```
module: 13
title: Team-Chat (intern)
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-2, LP-3, LP-4, LP-7, LP-9, LP-10, LP-13]
spec_style: requirements-pur
```

Modul 13 — Team-Chat (intern)

> ***Detail-Spec gemäß LP-9.*** *Interner Team-Chat als Quiply-Komplett-Ablösung. WebSocket-basierte Echtzeit-Kommunikation über drei Ebenen (Projekt-Channels, Gruppen-Channels, Direkt-Nachrichten) mit tiefer Verzahnung in CRM/Projekte/Tasks/Files. Kein Customer-Chat — Endkunden laufen über Customer-Portal (M15) bzw. WhatsApp (M11).*

Themenblöcke

```
A. Datenmodell (TeamChat, Member, Message, Pinned, Reaction, Thread, Read)
B. Menü-Struktur und Routes
C. Channel-Typen und Lifecycle (Projekt-Auto / Gruppen / DM)
D. Listen-View (Channel-Index)
E. Detail-View (Chat-Verlauf)
F. Threads, Reactions, Pinned, Mentions
G. Datei-Uploads, Link-Previews, Mark-as-Task
H. WebSocket-Server / Echtzeit-Übermittlung
I. Read-Receipts und Notifications
J. Mobile-First für Monteur (LP-2)
K. Activity-Stream-Integration
L. Permissions-Anwendung
M. Klartext-Hinweise/-Fehler
N. Phase 2 / spätere Features
```

Block A — Datenmodell

TeamChat

Repräsentiert einen Channel (Projekt-Chat, Gruppen-Chat oder DM).

- `id`, `tenant_id`, `created_at`, `updated_at`, `created_by_user_id`
- `kind` (Enum: `project | group | dm`)
- `name` (Pflicht für `group`, optional für `project` — Default = Project-Title, NULL bei `dm` — wird aus Members generiert)
- `description` (Markdown-Freitext, optional)
- `project_id` (FK auf Project — NUR bei `kind=project`, sonst NULL)
- `is_private` (Boolean — `group` kann öffentlich/sichtbar im Channel-Browser oder privat/nur per Einladung sein; `project` / `dm` immer privat)
- `archived_at` (Soft-Delete; bei Project-Abschluss automatisch gesetzt)
- `archived_reason` (Enum: `manual | project_completed | project_cancelled`, NULL solange aktiv)

- `last_message_at` (für Default-Sortierung im Channel-Index)
- `notification_default` (Enum: `all_messages | mentions_only | muted` — Default für neue Members)

Constraints:
- `kind=project` → `project_id NOT NULL`, `name` darf leer bleiben (UI fällt auf Project-Title zurück)
- `kind=dm` → genau 2 TeamChatMember-Einträge, `name` und `description` werden ignoriert
- `kind=group` → mind. 1 Member, `name` Pflicht
- Pro Project max. 1 aktiver Projekt-Chat (Unique-Constraint auf `project_id` WHERE `archived_at IS NULL`)
- Pro Tenant + Member-Paar max. 1 aktiver DM-Chat (Auto-Reuse statt Duplikat-Anlage)

Indizes:
- `tenant_id, kind, archived_at, last_message_at DESC` für Index-Sortierung
- `project_id` für Project-Verknüpfung
- Partial Unique Index auf `(project_id) WHERE archived_at IS NULL AND kind='project'`

TeamChatMember (M:N)

Mitgliedschaft eines Users in einem Channel.

- `id`, `tenant_id`, `chat_id`, `user_id`, `joined_at`, `joined_via_user_id` (wer hat eingeladen, NULL bei Auto-Add)
- `role_in_chat` (Enum: `admin | member` — `admin` darf Members ändern, Channel umbenennen, archivieren)
- `notification_pref` (Enum: `all_messages | mentions_only | muted` — Override des Channel-Defaults)
- `last_read_message_id` (FK auf TeamChatMessage — bestimmt Unread-Counter)
- `last_read_at`
- `is_pinned` (Boolean — User hat Channel im Index angeheftet)
- `left_at` (Soft-Leave; User kann re-joined werden)

Constraints:
- Unique `(chat_id, user_id) WHERE left_at IS NULL`
- Mind. 1 `admin` pro `group`-Channel; bei Last-Admin-Leave: Klartext-Block oder Auto-Promote eines anderen Members
- DM hat kein `admin`-Konzept (beide gleichberechtigt)

Indizes:
- `user_id, tenant_id, left_at` für *„meine Channels"*-Listen
- `chat_id, user_id` für Membership-Lookups

TeamChatMessage

Einzelne Nachricht im Channel oder Thread.

- `id`, `tenant_id`, `chat_id`, `created_at`, `updated_at`, `author_user_id`
- `body` (Markdown, max. 8000 Zeichen)
- `body_format` (Enum: `markdown | system` — `system` für automatische Nachrichten wie *„Max wurde hinzugefügt"*)
- `parent_message_id` (FK — wenn gesetzt, ist Nachricht Teil eines Threads; siehe Block F)
- `mentions[]` (JSON-Array aufgelöster Mentions, siehe Block F.4)
- `attachments[]` (FK-Liste auf File-Records aus M23)
- `link_previews[]` (JSON — siehe Block G.2)
- `edited_at` (NULL bis editiert)
- `deleted_at` (Soft-Delete; UI zeigt *„Nachricht gelöscht"*-Platzhalter)
- `deleted_by_user_id` (Author selbst oder Channel-Admin/Tenant-Admin)
- `system_event_kind` (NULL oder Enum: `member_added | member_removed | channel_renamed | channel_archived | project_stage_changed | acceptance_completed | ...`)
- `system_event_metadata` (JSON für strukturierte System-Nachrichten)

Constraints:
- `body` darf bei `attachments != []` leer sein (Reine-Datei-Nachricht)
- `body_format=system` darf NICHT von Usern erstellt werden (nur Backend)
- Editier-Fenster: 24h nach `created_at` (außer `tenant.admin`); danach Hard-Edit verboten

Indizes:
- `chat_id, parent_message_id, created_at DESC` für Channel-Verlauf-Pagination (Thread-Posts werden ausgefiltert mit `parent_message_id IS NULL`)
- `chat_id, parent_message_id, created_at ASC` für Thread-Verlauf
- GIN-Index auf `mentions` für *„meine Mentions"*-Query
- Trigram-Index auf `body` für In-Channel-Suche

TeamChatPinnedMessage

Eine Nachricht kann pro Channel angeheftet werden.

- `id`, `tenant_id`, `chat_id`, `message_id`, `pinned_by_user_id`, `pinned_at`
- `note` (optional Freitext, warum angeheftet)

Constraints:
- Unique `(chat_id, message_id)` (eine Nachricht max. einmal pro Channel angeheftet)
- Max. 50 Pinned pro Channel (Soft-Limit, Klartext bei Erreichen)
- Pinning erfordert `chat.admin` ODER Author-eigene Nachricht

TeamChatReaction

Emoji-Reaktion auf eine Nachricht.

