```
module: 17
title: Automation Engine
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-3, LP-4, LP-5, LP-6, LP-9, LP-10, LP-11]
spec_style: requirements-pur
```

Modul 17 — Automation Engine

> ***Detail-Spec gemäß LP-9.*** *USP-Säule. Löst Zapier+n8n+Supabase ab (LP-7). KI als Bedienoberfläche (LP-5). Anti-Agentur-Positionierung (LP-6) durch Klartext-Diagnostik + Selbst-Wartbarkeit.*

Themenblöcke

```
A. Datenmodell (Rule, Step, Execution, Credential, Template)
B. Trigger-Library (Phase 1, Geerkens-fokussiert)
C. Action-Library (Phase 1)
D. Visueller Flow-Editor (Graph-basiert)
E. KI-Erstellung per natürlicher Sprache (LP-5)
F. Dry-Run / Vorschau / Reversibilität (PM-3)
G. Schutz-Mechanismen (Loop-Guard, Rate-Limit, Anomalie-Detection – EC-1, EC-2)
H. Idempotenz + Resume-Mode (EC-3)
I. Templates (Branchen-modular)
J. Klartext-Diagnostik in Execution-Logs (LP-10)
K. Stage-Stuck-Eskalation als Built-in-Cron
L. Trigger-Volumen-Metering (für Pricing)
M. Permissions
```

Block A — Datenmodell

Architektur-Entscheidung

Graph-basierter Flow ab Phase 1. Knoten + Edges, parallele Branches möglich, Verzweigungen über If/Else. Mehr Komplexität, aber matched alte Codebase + mächtige Use-Cases.

Kern-Entitäten

```
AutomationRule           – eine Wenn-Dann-Regel pro Tenant
AutomationRuleStep       – einzelne Knoten in einer Rule
AutomationExecution      – Lauf einer Rule
AutomationStepExecution  – pro Step Status + Output
AutomationCredential     – Tenant-Credentials für Action-Knoten
AutomationTemplate       – vorgefertigte Workflows
```

AutomationRule

- `id`, `tenant_id`, `name`, `description`
- `flow_data` (JSON — Graph-Struktur: Nodes + Edges, kompatibel mit Flow-Library)

- `enabled` (Boolean)
- `created_by_user_id`, `created_via` (`manual | template | ki_natural_language`)
- `last_executed_at`, `executions_count_30d`
- `archived_at`

AutomationRuleStep

- `id`, `rule_id`, `step_kind` (`trigger | condition | action | flow_control`)
- `node_type` (z.B. `deal.stage_changed`, `condition.field_equals`, `action.send_email`, `flow.delay`)
- `config` (JSON — Knoten-spezifisch)
- `parent_step_ids[]` (für Multi-Parent-Merge nach parallelen Branches)
- `position` (für Editor-Layout)

AutomationExecution

- `id`, `tenant_id`, `rule_id`, `triggered_at`
- `trigger_payload` (JSON)
- `status` (`running | success | partial_success | failed | cancelled`)
- `started_at`, `finished_at`
- `summary_text` (Klartext)
- `dry_run` (Boolean)
- `triggered_by_user_id` (nullable)
- `idempotency_key`
- `reverse_window_expires_at` (24h nach Execution für Undo)

AutomationStepExecution

- `id`, `execution_id`, `step_id`
- `started_at`, `finished_at`
- `status` (`pending | running | success | skipped | failed`)
- `input` (JSON)
- `output` (JSON — für Folge-Steps)
- `summary_text` (Klartext)
- `error_text` (Klartext bei Fehler)
- `idempotency_key`

AutomationCredential

- `id`, `tenant_id`, `key`, `value` (verschlüsselt at-rest), `description`
- Apps aus M33a stellen ihre Connections automatisch zur Verfügung; AutomationCredential nur für nicht-App-Credentials

AutomationTemplate

- `id`, `tenant_id` (NULL bei Plattform-Templates), `name`, `description`
- `flow_data` (JSON)
- `category`
- `is_published`, `branchen_template_origin`

Block B — Trigger-Library (Phase 1, Geerkens-fokussiert)

Pro Trigger: Trigger-Payload-Daten stehen für Folge-Steps zur Verfügung. **Multi-Pipeline-Filter** bei pipeline-bezogenen Triggern: Pipeline + Stage als Multi-Select.

