```yaml
module: 38
title: Help-Center (öffentlich)
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-9, LP-10, LP-12, LP-13]
spec_style: requirements-pur
```

Modul 38 — Help-Center (öffentlich)

> ***Detail-Spec gemäß LP-9.*** *HiveGrids eigene öffentliche Hilfeseite für Tenant-User (Erste Schritte, Setup-Anleitungen, FAQ).* ***Nicht*** *zu verwechseln mit M16 Knowledge-Base (= tenant-internes Wiki, Phase 2). Phase 1 minimal: Markdown-Files im Repo, ~10-20 Artikel, nur Deutsch.*

Themenblöcke

```
A. Datenmodell (file-basiert, nicht DB)
B. Routen + URL-Struktur
C. Listen-View / Kategorien-Übersicht
D. Detail-View / Artikel-Seite
E. Suche
F. Verlinkung aus der App (Deep-Links)
G. Permissions
H. Phase 2
```

Block A — Datenmodell

A.1 — File-basierter Ansatz (Phase 1)

Help-Center-Inhalte werden **nicht in der DB** gepflegt, sondern als Markdown-Files im Repository abgelegt. Pflege erfolgt über Git-PRs durch das Plattform-Team. Keine Editor-UI in Phase 1.

Verzeichnis-Struktur (Vorgabe):

```
content/help/
  de/
    erste-schritte/
      _category.json          → Kategorie-Metadaten
      willkommen.md
      tenant-einrichten.md
    vertrieb/
      _category.json
      crm-kontakt-anlegen.md
      ...
    ...
```

A.2 — Markdown-File-Format

Jede `.md`-Datei beginnt mit YAML-Frontmatter:

```
---
slug: kwp-setup
title: KWP-Integration einrichten
category: integrationen
order: 10
description: Schritt-für-Schritt-Anleitung zur Verbindung von HiveGrid mit deiner KWP-Anlage.
last_updated: 2026-05-08
related_slugs: [kwp-troubleshooting, sandra-rechnungs-flow]
keywords: [kwp, erp, integration, anlage, setup]
---
```

Pflicht-Frontmatter-Felder: `slug`, `title`, `category`, `order`, `description`, `last_updated`.
Optional: `related_slugs`, `keywords`, `deprecated` (Boolean), `redirect_to`.

Body: Standard-Markdown. Erlaubt: Headings (h2-h4), Bold/Italic, Listen, Code-Blocks, Tabellen, Bilder, Links, Callout-Blocks (`> [!info]` / `> [!warning]`).

A.3 — Kategorie-Metadaten (`_category.json`)

```
{
  "slug": "integrationen",
  "label": "Integrationen",
  "description": "Anbindung externer Systeme (KWP, Google, AirCall, ...)",
  "icon": "plug",
  "order": 50
}
```

A.4 — Build-Time-Index

Build-Schritt erzeugt:
- Volltext-Index (Lunr.js / FlexSearch oder Postgres-FTS in einer Helper-Tabelle, falls Server-Side-Suche bevorzugt)
- Kategorien-Manifest (alle Kategorien + Artikel pro Kategorie, sortiert)
- Sitemap (`/help/sitemap.xml`)

Index wird im Build geprüft (jede Kategorie-Referenz aus Frontmatter muss existieren, jeder `related_slug` muss existieren, sonst Build-Fail).

A.5 — Phase-1-Artikel-Inventar (initial ~10-20)

Erste Schritte: Willkommen, Tenant einrichten, Erste Mitarbeiter einladen, Magic-Link-Login-FAQ
Vertrieb: Kontakte und Firmen, Lead-Inbox-Workflow, Sales-Pipeline-Grundlagen
Aufträge: Projekt anlegen, Plantafel nutzen, Acceptance-Protokoll-Anleitung
Mitarbeiter: HR-Hub-Übersicht, Krankmeldung/Urlaubsantrag stellen
Integrationen: KWP-Setup, Google-Kalender verbinden, AirCall-Setup, Email-Integration (Gmail/Outlook)
Mobile: Mobile-App-Setup, Offline-Modus auf Baustelle
Abrechnung: Stripe-Setup, Plan wechseln, Rechnungen einsehen
Tutorials: Form-Builder-Tutorial, Automation-Builder-Tutorial

(Konkrete Liste wird mit Plattform-Team beim Bau finalisiert.)

Block B — Routen

Sidebar-Hauptpunkt: *(keiner — Help-Center ist außerhalb des Tenant-Sidebar-Kontexts erreichbar)*.