- `id`, `tenant_id`, `message_id`, `user_id`, `emoji` (String, max. 16 Zeichen — Unicode-Emoji oder Tenant-Custom-Shortcode), `reacted_at`

Constraints:
- Unique `(message_id, user_id, emoji)` (gleiche Reaktion nicht doppelt — Toggle entfernt sie)
- Max. 20 verschiedene Emojis pro Nachricht (Anti-Spam)
- Max. 8 Reaktionen pro User pro Nachricht (Anti-Spam)
- Custom-Emoji-Shortcodes: Phase 2 — in Phase 1 nur Standard-Unicode

TeamChatThread

Eine Sub-Konversation, die an eine Eltern-Nachricht angehängt ist.

- `id`, `tenant_id`, `chat_id`, `parent_message_id` (Unique — eine Nachricht hat max. 1 Thread)
- `created_at`, `created_by_user_id`
- `last_reply_at`
- `reply_count` (denormalisiert für UI-Badge)
- `participant_user_ids[]` (denormalisiert)

Begründung: Threads sind eigene Entität (statt nur `parent_message_id` auf Messages), weil Aggregat-Daten (`reply_count`, `last_reply_at`, `participant_user_ids`) sonst pro Render aggregiert werden müssten — performance-kritisch bei Channel mit vielen Threads.

Constraints:
- Thread-Erstellung legt automatisch TeamChatThread-Record an, wenn erste Reply unter Eltern-Nachricht entsteht
- Thread-Replies sind TeamChatMessages mit `parent_message_id != NULL`

Indizes:
- `chat_id, last_reply_at DESC` für *„aktive Threads in diesem Channel"*

TeamChatRead (Read-Receipts)

Pro-User-Pro-Nachricht-Read-Marker. Wird nur für *gelesene* Markierungen geschrieben (kein Eintrag = ungelesen). Aggregat-Sicht „X von Y haben gelesen".

- `id`, `tenant_id`, `message_id`, `user_id`, `read_at`

Constraints:
- Unique `(message_id, user_id)`
- Wird automatisch geschrieben, wenn Client meldet *„Nachricht in View sichtbar mind. 1 Sekunde"*
- Bei DMs: gelesen-Status sofort sichtbar
- Bei Group/Project: Aggregat *„X von Y gelesen"*

Performance:
- Index auf `message_id, user_id` für Aggregat-Counts
- Bei sehr aktiven Channels (>100 Members): Read-Receipts werden pro Channel-Member-Setting deaktivierbar (Phase 2)

Verknüpfung mit anderen Modulen

- `TeamChat.project_id` → M4 Project
- `TeamChatMessage.author_user_id` → User
- `TeamChatMessage.attachments` → M23 File-Records
- `TeamChatMessage.mentions` → Auflösung auf Contacts/Companies/Deals/Projects/Users (siehe F.4)
- *„Mark as Task"* erzeugt Task in M26 mit Backlink auf Source-Message

---

Block B — Menü-Struktur und Routes

Sidebar-Hauptpunkt: *Chat* mit Unread-Badge (Summe Unread-Counter aller eigenen Channels).

Routen (Tenant-scoped):

```
/{tenant}/chat                                  → Channel-Index (Default-Landing)
/{tenant}/chat/new                              → Neuer Channel anlegen (Group oder DM)
/{tenant}/chat/:id                              → Channel-Detail (Verlauf + Compose)
/{tenant}/chat/:id/info                         → Channel-Settings (Members, Notifications,
Archivieren)
/{tenant}/chat/:id/thread/:thread_id            → Thread-Detail
/{tenant}/chat/:id/pinned                       → Übersicht aller Pinned Messages des Channels
/{tenant}/chat/:id/files                        → Übersicht aller Dateien aus diesem Channel
/{tenant}/chat/search                           → Cross-Channel-Suche (alle eigenen Channels)
/{tenant}/chat/mentions                         → *„Meine Mentions"*-Aggregat über alle Channels
/{tenant}/chat/archived                         → Archivierte Channels (auf die User Zugriff hat)

/{tenant}/admin/chat/                           → Tenant-Admin-Übersicht (alle Channels, Audit,
Cleanup-Tools)
/{tenant}/admin/chat/:id                        → Admin-Channel-Detail (Audit, Force-Member-Edit,
Force-Delete)
```

Mobile-Sichtbarkeit (LP-2 Baustellen-Modus):
- Index, Channel-Detail, Mention-Aggregat sind mobile-first
- Cross-Channel-Suche eingeschränkt auf Volltext-Modus (kein Power-Filter)
- Thread-View vereinfacht (Eltern-Message + Replies linear, kein Side-by-Side)

Block C — Channel-Typen und Lifecycle

C.1 — Projekt-Chat (`kind=project`)

Auto-Anlage:
- Bei Project-Anlage in M4 (Block F — alle Anlage-Quellen) wird automatisch ein TeamChat mit `kind=project` und `project_id=Project.id` erstellt
- Default-Members: alle ProjectMembers + assigned PL aus Project (M4)
- Default-Name: Project-Title (in UI, nicht persistiert; bleibt synchron)
- Beschreibung-Default: *„Projekt-Chat für [Project-Title], P-2026-XXXX"*
- Erste System-Nachricht: *„Projekt angelegt von [User] am [Datum]. PL: [Name]. Members: [Liste]."*

Member-Sync:
- ProjectMember-Add in M4 → Chat-Member-Add (mit System-Message)
- ProjectMember-Remove in M4 → Chat-Member auf `left_at` gesetzt + System-Message
- PL-Wechsel in M4 → System-Message im Chat
- Manuelles Member-Edit im Chat erlaubt: zusätzliche User können ohne ProjectMember-Status zum Chat eingeladen werden (z.B. Backoffice für Klärungs-Frage). Diese erscheinen NICHT als ProjectMember in M4.

Auto-Archive:
- Bei Stage-Wechsel auf `completed` ODER `cancelled` (M4 Block F) wird Chat 7 Tage später automatisch archiviert (Cron)
- Tenant-Admin-Setting pro Pipeline: Auto-Archive-Verzögerung in Tagen (Default 7, Range 0-90, oder „nie")
- Bei Project-Reaktivierung (M4): Chat wird automatisch reaktiviert
- Beim Auto-Archive-Schritt: Chat read-only, Members behalten Lese-Zugriff, System-Message *„Chat archiviert — Projekt abgeschlossen am [Datum]"*

Hard-Delete:
- Project-Hard-Delete (M4) → blockiert wenn aktive Acceptance-Protokolle existieren; sonst kaskadiert auch Chat-Hard-Delete (Audit-Trail bleibt im Activity-Stream)

C.2 — Gruppen-Chat (`kind=group`)

Anlage: Manuell unter `/{tenant}/chat/new` durch User mit `chat.create_channel`.