CRM (M1)

- `contact.created`, `contact.updated`, `contact.tag_added`

Lead (M2)

- `lead.received` (mit Source-Filter), `lead.status_changed`, `lead.assigned`

Sales (M3)

- `deal.created` (mit Pipeline-Filter)
- `deal.stage_changed` (mit Pipeline + From-/To-Stage-Filter)
- `deal.won`, `deal.lost` (mit Pipeline-Filter)
- `deal.stuck.in_stage`
- `deal.assignee_changed`
- `deal.volume_changed`

Projects (M4)

- `project.created` (mit Pipeline-Filter)
- `project.stage_changed` (mit Pipeline + Stage-Filter)
- `project.completed`, `project.cancelled` (mit Pipeline-Filter)
- `project.stuck.in_stage`
- `project.member_added`, `project.member_removed`
- `project.acceptance_completed` (mit `kind`-Filter)
- `schedule_entry.created`, `schedule_entry.changed`, `schedule_entry.cancelled`

Forms (M5)

- `form.submitted` (pro Form auswählbar)

Booking (M6)

- `booking.scheduled`, `booking.cancelled`, `booking.held`

Email (M9)

- `email.received_inbox` (pro Mailbox)
- `email.subject_pattern_matched` (Regex)

Telephony (M12)

- `call.inbound_completed`
- `call.outbound_completed`
- `call.missed`

Tasks (M26)

- `task.created`, `task.assigned`, `task.completed`
- `task.deadline_approaching`, `task.overdue`

Time-based (Built-in)

- `schedule.cron` (CRON-Expression)
- `schedule.delay` (Folge-Step nach Delay)

KWP (M19)

- `kwp.auftrag_pulled`
- `kwp.schema_drift_detected`

Apps Engine (M33a)

- `app.connection_status_changed`
- `app.sync_failed`

HR (M14)

- `hr.vacation_approved`
- `hr.sick_leave_reported`

Customer Portal (M15)

- `portal.kunde_uploaded_file`
- `portal.kunde_replied`

Block C — Action-Library (Phase 1)

CRM

- `contact.update`, `contact.add_tag` / `remove_tag`, `contact.assign_to_user`

Lead

- `lead.assign_to_user`, `lead.set_status`, `lead.promote_to_contact`

Sales (mit Pipeline-Targeting)

- `deal.create` (mit Ziel-Pipeline + Start-Stage)
- `deal.update_stage` (zu konkreter Stage oder semantisch `won | lost | next_open`)
- `deal.update_volume`, `deal.assign_to_user`, `deal.add_tag`
- `deal.move_to_pipeline` (Deal in andere Pipeline verschieben)

Projects (mit Pipeline-Targeting)

- `project.create` (mit Ziel-Pipeline + Start-Stage)
- `project.create_from_won_deal` (Won-Handoff mit konfigurierbarer Default-Mapping pro Sales-Pipeline)
- `project.update_stage`
- `project.add_member`
- `project.create_task`
- `project.create_schedule_entry`
- `project.start_acceptance`
- `project.move_to_pipeline`

Communication

- `email.send` (mit Template + dynamischen Vars)
- `chat.send_message` (in Channel)
- `notification.send_push`, `notification.send_in_app`

Tasks

- `task.create`, `task.assign_to_user`

KWP

- `kwp.push_auftrag`

HTTP / External

- `http.request` (GET/POST/PUT/DELETE)
- `webhook.send` (signiert)

Flow-Control

- `delay`, `if_else`, `parallel`, `merge`
- `loop` — **NICHT in Phase 1** (Loop-Guard-Komplexität)

KI (M18)

