```
module: 24
title: Globale Suche
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-1, LP-9, LP-10, LP-13]
spec_style: requirements-pur
```

Modul 24 — Globale Suche

> ***Detail-Spec gemäß LP-9.*** *Cmd+K-Stil-Suche überall verfügbar (Header der App). Cross-Entity, hybrid (Postgres-FTS + pgvector-Embeddings über EU-Provider), permission-aware. Adapter-Pattern: jedes Modul registriert seine durchsuchbaren Entitäten.*

Themenblöcke

```
A. Datenmodell (SearchableEntity-Index, RecentSearch, Embedding-Cache)
B. Menü-Struktur und Routes
C. Hybrid-Suche (FTS + Embeddings)
D. Adapter-Pattern (Modul-Registrierung)
E. UI-Anforderungen (Modal/Cmd+K + Vollbild-Suche)
F. Permissions / Sichtbarkeit
G. Recent Searches
H. Performance & Indexierung
I. Klartext-Hinweise
```

Block A — Datenmodell

SearchIndexEntry (zentrale Index-Tabelle)

Tenant-scoped, RLS-aktiv. Wird von jedem Modul-Adapter befüllt; ermöglicht eine einzige Such-Query über alle Entitäten.

- `id`, `tenant_id`, `created_at`, `updated_at`
- `entity_type` (Enum: `contact | company | lead | deal | project | file | form_submission | user | task | wiki_page` — wiki Phase 2)
- `entity_id` (Polymorpher FK auf das jeweilige Modul-Entity)
- `title` (Hauptsächlicher Treffer-Titel; z.B. Kontakt-Name, Deal-Titel)
- `subtitle` (optional; z.B. Firma beim Kontakt, Deal-Stage)
- `body_text` (gesamter durchsuchbarer Volltext-Inhalt; konkateniert aus relevanten Feldern)
- `tsvector_de` (Generated Column aus `title + subtitle + body_text` mit deutscher Konfiguration)
- `embedding` (vector(1024) — pgvector; siehe Block C für Provider)
- `embedding_generated_at` (Timestamp; NULL bei pending)
- `embedding_status` (Enum: `pending | generating | ready | failed | skipped`)
- `permission_subject` (JSONB — modulares Permissions-Subject für Filter-Anwendung; z.B. `{"assigned_to_user_id": "uuid", "module": "crm"}`)
- `last_activity_at` (für Sortier-Boost im Default-Ranking)
- `is_archived` (Boolean — archivierte Einträge sind suchbar mit Toggle, default ausgeblendet)

Indizes:
- GIN-Index auf `tsvector_de` für FTS
- IVFFlat / HNSW-Index auf `embedding` für Cosine-Similarity
- `(tenant_id, entity_type, entity_id)` unique
- `(tenant_id, is_archived, last_activity_at DESC)` für Sortier-Default

Sonderfall: `SearchIndexEntry` ist Sekundär-Index — Source-of-Truth bleibt das jeweilige Modul-Entity. Updates über Trigger oder Outbox-Pattern (siehe Block H).

RecentSearch

Pro User, nicht tenant-scoped (User-Setting; trotzdem `tenant_id` für Trennung).

- `id`, `tenant_id`, `user_id`, `created_at`
- `query_text`
- `applied_filters` (JSONB — gewählte Entity-Typen, Zeitraum, etc.)
- `result_count` (für Display)
- `clicked_entity_type`, `clicked_entity_id` (optional — falls User auf Treffer geklickt hat; für Personalisierung Phase 2)

Index: `(tenant_id, user_id, created_at DESC)`.

Retention: letzte 50 pro User, ältere automatisch gelöscht.

EmbeddingProviderJob

Worker-Queue-State für asynchrone Embedding-Generation.