Routen:

```
app.hivegrid.de/help/                              → Redirect auf /help/de/
app.hivegrid.de/help/{lang}/                       → Help-Home (Kategorien-Übersicht + Suche)
app.hivegrid.de/help/{lang}/{category}/            → Kategorie-Übersicht (Liste der Artikel)
app.hivegrid.de/help/{lang}/{category}/{slug}      → Artikel-Detail
app.hivegrid.de/help/{lang}/search?q=...           → Such-Ergebnisse
app.hivegrid.de/help/sitemap.xml                   → SEO-Sitemap
```

Phase 1 nur `lang = de`**.** Andere Sprachen würfeln 404 mit Klartext-Hinweis.

Öffentlich: keine Auth nötig, Routen sind public, kein Tenant-Context.

SEO-Defaults:
- Statisches Rendering (SSG oder SSR mit langem Cache-Header)
- Pro Artikel: `<title>` aus Frontmatter, `<meta name="description">` aus Frontmatter
- Open-Graph-Tags
- Canonical-URL pro Artikel

Block C — Kategorien-Übersicht (Help-Home + Kategorie-Seite)

Help-Home (`/help/de/`)

Zu zeigende Daten:
- Hero-Bereich mit Such-Funktion
- Kategorien-Liste mit Label, Description, Icon, Anzahl Artikel
- Optional: „Beliebteste Artikel"-Block (Phase 1: manuell-kuratierte Liste, Phase 2: View-Counts)

Benötigte UI-Elemente:
- Such-Eingabefeld (zentral platziert)
- Kategorie-Karten/-Liste (klickbar)
- Footer mit Link zu „Support kontaktieren" → triggert In-App-Feedback-Widget oder mailto: `support@hivegrid.de`

Kategorie-Seite (`/help/de/{category}/`)

Zu zeigende Daten:
- Kategorie-Label + Description
- Liste aller Artikel in dieser Kategorie, sortiert nach `order`
- Pro Artikel: Title, Description, ggf. `last_updated` -Datum

Benötigte UI-Elemente:
- Such-Eingabefeld
- Artikel-Liste (klickbar)
- Breadcrumb (Help / Kategorie)

Block D — Artikel-Detail (`/help/de/{category}/{slug}`)

Zu zeigende Daten:
- Title, Description, `last_updated` -Datum
- Markdown-Body (gerendert als HTML)
- Verwandte Artikel (`related_slugs`) als Liste
- „War dieser Artikel hilfreich?"-Feedback (optional, sehr einfach: Daumen hoch/runter, anonym tracked als Klicks ohne PII)

Benötigte UI-Elemente:
- Markdown-Renderer mit Support für: Headings, Listen, Bold/Italic, Code-Blocks (mit Syntax-Highlighting für Bash/JSON/YAML), Tabellen, Bilder (responsive), Links (External-Link-Indicator), Callout-Blocks (Info/Warning/Tip)
- Inhaltsverzeichnis (TOC) aus h2-Headings, klickbar/scrollbar
- Breadcrumb (Help / Kategorie / Artikel)
- Such-Eingabefeld (im Header)
- „Verwandte Artikel"-Sektion am Ende
- Optional: Edit-on-GitHub-Link (führt zur Markdown-Quelle für Plattform-Team)

Operationen:
- Lesen (read-only — kein Login, keine Edit-Funktion in Phase 1)
- Drucken (Browser-nativ)
- Link kopieren

Klartext-Hinweise:
- Artikel mit `deprecated: true`: Banner *„Dieser Artikel ist veraltet — siehe stattdessen [neuer Link]"*
- 404 (Slug existiert nicht): Klartext-Page *„Diesen Artikel gibt's nicht mehr. Vielleicht hilft die Suche?"* + Such-Eingabefeld

---

Block E — Suche

Such-Funktion

Verhalten:
- Eingabe → Such-Ergebnis-Page mit Treffern, ranked nach Relevanz
- Ranking: Title-Match > Description-Match > Keyword-Match > Body-Match
- Highlight des Such-Treffers im Snippet
- Empty-State bei keinen Treffern: *„Nichts gefunden zu '{query}'. Schreib uns: support@hivegrid.de"*

Benötigte UI-Elemente:
- Such-Eingabefeld (auf allen Help-Routen verfügbar)
- Ergebnis-Liste mit: Title, Kategorie-Label, Snippet (mit Highlight), Link
- Pagination (selten benötigt bei Phase-1-Volumen, aber sauber vorbereitet)

Performance:
- Index wird beim Build erzeugt
- Phase 1: Client-side-Suche via FlexSearch (~50KB Index für 20 Artikel) ODER Server-side über Postgres-FTS-View (nice-to-have)
- Debounce auf Eingabe (300ms)

