokalen Autonomieziel liegen. Als praktische Autoren-Referenzmuster unter `curricula/DE/Gymnasium/input/HE/upper-secondary/source-json/` dienen `DE_HES_S_GYM_2_PHYSIK.de.json.snapshot` und `DE_HES_S_GYM_2_MATHEMATIK.de.json.snapshot` (lokale `Übungen E-Phase/Q1`–`Q4` plus `Übungen Prozesskompetenzen`, sauber getrennt von den globalen Abitur-Containern; die aktiven Laufzeitlandschaften liegen unter `curricula/DE/Gymnasium/canonical/`).

- **Schritt 5 (Validierung & manuelle Reviews):**

```
cd app
npm run validate:graph
```

Ergänzend die manuellen Reviews für Atomizität (Atomizitäts-Review) und Voraussetzungen (Voraussetzungs-Review) durchlaufen.

- **Schritt 6 (Registrierung):** Wurzelcurricula in `curricula/curriculum_manifest.json` eintragen; Teilcurricula im Eltern-`contains` verankern.
- **Schritt 7 (QA-Checkliste):** Klare, in 1–3 Aufgaben prüfbare Atomziele, kanonische atomare `requires`, zyklenfreie `contains`/`requires`, geschlossene Routen von Motivation bis phasenlokaler Autonomie, stabile IDs, konsistente Metadaten (`phase`, `area`, `level`, `core`, `weight`) und abgeglichene Lokalisierung.

2 Curriculum-Pakete sicher provisionieren

Dieses Runbook beschreibt die lokale Installation, Verifikation, CAS-Aktivierung und das Rollback fertig gebauter `full-standalone-v1`-JSON-Pakete. Der Provisioner (`scripts/provision_curriculum_package.py`) ist die einzige schreibende Komponente des Package-Stores; die Skill Pilot-Runtime liest ausschließlich den exakt gepinnten aktiven Lock und verändert den Store niemals.

2.1 Sechsstufiges Sicherheitsmodell und Store-Layout

Ein eingehendes ZIP wird nie direkt als Laufzeitquelle konsumiert:

1. Unveränderte Eingabedatei mit stabiler Dateiidentität in eine private Quarantäne kopieren und äußeren SHA-256 sowie Bytezahl berechnen.
2. Unabhängigen Finished-Package-Validator v2 in einem getrennten Prozess aufrufen.
3. Ausschließlich das exakte, sichere Archivinventar in ein privates Staging-Verzeichnis extrahieren.
4. Manifest, `SHA256SUMS`, jeden Manifestrecord und den gesamten extrahierten Dateibaum erneut verifizieren.
5. Objekt, Validatorreport und Install-Record atomar unter `<outerZipSha256>` promoten.
6. Den aktiven Lock erst durch einen getrennten, expliziten Compare-and-Swap-(CAS-)Schritt umschalten.

Symlinks, Sonderdateien, Pfadkollisionen, ZIP64, Data Descriptors, verschachtelte Archive, unerwartete Dateien und Profilüberschreitungen scheitern fail-closed und lassen den bisherigen aktiven Lock unberührt.

```
<store>/
  .provisioner.lock
  quarantine/ & staging/
  objects/sha256/<outerZipSha256>/<archiveRoot>/
  validation-reports/<outerZipSha256>.json
  install-records/<outerZipSha256>.json
  locks/active.json & locks/history/<lockSha256>.json
```

`objects/`, Validatorreports und Install-Records sind nach der Promotion unveränderlich; es gibt weder Verzeichnis-Discovery noch eine ungepinnte `latest`-Auswahl.

2.2 Bedienfolge, Runtime-Konfiguration und Selbsttest

```
python3 -B scripts/provision_curriculum_package.py --help
python3 -B scripts/provision_curriculum_package.py activate --help
python3 -B scripts/provision_curriculum_package.py rollback --help
python3 -B scripts/provision_curriculum_package.py self-test
```

Normale Bedienfolge: (1) `install`, (2) `verify`, (3) `activate` mit `--expected-active-sha256` (`none` bei Erstaktivierung, sonst exakter SHA-256 des beobachteten Locks; kanonische Sortierung nach `packageId`, Abweisung doppelter IDs, inkompatibler Consumer-Bereiche oder widersprüchlicher Definitionen), (4) `status` und bei Bedarf (5) `rollback` per CAS auf einen historischen Lock.

