```
module: 25
title: Calendar-Sync (Google + Microsoft)
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-7, LP-9, LP-10, LP-11, LP-13]
spec_style: requirements-pur
```

# Modul 25 — Calendar-Sync

> ***Detail-Spec gemäß LP-9.*** *OAuth2-Connect von Google Calendar und Microsoft Outlook/365-Calendar* ***pro User*** *(nicht pro Tenant). Bidirektionale Synchronisation: HiveGrid-Termine (Booking, Projekt, Plantafel) gehen nach extern; externe Termine kommen read-only ins HiveGrid für Verfügbarkeits-Berechnung im Booking. Adapter-Pattern (LP-11): Google + MS implementieren gemeinsames* `CalendarAdapter` *-Interface.*

---

## Themenblöcke

```
A. Datenmodell (CalendarConnection, CalendarEvent, CalendarEventMapping, CalendarSyncCursor,
CalendarWebhookEvent)
B. Menü-Struktur und Routes
C. OAuth2-Connect-Flow (Google + Microsoft)
D. Sync-Richtungen und Mappings
E. Webhook-Sync (kein Polling)
F. Konflikt-Handling (Last-Write-Wins mit Hinweis)
G. Adapter-Pattern (CalendarAdapter-Interface)
H. Verfügbarkeits-Berechnung (Integration mit Booking, Modul 6)
I. Permissions-Anwendung
J. Klartext-Hinweise
K. Phase 2
```

---

## Block A — Datenmodell

### CalendarConnection

Pro User pro Provider eine Verbindung. Tenant-scoped (User kann pro Tenant unterschiedliche Konten verbinden), RLS-aktiv.

- `id`, `tenant_id`, `user_id`, `created_at`, `updated_at`
- `provider` (Enum: `google | microsoft`)
- `external_account_email` (z.B. `noah@geerkens.de` — User-sichtbar zur Kontrolle)
- `external_account_id` (Provider-interne Account-ID)
- `oauth_refresh_token` (verschlüsselt at-rest)
- `oauth_access_token` (verschlüsselt; kurzer Lebenszyklus)
- `oauth_access_token_expires_at`
- `oauth_scopes[]` (gewährte Scopes; bei Reduktion durch User → Klartext-Hinweis und Re-Connect)
- `webhook_subscription_id` (Provider-interne Subscription-ID; bei Google: `channel.id`, bei MS: `subscription.id`)
- `webhook_subscription_expires_at` (Provider-Limits: Google ~1 Woche, MS ~3 Tage; Auto-Renewal-Job nötig)

- `selected_calendars[]` (JSONB — User wählt welche externen Kalender synchronisiert werden; Default: Primary-Calendar)
- `sync_direction` (Enum: `bidirectional | hivegrid_to_external_only | external_to_hivegrid_only`; Default `bidirectional`)
- `status` (Enum: `active | paused | error | revoked`)
- `last_sync_at`
- `last_error_text` (Klartext, falls `status = error`)

**Constraints:** `(tenant_id, user_id, provider, external_account_id)` unique.

### CalendarEvent

Cache externer Events + interne Events (über Mappings verknüpft). Tenant-scoped, RLS-aktiv.

- `id`, `tenant_id`, `user_id`, `created_at`, `updated_at`
- `source` (Enum: `hivegrid_internal | external_synced`)
- `provider` (NULL bei `hivegrid_internal`, sonst `google | microsoft`)
- `external_event_id` (Provider-Event-ID, NULL bei rein internen Events ohne externe Spiegelung)
- `external_calendar_id` (welcher externer Kalender; NULL bei internen)
- `title`
- `description`
- `location`
- `starts_at`, `ends_at`, `is_all_day`
- `time_zone`
- `recurrence_rule` (RRULE-String, optional)
- `attendees` (JSONB — Liste mit Email + Status)
- `etag` / `version_token` (Provider-spezifisch — für Konflikt-Erkennung)
- `last_modified_external` (Provider-Modify-Timestamp)
- `is_busy_for_availability` (Boolean — fließt in Verfügbarkeits-Berechnung; bei externen Events Default true außer „Free"-Status)
- `internal_entity_link` (JSONB optional — z.B. `{ "type": "booking", "id": "uuid" }` oder `{ "type": "project", "id": "uuid" }` oder `{ "type": "plantafel_entry", "id": "uuid" }`)
- `archived_at` (Soft-Delete für gelöschte externe Events behalten wir kurz, dann purge)

**Indizes:**
- `(tenant_id, user_id, starts_at)` für Verfügbarkeits-Queries
- `(tenant_id, user_id, provider, external_event_id)` unique (für Update-Lookup)
- `(tenant_id, user_id, last_modified_external DESC)` für Sync-Cursor-Queries

### CalendarEventMapping

Verknüpft HiveGrid-interne Events mit externen Spiegel-Events.