- `id`, `tenant_id`, `created_at`, `updated_at`
- `search_index_entry_id` (FK)
- `provider` (Enum: `mistral | scaleway | alephalpha`)
- `status` (`pending | running | completed | failed | retrying`)
- `attempts`
- `last_error_text` (Klartext)
- `completed_at`

Block B — Menü-Struktur und Routes

Sidebar-Hauptpunkt: keiner (Suche ist global verfügbar im Header).

Routen:

```
/{tenant}/search                      → Vollbild-Such-View (für komplexe Filter, Saved Searches Phase 2)
/{tenant}/search?q=...&types=...      → Direkt-Link mit Query und Filtern (URL-state)
```

Embedded-Komponente:

- Such-Modal-Komponente (Cmd+K / Ctrl+K) ist global verfügbar in jedem authentifizierten Layout
- Triggered durch Tastenkürzel (Cmd+K / Ctrl+K) oder Klick auf Such-Anker im Header
- Auf Mobile: Such-Anker im Header, öffnet Vollbild-Modal

Block C — Hybrid-Suche (FTS + Embeddings)

Such-Strategie

Bei Query `q`:

1. **FTS-Pfad** (Postgres `tsvector` + `tsquery` mit deutscher Konfiguration; Trigram-Index als Ergänzung für Fuzzy-Match auf Namen)
   - Liefert exakte und Wortform-Treffer (Stemming, Umlaut-Normalisierung)
   - Sehr schnell, deterministisch
2. **Semantischer Pfad** (pgvector Cosine-Similarity)
   - Query wird zu Embedding über EU-Provider (siehe unten)
   - Top-N nächste Einträge nach Cosine-Distance
   - Liefert semantisch ähnliche Treffer auch ohne Wort-Übereinstimmung
3. **Fusion** (Reciprocal Rank Fusion oder einfaches Score-Merging mit gewichteter Summe)
   - Gewichtung Default: 70% FTS / 30% Semantisch (Tenant-Admin-konfigurierbar Phase 2)
   - Zusätzliche Boosts: aktuelle `last_activity_at`, exakter Match in `title`, User-eigene Entitäten

Embedding-Provider (LP-13: EU-Stack)

- **Primär:** Mistral Embeddings (mistral-embed)
- **Sekundär:** Scaleway-Hosted Embeddings
- **Tertiär:** AlephAlpha
- Embedding-Dimension einheitlich 1024 (Mistral-embed); falls Provider-Wechsel nötig → Re-Embedding-Job für gesamten Index
- Provider-Fallback automatisch bei Down (siehe AV #13 / EC-6)

Embedding-Generation-Strategie

- Bei `INSERT` / `UPDATE` einer Quell-Entity → Outbox-Eintrag → Worker holt Eintrag, baut `body_text` über Adapter (Block D), ruft Embedding-Provider, persistiert in `SearchIndexEntry.embedding`
- Sehr kurze Texte (< 10 Zeichen): `embedding_status = skipped` — nur FTS
- Bei Provider-Failure: Retry mit Backoff bis 3 Versuche, dann `failed` mit `last_error_text`
- Initial-Backfill bei Tenant-Migration: Bulk-Job über alle bestehenden Entitäten

Query-Embedding-Caching

- Häufige Such-Queries werden als Embedding gecacht (24h TTL) → reduziert Provider-Kosten
- Cache-Key: SHA256(query_text + provider_version)

Block D — Adapter-Pattern (Modul-Registrierung)

Anforderungen (LP-11)

Jedes Modul, das Entitäten durchsuchbar machen will, registriert einen **SearchAdapter**:

- Liefert pro Entity-Typ:
- Funktion `buildSearchableContent(entity) → { title, subtitle, body_text }` (welche Felder fließen in den Index)
- Funktion `buildPermissionSubject(entity) → JSONB` (welche Daten brauchen wir später für Permissions-Filter)
- Funktion `getRouteForEntity(entity) → string` (Ziel-Route bei Treffer-Klick)
- Funktion `getDisplayMetadata(entity) → { icon_kind, badge_text, ... }` (für UI-Anzeige; Designer entscheidet finales Aussehen)

