```
module: 1
title: CRM-Kern (Kontakte + Firmen + Leads)
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-7, LP-9, LP-10, LP-13]
spec_style: requirements-pur (siehe Memory feedback_spec_style_no_ui)
```

Modul 1 — CRM-Kern

> ***Detail-Spec gemäß LP-9.*** *Stammdaten-Grundlage, viele andere Module verlinken hier rein. Löst HubSpot-CRM-Tool ab (LP-7).*

Themenblöcke

```
A. Datenmodell (Contact, Company, Lead, ContactCompanyRelation)
B. Menü-Struktur und Routes
C. Listen-Views (Filter, Sortierung, Bulk-Actions)
D. Detail-Views
E. Activity-Stream
F. Custom-Felder
G. Suche & Duplikat-Erkennung
H. Tags
I. Permissions-Anwendung
```

Block A — Datenmodell

Drei Hauptentitäten + Verknüpfungstabellen

```
Contact                       – eine Person
Company                       – eine Firma (juristische Person, B2B)
Lead                          – unqualifizierter Eingang (eigene Entität, NICHT Contact mit
Status)
ContactCompanyRelation        – Verknüpfung Contact ↔ Company als M:N
ContactEmail / ContactPhone   – separate Tabellen für Mehrfach-Werte mit Typ
ContactHistory                – siehe Block E (Activity)
```

Contact

- `id`, `tenant_id`, `created_at`, `updated_at`, `created_by_user_id`
- `salutation` (Frau/Herr/Divers/Keine)
- `title` (Dr./Prof./...)
- `first_name`, `last_name` (Pflicht)
- (Email-Werte über `ContactEmail`)
- (Telefon-Werte über `ContactPhone`)
- `address_street`, `address_zip`, `address_city`, `address_country`
- `birthdate` (optional)

- `notes` (Markdown-Freitext)
- `language` (DE/EN, default DE — für Outbound-Kommunikation)
- `assigned_to_user_id` (optional — Hauptverantwortlicher)
- `lead_source` (Enum: `form | email | whatsapp | phone | manual | import | referral | other`)
- `is_endkunde` (Boolean — Privat oder B2B-Ansprechpartner)
- `archived_at` (Soft-Delete)

Company

- `id`, `tenant_id`, `created_at`, `updated_at`, `created_by_user_id`
- `name` (Pflicht)
- `legal_form` (GmbH, AG, Einzelunternehmer, GbR, UG, ...)
- `vat_id` (USt-Id), `tax_number` (Steuernummer)
- `industry` (optional Freitext)
- `website`
- `address_*` (Standort-Adresse)
- `notes`, `assigned_to_user_id`, `archived_at`

Lead (eigene Entität, NICHT Contact mit Status)

- `id`, `tenant_id`, `created_at`, `updated_at`
- `first_name`, `last_name`, `email`, `phone` (alle nullable — Form-Submissions sind manchmal dünn)
- `message` (Freitext aus Form / Email / Anruf-Notiz)
- `source` (Enum: `form | email | whatsapp | phone | manual | import | referral`)
- `status` (Enum: `new | qualified | converted | lost | spam`)
- `assigned_to_user_id`
- `converted_to_contact_id` (FK auf Contact, falls promoted — sonst null)
- `converted_to_deal_id` (FK auf Deal, falls bei Promotion auch Deal angelegt)
- `archived_at` — für DSGVO-Auto-Delete (z.B. 90 Tage nach `lost` / `spam`)

Begründung der Trennung: Lead ist semantisch *„potenzielle Person"* — Daten noch nicht qualifiziert/verifiziert. Bei Promotion zu Contact bestätigen wir: echte Person, ins CRM aufgenommen. DSGVO-Vorteil: Spam-Leads können ohne Audit-Trail-Last gelöscht werden.