- `ai.generate_text`, `ai.summarize`, `ai.classify`

Calendar (M25)

- `calendar.create_event`, `calendar.update_event`

Files (M23)

- `file.create_from_url`

Conditions (für If/Else)

- Field-Equals / Contains / Greater-Than / Less-Than
- Tag-Set / Tag-Not-Set
- Time-Of-Day / Day-Of-Week
- User-In-Group
- Custom-JS-Expression — **Phase 2** (Sicherheits-Risiko)

SHK-Template-Defaults: Sales-Pipeline → Project-Pipeline (Won-Handoff)

Sales-Pipeline	Default-Project-Pipeline
Anlagen-Verkauf	Anlagen-Installation
Wartungsverträge	Wartungseinsatz

Block D — Visueller Flow-Editor

Editor-Anforderungen (`/{tenant}/automations/:id/edit`)

Zu zeigende Daten:
- Canvas mit Graph-Struktur (Knoten + Verbindungen)
- Sidebar mit Knoten-Catalog (filterbar nach Modul-Quelle)
- Konfigurations-Bereich für ausgewählten Knoten
- Mini-Map für große Flows
- Toolbar: Zoom, Pan, Undo/Redo, Zentrieren, Auto-Layout

Benötigte UI-Elemente:
- Graph-Canvas-Komponente (Library-Wahl über Boilerplate)
- Knoten-Catalog-Panel mit Drag-&-Drop oder Klick-Hinzufügen
- Knoten-Konfigurations-Panel
- Verbindungs-Erstell-Funktion
- Validation-Anzeige
- Test-Run-Funktion (Dry-Run)
- Save / Activate / Deactivate-Funktion
- Versionierung-Anzeige (Phase 2)

Operationen:
- Knoten hinzufügen / konfigurieren / löschen / duplizieren
- Verbindungen erstellen / löschen
- Knoten verschieben / Auto-Layout
- Test-Run mit Sample-Payload
- Save (versionierter Snapshot)
- Activate / Deactivate
- Rule duplizieren

Klartext im Knoten-Label:
- Trigger: *„Wenn Deal in Pipeline ‚Anlagen-Verkauf' → Stage ‚Verhandlung'"*
- Condition: *„Wenn Volumen > 10.000 EUR"*
- Action: *„Push an Noah G. + Vertriebsleiter"*
- Delay: *„Warte 24 Stunden"*

Validation-Hinweise:
- Knoten ohne Verbindung: *„Knoten ‚Email senden' ist nicht verbunden — wird nie ausgeführt"*
- Cycle erkannt: *„Diese Verbindung würde eine Schleife bilden — bitte anders verkabeln"*
- Knoten unvollständig: *„Bei ‚Email senden' fehlt der Empfänger"*

Block E — KI-Erstellung per natürlicher Sprache (LP-5)

User sagt z.B.: *„Wenn jemand das Kontaktformular ausfüllt und das Volumen über 10.000 EUR ist, schicke mir und dem Vertriebsleiter eine Push-Benachrichtigung und lege den Lead in der Premium-Pipeline an."*

KI-Pipeline (intern)

1. **Intent-Klassifikation:** neue Automation oder Edit?
2. **Entity-Extraction:** Trigger, Filter, Actions
3. **Schema-Validation:** existieren referenzierte Pipelines, User, Forms? (EC-4)
4. **Rückfrage bei Mehrdeutigkeit:** *„Du meinst Klaus Müller (Vertriebsleiter Süd) oder Klaus Schmidt (Bauleiter Nord)?"* (EC-4)

5. **Flow-Generierung:** Graph-Struktur als JSON
6. **Vorschau-Rendering:** Flow im Editor + Klartext-Beschreibung
7. **Risiko-Hinweise:** *„Achtung: ‚Volumen über 10.000' kann auch bei Wartungsverträgen mit hohem Vertragswert greifen — nur Anlagen-Verkauf-Pipeline einschränken?"* (EC-4)

Edit-Mode

Bestehende Automation: *„Füge eine Wartezeit von 1 Tag zwischen der Email und der Push-Benachrichtigung ein"* → KI baut Diff-Vorschlag.