Erst danach wird das Backend unter getrennter schreibgeschützter OS-Identität auf den Store konfiguriert (ohne Fallback auf Repository, Classpath oder Netzwerk):

```
skillpilot:
  curriculum:
    source: package
    consumer-version: 0.1.0
    packages:
      store-directory: /srv/skillpilot/curriculum-packages
      active-lock: locks/active.json
```

2.3 Bewusste Grenzen

- Eine technisch gültige Installation ersetzt keine Publikations-, Quellen- oder Rechtefreigabe; kryptographische Publisher-Signaturen folgen mit `DPK-011`.
- `embedded-fragment` bleibt bis zur vollständigen fachübergreifenden Closure- und Consumer-Lane fail-closed.
- Der checkout-freie Betrieb (15 Funktionsprüfungen unter `--clearenv`, Loopback-Netzwerk und Catalog-404-Fail-Closed-Test) ist seit `DPK-007` über denselben Store-Pfad bewiesen (Curriculum-Paketbereitschaft).

3 SkillPilot-Fachexport-Paket

Diese Pipeline erzeugt ein reproduzierbares Legacy-Fachexport-ZIP (`legacy-subject-export`, Zielstatus `not-ready-legacy`, `standaloneProfileReady: false` gemäß Curriculum-Paketbereitschaft) für eine einzelne Fachlandschaft als geprüfter Publikations- und Roundtrip-Träger (4), jedoch nicht als alleinige `full-standalone-v1`-Laufzeiteingabe (Release-Modell-Konformität). Unterstützte `DE/Gymnasium`-Presets decken aktuell alle Fächer ab `M5` ab: `Biologie`, `Chemie`, `Deutsch`, `Geschichte`, `Informatik`, `Latein`, `Mathematik`, `Physik`, `Politik und Wirtschaft` sowie `Wirtschaftswissenschaften`.

3.1 Bau, unabhängige Validierung, Link-Audit und CI-Release-Gate

Einzelbau, Batch-Bau aller `M5`/`M6`-Fächer (`tmp/exports/m5-subject-export-summary.{json,md}`), unabhängige ZIP-Validierung (`tmp/exports/validation/` + `tmp/exports/readiness/`), Live-Quelllink-Audit (`--fail-on-broken` für strikte Vorabprüfung) und Reproduzierbarkeitsbeweis (`SOURCE_DATE_EPOCH`, unkomprimierte sortierte Einträge mit Modus `0644`):

```
cd app
npm run export:subject-package -- --subject Mathematik --version 0.1.0
npm run export:m5-subject-packages -- --version 0.1.0
npm run export:subject-packages:validate -- --dir tmp/exports
npm run export:subject-packages:audit-links -- --dir tmp/exports
npm run export:subject-release-gate -- --version 0.1.0 --strict-source-links
npm run export:subject-release-gate:ci
npm run export:m5-subject-packages:check-reproducible -- --version 0.1.0
SOURCE_DATE_EPOCH=1767225600 npm run export:subject-package -- \
  --subject Mathematik --version 0.1.0
```

`export:subject-release-gate:ci` läuft wöchentlich/manuell in `.github/workflows/owl-ci.yml` mit `--enforce-committed-quality-status --enforce-clean-source-tree` und schreibt Release-Gate- sowie Publikations-Indizes unter `tmp/exports/{release-gate,publication}/`. Für Reviewer-Übergaben löst `export:subject-package:trace-source` Quell- oder kanonische Ziel-IDs direkt aus dem fertigen ZIP auf:

```
cd app
ZIP=../tmp/exports/skillpilot-de-gymnasium-mathematik-v0.1.0.zip
npm run export:subject-package:trace-source -- \
  --zip "$ZIP" --source-goal-id he-math-sekii-e-1-b01-a01-376ffbbc
npm run export:subject-package:trace-source -- \
  --zip "$ZIP" --canonical-goal-id 502ecaa7-cca6-5c51-a1cc-da09a7b2382c
```