Konfiguration beim Anlegen:
- Name (Pflicht)
- Description (optional)
- Privacy (`is_private`): privat (nur Members sehen) oder öffentlich (im Channel-Browser sichtbar, jeder mit `chat.read_own_channels` kann beitreten)
- Initial-Members: Multi-Select (Users + Groups aus M27 Block C.4 — Group-Auswahl löst alle Member ein)
- Notification-Default für neue Members
- Optional: aus Group-Definition (M27 C.4) auto-syncen → bei Group-Mitglieder-Änderung wird Chat-Member-Liste angeglichen (Toggle, Phase 1 ja)

Use-Cases (Geerkens-Beispiele): *„Team Süd"*, *„Außendienst-Allgemein"*, *„Lehrlinge"*, *„Notfall-Pikett"*.

Member-Verwaltung:
- Channel-Admins können einladen/entfernen
- *„Channel verlassen"* möglich für jeden Member (Last-Admin-Block)
- Bulk-Add aus Group oder via Multi-Select

Archivierung:
- Manuell durch Channel-Admin oder Tenant-Admin
- System-Message: *„Channel archiviert von [User] am [Datum]"*
- Read-only ab Archivierung; Reaktivierung möglich

C.3 — Direkt-Nachricht (`kind=dm`)

Anlage:
- User klickt *„Neue DM"* → User-Picker → System sucht existierenden DM zwischen den beiden Usern
- Falls existiert: Reuse + Navigation
- Falls nicht: Anlage mit beiden Members + erste leere Konversation

Eigenschaften:
- Genau 2 Members, immer
- Privat (nicht im Channel-Browser sichtbar)
- Read-Receipts immer aktiv und sichtbar pro Nachricht
- Tenant-Admin sieht DMs **nicht** standardmäßig (Privacy); nur bei explizitem Audit-Antrag mit Begründung (Audit-Log)
- Keine Pinned Messages, keine Member-Verwaltung
- Threads erlaubt
- Reactions erlaubt

Group-DMs (3+ User): Phase 2. In Phase 1 wird empfohlen, Group-Channel anzulegen.

Lebenszyklus:
- DMs werden NICHT auto-archiviert
- Manuelle Archivierung durch einen der beiden Member möglich (beim anderen wird Chat dann auch archiviert)
- Bei User-Deaktivierung (M27 D.2): DM wird auf der Seite des verbleibenden Users sichtbar als *„DM mit ehemaligem Mitarbeiter"* (read-only)

Block D — Listen-View (Channel-Index)

`/{tenant}/chat`

Zu zeigende Daten pro Channel-Eintrag:
- Channel-Name (bei DM: Name des anderen Users)
- Avatar/Initialen (bei DM: anderer User; bei Group: Channel-Icon; bei Project: Project-spezifischer Indikator)
- Letzter Nachrichten-Auszug (Body-Preview, max. 80 Zeichen, Mentions hervorgehoben)
- Letzter Nachrichten-Zeitstempel (relativ wenn < 7 Tage)
- Unread-Counter (Badge mit Zahl)
- Mention-Indikator (separat von Unread, falls in den ungelesenen Nachrichten Mentions auf den User existieren)
- Pin-Indikator (User-Pin, siehe TeamChatMember.is_pinned)
- Channel-Typ (visuell unterscheidbar: Projekt / Gruppe / DM)
- Bei Projekt-Chats: Project-Stage als zusätzlicher Hinweis
- Mute-Indikator (wenn `notification_pref=muted`)

Benötigte UI-Elemente:
- Listen-Anzeige aller Channels des Users (default sortiert nach `last_message_at DESC`, gepinnte zuerst)
- Such-Funktion (Channel-Namen + Member-Namen + Volltext über letzte Nachrichten)
- Filter-Funktionen:
- Channel-Typ (Projekt / Gruppe / DM / Alle)
- Nur ungelesene
- Nur mit Mentions
- Verknüpfter Project-Status (aktiv / archiviert / abgeschlossen)
- Tag (über Project-Tags bei Projekt-Chats)
- Toggle „Archivierte anzeigen"
- Aktions-Funktionen pro Eintrag: Anpinnen/Lösen, Mute/Unmute, Mark-as-read, Archivieren (mit Permission)
- Quick-Action-Funktion *„Neuer Channel"* / *„Neue DM"*
- Cross-Channel-Suche-Funktion (führt nach `/{tenant}/chat/search`)

- Sektion *„Meine Mentions"* als Direkt-Link
- Channel-Browser-Funktion für öffentliche Group-Channels (Discover)

Operationen:
- Klick öffnet Channel-Detail
- Pin / Unpin (User-spezifisch)
- Mute (per Channel)
- Mark all as read
- Archivieren (mit Permission)

Performance:
- Server-side gefiltert nach Member-Sichtbarkeit (User sieht nur eigene Channels + sichtbare öffentliche Groups)
- Index auf `(user_id, tenant_id, left_at, is_pinned, last_message_at DESC)` über TeamChatMember + JOIN
- Pagination: 50 Channels pro Page, *„Mehr laden"*
- WebSocket-Push bei neuer Nachricht aktualisiert Index in Echtzeit

---

Block E — Detail-View (Chat-Verlauf)

`/{tenant}/chat/:id`

Zu zeigende Daten:
- Channel-Header: Name, Typ-Indikator, Member-Anzahl, bei Projekt-Chat Link auf Project-Detail (M4)
- Pinned-Messages-Sektion (max. 3 sichtbar inline + Link auf Pinned-Übersicht)
- Chat-Verlauf (chronologisch absteigend von neu nach alt beim Scrollen, neuste unten)
- Pro Nachricht: Author-Avatar + Name, Zeitstempel, Body (Markdown gerendert), Reactions, Thread-Indikator (Anzahl Replies + letzter Reply-Zeitpunkt), Mentions hervorgehoben, Attachments inline gerendert (Bild-Thumbnails, Dateinamen mit Icon, Link-Previews)
- Edited-Marker bei `edited_at != NULL`
- Lösch-Platzhalter bei `deleted_at != NULL`
- System-Messages visuell anders dargestellt (z.B. zentriert, neutraler Stil — Designer)
- Read-Receipts-Aggregat unter eigener Nachricht (z.B. *„3 von 5 gelesen"*; bei DM einfach gelesen/ungelesen)
- Date-Bucket-Separatoren (heute / gestern / Montag, 5. Mai / ...)
- Compose-Bereich am unteren Rand des Verlaufs
- Markdown-Eingabe
- Datei-Anhang-Funktion (M23-Upload)
- Emoji-Picker
- @-Mention-Auto-Complete (siehe F.4)
- Send-Funktion (Tastatur-Shortcut Enter, Shift+Enter für Zeilenumbruch — aber mobile-friendly Variante)
- Draft-Auto-Save (lokal pro Channel; bleibt erhalten bei Tab-Wechsel)
- Sidebar/Sektion mit Channel-Info: Members-Liste, Pinned, Files, Settings-Funktion (für Channel-Admins)
- Typing-Indikator (*„Max tippt..."*) — siehe Block H