ContactCompanyRelation (M:N)

- `contact_id`, `company_id`
- `role` (Freitext, z.B. „Geschäftsführer", „Hausmeister", „Bauherr")
- `is_primary` (Default-Firma für diesen Contact)
- `since_date`, `until_date` (optional)

Begründung M:N: eine Person kann bei mehreren Firmen sein (Holdings, Wechsel, Doppelfunktionen).

ContactEmail / ContactPhone

- `contact_id`, `value`, `label` (Enum: `private | business | billing | other`), `is_primary`
- Index für Live-Validation und Auto-Complete

Lead-Promotions-Logik

User klickt *„Als Kontakt übernehmen"* → Modal:

1. Felder vorausgefüllt aus Lead-Daten, editierbar
2. Optional: existierende Firma suchen oder neue anlegen + Rolle wählen
3. Optional: Deal anlegen (Pipeline + Stage + Volumen — Default ja)

4. Auto-Email-Match-Check: existiert bereits Contact mit dieser Email?
   - **(a) Existing Contact aktualisieren** (Default — leere Felder ergänzen, neuer Activity-Eintrag, `Lead.converted_to_contact_id` = existing)
   - **(b) Trotzdem neuen Contact anlegen**
   - **(c) Manuell zu anderem existing Contact zuordnen** (z.B. private vs. geschäftliche Email)
   - **(d) Als Spam markieren**
5. Bestätigen → Transaction: Contact + ggf. Company + ContactCompanyRelation + ggf. Deal angelegt; Lead bekommt `status = converted` + FKs

Statuse die ein Lead annehmen kann ohne Promotion:
- `qualified` (manuell, bleibt im Auge)
- `lost` (archiviert, für Re-Engagement-Statistik)
- `spam` (markiert für DSGVO-Auto-Lösch in 30 Tagen)
- Hard-Delete (sofort, ohne Grace)

Block B — Menü-Struktur und Routes

Sidebar-Hauptpunkt: *CRM* mit Sub-Punkten Kontakte / Firmen / Leads.

Routen:

```
/{tenant}/contacts                          → Kontakt-Liste
/{tenant}/contacts/new                      → Neuer Kontakt
/{tenant}/contacts/:id                      → Kontakt-Detail

/{tenant}/contacts/companies                → Firmen-Liste
/{tenant}/contacts/companies/new            → Neue Firma
/{tenant}/contacts/companies/:id            → Firma-Detail

/{tenant}/leads                             → Lead-Inbox (siehe Modul 2)
/{tenant}/leads/:id                         → Lead-Detail (mit Promotion)

/{tenant}/admin/custom-fields/              → Custom-Felder-Verwaltung
/{tenant}/admin/tags/                       → Tag-Verwaltung
/{tenant}/admin/duplicates/                 → Bulk-Duplikat-Tool
```

Block C — Listen-Views

Kontakt-Liste

Zu zeigende Daten pro Eintrag:
- Kontakt-Identität (Avatar/Initialen, Vor- + Nachname)
- Primäre Firma (aus ContactCompanyRelation mit `is_primary=true`)
- Primäre Email
- Primäres Telefon
- Tags
- Verantwortlicher (User)
- Letzte Activity (Datum + Typ)
- Erstellt-Datum (optional sichtbar)

Benötigte UI-Elemente:
- Tabellarische Listen-Anzeige
- Spalten-Header klickbar für Sortierung

- Such-Funktion (Volltext über Name/Email/Telefon/Notes/Custom-Felder)
- Filter-Komponenten:
- Verantwortlicher (Multi-Select)
- Tags (Multi-Select mit AND/OR-Toggle)
- Firma (Auto-Complete)
- Erstellt zwischen (Date-Range-Picker)
- Verknüpft mit Deal/Projekt (Boolean)
- Custom-Felder (siehe Block F)
- Toggle „Archivierte anzeigen"
- Tab-Navigation auf der Page für: Kontakte / Firmen / Leads
- Bulk-Auswahl (Checkboxes pro Zeile + Master-Checkbox)
- Pagination (Cursor-basiert, „Mehr laden", 50 pro Page)
- Spalten-Konfiguration (User wählt sichtbare Spalten — speichert pro User)