3.2 Paketstruktur, Quellen-/Merkkarten-Evidenz, Lizenzkategorien und neue Fächer

Jedes Archiv entpackt Windows-`MAX_PATH`-sicher unter `skillpilot-de-gymnasium-<fach>-v<version>/` ohne Repository-Pfade oder Lernstände (der begleitende `*-release-report.md` liegt außerhalb des ZIPs):

- **Fach- & Quellendaten:** `data/canonical/`, `data/views/`, `data/mappings/` (alle 16 Länder; unvollständige Länderabdeckung nur explizit per `--allow-missing-states`), `data/cards/` + `card-index.json`, `data/resources/goal-visualizations.json` + nur aktiv verlinkte Bilder unter `assets/goal-visualizations/`, `data/dependencies/external-goal-references.json` sowie die verpflichtende HTTP(S)-Quellenbrücke `data/sources/source-

`index.json` ← `data/sources/source-goal-references.json` (`metadata/provenance-report.md`).

- **Qualität, Schemata & Checksummen:** `metadata/quality/memory-card-review/` (`*.md`, `*.config.json`, `*.review.jsonl`, `*.cards.review.jsonl` für `CQR-302`), `schemas/`, `metadata/manifest.json`, `metadata/validation-report.json`, `metadata/SHA256SUMS`.
- **Dateibasierte Lizenzkategorien (`LICENSE.md`, `NOTICE.md`, `licenses/`):** Gemäß `LICENSING.md` trägt jede Datei in `manifest.json` eine `licenseCategory`: `skillpilot-software-apache-2.0`, `skillpilot-data-cc-by-4.0`, `official-source-provenance-only`, `generated-package-metadata` oder `goal-visualization-ai-generated-curated` (Provenienzlabel, kein SPDX-Bezeichner und keine Drittrechte-Freigabe).
- **Wiederverwendung für neue Fächer ohne Preset:** Explizite Selektoren `--subject`, `--subject-slug`, `--composition-dir` und wiederholte `--mapping-token`-Parameter übergeben.

4 MEM/FWU-Core-Roundtrip-Pipeline

Diese Pipeline beweist, dass ein SkillPilot-Publikationspaket in ein Core-first-RDF/OWL-Bündel auf Basis der FWU-Lehrplan-Ontologie (`tmp/lehrplan-ontologie`, `src/ontology/components/lehrplan-core.owl`, IRI `https://w3id.org/lehrplan/ontology/lp/components/lehrplan-core.owl`) transformiert und anschließend **ohne Zugriff auf das Original-JSON oder den Checkout** verlustfrei rekonstruiert werden kann (Duale Release-Verträge, Menschliche Review-Gates).