- Liste der `triggers` (welche DB-Events lösen Re-Indexierung aus)
- Definiert seinen `entity_type`-Wert (Enum-Eintrag)
- Definiert Permissions-Filter-Funktion: `filterVisibleForUser(searchEntries, userPermissions) → searchEntries[]`

Modul-Initial-Registrierungen (Phase 1)

Modul	Entity-Typen
Modul 1 (CRM)	`contact`, `company`, `lead`
Modul 3 (Sales-Pipeline)	`deal`
Modul 4 (Projekt-Pipeline)	`project`, `order`
Modul 5 (Form-Builder)	`form_submission`
Modul 14 (HR-Hub)	`user` (Mitarbeiter) — eingeschränkte Sichtbarkeit
Modul 23 (Files)	`file`
Modul 26 (Tasks)	`task`

Phase 2 Erweiterungen

- Modul 16 (Wiki) → `wiki_page`
- Modul 9 (Email-Inbox) → `email_message` (mit DSGVO-spezifischer Sichtbarkeit)
- Modul 13 (Team-Chat) → `chat_message` (sehr restriktive Permissions)

Re-Indexierungs-Trigger

- INSERT der Quell-Entity → SearchIndexEntry anlegen (Outbox)
- UPDATE relevanter Felder (laut Adapter) → SearchIndexEntry aktualisieren + Embedding-Re-Generation
- DELETE / Archivierung → SearchIndexEntry löschen oder `is_archived = true` setzen
- Permission-relevante Änderung (z.B. `assigned_to_user_id`) → `permission_subject` aktualisieren (kein Re-Embedding nötig)

Block E — UI-Anforderungen

Such-Modal (Cmd+K)

Zu zeigende Daten:

- Eingabefeld für Query-Text mit Live-Suche
- Trefferliste, gruppiert nach `entity_type`
- Pro Treffer: Titel, Subtitle, Entity-Typ-Indikator (Designer wählt visuelles Zeichen), Kontext-Snippet (Highlight der Match-Stelle aus `body_text`), letzte Aktivität (relativ)
- Bei leerem Query: Recent-Searches (siehe Block G) + Tipps („Tippe um zu suchen", „Cmd+K für Suche überall")
- Bei keinem Treffer: Klartext + Vorschläge (siehe Block I)
- Loading-Indikator während Suche

Benötigte UI-Elemente:

- Such-Eingabefeld mit Auto-Fokus beim Öffnen
- Trefferliste mit Tastatur-Navigation (Pfeiltasten + Enter zum Öffnen, Cmd+Enter für „in neuem Tab")
- Filter-Chips für Entity-Typ-Auswahl (z.B. „Nur Kontakte", „Nur Files") — Multi-Select

- Mehr-Treffer-Funktion („alle {n} Treffer zeigen" → leitet auf `/{tenant}/search` mit Query)
- Schließen-Funktion (Esc oder Klick außerhalb)
- Keyboard-Shortcut-Anzeige (z.B. „↑↓ navigieren • Enter öffnen • Esc schließen")

Operationen:

- Enter / Klick auf Treffer → öffnet die Ziel-Route (über Adapter `getRouteForEntity`)
- Filter-Chip-Klick → schränkt Treffer auf gewählte Typen ein
- „Alle Treffer anzeigen" → leitet auf `/search` mit Query und Filtern als URL-Param

Vollbild-Such-View (`/{tenant}/search`)

Zu zeigende Daten:

- Such-Eingabefeld (vorausgefüllt aus URL-Param)
- Vollständige Trefferliste mit Pagination (Cursor-basiert, 50 pro Page)
- Pro Treffer: alle Daten wie im Modal + erweiterte Metadaten (Erstellt-Datum, Verantwortlicher, Tags)
- Filter-Sidebar/Sektion mit:
- Entity-Typ (Multi-Select)
- Erstellt-Zeitraum (Date-Range)
- Verantwortlicher (Multi-Select)
- Tags (Multi-Select)
- Modul-spezifische Filter (Adapter-erweitert: z.B. Deal-Stage bei `deal`-Filter)
- Toggle „Archivierte einbeziehen"
- Trefferanzahl-Anzeige (Klartext: „1.247 Treffer in 0,3 Sek")
- Sortier-Optionen: Relevanz (Default) / Letzte Aktivität / Erstellt-Datum