Sicherheits-Layer

- **Action-Boundary** (EC-5): KI führt nicht direkt aus, bereitet Flow vor — Bestätigung Pflicht
- **Schema-Validation** vor Vorschau (EC-4)
- **Klartext-Risiko-Hinweise** (EC-4)
- **Promptfoo-Eval-Suite** als CI-Pflicht (EC-5)

UI-Anforderungen

- Eingabe-Funktion (Text oder Voice via Whisper-EU M18)
- Live-Streaming-Antwort
- Vorschau-Rendering im Flow-Editor
- Bestätigungs-/Anpassungs-Funktion
- Beispiel-Prompts als Vorschläge (Onboarding)

Block F — Dry-Run / Vorschau / Reversibilität (PM-3)

Dry-Run-Mode

- Engine simuliert Rule mit Sample-Payload
- **Keine echten Side-Effects**
- Klartext-Anzeige pro Step:
- *„Diese Automation würde jetzt 1.400 Personen treffen. Hier die ersten 5: ..."*
- *„Email würde versandt an: noah@geerkens.de mit Subject ‚Premium-Lead'"*
- Bestätigung Pflicht: *„Trotzdem aktivieren?"*

Vor-Aktivierungs-Pflicht

- Rule mit > 100 Datensätzen pro Schritt → Dry-Run-Bestätigung Pflicht
- Schwellenwert konfigurierbar pro Tenant (Default 100)

Reversibilität (24h-Reverse-Window)

- Jede Execution hat `reverse_window_expires_at = triggered_at + 24h`
- Innerhalb 24h reversible Wirkungen rückgängig:
- Mails: nicht zurückziehbar, aber Empfänger-Liste anzeigbar mit Klartext
- Angelegte Entitäten: hard-löschen mit Audit
- Status-Änderungen: zurücksetzen
- Tag-Änderungen: rückgängig
- Klartext: *„Du hast noch 23h 12min, um diese Ausführung zurückzunehmen."*
- Multi-Step-Confirm + Audit-Trail-Eintrag pro Reverse

Block G — Schutz-Mechanismen (EC-1, EC-2)

Rate-Limit (EC-1)

- **Hard-Limit pro Trigger:** Default 1.000 Auslösungen/h, Tenant-konfigurierbar
- **Anomalie-Detection:** 10x normale Rate → Pause + Klartext-Notification: *„Die Automation ‚Lead-Begrüßung' wurde 2.000-mal in 5 Min ausgelöst — typisch sind 5-10 pro Tag. Wir haben sie pausiert. Bitte prüfen."*

Loop-Guard (EC-2)

- **Hardes Limit:** max. 5 Auto-Trigger pro Datensatz pro 60 Sek
- **Klartext-Stopp** bei Loop-Erkennung: *„Automation A löst B aus, B löst A aus — Schleife erkannt, beide pausiert. [Visualisierung] [Beheben]"*
- **Pre-Save-Validation** beim Erstellen: System prüft Abhängigkeits-Graph, warnt vor Schleifen

---

Block H — Idempotenz + Resume-Mode (EC-3)

Idempotenz pro Action

- Jeder Action-Step hat `idempotency_key`
- Action prüft vor Ausführung — wenn schon erledigt, skip mit Status `skipped`

Resume-Mode

- Retry beginnt am fehlgeschlagenen Step, nicht von vorne
- Klartext: *„Automation X war zu 60% durch (3/5 Schritte erfolgreich). Schritt 4 (KWP-Sync) fehlgeschlagen. Auto-Retry in 5 Min. [Manuell wiederholen] [Ab Schritt 4 weitermachen] [Abbrechen]"*

Auto-Retry-Strategie

- Exponential Backoff: 1 Min, 5 Min, 15 Min, 1 Std, 6 Std, 24 Std
- Nach 6 Retries → `failed` mit Klartext-Email an Tenant-Admin
- Manueller Retry möglich