- `id`, `tenant_id`, `created_at`, `updated_at`
- `internal_event_id` (FK auf `CalendarEvent` mit `source = hivegrid_internal`)
- `external_event_id` (Provider-Event-ID)
- `provider`
- `external_calendar_id`
- `connection_id` (FK auf `CalendarConnection`)
- `last_synced_at`
- `last_known_external_etag`
- `last_known_internal_version_token`

**Constraints:** ein `internal_event_id` kann mehrere Mappings haben (z.B. wenn User Google + MS gleichzeitig verbunden) — also `(internal_event_id, provider, external_calendar_id)` unique.

### CalendarSyncCursor

Pro Connection und Kalender ein Sync-Cursor (Provider-spezifisch).

- `id`, `tenant_id`, `connection_id`, `external_calendar_id`, `created_at`, `updated_at`
- `cursor_token` (Google: `syncToken`; MS: `deltaLink`)
- `last_full_sync_at`

**Constraints:** `(connection_id, external_calendar_id)` unique.

### CalendarWebhookEvent

Eingehende Webhook-Notifications werden persistiert (für Idempotenz und Retry).

- `id`, `tenant_id`, `connection_id`, `received_at`
- `provider`
- `notification_payload` (JSONB)
- `processing_status` (Enum: `pending | processing | completed | failed | skipped`)
- `processing_error_text` (Klartext)
- `processed_at`
- `external_resource_state` (Provider-spezifisch — z.B. Google's `X-Goog-Resource-State`)

**Indizes:** `(tenant_id, processing_status, received_at)` für Worker-Pickup.

---

## Block B — Menü-Struktur und Routes

**Sidebar-Hauptpunkt:** keiner direkt — Settings-Bereich.

**Routen:**

```
/{tenant}/settings/integrations/calendar/                → Übersicht Verbindungen + Connect-Buttons
/{tenant}/settings/integrations/calendar/google/connect    → OAuth-Initiate Google
/{tenant}/settings/integrations/calendar/google/callback   → OAuth-Callback Google
/{tenant}/settings/integrations/calendar/microsoft/connect → OAuth-Initiate Microsoft
/{tenant}/settings/integrations/calendar/microsoft/callback→ OAuth-Callback Microsoft
/{tenant}/settings/integrations/calendar/:connection_id    → Detail-View einer Verbindung (Kalender-Auswahl, Sync-Status, Logs)
```

**Embedded-Anwendung:**

- Verfügbarkeits-Berechnung im Booking (Modul 6) liest `CalendarEvent` mit `is_busy_for_availability = true`
- Plantafel (Modul 4) zeigt externe Events des zugewiesenen Mitarbeiters als „belegt"
- Termine im Customer-Portal (Modul 15) werden nicht direkt angezeigt, aber Booking-Slots berücksichtigen sie

---

## Block C — OAuth2-Connect-Flow

### Google (Provider 1)

**Scopes (Phase 1):**
- `https://www.googleapis.com/auth/calendar` (Read/Write)
- `https://www.googleapis.com/auth/calendar.events` (Events Read/Write)
- `openid email profile` (für `external_account_email`)

**Flow-Schritte:**

1. User klickt „Google Calendar verbinden" auf Settings-View
2. App leitet zu Google-Consent-Screen mit `state`-Token (CSRF-Schutz, mit `tenant_id + user_id` signiert)
3. User autorisiert, Google leitet zurück zu `/google/callback?code=...&state=...`
4. Server validiert `state`, tauscht `code` gegen Tokens via Google-Token-Endpoint
5. Server holt User-Info (`external_account_email`, `external_account_id`)
6. `CalendarConnection` wird angelegt (oder bei Re-Connect aktualisiert)
7. Initial-Discovery: Liste aller Kalender → User wählt welche synchronisiert werden (Default: Primary)
8. Initial-Sync (Full-Pull der gewählten Kalender; siehe Block D)
9. Webhook-Subscription (Google Calendar Push Notifications) wird angelegt
10. UI zeigt: *„Verbunden mit {email}. Letzte Synchronisation: gerade eben."*