Operationen:
- Sortierung pro Spalte (Default: Letzter Activity-Wert absteigend)
- Filter anwenden / kombinieren / zurücksetzen
- Bulk-Actions:
- Tag hinzufügen / entfernen
- Verantwortlicher zuweisen
- Archivieren
- Export als CSV (gefilterte Auswahl)
- Klick auf Kontakt-Zeile → Detail-View

Performance-Anforderungen:
- Server-side-Filter (kein Client-side-JS-Filtern bei 10.000+ Einträgen)
- Index auf `tenant_id, archived_at, last_activity_at` für Default-Sortierung
- Trigram-Index auf Namens-Feldern für Fuzzy-Suche

Firmen-Liste

Analog zur Kontakt-Liste mit Spalten:
- Firma-Name
- Branche
- Anzahl verknüpfter Kontakte
- Verantwortlicher
- Letzte Activity
- Erstellt-Datum

Lead-Inbox

Spec siehe Modul 2.

Block D — Detail-Views

Kontakt-Detail (`/{tenant}/contacts/:id`)

Zu zeigende Daten:
- Stammdaten des Kontakts (alle Contact-Felder + Email/Phone-Listen)
- Verknüpfte Firmen mit Rolle pro Verknüpfung
- Tags
- Verantwortlicher User
- Verknüpfte Deals
- Verknüpfte Projekte

- Verknüpfte Files
- Custom-Felder (siehe Block F)
- Notizen (Markdown-Freitext)
- Activity-Stream (siehe Block E)

Benötigte UI-Elemente:
- Sektionierte Anzeige der Stammdaten + verknüpfter Entitäten + Notizen + Custom-Felder
- Activity-Stream-Komponente (siehe Block E für Spec)
- Inline-Edit-Funktion für jedes Stammdatenfeld (mit Save/Cancel)
- Action-Buttons für die Operationen (siehe unten)
- Markdown-Editor für Notizen
- Suchbares Multi-Select für Verknüpfungs-Verwaltung (Firmen ↔ Kontakt mit Rollen-Auswahl)
- Tag-Editor (Auto-Complete + neue Tags ad-hoc anlegbar)
- Tab- oder Sektionen-Navigation für: Übersicht / Deals / Projekte / Files / Custom-Felder

Operationen:
- Stammdaten editieren
- Click-to-Call (via AirCall-Integration falls verbunden, sonst `tel:`-Link)
- Email schreiben (Compose-Funktion mit vorausgefülltem Empfänger; sendet über verbundene Mailbox)
- Termin buchen (Booking-Slot-Picker)
- Deal anlegen (Kontakt vorausgewählt)
- Aufgabe anlegen (mit Kontakt-Verknüpfung)
- Tag hinzufügen / entfernen
- Verantwortlichen ändern
- Firma verknüpfen / lösen + Rolle ändern + Primary-Status setzen
- Email/Phone hinzufügen / als Primary setzen / löschen
- Archivieren / Reaktivieren
- Hard-Delete (mit Klartext-Warnung über offene Verknüpfungen)
- DSGVO-Daten-Export (alle Daten als JSON)

Permissions-Wirkungen:
- Ohne `crm.read_*`: Route nicht erreichbar (404)
- `crm.read_own`: nur sichtbar wenn `assigned_to_user_id = self`
- Schreib-Operationen: erfordern `crm.write`
- Hard-Delete und DSGVO-Export: erfordern `crm.admin`

Klartext-Hinweise:
- Kontakt ohne Email/Telefon: *„Keine Kontaktdaten hinterlegt — Anrufen/Mailen erst nach Eingabe möglich"* mit Direkt-Editor
- Archivierter Kontakt: Banner *„Kontakt ist archiviert — wieder aktivieren um zu bearbeiten"*
- Löschen mit offenen Verknüpfungen: Multi-Step-Confirm mit Anzahl Verknüpfungen pro Typ

Firma-Detail (`/{tenant}/contacts/companies/:id`)