4.1 Zwei koexistierende Lanes (`DPK-007a/008c/008d` vs. Slim-Pilot)

1. **Produktive `fwu-owl-v1`-Release-Konformität (`DPK-008a`–`008d`):** Nimmt das eingefrorene `DPK-007a`-JSON-Artefakt (`skillpilot-curriculum-de-gymnasium-mathematik-0.1.0-conformance.3.json.zip`, `1.738.161.217` Byte, SHA-256 `403cc0bc6004da549c8b9ed9fafad222fe0ddda1107806fe087cfa871a6dbcf9`, `contentDigest` `sha256:e83936aaf3645ff5f6e8132c4a801bd4bd66f55d3c0304a5deda3d6a5d194101`), erzeugt den byteidentischen 2,36-GB-FWU-OWL-Träger (`DPK-008c`: `2.362.455.128` Byte, SHA-256 `abab1d8aac3e9394af26c614bbf231954ba45ab11f725dd0f93f088820dc3f94`, Manifest `29f308424d1aeba9095f0e800253acadcdfaca0562dfa1fc37741c77c76023b3`, `819` Einträge, `757` Bild-Sidecars, `824.452` Tripel über 8 RDF-Segmente, 18/18 Gates bestanden, Placement-Schlüssel `goalId@unitId@index`) und übersetzt diesen in `DPK-008d` isoliert unter `bubblewrap --clearenv` (`python -I -S`, kein Netzwerk, Host-Rawtrace-ELF-Audit) in zwei byteidentische JSON-ZIPs zurück (`1.737.571.471` Byte, SHA-256 `7dcd233dd495900f6d6bd971ff6e86bdcdcfe5701f3522ed480be8336de23195`, Manifest `8d7970435431ff78743d0bf413a54bbebee009bbaafe35f1283ebcb89b4f2ff0`, 111/111 Normalform-Oracles; das öffentliche Äquivalenzurteil folgt erst in `DPK-009`).
2. **Legacy-Slim-Pilot (`tmp/exports/skillpilot-de-gymnasium-mathematik-v0.1.0.zip`):** Schnelle Regressions-Lane unter `tmp/roundtrip/mem-fwu/.../slim/` (`declarations.nt` mit exakt `526` Deklarationstripeln, `landscape.nt`, `assets.nt`, `sources.nt`, `mappings.nt`, `views.nt`, `cards.nt`, abgeleitetes `bundle.nt`, `catalog-v001.xml`, `skillpilot-mem-fwu-profile.ttl`, `manifest.json`). Da der Legacy-Comparator `null`-Felder verwirft und Arrays teilweise als Mengen vergleicht, gelten seine Berichte ausschließlich als Regressionsnachweis (`not-ready-legacy`), niemals als öffentliches Äquivalenzurteil.

4.2 Core-First-Abbildungsregeln und Bild-Sidecars

SkillPilot-Konzept	Primäre FWU-Core-Darstellung	Anwendungs-Erweiterung / Rückübersetzungsanker
Fachliches Atomziel (`curricularAtomic`)	`LP_0000336` (*Fachbezogene Kompetenzspezifikation*; ohne Bereichs-Eltern zusätzlich unter deterministischem Fallback-Bereich `LP_0000349`)	`sp:LearningGoal` + `sp:AtomicGoal` (Fallback-Bereich wird bei Rekonstruktion ignoriert)
Fachlicher Cluster (`curricularArea`)	`LP_0000349` (*CE-Bereich*)	`sp:LearningGoal` + `sp:ClusterGoal`
Prozessachsen `K1`–`K6` / Leitideen `L1`–`L5`	27 `LP_0030265`- bzw. 5 `LP_0000268`-Ressourcen; Verweis über `LP_0030065` (`LP_0030071`/`LP_0030072`)	`sp:referenceRole` trennt authored `competencyRefs` von abgeleiteten `dimensionTags`; `sp:dimensionTagsJson` sichert Verlustfreiheit

SkillPilot-Konzept	Primäre FWU-Core-Darstellung	Anwendungs-Erweiterung / Rückübersetzungsanker
Jahrgang / Schulstufe	`LP_0000026` (`LP_2000005`–`LP_2000010`), `LP_0000047` (Sek I/II)	`sp:ProgramUnit` & Placement-Metadaten (`goalId@unitId@index`)
Fachliches `contains`	`BFO_0000051` (*hat Teil*, transitiv)	Direktes `sp:containsGoal` verhindert falsche direkte Kinder durch BFO-Inferenz
Fachliches vs. Laufzeit-`requires`	Reifiziertes `LP_0000554` (*Didaktische Voraussetzung*, aus `FWU-DE/lehrplan-ontologie#9`)	`sp:didacticRequires` für Übungs-, Prüfungs-, Memory- und Orientierungsknoten
Titel / Beschreibung / Kurzschlüssel	`LP_0030056` / `LP_0030051` / `LP_0030057` → `LP_0000344`	`rdfs:label`, `dcterms:description`, `sp:shortKey`
Bild auf Fachziel vs. Laufzeitziel	`LP_0030065` auf `schema:ImageObject` mit Position `LP_0000460`	`sp:GoalVisualizationReference` für Laufzeitziele; gehashte Binär-Sidecars (nie Base64 in `.nt`; Limits: `64 MiB`/Bild, `8 GiB` gesamt)
Quellen & Mappings	`LP_0000438` (*Lehrplan*), `LP_0000261` (*Curriculares Element*), `LP_0030065`	`sp:OfficialSourceDocument` für Zusatzdokumente; `exact`/`partial`-Prädikat; Qualitätsdaten bleiben rein in `sp:*`