### Microsoft (Provider 2)

**Scopes (Phase 1):**
- `Calendars.ReadWrite`
- `User.Read` (für Email + Profile)
- `offline_access` (für Refresh-Tokens)

**Flow-Schritte:** analog Google, gegen Microsoft-Identity-Platform (`login.microsoftonline.com`).

**Tenant-Setup:** App muss als Multi-Tenant-App in Azure AD registriert sein. Klartext-Hinweis falls User-Org-Admin Consent nicht gegeben hat: *„Dein IT-Admin muss diese Verbindung zuerst genehmigen. [Link zur Admin-Consent-URL]"*

### Disconnect

- User klickt „Verbindung trennen" → Server widerruft Tokens beim Provider, löscht Webhook-Subscription, setzt `CalendarConnection.status = revoked`
- Optional: alle synchronisierten externen Events aus HiveGrid entfernen (User-Wahl mit Klartext-Erklärung)
- Mappings (für interne Events mit externer Spiegelung) werden gelöscht; interne Events bleiben in HiveGrid

### Re-Connect

- Bei `status = error` (z.B. Token revoked, Scope reduziert): Klartext-Banner mit „Erneut verbinden"-Funktion
- Token-Refresh läuft automatisch im Hintergrund; bei Fehler → `status = error` + Email-Notification an User

### Token-Storage

- Refresh-Tokens verschlüsselt at-rest (envelope encryption mit Tenant-spezifischem Schlüssel via KMS)
- Access-Tokens nur kurz gehalten, Re-Fetch bei Bedarf
- Logging niemals den Token-Klartext

---

## Block D — Sync-Richtungen und Mappings

### Richtung 1: HiveGrid → externer Kalender (bidirektional, Schreib-Pfad)

**Welche HiveGrid-Events werden gespiegelt:**

- **Booking-Termine** (Modul 6): wenn ein Booking gegen einen User mit aktiver `CalendarConnection` gebucht wird, wird automatisch ein Spiegel-Event im externen Kalender erstellt
- **Projekt-Termine** (Modul 4): User-konfigurierbar pro Projekt-Termin („In meinen Kalender eintragen" Toggle)
- **Plantafel-Einträge** (Modul 4): wenn User Plantafel-Einträge zugewiesen bekommt, automatisch oder per User-Setting

- **Tasks mit Fälligkeit** (Modul 26): optional als All-Day-Event; User-Setting Default aus

**Operationen:**

- **Create** in HiveGrid → API-Call zum Provider (Google/MS Insert), `external_event_id` und `etag` werden gespeichert
- **Update** in HiveGrid → Provider-Update-Call mit `If-Match`-Header (Optimistic Locking via etag)
- **Delete** in HiveGrid → Provider-Delete-Call

**Mehrfach-Provider:** wenn User Google + MS gleichzeitig verbunden hat → Spiegel-Event in BEIDEN Kalendern (zwei Mappings pro internem Event)

**Welche User-Felder werden synchronisiert:**

- `title` — von HiveGrid-Event-Titel (z.B. „Kundentermin Schmidt", „Projekt Heizung Schmidt — Montage Tag 2")
- `description` — Kontext-Info + Link zurück zu HiveGrid-Entity (z.B. „Geöffnet in HiveGrid: {url}")
- `location` — Adresse aus Projekt/Kunde
- `starts_at` / `ends_at` / `time_zone`
- `attendees` — bei Booking: Endkunde wird als Attendee eingeladen (mit Provider-eigener Einladungs-Mail; konfigurierbar per Booking-Page-Setting)

## Richtung 2: externer Kalender → HiveGrid (read-only)

**Zweck:** Verfügbarkeits-Berechnung für Booking + Plantafel. **Kein** Schreib-Zugriff auf externe Events außer für die von HiveGrid selbst erstellten (Mappings).