Benötigte UI-Elemente:
- Sektionierte Anzeige des Verlaufs mit Date-Bucketing
- Inline-Compose mit Markdown-Editor
- Emoji-Picker mit Suchfunktion
- @-Mention-Auto-Complete (Such-Funktion über User/Project/Deal/Contact)
- Datei-Drop-Zone (Drag-&-Drop)
- Per-Message-Aktions-Funktionen (Hover/Tap-Menü):
- Reagieren (Emoji-Picker)
- Antworten im Thread
- Anpinnen (mit Permission)
- Mark as Task

- Kopieren / Permalink
- Editieren (nur eigene + innerhalb 24h)
- Löschen (nur eigene + Channel-/Tenant-Admin)
- Scroll-to-Bottom-Funktion bei aktiver Lese-Position weit oben
- New-Messages-Indikator wenn unten weiter Nachrichten ankommen
- Thread-Side-Panel oder eigene Route für Thread-Detail

Operationen:
- Nachricht senden
- Datei anhängen
- Emoji einfügen
- @-Mention setzen (User / Projekt / Deal / Contact / Firma)
- Reagieren
- In Thread antworten
- Pinnen / Lösen (mit Permission)
- Mark as Task
- Editieren (mit Author + 24h-Fenster ODER `tenant.admin`)
- Löschen (Author / Channel-Admin / Tenant-Admin)
- Permalink kopieren
- Member hinzufügen / entfernen (Channel-Admin)
- Channel-Settings öffnen
- Channel verlassen
- Mark all as read

Performance-Anforderungen:
- Initial-Load: 50 letzte Nachrichten
- Infinite-Scroll nach oben mit Cursor-basierter Pagination (50er-Batches)
- WebSocket für neue Nachrichten / Edits / Deletes / Reactions / Read-Receipts (Block H)
- Optimistic-UI: Nachricht sofort gerendert nach Send, Server-Bestätigung aktualisiert ID
- Attachments lazy-geladen bei Eintritt in Viewport
- Read-Receipt-Schreiben gebatcht (alle 2 Sekunden, max. 50 Messages pro Batch)

Block F — Threads, Reactions, Pinned, Mentions

F.1 — Threads

Verhalten:
- Klick auf *„Antworten im Thread"* an einer Nachricht öffnet Thread-Side-Panel oder eigene Route `/{tenant}/chat/:id/thread/:thread_id`
- Thread enthält: Eltern-Nachricht + Replies in chronologischer Reihenfolge (älteste zuerst)
- Thread-Reply ist eine TeamChatMessage mit `parent_message_id = parent.id`; ergänzt die TeamChatThread-Aggregate (`reply_count`, `last_reply_at`, `participant_user_ids`)
- Im Channel-Verlauf zeigt die Eltern-Nachricht einen Thread-Indikator: *„X Replies"* + Avatare der Participants + letzter Reply-Zeitpunkt
- Replies erscheinen NICHT im Haupt-Verlauf (Default), außer User aktiviert *„Auch im Hauptverlauf zeigen"* per Reply (Slack-Pattern, Toggle pro Reply)
- Thread-Notifications: Reply benachrichtigt nur Thread-Participants + bei Mentions
- Mentions in Thread-Replies funktionieren wie im Hauptverlauf

Operationen:
- Reply schreiben
- Thread folgen / nicht folgen (für Notifications)

- Thread-Permalink kopieren
- Eigene Reply editieren / löschen

F.2 — Reactions

Verhalten:
- Klick auf Reagieren-Funktion an einer Nachricht öffnet Emoji-Picker
- Reaktion wird sofort übermittelt (Optimistic-UI)
- Anzeige als Badge unterhalb der Nachricht: Emoji + Counter (Klick zeigt Liste der reagierenden User)
- Klick auf eigene Reaktion entfernt sie (Toggle)
- Eigene + andere Reaktionen sichtbar; eigene visuell hervorgehoben

Anti-Spam-Konstraints:
- Max. 20 verschiedene Emojis pro Nachricht
- Max. 8 eigene Reaktionen pro Nachricht
- Bei Erreichen: Klartext-Hinweis

F.3 — Pinned Messages

Verhalten:
- Per-Message-Aktion *„Anpinnen"*
- Permission: `chat.admin` für den Channel ODER Author der Nachricht (eigene Notiz pinnen)
- Anzeige: Pinned-Sektion oben im Verlauf (max. 3 inline) + Übersicht unter `/{tenant}/chat/:id/pinned`
- *„Lösen"*-Funktion an gepinnter Nachricht
- Optional: Note/Begründung beim Pinnen

Use-Cases (SHK): Wichtige Telefonnummern, Adresshinweise, Material-Bestelllisten, Zugangsdaten Baustelle, kritische Termine.

F.4 — Mentions mit Auto-Verknüpfung

Mention-Typen:

Trigger	Picker zeigt	Auflösung-Ziel	Effekt
`@username`	Channel-Members + alle aktiven Tenant-User (mit Permission-Filter)	User	Notification für gementen User
`@projekt:` oder `#P-...`	Aktive Projekte mit `read_*`	M4 Project	Klickbarer Link, Auto-Activity im Project-Stream
`@deal:` oder `#D-...`	Aktive Deals mit `read_*`	M3 Deal	Klickbarer Link, Auto-Activity im Deal-Stream
`@kontakt:`	Sichtbare Contacts	M1 Contact	Klickbarer Link, Auto-Activity im Contact-Stream
`@firma:`	Sichtbare Companies	M1 Company	Klickbarer Link, Auto-Activity
`@channel`	(Spezial-Trigger)	alle Channel-Members	Notification an alle (mit Confirm bei großen Channels)
`@here`	(Spezial-Trigger)	alle aktuell *online* Members	Notification an Online-Members

Auto-Complete-Verhalten:
- Tipp-Trigger ist `@`
- Debounced Such-Lookup (200ms)
- Vorschläge gruppiert nach Typ (User zuerst, dann Projekte/Deals/Kontakte/Firmen)
- Top-10-Treffer

- Permission-aware: nicht-sichtbare Entities erscheinen nicht
- Bei keinem Treffer: Klartext *„Keine Treffer für '[query]'"* — KEIN Auto-Anlegen aus Mentions

Auto-Verknüpfung:
- Mention auf Project/Deal/Contact/Company in einer Nachricht erzeugt eine Activity im jeweiligen Stream (`subject = mentionTarget` , `summary = „[User] hat im Chat [Channel] über dich gesprochen: [Snippet]"`)
- Klick auf Mention im gerenderten Body navigiert zur jeweiligen Detail-View
- Bei Projekt-Chat-Mention auf den eigenen Project: keine Duplikat-Activity (Verlinkung implizit)