Zu zeigende Daten:
- Stammdaten (alle Company-Felder)
- Verknüpfte Kontakte mit Rolle, primärer Ansprechpartner markiert
- Tags, Verantwortlicher
- Verknüpfte Deals und Projekte
- Files
- Custom-Felder
- Notizen
- Activity-Stream (Aggregat aller Aktivitäten der verknüpften Kontakte + firmen-eigene Activities)

Benötigte UI-Elemente und Operationen: analog Kontakt-Detail-View, plus:
- Kontakt-Verknüpfungs-Verwaltung (mit Rolle, primärer Ansprechpartner setzbar)

Permissions: analog Contact-View über `crm.*`.

Lead-Detail (`/{tenant}/leads/:id`)

Zu zeigende Daten:
- Lead-Felder (Name, Email, Phone, Source, Message, Status, Verantwortlicher)
- Auto-Match-Hinweis: existiert bereits Contact mit dieser Email?
- Activity-Stream des Leads (Form-Submission, Status-Änderungen, Email-Antworten)

Benötigte UI-Elemente:
- Anzeige der Lead-Felder
- Inline-Edit
- Promotion-Funktion (öffnet Promotions-Modal aus Block A)
- Status-Wechsler (`new | qualified | converted | lost | spam`)

Operationen:
- Promote zu Contact (siehe Block A)
- Status ändern
- Antwort senden (Email/WhatsApp falls Daten vorhanden)
- Verantwortlichen zuweisen
- Manuell als Spam markieren → triggert DSGVO-Auto-Lösch in 30 Tagen
- Hard-Delete

Klartext-Hinweise:
- Status `converted` : Banner *„Dieser Lead wurde zu Kontakt [Name] promoviert am [Datum]"* mit Link zum Contact
- Status `spam` : Banner *„Wird automatisch gelöscht am [Datum]"*

Permissions: über `leads.*` (Detail-Spec in Modul 2).

Block E — Activity-Stream

Datenmodell

Zentrale `Activity` -Tabelle (Event-Sourcing-Pattern): Jedes Modul, das Aktivitäten erzeugt, schreibt einen Eintrag. Ein Activity-Stream-View liest dann aus einer Quelle.

Felder:
- `id` , `tenant_id` , `created_at`
- `actor_user_id` (NULL bei System/Automation)
- `actor_kind` (`user | system | automation | ki_agent`)
- `activity_type` (Enum, siehe unten)
- `subject_type` (`contact | company | lead | deal | project | ...`)
- `subject_id` (FK auf das betroffene Hauptobjekt)
- `related_entity_refs[]` (JSON-Array — weitere verknüpfte Entitäten)
- `metadata` (JSON — Activity-spezifische Daten)
- `summary_text` (Klartext-Zusammenfassung in DE — Pflicht)

Activity-Typen (Enum)

- `call_inbound` / `call_outbound`
- `email_sent` / `email_received`
- `whatsapp_sent` / `whatsapp_received` (Phase 2)
- `chat_message` (Team-Chat in zugeordneten Channels)
- `note_added`
- `tag_added` / `tag_removed`

- `assignee_changed`
- `deal_stage_changed` / `project_stage_changed`
- `file_uploaded`
- `form_submitted`
- `task_created` / `task_completed`
- `appointment_scheduled` / `appointment_held`
- `lead_promoted` (im resultierenden Contact)
- `deal_won` / `deal_lost`
- `automation_triggered`
- `ki_action`
- `audit_event` (sichtbar nur mit `audit.read`)

Activity-Stream-View (Komponenten-Spec)

Zu zeigende Daten pro Activity:
- Datum + Uhrzeit (relativ wenn < 7 Tage, absolut sonst)
- Actor (User-Avatar + Name, oder „System" / „Automation X" / „KI")
- `summary_text` als Hauptinhalt
- Activity-Typ (visuelles Zeichen — Designer entscheidet)
- Klick öffnet Detail-Anzeige mit vollen `metadata`