Block I — Templates

Plattform-Templates (Code-Registry, branchen-modular)

SHK-Standard-Templates (Vorschlag):
- *„Lead-Begrüßung"* — neuer Lead → Auto-Email mit Wartungstermin-Vorschlag
- *„Won-Handoff zur Installation"* — Deal Won → Auto-Project + Push an PL
- *„Wartungstermin-Reminder"* — 1 Woche vor Termin Reminder (Phase 2 für WhatsApp)
- *„Stuck-Deal-Eskalation"* — Deal > 14 Tage in „Angebot" → Push an Vertriebsleiter
- *„Anrufe-Rückruf"* — verpasster Anruf → Task für Vertriebler mit 24h-Deadline
- *„Acceptance-Komplett-Workflow"* — `acceptance_completed` mit `kind=full` → Email + Customer-Portal-Update + KWP-Rechnungs-Trigger
- *„Mängel-Tasks-Generator"* — `acceptance_completed` mit `kind=with_defects` → Tasks für jeden Mangel mit 7-Tage-Deadline

Tenant-Templates (User-definiert)

- Tenant kann eigene Templates anlegen
- Phase 2: Sharing über Marketplace (M33c)

Template-Catalog (`/{tenant}/automations/templates`)

- Liste aller Templates (Plattform + Tenant), gruppiert nach Kategorie
- Pro Template: Beschreibung + Beispiel-Use-Case + Anzahl Tenants die es nutzen

Operationen:
- Template auswählen → Rule-Instanz aus Template
- Eigenes Template anlegen aus existierender Rule
- Template editieren / archivieren

Block J — Klartext-Diagnostik in Execution-Logs (LP-10)

Execution-Detail-View (`/{tenant}/automations/:id/executions/:exec_id`)

Zu zeigende Daten:
- `summary_text` (Top-Level)
- `triggered_at` , `finished_at` , Dauer
- Trigger-Payload (User-readable)
- Step-by-Step-Verlauf mit Status, Klartext, Input/Output, Dauer
- `dry_run` -Marker
- `reverse_window` -Status

Benötigte UI-Elemente:
- Step-by-Step-Liste oder Timeline
- Pro Step: Detail-Panel mit Input/Output
- Filter nach Status
- Funktion *„Wirkungen rückgängig machen"* (innerhalb 24h)
- Funktion *„Manuell retry"*
- Funktion *„Detail-Logs für Plattform-Admin"* (zeigt `tech_details`)

Execution-Liste (`/{tenant}/automations/:id/executions`)

- Letzte N Executions
- Filter nach Status, Datum, Trigger-Typ
- Klick öffnet Detail

Klartext-Beispiele

- Erfolgreich: *„Lead-Begrüßung wurde ausgeführt. Email an max.mueller@example.com gesendet. Dauer: 0.8s."*
- Halb-Erfolg: *„Won-Handoff: Project erstellt ✓, Push gesendet ✓, KWP-Sync fehlgeschlagen ✗ — Verbindung zu KWP nicht erreichbar. Auto-Retry in 5 Min."*
- Fehler: *„Fehler beim Schritt ‚Email senden': Empfänger-Email nicht zustellbar. [Empfänger prüfen]"*
- Skipped: *„Schritt ‚Email senden' wurde übersprungen, weil ‚Bedingung Volumen > 10.000' nicht erfüllt war (tatsächlich: 2.500)."*

Block K — Stage-Stuck-Eskalation (Built-in-Cron)

- Daily-Cron checkt offene Deals/Projects auf Stage-Stuck (länger als `stage.auto_archive_after_days`)
- Bei Stuck: Trigger-Event `deal.stuck.in_stage` / `project.stuck.in_stage` mit Metadata
- Tenant kann eigene Automations darauf hängen
- Default-Template *„Stuck-Deal-Eskalation"* nutzt das