Speicherung:
- `TeamChatMessage.mentions[]` ist ein JSON-Array von Resolved-Mentions: `[{kind: 'user', id, label}, {kind: 'project', id, label, project_number}, ...]`
- Body enthält Mention als spezielle Markdown-Syntax (z.B. `<@user:abc>` , `<@project:xyz>`); UI rendert daraus klickbare Chips

Mass-Mentions:
- `@channel` in Channels mit > 25 Members triggert Klartext-Confirm: *„Damit benachrichtigst du 47 Personen. Trotzdem?"*
- `@here` zeigt aktuell-online-Anzahl im Confirm

Block G — Datei-Uploads, Link-Previews, Mark-as-Task

G.1 — Datei-Uploads

Integration mit M23 (Files):
- Dateien aus dem Compose-Bereich werden in den File-Namespace des Channels hochgeladen (Storage-Pfad `tenants/{tenantId}/chat/{chatId}/...`)
- Bei Projekt-Chat: zusätzlich Verknüpfung mit Project-File-Namespace (Datei taucht in M4 Files-Tab auf)
- File-Record mit Backlink auf Source-Message
- Multi-File-Upload (Drag-&-Drop oder File-Picker)
- Foto-Direct-Upload aus Mobile (Kamera) — siehe Block J
- Größen-/Typ-Limits aus M23 (Tenant-Setting)
- Anti-Virus-Scan aus M23

Anzeige im Verlauf:
- Bilder als Thumbnail inline (Klick öffnet Lightbox)
- Videos als Thumbnail mit Play-Button (Inline-Player oder externer Open)
- PDFs / Office-Dateien als File-Card mit Vorschau-Link
- Andere Datei-Typen: generisches File-Icon + Name + Größe
- Download-Funktion pro Datei
- Im Files-Tab des Channels: aggregierte Liste aller Dateien

Operationen:
- Hochladen, Herunterladen
- Löschen (nur Author + 24h-Fenster ODER Channel-/Tenant-Admin) — entfernt File und Backlink
- *„In Project-Files speichern"* (bei Nicht-Projekt-Chats: Funktion kopiert Datei in einen Ziel-Project-Namespace)

G.2 — Link-Previews

Verhalten:
- Beim Senden einer Nachricht erkennt Backend URLs im Body (Regex)
- Asynchroner Job zieht OG-Tags / oEmbed (Title, Description, Bild, Site-Name)
- Preview wird in `link_previews[]` gespeichert (max. 3 Previews pro Message; weitere ignoriert)
- WebSocket-Update an Channel: Preview erscheint nach kurzem Delay
- Bei Fehler: kein Preview (Klartext im Hover *„Vorschau nicht verfügbar"*)

Privacy-Einstellungen (Tenant-Admin):
- Toggle *„Link-Previews aktivieren"* (Default an)
- Whitelist/Blacklist von Domains (Phase 2)
- EU-Provider für Link-Preview-Service (LP-13): Eigener Microservice statt Drittanbieter

Klartext-Hinweise:
- Preview-Service down: *„Vorschau wird später nachgeladen"*
- Geblockte Domain: kein Preview, keine UI-Erwähnung

G.3 — Mark as Task

Verhalten:
- Per-Message-Aktion *„Mark as Task"* öffnet Task-Anlage-Modal mit:
- Task-Titel (Default: erste Zeile der Message bzw. erste 80 Zeichen)
- Beschreibung (Default: Message-Body komplett)
- Zugewiesener User (Default: Author falls Mention vorhanden — sonst leer)
- Deadline (optional)
- Verknüpftes Subject: Auto-übernommen aus Channel (Projekt-Chat → linkt auf Project; sonst leer / manuell wählbar)
- Bestätigen → Task wird in M26 angelegt mit:
- Backlink-Field auf Source-Message (TeamChatMessage-ID)
- Activity *„Task erstellt aus Chat"* im verknüpften Subject-Stream
- System-Reply unter der Source-Message: *„[User] hat aus dieser Nachricht eine Aufgabe erstellt: [Task-Titel] → [Link]"*

Permissions:
- `tasks.write` Pflicht
- Wenn fehlend: Funktion ausgeblendet; Tooltip *„Aufgaben erstellen erfordert erweiterte Berechtigungen"*

Block H — WebSocket-Server / Echtzeit-Übermittlung

H.1 — Architektur

Server-Komponente:
- Eigener WebSocket-Endpoint `/ws/chat` (separat von HTTP-API, gleicher Auth-Mechanismus über Session-Cookie + Tenant-Context)
- Ein WebSocket-Connection pro Browser-Tab pro User
- Connection-Auth bei Handshake: Session-Cookie + Tenant-ID validiert
- Rooms-Modell: pro Channel ein Room; User joined Room beim Öffnen des Channels und beim Hintergrund-Subscribe für eigene Channels (für Index-Updates)

Skalierung:
- Redis-Pub/Sub als Message-Broker zwischen Server-Instanzen (Multi-Pod-Setup)
- Sticky-Sessions am Load-Balancer (oder Token-basiertes Re-Connect)
- Backpressure: Bei langsamen Clients verwirft Server alte Events; Client lädt bei Reconnect verpasste Nachrichten via REST-Cursor

H.2 — Übermittelte Events

Event	Trigger	Empfänger
`message.new`	Neue Nachricht	Alle Channel-Members
`message.edited`	Nachricht editiert	Alle Channel-Members
`message.deleted`	Nachricht gelöscht	Alle Channel-Members
`reaction.added` / `reaction.removed`	Reaktion-Toggle	Alle Channel-Members
`thread.reply`	Thread-Reply	Channel-Members + Thread-Participants
`read.receipt`	Lese-Marker	Channel-Members
`typing.start` / `typing.stop`	Tipp-Indikator	Channel-Members (außer Sender)
`channel.member_added` / `member_removed`	Membership-Änderung	Alle Channel-Members
`channel.archived`	Archivierung	Alle Channel-Members
`channel.pinned_changed`	Pin/Unpin	Alle Channel-Members
`presence.online` / `presence.offline`	User wechselt Status	Alle User mit gemeinsamem Channel
`link_preview.ready`	Preview asynchron fertig	Alle Channel-Members

H.3 — Reconnect-Strategie

- Client speichert letzten gesehenen Event-Cursor pro Channel
- Bei Reconnect: REST-Call holt verpasste Nachrichten/Events seit Cursor
- WebSocket-Verbindungs-Verlust > 30 Sekunden → Banner *„Verbindung unterbrochen — versuche Neuaufbau..."*
- Connection-Pooling pro Tab (Single-Connection für alle Channels)

H.4 — Typing-Indikator

- Client sendet `typing.start` beim ersten Tastendruck im Compose
- Throttle: Re-Senden alle 3 Sekunden, solange weiter getippt wird
- `typing.stop` nach 5 Sekunden Inaktivität ODER Send
- UI zeigt *„X tippt..."* / *„X und Y tippen..."* / *„Mehrere tippen..."*

H.5 — Presence

- Pro User Online-Status: `online | away | offline`
- `online` = aktiver Tab + WebSocket connected
- `away` = WebSocket connected, kein Tab-Focus seit 5 Min
- `offline` = keine Connection
- Channel-Members-Liste zeigt Status-Indikator