**Operationen:**

- **Initial-Sync:** Full-List der nächsten 60 Tage + vergangenen 7 Tage; danach inkrementeller Sync via Sync-Token
- **Inkrementeller Sync:** über `syncToken` (Google) bzw. `deltaLink` (MS) — Provider liefert nur Änderungen seit letztem Cursor
- **Webhook-getriggert** (siehe Block E): Notification → Worker holt Delta-Liste und aktualisiert `CalendarEvent`-Cache

**Welche externen Events werden gespiegelt:**

- Alle Events der vom User ausgewählten Kalender im Sync-Fenster (rolling window: -7 Tage bis +60 Tage; Phase 2: konfigurierbar)
- Private Events des Users (mit `private`-Markierung) werden gespiegelt aber `title` wird auf „Privat" reduziert; `is_busy_for_availability` bleibt true

**Welche externen Events werden NICHT gespiegelt:**

- Events mit Status „Free" beim Provider (haben `is_busy_for_availability = false`)
- Abgesagte Events
- Events außerhalb des Sync-Fensters

## Sicherheit & Privacy

- Externe Event-Inhalte sind sichtbar **nur** für den User selbst und Tenant-Admin (mit `calendar.admin`); nicht für andere Tenant-User
- Im Booking/Plantafel-Kontext werden externe Events nur als „belegt zwischen X und Y" angezeigt — kein Titel/Inhalt-Leak
- DSGVO: bei Account-Löschung des Users werden alle externen Event-Cache-Einträge hart gelöscht; bei Disconnect optional

---

# Block E — Webhook-Sync (kein Polling)

## Anforderungen

- **Kein Polling** — Provider pushen Notifications via Webhook
- **Google Calendar Push Notifications:** Channel-basiert, eine Subscription pro Kalender, Renewal vor Ablauf
- **Microsoft Graph Subscriptions:** ähnlich, Renewal alle 3 Tage
- **Renewal-Job:** Cron läuft alle 6h, erneuert Subscriptions die in den nächsten 24h ablaufen
- Webhook-Endpoints sind öffentlich erreichbar (keine Auth aus Provider-Sicht), aber:
- Provider-Signature-Verifikation (Google: `X-Goog-Channel-Token`, MS: `clientState`)
- Eindeutige Channel-IDs werden mit `connection_id` gemappt

## Webhook-Verarbeitung

1. Provider sendet POST an `/webhooks/calendar/{provider}` mit Notification
2. Server validiert Signature, schlägt `connection_id` über Channel-/Subscription-ID nach
3. Eintrag in `CalendarWebhookEvent` mit `processing_status = pending`
4. Sofortige 200-OK-Antwort an Provider (Provider-Limit auf <10s Response)
5. Async-Worker holt `pending`-Events und verarbeitet:
   - Inkrementeller Sync via Cursor-Token
   - Updates auf `CalendarEvent` schreiben
   - Bei Updates an Events mit Mapping → Konflikt-Check (siehe Block F)
6. Status-Update auf `completed` oder `failed`

## Idempotenz

- Provider können Notifications duplizieren — Worker prüft `last_modified_external` und skippt wenn bereits verarbeitet
- Bei Cursor-Invalidierung (z.B. nach langem Offline-Sein) → Full-Resync (Provider liefert 410-Gone, App reagiert mit Full-List-Pull)

---

# Block F — Konflikt-Handling

## Last-Write-Wins (LP-7-konform; einfach für Phase 1)

**Szenarien:**

1. **HiveGrid-Update + externes Update zwischen den Syncs:**
   - User editiert Termin in HiveGrid → App pusht zu Provider mit `If-Match: etag`
   - Falls Provider-etag inzwischen geändert (paralleler Edit im Google Calendar UI) → Provider antwortet 412 Precondition Failed
   - App löst Konflikt mit **„externer Stand gewinnt"** als Default (HiveGrid-Edit wird verworfen) UND zeigt Klartext-Notification an User: *„Dein Termin ‚{titel}' wurde gleichzeitig in {provider} geändert — die externe Version wurde übernommen. Du kannst die Änderung manuell wieder einarbeiten. [Original-HiveGrid-Stand anzeigen]"*
   - User hat dann via In-App-Notification die Möglichkeit, manuell nachzuziehen
2. **Externer Termin gelöscht, HiveGrid hat noch Mapping:**
   - Mapping wird entfernt, internes Event bleibt unverändert (z.B. Booking-Termin existiert weiter, externer Spiegel ist weg) — Klartext: *„Der Termin wurde im externen Kalender gelöscht. In HiveGrid bleibt er bestehen. [Im externen Kalender wieder anlegen]"*

3. **HiveGrid-Termin gelöscht:**
   - App pusht Delete an alle gemappten externen Events
   - Falls externe Events bereits geändert/gelöscht → still ignorieren