Benötigte UI-Elemente:

- Such-Feld
- Filter-Komponenten (siehe Liste oben)
- Trefferliste mit Pagination
- Sortier-Auswahl
- Klick auf Treffer → Ziel-Route

Mobile-Anforderungen

- Such-Modal als Vollbild-Layer
- Tastatur öffnet automatisch
- Touch-Optimierte Trefferliste

Performance-Anforderungen

- Debounce: 300ms zwischen Tippen und Query (kein Server-Hit pro Tastendruck)
- Initial-Response binnen 500ms (FTS-Pfad)
- Semantischer Pfad darf länger laufen (Embedding-Provider-Latenz) — Trefferliste wird nachgeladen wenn Embeddings da
- „Loading more results"-Indikator während Embedding-Pfad noch läuft

---

Block F — Permissions / Sichtbarkeit

Anforderungen (AV #5: Permissions-aware-Default)

- Suche darf **niemals** Treffer anzeigen, die der User regulär nicht sehen darf

- Filter wird **server-side** angewendet, bevor Treffer an den Client gehen (kein Client-side-Hidings — User kann sonst Existenz aus Trefferanzahl ableiten)
- `permission_subject`-JSONB pro `SearchIndexEntry` enthält die Daten, die der modulspezifische Filter braucht (z.B. `assigned_to_user_id`, `project_team_member_ids`)
- Pro Adapter: `filterVisibleForUser(entries, userPermissions)` reduziert Treffer auf erlaubte

Beispiele

- User mit `crm.read_own`: nur Contacts/Deals/Leads mit `assigned_to_user_id = self`
- User mit `projects.read_own`: nur Projekte, in deren Team er Mitglied ist
- User ohne `files.read_*`: keine Files in den Treffern
- User ohne `hr.read_all`: keine `user`-Treffer (bzw. nur eigener User-Eintrag)

Trefferanzahl-Konsistenz

- Trefferanzahl bezieht sich **immer** auf erlaubte Treffer (nicht auf Index-Gesamtmenge)
- Klartext bei massiv reduzierter Anzeige: *„Es gibt weitere Treffer, die du aufgrund deiner Berechtigungen nicht siehst."* (nur wenn signifikant — sonst Verwirrung; konkrete Anzahl bewusst nicht genannt)

Audit-Log

- Suchanfragen mit Inhalt und Trefferanzahl werden im User-Activity-Trail (Modul 21) geloggt
- Bei sensiblen Themen (HR-Suche, Lohnabrechnungen) optional erweiterte Audit-Granularität — Phase 2

Block G — Recent Searches

Anforderungen

- Pro User die letzten 50 Suchanfragen gespeichert (siehe Datenmodell `RecentSearch`)
- Anzeige im Modal bei leerem Query
- Klick auf Recent-Search-Eintrag → setzt Query und Filter neu, führt Suche aus

Operationen

- Einzelne Recent-Search löschen
- Alle Recent-Searches löschen (User-Setting in `/settings/`)
- Auto-Cleanup: ältere als 50 oder älter als 90 Tage werden automatisch entfernt

Privacy

- Recent-Searches sind privat pro User — kein Tenant-Admin-Zugriff
- Bei DSGVO-Export (Modul 35) eines Users werden die eigenen Recent-Searches mit ausgegeben
- Bei Account-Löschung werden Recent-Searches hart gelöscht

Block H — Performance & Indexierung

Indexierungs-Architektur

- **Outbox-Pattern:** schreibende Operationen in Modul-Tabellen schreiben Events in `search_outbox` (Tenant-scoped); Worker konsumiert Events und aktualisiert `SearchIndexEntry`