Benötigte UI-Elemente:
- Chronologische Liste, gruppiert nach Tag-Bucket (heute / gestern / diese Woche / älter)
- Filter-Funktion nach Activity-Typ (Multi-Select)
- Filter-Funktion nach Actor
- Filter-Funktion nach Zeitraum
- Pagination/„Mehr laden"-Mechanik (Cursor-basiert)
- Suche im `summary_text` und `metadata`
- Detail-Anzeige für Single-Activity (zeigt vollen Kontext: Email-Body, Anruf-Notiz, Stage-Diff)

Operationen:
- Activities sind read-only — können nicht direkt erstellt oder bearbeitet werden
- Notizen sind die einzige Activity, die direkt erzeugbar ist (über „Notiz hinzufügen"-Funktion am Contact/Deal/Projekt)
- Activities können **nicht gelöscht** werden (Audit-Trail-Integrität)

Permissions-Wirkungen:
- User sieht nur Activities, deren `subject` er sehen darf (gleiche Permissions wie Subject-Modul)
- `audit_event`-Typen ausgeblendet ohne `audit.read`
- Activities mit sensitiven `metadata` zeigen nur `summary_text` ohne Detail-Drawer für User ohne `crm.read_full`

Klartext-Hinweise:
- Leerer Stream: *„Noch keine Aktivitäten — sobald Anrufe, Mails oder Updates passieren, erscheinen sie hier."*
- Permissions-Block: *„Es gibt Aktivitäten zu diesem Kontakt, die du nicht einsehen darfst."*

Performance:
- Index auf `tenant_id, subject_type, subject_id, created_at DESC`
- Top 50 Activities laden, Rest auf „Mehr laden"
- Bei Firma-Detail: Aggregat aller Activities der verknüpften Kontakte + firmen-eigene Activities
- Bei Dashboard-Overview: getrennte Aggregat-View mit Caching

---

Block F — Custom-Felder

Datenmodell

`CustomFieldDefinition`:
- `id`, `tenant_id`
- `applicable_to` (Enum: `contact | company | lead`)
- `field_key` (technischer Name, snake_case, unique pro applicable_to + tenant)
- `label` (User-sichtbarer Name)
- `description` (optional, Tooltip-Text)
- `field_type` (Enum: `text | number | date | select | multiselect | boolean | url | email`)
- `options[]` (nur für select/multiselect)
- `required` (Boolean)
- `default_value` (JSON — Phase 2)
- `position` (Sortierung)
- `is_template_origin` (kam aus Branchen-Template?)
- `archived_at`

`CustomFieldValue`:
- `id`, `tenant_id`, `field_definition_id`
- `entity_type`, `entity_id`
- `value` (JSONB — Typ-abhängig)
- `updated_at`, `updated_by_user_id`

Verwaltungs-Route (`/{tenant}/admin/custom-fields/`)

Zu zeigende Daten:
- Liste aller Custom-Felder, gruppiert nach `applicable_to`
- Pro Feld: Label, Typ, Anzahl Entities mit Wert, Quelle (Template/Custom), Status

Benötigte UI-Elemente:
- Sektion pro Entity-Typ
- Liste mit Drag-&-Drop für Sortierung
- Add-Funktion pro Sektion
- Edit-Funktion pro Feld
- Archivieren / Reaktivieren / Hard-Delete

Operationen:
- Custom-Feld definieren (Label, Typ, Optionen, Required)
- Umbenennen (`field_key` bleibt fix für Datenkonsistenz, nur `label` änderbar)
- Archivieren (Werte bleiben, Feld in UI ausgeblendet)
- Hard-Delete (mit Warnung über Anzahl betroffener Werte)
- Sortierung anpassen

Permissions: `tenant.admin` ODER `crm.admin`.