Kompakte JSON-Metadaten nutzen `xsd:string` (kein privater Datentyp, damit OWL 2 DL erhalten bleibt; `sp:fieldState` bewahrt leere Kollektionen). Während Legacy-Bundles OCR-Steuerzeichen über `sp:quarantinedRecordJson` kapseln konnten, weist der `DPK-004`-Release-Compiler C0-Steuerzeichen und ungepaarte UTF-16-Surrogate strikt ab. Die gepinnte Core-Datei (OWL Functional Syntax, `text/owl-functional`) sowie Anwendungsontologie, SHACL-Shapes (ohne SPARQL/JS) und XML-Katalog unter `contracts/curriculum-package/v1/ontology/` bilden externe Trust-Roots.

4.3 Produktions- und Pilot-Kommandos

Gepinnte Offline-Toolchain (`jsonschema 4.26.0`, `pySHACL 0.30.1`, `RDFLib 7.6.0`, `owlrl 7.6.2`, `ROBOT 1.9.10`, `HermiT 1.4.5.456`, Amazon Corretto), Selbsttests und die schweren 2,36-GB-Produktionsläufe vom Repository-Root:

```
python3 -m pip install -r scripts/curriculum_fwu_owl_validation_requirements.txt
bash scripts/provision_pinned_robot.sh
python3 -B scripts/check_curriculum_fwu_owl_validation_tools.py \
  --report tmp/curriculum-release-model/fwu-owl-validation/tools-report.json
python3 -B scripts/validate_fwu_owl_curriculum_package.py --self-test
python3 -B scripts/reconstruct_json_curriculum_package_from_fwu_owl.py --self-test
python3 -B scripts/run_fwu_owl_reverse_compiler_hermetic.py --self-test
python3 -B scripts/validate_curriculum_fwu_owl_reverse_compilation_contract.py \
  --self-test
bash scripts/run_curriculum_fwu_owl_package_conformance.sh
bash scripts/run_curriculum_fwu_owl_reverse_conformance.sh
```

Ausführung des schnellen Legacy-Slim-Piloten aus `app/` (semantisch, OWL-2-DL-/HermiT-Reasoning und optionaler technischer Zeilen-Träger):

```
cd app
npm run export:subject-package -- --subject Mathematik --version 0.1.0
npm run export:subject-packages:validate -- \
  --zip ../tmp/exports/skillpilot-de-gymnasium-mathematik-v0.1.0.zip
npm run roundtrip:mem-fwu:semantic
ROBOT_JAR=/abs/path/to/robot.jar npm run roundtrip:mem-fwu:owl:reason
npm run roundtrip:mem-fwu
```

5 Klausurbeispiel-Produktionspipeline

Dieses Kapitel definiert die allgemeine, wiederholbare Blaupause zur Erstellung qualitätsgesicherter Klausur- und Abitur-Beispielpakete aus amtlichen Prüfungsvorgaben, bevor diese stabilisierten Pakete als `examData` in den Skill-Graph übernommen werden (1).

5.1 Verzeichnisstruktur und Blueprint-Schema

Fachübergreifende Steuerdateien liegen unter `curricula/<...>/abi/<Subject>/` (`exam-example-authoring-guide.md`, `input/references.md`, `*_exam_blueprint*.json`); jedes Beispielpaket erhält einen eigenen Ordner wie `Klausurbeispiel2026_1/`:

```
curricula/<...>/abi/<Subject>/Klausurbeispiel2026_1/
  checkliste.md
  <Subject>_<Region>_<Year>_Klausurbeispiel_<n>.md
  <Subject>_<Region>_<Year>_Klausurbeispiel_<n>_Musterloesung.md
  Schritt5_Abschluss_QS.md
  findings.md
  Schritt7_Finding_Bearbeitung.md
  Schritt8_ReQS.md
```