H.6 — Architektur-Vorschriften

- LP-13: WebSocket-Server im EU-Hosting (kein Pusher.com / kein Drittanbieter)
- AV: Auth-Validation auch pro Event (nicht nur bei Handshake) — Permission-Änderung wirkt sofort
- Audit-relevante Events (member_added, channel_archived) gehen zusätzlich in Activity-Stream + Audit-Log

---

Block I — Read-Receipts und Notifications

I.1 — Read-Receipts

- Client beobachtet Viewport: Nachricht > 1 Sekunde sichtbar → in Read-Batch sammeln
- Batch alle 2 Sekunden an Server (oder beim Verlassen des Channels)
- Server schreibt TeamChatRead-Records + emittiert `read.receipt`-Event
- UI-Anzeige:
- DM: Pro-Message *„gelesen um [Zeit]"* unterhalb eigener Nachricht
- Group/Project: *„X von Y gelesen"* unterhalb eigener Nachricht; Klick zeigt Liste der Gelesen/Ungelesen-User
- TeamChatMember.last_read_message_id wird auf den höchsten gelesenen Message-Cursor gesetzt → bestimmt Unread-Counter im Index

I.2 — Notifications (M22)

Trigger:

Event	Default-Notification	Konfigurierbar pro Channel
Neue Message in Channel	abhängig von `notification_pref`	ja
@-Mention auf User	immer	nein (immer aktiv außer `muted`)
@here / @channel mit Online-Status	bei `notification_pref != muted`	ja
Neue DM	immer (außer `muted`)	nein
Thread-Reply (User folgt Thread)	immer	nein (per-Thread-Toggle)
Channel-Member-Add (Self)	immer	nein
Channel-Archive	nicht notifiziert	nein (System-Message reicht)

Channel-Notification-Modi:
- `all_messages` — jede Message benachrichtigt
- `mentions_only` — nur bei eigenen Mentions / DMs / @here / @channel
- `muted` — keine Notifications, nur visueller Unread-Counter

Channels:
- In-App Notification (Modul 22): Bell-Center mit Mention-Aggregat
- Push-Notification (Web-Push + Mobile-App): siehe Block J
- Email-Digest (Phase 2): tägliche Zusammenfassung verpasster Mentions/DMs

Anti-Spam-Bündelung (M22):
- Bei > 5 Nachrichten in < 1 Minute aus demselben Channel: Bündeln zu *„X neue Nachrichten in [Channel]"*
- Pro User pro Stunde max. 10 Push-Notifications (Rest gebündelt)
- Mentions umgehen Bündelung (immer einzeln zugestellt)

Quiet-Hours:
- Pro User unter `/{tenant}/settings/notifications/`: Stille-Zeiten (z.B. 20:00-7:00, Sa+So)
- Während Stille-Zeit: nur DMs + Mentions kommen durch (konfigurierbar)
- Manueller Override *„Auch in Stille-Zeit benachrichtigen"* pro Channel

Override-Hierarchie (höchste Priorität zuerst):
1. Channel-Mute → unterdrückt alles außer DMs und @-Mentions
2. Quiet-Hours → unterdrückt alles außer expliziter Override
3. User-Default-Notification-Pref → System-Default `mentions_only`
4. Channel-Default → bei neuen Channel-Beitritten

Block J — Mobile-First für Monteur (LP-2)

J.1 — Vereinfachte Chat-View

Ziel: Monteur auf Baustelle nutzt Smartphone, oft mit Handschuhen, schlechtes Licht, schnelle Bedienung.

Anforderungen:
- Compose-Funktion prominent erreichbar (Daumen-Zone)
- One-Tap *„Foto senden"*-Funktion (öffnet Kamera direkt; nach Aufnahme Vorschau + Send-Funktion ohne Modal-Verkettung)
- Voice-Note-Funktion (Halten zum Aufnehmen, Loslassen zum Senden) — Audio-File als Attachment + optionaler Whisper-EU-Transkript (M18) im Body
- Reduzierte Channel-Liste: Default-Filter *„Aktive Projekt-Chats + DMs mit Mentions"*
- Vereinfachte Mention-Auto-Complete (User-Liste prominent, Project/Deal/Contact unter *„Mehr"*)
- Pinned Messages prominent im Header (für Adressen, Material-Listen, Notrufnummern)
- Read-Receipts vorhanden, aber dezenter dargestellt
- Threads vereinfacht (lineare Reply-Liste statt Side-by-Side)
- Push-Notifications mit Quick-Reply-Funktion (direkt aus Notification antworten)

J.2 — Push-Notifications

Konfiguration:
- Pro User pro Gerät registriertes Push-Token (Web-Push API + native Mobile-App in Phase 2)
- Push für eigene Channels mit `notification_pref != muted`
- Mention-Push immer (unabhängig von Stille-Zeit konfigurierbar)
- Click auf Push-Notification öffnet direkt Channel-Detail an gemeinter Message-Position

Datenschutz:
- Push-Payload enthält NUR: Channel-Name (Projekt-Title generisch *„P-2026-XXXX"*), Message-Sender-Name, Snippet (max. 100 Zeichen)
- Keine Adressen, keine PII außer Sender-Vorname
- Tenant-Setting *„Push-Vorschau im Lock-Screen"* (Default an, Off = nur „Neue Nachricht")

J.3 — Foto-Direct-Upload

- Kamera-Zugriff aus PWA / Mobile-App
- Foto wird komprimiert (max. 2048px lange Kante, JPEG-Qualität 85)
- Optional: Auto-Caption mit Datum/Uhrzeit/GPS-Koordinaten als Footer-Text (Tenant-Toggle, Default aus, GPS nur mit User-Permission)
- Multi-Foto-Upload möglich (Galerie-Auswahl)
- Bei Projekt-Chat: Foto landet zusätzlich im Project-File-Namespace (M4/M23)

Block K — Activity-Stream-Integration

K.1 — Erzeugte Activities

Activity-Typ	Bedingung	Subject
`chat.message`	Neue Nachricht in Channel mit Bezug	linkt auf zugeordnetes Subject (Project/Deal/Contact bei Projekt-Chat / bei Mentions)
`chat.mention`	@-Mention auf Project/Deal/Contact/Company	Mention-Target
`chat.channel_created`	Channel angelegt	Channel selbst (oder verknüpftes Project)
`chat.channel_archived`	Channel archiviert	Channel / Project
`chat.task_created_from_message`	Mark-as-Task	Source-Message + Task
`chat.member_added` / `chat.member_removed`	Membership-Änderung	Channel