Anwendung in Detail-Views

Zu zeigende Daten:
- Sektion „Custom-Felder" zeigt alle aktiven Definitionen + Werte für die aktuelle Entity
- Felder in definierter Reihenfolge

Benötigte UI-Elemente:
- Pro Feld: Label + Eingabe-Komponente passend zum Typ (Text-Input, Number-Input, Date-Picker, Select, Multi-Select-Chips, Toggle, URL-Input, Email-Input)

- Pflichtfeld-Kennzeichnung
- Tooltip-Anzeige für `description`

Operationen:
- Wert eintragen / ändern / löschen
- Permission-gegated über `crm.write`

Anwendung in Listen-Views (Phase 1)

- Custom-Felder als **Filter** verwendbar (Add-Filter wählt aus aktiven Definitionen)
- Custom-Felder als **Spalten** sichtbar machbar (User-spezifische Spalten-Auswahl)

Phase 2

- Custom-Felder als Trigger-Bedingung in Automation-Engine
- Erweiterte Validierung (Regex, Zahlen-Range, abhängige Felder)
- Permissions pro Custom-Feld (z.B. *„nur HR sieht 'Personalnummer'"*)

SHK-Branchen-Template (beim Tenant-Anlegen geseedet)

Für Contact:
- *„Heizungsanlage Hersteller"* (text)
- *„Heizungsanlage Modell"* (text)
- *„Letzte Wartung"* (date)
- *„Wartungsvertrag"* (boolean)

Für Company:
- *„Anzahl Anlagen"* (number)
- *„Vertragsklasse"* (select: Standard / Premium / Custom)

Für Lead:
- *„Anlagen-Typ Interesse"* (select: Gas / Wärmepumpe / Solar / Mischanlage / Unklar)

(SHK-Defaults zur späteren Konkretisierung mit Noah)

Klartext-Hinweise

- Pflichtfeld leer beim Speichern: *„Das Feld ‚Heizungsanlage Typ' ist Pflicht."*
- Feld archiviert mit Wert vorhanden: *„Dieses Feld wurde archiviert — Wert bleibt gespeichert, ist aber nicht mehr editierbar. [Wieder aktivieren]"*
- Hard-Löschen mit Werten: *„15 Kontakte haben einen Wert in diesem Feld. Beim Löschen gehen Werte verloren. Stattdessen archivieren? [Archivieren] [Trotzdem löschen]"*

Block G — Suche & Duplikat-Erkennung

Such-Funktionen (drei Kontexte)

(1) Listen-View-Suche (CRM-spezifisch):
- Volltext über Name, Email, Telefon, Notizen, Custom-Felder
- Server-side mit Postgres-FTS
- Permission-aware

(2) Globale Suche (Modul 24):
- CRM-Modul registriert Contact/Company/Lead als suchbare Entitäten
- Hybrid (Volltext + Embeddings für semantische Treffer)

(3) Auto-Complete in Form-Feldern:
- Debounced (300ms), max. 10 Treffer
- Bei keinem Treffer: *„Keinen Kontakt gefunden — neuen anlegen?"* mit Direkt-Anlage

Performance:
- Index auf `tenant_id, last_name, first_name` und `tenant_id, primary_email`
- Trigram-Index (`pg_trgm`) für Fuzzy-Match
- Embeddings über pgvector (Modul 18)

Duplikat-Erkennung

Match-Kriterien (von hart zu weich):
1. **Hard:** Email exakt (case-insensitive)
2. **Hard:** Telefon exakt (normalisiert)
3. **Soft:** Vor- + Nachname identisch + Firma identisch
4. **Soft:** Vor- + Nachname identisch + Adresse PLZ identisch

Inline-Hinweis beim Anlegen:
- Live-Check beim Tippen von Email/Telefon (debounced)
- Hard-Match: Klartext-Hinweis *„Es gibt bereits einen Kontakt mit dieser Email: [Name]. [Diesen öffnen] [Trotzdem neu anlegen] [Daten zusammenführen]"*
- Soft-Match: dezenter Hinweis im Save-Bereich *„Möglicher Duplikat: [Name] [Anschauen]"* — User kann ignorieren

Bulk-Duplikat-Tool (`/{tenant}/admin/duplicates/`):

