```
type: schema-vorlage
purpose: Einheitliche Struktur für alle Modul-Detail-Specs in HiveGrid 2.0
spec_style: requirements-pur (KEIN UI-Layout, KEIN ASCII-Mockup, KEINE Anordnungs-Beschreibung)
last_updated: 2026-05-08
```

Schema-Vorlage für Detail-Specs

> ***Stilregel (LP-9 + Memory*** `feedback_spec_style_no_ui` ***):*** *Specs beschreiben was möglich sein muss, nicht wie es visuell aussieht. UI-Elemente werden als* ***Anforderungs-Listen*** *dokumentiert ("benötigt: Tab-Navigation für X/Y/Z", "benötigt: Auto-Complete-Funktion") — niemals als Layout-Skizze. Designer hat freie Hand für Anordnung/Optik/Farben/Spacing.*

Frontmatter (Pflicht)

```
---
module: <Nummer>
title: <Modul-Name>
phase: 1 | 2
status: detail-spec-draft | reviewed | approved | implemented
last_updated: YYYY-MM-DD
session_source: <Quell-Datei>
related_lps: [LP-X, LP-Y, ...]
spec_style: requirements-pur
---
```

Pflicht-Sections (in dieser Reihenfolge)

1. Header

- `# Modul X – Titel`
- 1-2-Satz-Kurzbeschreibung (was tut das Modul, welcher LP-Bezug, was wird abgelöst)
- Themenblöcke-Übersicht als Code-Block

2. Datenmodell

Pro Entität:
- Tabellenname + Beziehungs-Kontext (M:N? Soft-Delete? RLS?)
- Felder als Liste mit Typ, Pflicht/Optional, Anmerkungen
- Indizes (für Performance)
- Sonderfälle/Constraints

3. Menü-Struktur und Routes

- Sidebar-Hauptpunkt
- Liste aller Routen mit Zweck
- Permission-Anforderung pro Route (kommt in Section 8 nochmal konsolidiert, hier als Hinweis)

4. Listen-Views (falls relevant)

- Zu zeigende Daten pro Eintrag

- Benötigte UI-Elemente (Filter, Sortierung, Bulk-Actions, Pagination — als Anforderungs-Liste)
- Operationen
- Performance-Anforderungen (Indizes, Pagination-Strategie)

5. Detail-Views (falls relevant)

- Zu zeigende Daten
- Benötigte UI-Elemente (als Anforderungs-Liste, **kein** Layout)
- Operationen (was kann der User tun)
- Sub-Sections falls relevant (z.B. Tabs für verschiedene Aspekte)

6. Spezifische Workflows / Sub-Funktionen

- Komplexe Flows (z.B. „Lead-Promotion", „Bulk-Reassign", „2FA-Setup")
- Pro Workflow: Schritte, Edge-Cases, Klartext-Hinweise

7. Activity-Stream-Integration (falls Modul Activities erzeugt)

- Welche Activity-Typen produziert das Modul
- Verknüpfungen mit anderen Modulen

8. Permissions-Anwendung

- Permission-IDs als Tabelle
- Bundle-Mapping (Variante 2)
- Anwendung in Routes (Permission-Mapping)
- Anwendung in Action-Buttons / Operationen
- Owner-Check-Logik (für `read_own`-Permissions)

9. Klartext-Hinweise/-Fehler

- Liste der wichtigsten User-facing Klartext-Texte
- Fehler-Cases mit Lösungs-Vorschlag (LP-10)

10. Phase 2 / spätere Features

- Was bewusst NICHT in Phase 1 ist
- Begründung wenn nicht offensichtlich

11. Architektur-Vorschriften (Querschnitt)

- Welche der 20 globalen AVs (siehe Brainstorming-Session) für dieses Modul besonders relevant sind

12. Detail-Spec-Status

- DRAFT / REVIEWED / APPROVED
- Offene Detail-Punkte (nicht-blockierend)
- Verweise auf abhängige Module

Was NICHT in Specs gehört

❌ ASCII-Layouts wie:

```
┌─────┐
| Box |
└─────┘
```

❌ Anordnungs-Beschreibungen wie *„Sidebar links, Stream rechts"*

❌ Konkrete Pixel-/CSS-Werte oder Farben

❌ Konkrete Button-Reihenfolgen oder -Gruppierungen

❌ Wireframes oder Mockup-Beschreibungen

✅ Stattdessen: *„Folgende UI-Elemente werden benötigt: [Liste]"*

Beispiel — falsch vs. richtig

Falsch (UI-Layout):

> *Header oben mit Avatar links, Name daneben. Drei Action-Buttons (Anrufen, Email, Termin) rechts oben. Tab-Navigation darunter. Activity-Stream in der rechten Spalte sticky.*

Richtig (Requirements):

> *Folgende UI-Elemente werden benötigt:*
> *- Anzeige von Avatar + Name + Verantwortlicher + Tags + primäre Email/Phone*
> *- Action-Funktionen für: Anrufen (Click-to-Call), Email schreiben, Termin buchen, Deal anlegen, Aufgabe anlegen, Bearbeiten, Archivieren, Löschen*
> *- Tab- oder Sektions-Navigation für: Übersicht, Deals, Projekte, Files, Custom-Felder*
> *- Activity-Stream-Komponente (Spec siehe Modul X)*