Temporäre Ähnlichkeits- oder Originalitätsberichte gehören ausschließlich unter `tmp/`. Der Planungs-Blueprint hält Abdeckung, Auswahlregeln und Komplementarität zu früheren Beispielen vorab fest:

```
{
  ëxamSpecId: ßubject-region-year-example",
  "locale: "de-DE",
  ßubject: "Physik",
  "frameworkRefs: [Öfficial exam hint", "Curriculum"],
  ßelectionModes: {},
  "contentFocus: {},
  "proposalMatrix: [],
  "globalCoverageChecks: []
}
```

5.2 Zehnstufiger Workflow (Schritt 0 bis Schritt 9)

1. **Schritt 0 (Quellen registrieren):** Erlasse, Kerncurricula, Stilkorpora und frühere Skill Pilot-Beispiele in `input/references.md` erfassen und amtliche Rechtsquellen strikt von Stilreferenzen trennen.
2. **Schritt 1 (QA-Checkliste anlegen):** Prüfbare Kriterien für Formalia, Auswahlmodus, Themenabdeckung, Anforderungsniveau, Originalität, Hilfsmittel und Konsistenz zwischen Aufgabe, Lösung und Bewertung definieren (`checkliste.md`).
3. **Schritt 2 (Blueprint entwerfen):** `GK`/`LK`-Varianten, Auswahlregeln, Teilaufgaben-Matrix und Komplementarität/Originalität gegenüber bestehenden Beispielen planen.
4. **Schritt 3 (Aufgabenblatt verfassen):** Eigenständige, nicht kopierte Aufgaben mit realistischem Kontext, sauberer Operatorensprache, konsistenten Einheiten/Daten und rein lernendenseitiger Formulierung erstellen.
5. **Schritt 4 (Musterlösung & Bewertung erstellen):** Vollständigen Lösungsweg je Teilaufgabe, transparente `BE`-Zuordnung, Begründungen sowie Rundungs-/Toleranzregeln dokumentieren.

6. **Schritt 5 (Abschluss-QS durchführen):** Punkt-für-Punkt-Bewertung gegen die Checkliste samt Restrisiken und Status (`draft-ready`, `release candidate`) in `Schritt5_Abschluss_QS.md` festhalten.
7. **Schritt 6 (Findings erfassen):** Review-Rückmeldungen strukturiert in `findings.md` aufnehmen (ID, Quelle, Beschreibung, Priorität, betroffene Dateien, Akzeptanzkriterium, Status).
8. **Schritt 7 (Findings abarbeiten):** Umgesetzte Änderungen, Ort, Begründung, Ähnlichkeits-/Originalitätsprüfungen und Folgerisiken in `Schritt7_Finding_Bearbeitung.md` protokollieren.
9. **Schritt 8 (Re-QS nach Findings):** Status-Delta (`behoben`, `teilweise behoben`, `offen`), betroffene Checklistenpunkte und aktualisierten Releasestatus in `Schritt8_ReQS.md` bestätigen.
10. **Schritt 9 (Graph-Integration):** Erst nach inhaltlicher Stabilisierung neue Prüfungsknoten mit `examData.taskContent`, `examData.solutionContent` und `examData.scoring` in das Landschafts-JSON eintragen (`docs/concept/skill-graph/node-types.md`). Fachspezifische Guides (`exam-example-authoring-guide.md`) instanziieren diesen Prozess mit konkreten Quellen, Pfaden und aktivem Schritt, ohne den allgemeinen Workflow zu duplizieren.

6 Produktionspipeline für Sek-I-Mathematikprüfungen

Dieses Kapitel definiert den geprüften Produktionsworkflow für kanonische Gymnasium-Mathematikprüfungen der Sekundarstufe I (Jahrgangsstufen 5 bis 10). Jede sichtbare Jahrgangsstufe erhält genau einen Ordner `Prüfungen Jahrgangsstufe <n>` mit einzelnen atomaren Prüfungsaufgaben darunter; weder ein Sammel-Capstone `Sek-I-Abschlussaufgaben Mathematik` noch ein pauschaler Zweig `Übungen Sekundarstufe I` darf als lernendenseitiger Routenendpunkt genutzt werden.