Zu zeigende Daten:
- Liste aller erkannten Duplikat-Paare
- Pro Paar: Match-Kriterium, beide Kontakte mit Stammdaten-Vorschau, Anzahl Activities/Verknüpfungen pro Seite

Benötigte UI-Elemente:
- Filter nach Match-Kriterium
- Side-by-Side-Anzeige
- Merge-Funktion mit Field-by-Field-Auswahl
- Skip-Funktion (Paar als „kein Duplikat" markieren)

Merge-Logik:
- Stammdaten: Field-by-Field-Auswahl
- Verknüpfungen (Deals, Projekte, Files, Activities): vom Verlierer auf Gewinner umgehängt
- Tags: vereinigt
- Custom-Felder: Field-by-Field
- Verlierer: hard-gelöscht oder archiviert (Tenant-Admin-Setting)
- Audit-Trail: Merge-Event mit Vorher-Nachher-Diff

Permissions: Bulk-Tool nur für `crm.admin`. Inline-Hinweis für jeden mit `crm.write`.

Phase 2

- Smart-Match mit KI
- Auto-Merge-Regeln (Tenant-Admin definiert: *„bei Hard-Email-Match automatisch mergen"*)
- Schedule-basiertes Duplikat-Scanning

Block H — Tags

Datenmodell

`Tag`:
- `id`, `tenant_id`, `created_at`
- `name` (unique pro Tenant + applicable_to)
- `color` (8 vordefinierte Farben für Konsistenz)
- `applicable_to[]` (mehrere Entity-Typen möglich)

`EntityTag`:
- `tag_id`, `entity_type`, `entity_id`, `tenant_id`, `applied_at`, `applied_by_user_id`

Ad-hoc-Erstellung

- User tippt im Tag-Editor einen neuen Tag → bei keinem Match Vorschlag *„+ Tag ‚Bestandskunde' anlegen"*
- Klick → Tag wird angelegt + sofort angewendet
- Permission `crm.write` ausreichend

Verwaltungs-Route (`/{tenant}/admin/tags/`)

Zu zeigende Daten:
- Liste aller Tags pro Entity-Typ
- Pro Tag: Name, Farbe, Anzahl Verknüpfungen, Anwendbar auf welche Typen, Erstellt-Datum

Benötigte UI-Elemente:
- Listen-Ansicht mit Such-/Filter-Funktion
- Edit-Funktion pro Tag
- Merge-Funktion (zwei Tags zusammenführen)
- Bulk-Delete

Operationen:
- Umbenennen (alle Verwendungen mit-aktualisiert)
- Farbe ändern
- Mergen (Verlierer-Verknüpfungen werden umgehängt)
- Hard-Löschen (mit Warnung)
- Auto-Cleanup-Toggle: *„Tags ohne Verwendung nach X Tagen archivieren"* (Default 90)

Anwendung

- **Listen-Views:** als Filter (Multi-Select mit AND/OR)
- **Detail-Views:** Tag-Editor pro Entity (Auto-Complete + ad-hoc anlegbar)
- **Automation-Engine** (Modul 17): Tags als Trigger-Bedingung und Action-Resultat
- Trigger: *„wenn Tag X gesetzt"*
- Action: *„Tag Y setzen / entfernen"*

Klartext-Hinweise

- Tag-Merge: *„‚VIP' (3 Verknüpfungen) wird in ‚Vip' (45 Verknüpfungen) zusammengeführt. ‚VIP' wird gelöscht."*
- Tag-Hard-Löschen: *„Dieser Tag ist 67 Mal verwendet. Beim Löschen werden alle Verknüpfungen entfernt — Entities bleiben."*