- **Trigger-Alternative für Phase 1:** Postgres-Trigger auf relevanten Tabellen, die direkt synchron `SearchIndexEntry` updaten (FTS-Felder); Embedding-Generation läuft asynchron
- Entscheidung: Outbox bevorzugt, weil weniger Trigger-Last und besser für hohe Schreibraten

Index-Größen-Schätzung

- Geerkens-Größe: ~5.000 Kontakte + ~3.000 Deals + ~2.000 Projekte + ~10.000 Files + ~50 User = ~20.000 Einträge → unproblematisch
- 1.024-dim Embeddings × 20.000 = ~80 MB pro Tenant
- HNSW-Index zusätzlich ~30%

Query-Performance-Ziele

- FTS-Pfad: < 100ms p95
- Embedding-Pfad: < 300ms p95 (bei warmem Cache); 1-3s bei Cold-Cache (erstes Embedding der Query)
- Modal-Initial-Response: < 500ms (FTS-only first paint, Embeddings nachladen)

Indexierungs-Latenz

- Update einer Entity → erscheint in Suche binnen 5-10 Sekunden (Outbox-Worker-Tick)
- Klartext-Hinweis falls User direkt nach Anlegen sucht und nichts findet: *„Neue Einträge erscheinen kurz nach Anlegen in der Suche."*

Skalierung

- pgvector mit IVFFlat-Index reicht für Phase-1-Größen
- Bei Wachstum (10M+ Einträge global): Migration zu HNSW oder Sharding pro Tenant erwägen — Phase 2/3
- Tenant-Pooling auf DB-Ebene; keine Cross-Tenant-Queries möglich

---

Block I — Klartext-Hinweise / Fehler-Cases (LP-10)

- **Keine Treffer:** *„Keine Treffer für ‚{query}'. Versuch andere Begriffe oder erweitere den Zeitraum-Filter."*
- **Permission-bedingte Reduktion:** *„Es gibt weitere Treffer, die du aufgrund deiner Berechtigungen nicht siehst."* (nur bei signifikantem Unterschied einblenden)
- **Embedding-Provider-Down:** *„Erweiterte semantische Suche ist gerade nicht verfügbar — Treffer kommen nur aus der Volltext-Suche."* (Banner im Modal; Treffer trotzdem sinnvoll, nur weniger semantisch)
- **Zu kurze Eingabe (<2 Zeichen):** *„Tippe mindestens 2 Zeichen für die Suche."*
- **Indexierungs-Verzögerung:** *„Neue Einträge erscheinen ein paar Sekunden nach dem Anlegen in der Suche."* (nur bei Verdacht, dass User darauf wartet — z.B. wenn Query exakt einem gerade angelegten Namen entspricht und keine Treffer da sind)
- **Re-Indexierungs-Lauf (Admin-Trigger):** *„Such-Index wird neu aufgebaut. Suche kann temporär unvollständig sein. Status: {progress}%"* (in `/admin/`-Bereich, Phase 2)
- **Server-Fehler:** *„Suche ist gerade nicht erreichbar. Bitte in einer Minute nochmal versuchen."*

Permissions-Anwendung

Permission-IDs

Permission	Erlaubt
`search.use`	Globale Suche nutzen (Default für alle Tenant-User)
`search.admin`	Re-Indexierungs-Lauf triggern, Index-Statistiken sehen

Hinweis: Die Suche selbst hat sehr leichtgewichtige Permissions, weil die eigentliche Sichtbarkeit über die **Modul-Permissions** der jeweiligen Entitäten bestimmt wird (siehe Block F).