Block F — Verlinkung aus der App (Deep-Links)

Hilfe-Icon-Pattern

Jede App-Route kann ein optionales Hilfe-Icon zeigen (Plazierung Designer-Entscheidung). Klick öffnet:
- Deep-Link in neuem Tab: `app.hivegrid.de/help/de/{category}/{slug}`
- Deep-Link wird pro Route in einer Mapping-Konfiguration gepflegt

Konfigurations-Beispiel (Code-internal):

```
{
  "/{tenant}/automation/builder": "automatisierungen/automation-builder-tutorial",
  "/{tenant}/integrations/kwp": "integrationen/kwp-setup",
  "/{tenant}/forms/builder": "tutorials/form-builder-tutorial",
  ...
}
```

Klartext-Fallback: Falls für eine Route kein Mapping existiert, führt das Hilfe-Icon zu `/help/de/` (Help-Home).

Block G — Permissions-Anwendung

Keine Permissions — Help-Center ist öffentlich erreichbar ohne Login.

Konsequenz für Routing:
- `/help/*` ist *vor* dem Auth-Middleware-Layer gemounted
- Kein Tenant-Context, kein RLS-Setting
- Standard-Rate-Limiting per IP (Schutz gegen Crawler-Missbrauch, z.B. 1.000 Req/Min/IP)

Block H — Klartext-Hinweise

- Sprache nicht verfügbar (`/help/en/...` in Phase 1): Redirect auf `/help/de/` mit Klartext-Banner *„Englischsprachige Hilfe ist in Vorbereitung. Hier ist die deutsche Version."*
- Such-Empty-State: *„Nichts gefunden zu '{query}'. Probier andere Begriffe — oder schreib uns direkt an support@hivegrid.de."*
- Artikel-404: *„Diese Hilfe-Seite gibt's nicht mehr. [Zur Übersicht] oder [Suche starten]."*

Block I — Phase 2 / spätere Features

- **DB-getriebene Pflege durch Plattform-Admin** (Editor-UI in `admin.hivegrid.de/help/`)
- **Mehrsprachigkeit:** EN/TR/PL via DeepL-Pipeline (LP-12). Aus DE-Quelltext werden Übersetzungen generiert, manuell verifiziert, dann publiziert. Frontmatter-Feld `translations: [en, tr, pl]`.
- **KI-gestützte Artikel-Suche (RAG):** Help-Center-Inhalte als Embeddings-Korpus für KI-Assistent in der App; User kann „Wie geht X?" fragen und bekommt Antwort mit Artikel-Zitat
- **View-Counts + Helpful-Votes-Statistik** für Plattform-Team (welche Artikel werden gelesen, welche helfen)
- **Versionierung:** alte Artikel-Versionen sichtbar bei Deprecation
- **Kontextuelle Inline-Hilfe:** statt Deep-Link in neuem Tab → Side-Panel mit Artikel direkt in der App
- **Video-Tutorials einbettbar** (YouTube/Loom-Embeds)
- **„Was ist neu?"-Changelog-Feed** auf Help-Home

Block J — Architektur-Vorschriften (Querschnitt)

- **LP-9:** Spec-Tiefe — Mapping-Tabelle Route→Slug ist explizit definiert, nicht „mal so anlegen"
- **LP-10:** Klartext-Diagnostik — auch im Help-Center selbst (404-/Empty-State-Texte)
- **LP-12:** i18n-sauber — alle UI-Strings aus Locale-Files; Phase 1 nur DE-Locale, Library bereits mehrsprachig vorbereitet
- **LP-13:** EU-Stack — Help-Content auf Scaleway-Hosting, kein US-CDN für Pflege-Content

Detail-Spec-Status: DRAFT

Phase-1-Modul bewusst minimal. **Vor Bau:**
- Konkretes Verzeichnis-Layout im Code-Setup (`content/help/de/...`) festlegen
- Initial-Artikel-Inventar mit Plattform-Team finalisieren (welche 15-20 Artikel zum Launch nötig)
- Markdown-Renderer-Library wählen (z.B. `unified` + `remark-gfm` + `rehype-shiki`)
- Such-Strategie entscheiden: client-side (FlexSearch) vs. server-side (Postgres-FTS)
- Mapping-Konfiguration Route→Slug als File im Repo oder als Konstante im Code

Offene Detail-Punkte (nicht-blockierend):
- Konkretes Auswahl der Initial-Artikel
- Routing-Setup für Public-Route außerhalb Auth-Middleware
- SEO-Optimierung-Details (sitemap.xml, robots.txt)
- Optionales Helpful-Voting-Tracking (anonym ohne Cookies)