Block I — Permissions-Anwendung

Permission-IDs

Permission	Erlaubt
`crm.read_own`	Kontakte/Firmen/Leads sehen, bei denen User = `assigned_to_user_id`
`crm.read_all`	Alle Kontakte/Firmen/Leads sehen
`crm.write`	Anlegen + Bearbeiten (Permissions-aware: nur eigene wenn nur `read_own`)
`crm.delete`	Hard-Delete
`crm.admin`	Custom-Felder verwalten, Bulk-Duplikat-Tool, Tag-Verwaltung, DSGVO-Export, Bulk-Operations
`leads.read_own` / `read_all` / `write` / `admin`	Analog für Leads (Detail in Modul 2)

Bundle-Mapping (Variante 2)

Bundle	Atomare Permissions
`crm.none`	(keine)
`crm.read_own`	`crm.read_own`
`crm.read_all`	`crm.read_own` + `crm.read_all`
`crm.write`	`crm.read_all` + `crm.write`
`crm.admin`	`crm.read_all` + `crm.write` + `crm.delete` + `crm.admin`

Anwendung in Routes

- `/{tenant}/contacts` → `crm.read_own` oder `crm.read_all`
- `/{tenant}/contacts/new` → `crm.write`
- `/{tenant}/contacts/:id` → `crm.read_own` (mit Owner-Check) oder `crm.read_all`
- `/{tenant}/contacts/:id/edit` → `crm.write` (mit Owner-Check bei `read_own`)
- `/{tenant}/contacts/companies` → analog
- `/{tenant}/leads` → `leads.read_own` oder `leads.read_all`
- `/{tenant}/admin/custom-fields/` → `tenant.admin` oder `crm.admin`
- `/{tenant}/admin/tags/` → `tenant.admin` oder `crm.admin`
- `/{tenant}/admin/duplicates/` → `crm.admin`

Anwendung in Action-Buttons (Detail-View)

- Inline-Edit: `crm.write`
- Click-to-Call / Email schreiben: `crm.read_*` (Anzeige reicht)
- Hard-Delete: `crm.delete`
- DSGVO-Export: `crm.admin`
- Custom-Field-Wert ändern: `crm.write`

Anwendung in Listen-Filter

- Server-side-Filter wendet automatisch Permission-Filter an (AV #5: *Permissions-aware-Default*)

- User mit `crm.read_own`: nur Einträge mit `assigned_to_user_id = self`
- Bei Wechsel des Verantwortlichen während User aktiv ist: Realtime-Update entfernt den Eintrag aus der Liste

Klartext-Fehler

- Zugriff auf nicht-eigene Entity mit `crm.read_own`: 404 (nicht 403, um Existenz nicht zu offenbaren)
- Edit-Versuch ohne `crm.write`: Inline-Edit-Funktion einfach nicht sichtbar (Buttons werden gar nicht gerendert)
- Bulk-Action ohne Permission: Action-Funktion ausgegraut mit Tooltip *„Hierfür benötigst du erweiterte Berechtigungen"*

Architektur-Vorschriften (Querschnitt)

- **LP-3, AV #3:** SHK-Custom-Felder + Tag-Defaults sind Branchen-Template-Seeding, nicht Hardcode
- **LP-10, AV #1:** Klartext-Diagnostik überall (Beispiele in jedem Block)
- **AV #5:** Permissions-aware-Default in allen Listen/Suchen
- **AV #6:** Audit-Trail-Default für alle schreibenden Aktionen
- **LP-13:** EU-Stack — Embeddings über EU-Provider (Mistral/Scaleway)
- **LP-9 + Stil-Vorgabe:** UI-Element-Anforderungen als Liste, kein Layout

---

Detail-Spec-Status: DRAFT

Erste vollständige Detail-Spec für Modul 1. **Vor Bau:** Verifizierung mit Noah (zusammen mit Pflichtenheft-Sweep). Bei Implementierung dann konkrete Index-Definitionen, Migrations-SQL und SHK-Default-Werte ergänzen.

Offene Detail-Punkte (nicht-blockierend):
- Konkrete SHK-Custom-Field-Defaults mit Noah finalisieren
- Konkrete Index-Strategie pro Tabelle (Implementierung)
- Konkrete Migrations-Skripte für Branchen-Template-Seeding
- Subframe-UI-Designs (separat)