Bundle-Mapping

Bundle	Atomare Permissions
`search.none`	(keine — Suche im Header ausgeblendet)
`search.use`	`search.use`
`search.admin`	`search.use` + `search.admin`

Anwendung in Routes

- `/{tenant}/search` → `search.use`
- Such-Modal global → `search.use`
- `/{tenant}/admin/search/` (Phase 2 Index-Verwaltung) → `search.admin` oder `tenant.admin`

Phase 2 / spätere Features

- **Saved Searches:** User speichert Query + Filter unter Namen, ruft sie aus Dropdown auf
- **Erweiterte Operatoren:** `contact:max@firma.de status:active type:deal` (Lucene-light-Syntax)
- **Search-Analytics:** Tenant-Admin sieht häufige Queries, Null-Treffer-Queries (Hinweis auf fehlende Inhalte), Klick-Through-Rate
- **Personalisiertes Ranking:** Treffer-Boost basierend auf User-Klick-Historie (Recent-Searches mit `clicked_entity_*`-Daten)
- **Cross-Tenant-Suche** (für Plattform-Admin in `admin.hivegrid.de`) — sehr restriktiv
- **Wiki-/Knowledge-Base-Inhalte** indexieren (sobald Modul 16 in Phase 2)
- **Email-Inhalte / Chat-Nachrichten** indexieren (mit DSGVO-spezifischer Permissions-Schicht)
- **OCR-Texte aus Files** (sobald Modul 23 OCR hat) ins `body_text` einfließen lassen
- **Tenant-Admin-konfigurierbare Gewichtung** zwischen FTS und Semantik
- **Multi-Sprache:** weitere `tsvector`-Konfigurationen je nach `tenant.language` und Phase-2-i18n-Roadmap (LP-12)

Architektur-Vorschriften (Querschnitt)

- **LP-9, AV #1:** Klartext-Diagnostik (siehe Block I)
- **LP-11, AV #2:** Adapter-Pattern für Modul-Registrierung (siehe Block D)
- **LP-13, AV #12:** EU-Stack — Embeddings ausschließlich über Mistral / Scaleway / AlephAlpha

- **AV #5:** Permissions-aware-Default — server-side-Filter, niemals Treffer durchsickern lassen
- **AV #6:** Audit-Trail-Default — Suchanfragen werden geloggt
- **AV #13:** Multi-Provider-Fallback für Embeddings
- **LP-1:** Tenant-Isolation — Index pro Tenant via `tenant_id` und RLS
- **LP-9:** Requirements-pur Spec ohne UI-Layout

Abhängige Module

- **Modul 1, 3, 4, 5, 14, 23, 26:** liefern Adapter und werden durchsucht
- **Modul 17 (Automation Engine):** stellt Worker-Queue für asynchrone Embedding-Generation bereit
- **Modul 18 (KI-Layer):** stellt LLM-Gateway für Embedding-Provider-Calls bereit (LP-13, EU-Provider, Multi-Provider-Fallback)
- **Modul 21 (Audit-Log):** Such-Anfragen werden geloggt
- **Modul 27 (Auth/Permissions):** liefert User-Permissions, die in `filterVisibleForUser` einfließen
- **Modul 35 (DSGVO):** Recent-Searches im DSGVO-Export

Detail-Spec-Status: DRAFT

Erste vollständige Detail-Spec für Modul 24. **Vor Bau:** Verifizierung mit Noah und Abgleich mit Modul 18 (LLM-Gateway-Spec) und Modul 17 (Worker-Queue-Spec).

Offene Detail-Punkte (nicht-blockierend):

- Embedding-Dimension Standardisierung (1024 vs. 1536 vs. 768) — Provider-übergreifende Strategie
- HNSW vs. IVFFlat — initial IVFFlat, Migration zu HNSW bei Skalierungs-Bedarf
- Permission-Filter-Performance bei großen Tenants (10.000+ Treffer pro Query) — ggf. Materialized View pro User
- Konkrete Outbox-Implementation (separate Tabelle vs. logisches Replication Slot)
- Klartext-Schwellwert für „Es gibt weitere Treffer, die du nicht siehst"-Hinweis (Datenleak-Risiko vs. Klartext-Pflicht)
- Recent-Searches: User-Opt-Out-Möglichkeit (Privacy-Setting) — Default an oder aus