Konfigurierbar pro Stage:
- `auto_archive_after_days`
- Tenant-Admin passt Schwellenwerte an

Block L — Trigger-Volumen-Metering

- Pro Tenant pro Monat: Anzahl `AutomationExecution`-Einträge
- Visualisierung in `/{tenant}/admin/usage/`
- Soft-Limit-Warnung bei 80% des Plan-Limits
- Hard-Limit bei Überschreitung: neue Executions werden gequeued + Klartext: *„Trigger-Volumen für diesen Monat überschritten. Automations laufen ab nächstem Monat wieder. Erweiterung im Plan möglich (extern auf hivegrid.de/pricing)."*
- Konkrete Limits pro Plan kommen mit Pricing-Definition (geparkt)

Block M — Permissions

Permission	Erlaubt
`automation.read`	Liste sehen, Logs sehen
`automation.test`	Dry-Run / Test-Run
`automation.write`	Rules anlegen + bearbeiten
`automation.activate`	Rules aktivieren / deaktivieren
`automation.delete`	Rules löschen
`automation.reverse`	Wirkungen rückgängig machen (innerhalb 24h)
`automation.use_ki`	KI-Erstellung per Sprache
`automation.admin`	Volle Verwaltung + Templates + Trigger-Volumen-Override

Bundle-Mapping

Bundle	Atomare Permissions
`automation.basic`	`read`
`automation.builder`	obigen + `test` + `write` + `use_ki`
`automation.publisher`	obigen + `activate` + `reverse`
`automation.admin`	alle obigen + `delete` + `admin`

Default-Rollen-Zuweisung (SHK-Template)

Rolle	Permission
GF	`automation.admin`
Vertriebsleiter	`automation.publisher`
Vertriebler	`automation.basic`
PL	`automation.builder` (bauen, aber Aktivierung durch GF)
Büro	`automation.basic`
Marketing	`automation.publisher`
Monteur	(keine)

Klartext-Fehler

- Aktivierung ohne `activate`: *„Du kannst Automations bauen, aber das Aktivieren erfordert Freigabe durch einen Admin. [Freigabe anfordern]"*
- Reverse nach 24h: *„Diese Ausführung kann nicht mehr zurückgenommen werden — das 24h-Fenster ist abgelaufen."*
- KI-Nutzung ohne Permission: *„Die KI-gestützte Erstellung ist im Cloud-Plus-Plan verfügbar."*

Architektur-Vorschriften (Querschnitt)

- **LP-5:** KI als Bedienoberfläche der Engine (Block E)
- **LP-6:** Anti-Agentur — Selbst-Wartbarkeit über Klartext + KI-Erstellung
- **LP-10, AV #1:** Klartext-Diagnostik zentral (Block J)
- **LP-11, AV #2:** Adapter-Pattern (Knoten konsumieren Apps aus M33a)
- **AV #5:** Permissions-aware-Default
- **AV #6:** Audit-Trail für Rule-Änderungen
- **AV #8:** Idempotenz pro Action (Block H)
- **AV #10:** Dry-Run + Reversibilität (Block F)
- **AV #14:** Promptfoo-Eval-Suite für KI-Erstellung
- **AV #16:** Rate-Limiting + Anomalie-Detection (Block G)
- **AV #17:** Loop-Guard mit Klartext (Block G)

Detail-Spec-Status: DRAFT

Modul 17 komplett spec'd — größte und wichtigste USP-Säule.

Offene Detail-Punkte:
- Konkrete Flow-Editor-Library-Wahl (Boilerplate-Entscheidung: React Flow / Reaflow / X6 / Custom)
- Konkrete Pre-Save-Loop-Validation-Algorithmus
- Konkrete Anomalie-Detection-Schwellwerte pro Trigger-Typ
- Sample-Payload-Quelle für Dry-Run (echter Datensatz vs. synthetisch)
- KI-Prompt-Templates pro Use-Case (Implementierungs-Detail)
- Konkrete Trigger-Volumen-Limits pro Plan (kommt mit Pricing)