K.2 — Aggregat-Wirkung

- **Projekt-Detail (M4):** Activity-Stream zeigt `chat.message`-Aktivitäten des Projekt-Chats unter Filter *„Chat"* (Auto-Filter aktiv); Default ausgeblendet, da hohes Volumen — User kann *„Chat-Nachrichten einblenden"* togglen
- **Deal-Detail (M3):** Mentions auf Deal aus Chat erscheinen im Stream (selten, niedrigere Volumen)
- **Contact-Detail (M1):** Mentions auf Contact erscheinen
- **Globale Aggregate (Dashboard):** Summary-Kacheln zeigen *„X Mentions / Tag"*, *„Y aktive Chats"*

K.3 — Datenschutz

- `chat.message`-Activities tragen NUR `summary_text` (Sender + Channel + Snippet 80 Zeichen) + `metadata` mit Message-ID
- Klick auf Activity navigiert in den Channel — Permission-aware (User ohne Channel-Zugang sieht Activity-Eintrag, aber Klick führt zu *„Du bist kein Mitglied dieses Channels"*-Hinweis)
- Activities respektieren Channel-Privacy: bei privaten Channels wird Activity NUR Channel-Members angezeigt (nicht z.B. allen mit `crm.read_all`)

Block L — Permissions-Anwendung

L.1 — Permission-IDs

Permission	Erlaubt
`chat.read_own_channels`	Eigene Channels (Member-Status) sehen
`chat.write`	Nachrichten senden, reagieren, Threads, Datei-Uploads, Mark-as-Task in eigenen Channels
`chat.create_channel`	Group-Channels anlegen
`chat.create_dm`	Direkt-Nachrichten starten
`chat.admin`	Channel-Settings (alle Channels), Force-Member-Edit, Force-Archive, Force-Delete-Message
`chat.read_all_channels`	Tenant-Admin-Sicht: alle Channels lesen (Audit) — Nutzung wird audit-loggt

Permission-Verhalten innerhalb Channel:
- Channel-Admin (TeamChatMember.role_in_chat=admin) hat Channel-Local-Admin-Rechte ohne `chat.admin` global zu brauchen
- `chat.admin` sticht Channel-Admin-Status aus

L.2 — Bundle-Mapping (Variante 2)

Bundle	Atomare Permissions
`chat.none`	(keine — User sieht Sidebar-Punkt nicht)
`chat.read_own`	`chat.read_own_channels`
`chat.write`	obigen + `chat.write` + `chat.create_dm`
`chat.member`	obigen + `chat.create_channel`
`chat.admin`	obigen + `chat.admin` + `chat.read_all_channels`

L.3 — SHK-Template-Default-Rollen

Rolle	Chat-Bundle
GF	`chat.admin`
Vertriebsleiter	`chat.member`
Vertriebler	`chat.write`
Projektleiter / Bauleiter	`chat.member`
Büro / Backoffice	`chat.member`
Marketing	`chat.write`
Monteur	`chat.write` (kein Group-Channel-Anlage; DM aktiv; Projekt-Chats werden auto-zugeordnet)

L.4 — Anwendung in Routes

- `/{tenant}/chat` → `chat.read_own_channels`
- `/{tenant}/chat/new` → `chat.create_channel` (Group) oder `chat.create_dm` (DM)
- `/{tenant}/chat/:id` → Membership-Check (User muss TeamChatMember sein) ODER `chat.read_all_channels`
- `/{tenant}/chat/:id/info` → Channel-Member oder `chat.admin`
- `/{tenant}/chat/:id/thread/:thread_id` → Membership-Check
- `/{tenant}/chat/mentions` → `chat.read_own_channels`
- `/{tenant}/chat/archived` → `chat.read_own_channels` (zeigt nur archivierte Channels, in denen User Member war)
- `/{tenant}/admin/chat/` → `tenant.admin` ODER `chat.admin`

L.5 — Action-Buttons / Operationen

- Senden / Reagieren / Thread-Reply / Mention: `chat.write` + Membership
- Pinnen / Entfernen-Pin: `chat.admin` global ODER Channel-Admin ODER (eigene Message)
- Editieren: Author + 24h-Fenster ODER `tenant.admin`
- Löschen: Author + 24h-Fenster ODER Channel-Admin ODER `chat.admin`
- Member hinzufügen: Channel-Admin ODER `chat.admin`
- Channel umbenennen / Description ändern: Channel-Admin ODER `chat.admin`

- Channel archivieren: Channel-Admin ODER `chat.admin` ODER (bei Projekt-Chat) PL über Cascading aus M4
- Channel hard-löschen: nur `chat.admin` (mit Klartext-Confirm + Audit-Log)
- Mark as Task: zusätzlich `tasks.write`

L.6 — Owner-Check-Logik

- `chat.read_own_channels`: User darf Channels sehen, in denen er aktiver TeamChatMember ist (`left_at IS NULL`)
- DMs: nur die zwei Members
- Projekt-Chats: ProjectMember-Sync (Member-Add in M4 = Member-Add in Chat); wer kein ProjectMember mehr ist, behält Lese-Zugriff bis Auto-Archive (Audit), aber kann nicht mehr schreiben
- Öffentliche Group-Channels (`is_private=false`): User mit `chat.read_own_channels` sehen Channel im Browser und können beitreten

L.7 — `read_all_channels` (Tenant-Admin-Audit)

- Sicht auf alle Channels inkl. DMs
- Jede Lese-Aktion mit dieser Permission wird zusätzlich audit-loggt (Audit-Trail-Modul) mit Klartext-Begründungs-Pflicht (User muss bei erstmaligem Zugriff pro Session Begründung eingeben)
- Admin kann nicht im fremden Namen schreiben, nur lesen
- DMs werden mit Banner *„Diese DM wird im Audit-Modus angezeigt"* markiert

---

Block M — Klartext-Hinweise / -Fehler

- Leerer Channel-Index: *„Du bist noch in keinem Channel. Lege einen neuen an oder warte auf eine Einladung."*
- Leerer Channel: *„Noch keine Nachrichten. Schreib die erste!"*
- Keine Permission auf Channel: *„Du bist kein Mitglied dieses Channels."* (404 statt 403, um Existenz nicht zu offenbaren — außer bei öffentlichen Group-Channels)
- Channel archiviert: Banner *„Dieser Channel ist archiviert. Du kannst lesen, aber nicht mehr schreiben. [Reaktivieren]"* (Reaktivieren mit Permission)
- Verbindung verloren: *„Verbindung unterbrochen — versuche Neuaufbau..."*
- Reconnect erfolgreich: kurzer Hinweis *„Wieder verbunden"* + Auto-Sync verpasster Nachrichten
- Mass-Mention-Confirm: *„Damit benachrichtigst du 47 Personen. Trotzdem?"*
- Editieren nach 24h: *„Editieren ist nur innerhalb 24 Stunden möglich."*
- Datei zu groß: *„Datei [Name] ist zu groß ([X] MB von max. [Y] MB)."*
- Datei-Typ blockiert: *„Dateityp [.exe] ist nicht erlaubt."*
- Last-Admin-Leave: *„Du bist der einzige Admin in diesem Channel. Bitte einen anderen Member zum Admin ernennen oder den Channel archivieren, bevor du gehst."*
- Last-Admin-Member entfernen: analog
- Mark-as-Task ohne Permission: *„Aufgaben erstellen erfordert erweiterte Berechtigungen."* (Funktion ausgeblendet im Default; nur on-Hover/-Tap-Hinweis)
- Tenant-Admin-DM-Audit-Begründung-Pflicht: Modal mit Pflicht-Textfeld *„Bitte begründe den Audit-Zugriff (z.B. Compliance-Anfrage, Vorfall-Untersuchung)"*
- Push-Notification-Berechtigung verweigert: *„Browser-Benachrichtigungen sind deaktiviert. In den Browser-Einstellungen aktivieren um wichtige Mentions zu erhalten."*
- Quiet-Hours aktiv: kleine Anzeige im Settings-Bereich *„Stille-Zeit bis [Zeit] — nur DMs und Mentions kommen durch."*
- Reaktion-Limit erreicht: *„Du hast bereits 8 Reaktionen auf diese Nachricht. Entferne eine, um eine neue hinzuzufügen."*
- Pinned-Limit: *„Maximal 50 Nachrichten pro Channel können angeheftet werden. Bitte ältere Pins lösen."*