6.1 Quellenhygiene und Arbeitspaket-Layout

Externe Referenzaufgaben dienen ausschließlich der Stil-, Qualitäts- und Schwierigkeitskalibrierung — niemals als Kopier-, Übersetzungs- oder Paraphrasiervorlage:

- Heruntergeladene Beispiele bleiben lokal unter `tmp/seki-math-exam-reference-pool/` (inkl. `sources.json` oder `README.md` mit URL, Anbieter, Abrufdatum, Jahrgangsbezug und Verwendungszweck) und werden **nie** in Git eingecheckt.
- Bevorzugte Referenzpools: IQB Sek I / VERA-8, Landesbildungsserver / ZSL / Lehrerfortbildungsserver, kuratierte Hochschul-/Landesdatenbanken wie SMART sowie amtliche Lehrplanseiten (keine privaten Arbeitsblätter mit unklaren Rechten, Lösungs-Scraper oder ungeprüften KI-Listen).
- Bei externen Designideen darf höchstens das abstrakte Konstruktionsprinzip notiert werden, niemals Wortlaut, Zahlen, Rahmengeschichte, Grafik oder Lösungsweg.

Gemeinsame Checkliste ist `curricula/DE/Gymnasium/assessments/mathematik/seki/quality-checklist.md`; je Jahrgangsstufe wird ein Arbeitspaket angelegt (flache Altdateien wie `j5_terminal_autonomy_v1.md` gelten nach Promotion als abgelöst):

```
curricula/DE/Gymnasium/assessments/mathematik/seki/j5/
  blueprint.md & draft_v1.md & solution_v1.md
  internal_qa_v1.md & external_review.md
  findings.md & finding_resolution.md
  re_qa.md & release_candidate.md
```

6.2 Zehnstufiger Gate-Workflow und Releaseregel

Schritt	Pflichtergebnis & Freigabe-Gate
0. Referenzpool	Mindestens zwei unabhängige hochwertige Quellfamilien unter `tmp/seki-math-exam-reference-pool/` gesichtet; keine Fremdaufgaben im Git.
1. Checkliste	`quality-checklist.md` vor Entwurfsbeginn durch Product Owner oder externes Review bestätigt.
2. Blueprint	Jahrgang, Bearbeitungszeit, abgedeckte kanonische Ziel-IDs, Leitideen/Prozesskompetenzen, Punkte-/Anforderungsmatrix, explizite Nicht-Abdeckung und Hilfsmittel dokumentiert.
3. Roh-Entwurf	Altersgerechter Kontext, zugänglicher Einstieg plus Begründungs-/Prüf-/Modellgrenzen-Impuls je Aufgabe, unabhängige Teilaufgaben ohne versteckte Annahmen (`draft_v1.md`).
4. Lösung & BE	Vollständiger Lösungsweg, transparente `BE`-Vergabe auf beobachtbare Schülerleistung, zulässige Alternativwege und typische Fehlerhinweise (`solution_v1.md`).

Schritt	Pflichtergebnis & Freigabe-Gate
5. Interne QS	`internal_qa_v1.md` mit Status `pass`, `fail`, `blocked` oder `not_applicable` je Punkt; null `fail`/`blocked` in releasekritischen Abschnitten.
6. Externes Review	`external_review.md` mit positivem Votum oder konkretem Sanierungsplan für alle Blocker.
7. Findings-Iteration	`findings.md` & `finding_resolution.md`; jeder Befund ist begründet `resolved`, `accepted_risk` oder `out_of_scope`.
8. Re-QS & Kandidat	`re_qa.md` & `release_candidate.md`: Checkliste komplett, externe Blocker gelöst, Originalität und Quellenhygiene bestätigt.
9. Graph-Promotion	Kanonische Knoten aktualisieren (`examData.reviewStatus = "released"`, präzise `examData.coveredGoalIds`, exakt geprüfte `requires`), Quellen-/ Rationale-Berichte erneuern und vollständige CI samt Kompositionsansichten unter `Prüfungen Jahrgangsstufe <n>` bestehen.