- Auto-Archive-Vorwarnung: System-Message 24h vor Auto-Archive *„Dieser Projekt-Chat wird morgen automatisch archiviert (Projekt abgeschlossen am [Datum]). [Archivierung verschieben]"*

Block N — Phase 2 / spätere Features

Bewusst NICHT in Phase 1:

- **Group-DMs (3+ User ohne Group-Channel):** in Phase 1 stattdessen Group-Channel anlegen
- **Custom-Emojis:** nur Standard-Unicode in Phase 1
- **Email-Digest:** tägliche Zusammenfassung verpasster Mentions/DMs
- **Voice-/Video-Calls** im Chat (Phase 2 oder später; bleibt bei AirCall/Externe)
- **Slack-/Teams-Bridge:** keine Phase-1-Integration
- **Bots / Slash-Commands:** kein /-Command-Framework in Phase 1; nur Mark-as-Task als Direkt-Aktion
- **End-to-End-Verschlüsselung:** Phase 1 nutzt Server-side-Verschlüsselung at rest, kein E2EE
- **Search-Power-Filter:** Cross-Channel-Suche in Phase 1 nur Volltext + Channel-Filter; erweiterte Filter (nach Author, Date-Range, Mention-Type) Phase 2
- **Message-Scheduling:** *„Später senden"*-Funktion Phase 2
- **Pinned-Reorder:** Drag-Reorder von Pinned-Messages Phase 2 (in Phase 1 Sortierung nach Pin-Datum DESC)
- **Tenant-übergreifende Channels (Inter-Tenant-Chat):** nicht angedacht
- **KI-Auto-Übersetzung (M18):** Phase 2 — automatische DE↔PL/EN/UA-Übersetzung von Chat-Messages für mehrsprachige Teams (PL-Monteur schreibt PL, Büro liest DE). Toggle pro User pro Channel; Übersetzung als Hover/Tap-Layer unter Original-Body. Sprach-Detect mit Fallback auf User-`Contact.language`
- **KI-Chat-Summarizer:** *„Was war heute wichtig?"*-Zusammenfassung über alle eigenen Channels (M18, Phase 2)
- **KI-Smart-Reply-Vorschläge:** Phase 2
- **Voice-Note-Auto-Transcript-Search:** Phase 1 hat Voice-Notes als Audio + optional Transcript; Volltext-Suche in Transcripts erst Phase 2
- **Native Mobile-Apps:** Phase 1 = PWA mit Web-Push; Native iOS/Android Phase 2
- **Push-Notification-Quick-Reply native:** in PWA nur Click-to-Open; native Quick-Reply Phase 2

Architektur-Vorschriften (Querschnitt)

- **LP-2:** Mobile-First für Chat-Detail, Foto-Upload, Voice-Note, Push-Quick-Reply (Phase 2)
- **LP-3, AV #3:** Chat-Notifications-Defaults und Auto-Archive-Verzögerung sind Branchen-Template-konfigurierbar (SHK-Default 7 Tage)
- **LP-7:** Quiply komplett abgelöst (Mitarbeiter-App, HR + Chat → HiveGrid)
- **LP-9 + Stil-Vorgabe:** UI-Element-Anforderungen als Liste, kein Layout
- **LP-10, AV #1:** Klartext-Diagnostik in jedem Block
- **LP-13:** WebSocket-Server EU-Hosting, Link-Preview-Service EU-self-hosted; Whisper-EU-Transkription für Voice-Notes; KI-Übersetzung Phase 2 über EU-Provider
- **AV #5:** Permissions-aware-Default in Channel-Listen (server-side gefiltert), Mention-Auto-Complete (nur sichtbare Entities), Activity-Anzeige (Permissions-respektierend)
- **AV #6:** Audit-Trail-Default für alle schreibenden Aktionen (Member-Edits, Channel-Archive, Force-Delete, Tenant-Admin-DM-Audit)
- **AV #20:** Action-Boundary für KI — KI-Übersetzung (Phase 2) ist passive Render-Layer, kein Auto-Send. KI-Smart-Reply-Vorschläge nur als Vorschlag, niemals automatisch versendet

---

Detail-Spec-Status: DRAFT

Erste vollständige Detail-Spec für Modul 13. **Vor Bau:** Verifizierung mit Noah (zusammen mit Pflichtenheft-Sweep, speziell Notification-Defaults und Auto-Archive-Verzögerung pro Pipeline). Bei Implementierung dann konkrete WebSocket-Event-Schemata, Markdown-Renderer-Spec und Mention-Resolver-API ergänzen.

Offene Detail-Punkte (nicht-blockierend, beim Bau zu klären):
- Konkrete Auto-Archive-Verzögerungs-Defaults pro SHK-Pipeline (Anlagen-Installation 14d? Wartung 7d? Notfall 3d?)
- Voice-Note: Maximum-Aufnahme-Dauer (Vorschlag 5 Min) und Storage-Limit pro Tenant
- Anti-Spam-Bündelung-Schwellen mit M22 abstimmen (5 Messages/Min vs. 3 vs. 10)
- Tenant-Admin-DM-Audit-Begründungs-UI: dropdown-vorgegeben oder Freitext?
- WebSocket-Reconnect-Backoff-Strategie (Exponential vs. linear) — Implementierungs-Detail
- Read-Receipt-Disable-Pro-User: ob Phase 1 oder Phase 2 (User möchte ggf. nicht zeigen, dass er gelesen hat)
- Group-DM-Konvertierung in Group-Channel: ob User-Workflow Phase 1 oder Phase 2
- Concrete File-Type-Whitelist für Chat-Uploads (überlappt mit M23)
- Subframe-UI-Designs (separat)
- Konkrete Migrations-Skripte für Quiply-Daten-Import (separates Migrations-Modul)