### Konflikt-Sichtbarkeit im UI

- In `CalendarConnection`-Detail-View: Liste der letzten 50 Konflikte mit Klartext-Beschreibung und Resolution
- In Notifications (Modul 22): bei wichtigen Konflikten Push/Email an User

### Optimistic Locking auf HiveGrid-Seite

- Interne Events haben `version_token` (Increment bei jedem Update)
- Mapping speichert `last_known_internal_version_token` — Worker prüft, dass Webhook-Updates nicht inzwischen veraltete Stände schreiben

---

## Block G — Adapter-Pattern (CalendarAdapter-Interface)

### Anforderungen (LP-11, AV #2)

Gemeinsames Interface für Google + Microsoft (und ggf. Phase-2-Provider wie Apple/iCloud, FastMail):

- `initiateOAuth(redirect_url) → authorizationUrl`
- `handleOAuthCallback(code, state) → { access_token, refresh_token, account_id, account_email, scopes }`
- `refreshAccessToken(connection) → { access_token, expires_at }`
- `listCalendars(connection) → Calendar[]`
- `subscribeWebhook(connection, calendar_id) → { subscription_id, expires_at }`
- `unsubscribeWebhook(connection, subscription_id) → void`
- `renewWebhook(connection, subscription_id) → { new_subscription_id, expires_at }`
- `fetchEventsInitial(connection, calendar_id, time_window) → { events[], cursor_token }`
- `fetchEventsDelta(connection, calendar_id, cursor_token) → { events[], next_cursor_token, full_resync_required }`
- `createEvent(connection, calendar_id, event_data) → { external_event_id, etag }`
- `updateEvent(connection, calendar_id, external_event_id, event_data, if_match_etag) → { etag }` (412 wirft `ConflictError`)
- `deleteEvent(connection, calendar_id, external_event_id, if_match_etag) → void`
- `verifyWebhookSignature(headers, body) → boolean`

### Klartext-Diagnostik pro Adapter (LP-10)

Adapter müssen Provider-spezifische Fehler in einheitliche Klartext-Codes übersetzen:

- `auth_token_expired` → *„Verbindung zu {provider} ist abgelaufen. Bitte neu verbinden."*
- `auth_scope_missing` → *„Berechtigung im {provider}-Konto wurde reduziert. Bitte neu verbinden."*
- `rate_limit_exceeded` → *„{provider} bremst gerade unsere Anfragen — wir versuchen es automatisch nochmal in {minutes} Minuten."*
- `calendar_not_found` → *„Der ausgewählte Kalender existiert nicht mehr im externen Konto."*
- `webhook_subscription_failed` → *„Push-Benachrichtigungen von {provider} sind gerade gestört. Synchronisation läuft mit Verzögerung."*
- `provider_outage` → *„{provider} ist gerade nicht erreichbar — Synchronisation pausiert."*

---

## Block H — Verfügbarkeits-Berechnung (Booking-Integration)

### Anforderungen

- Booking-Page (Modul 6) zeigt verfügbare Slots eines Users basierend auf:
- Working-Hours des Users (Modul 6 Settings)
- HiveGrid-eigene Termine des Users ( `source = hivegrid_internal` )
- Externe Events mit `is_busy_for_availability = true`
- Plantafel-Einträge (Modul 4)
- Verfügbarkeits-Query: gegeben User + Zeitraum + Slot-Dauer → Liste freier Slots
- Performance: Slots werden für die nächsten 30 Tage gecacht; Cache-Invalidierung bei jedem CalendarEvent-Update

### Anti-Doppel-Buchung

- Wenn ein externer Termin während eines reservierten Booking-Slots erscheint (Sync-Verzögerung): Klartext-Notification an Booker mit Re-Booking-Vorschlag
- Realistisch nur bei Sync-Latenz > Booking-Duration; Phase-1-Toleranz akzeptabel

### Slot-Filterung

- Booking-Page lädt nicht direkt aus `CalendarEvent` (DSGVO/Privacy), sondern erhält nur `boolean is_available`-Antwort pro Slot
- Endkunde sieht nie Titel/Inhalt externer Events

---

## Block I — Permissions-Anwendung

### Permission-IDs

| Permission | Erlaubt |
|---|---|
| `calendar.connect_own` | Eigenen Kalender verbinden, eigene Connection verwalten |
| `calendar.admin` | Alle Connections im Tenant sehen, gestörte Verbindungen anzeigen, Auto-Disconnect bei User-Offboarding |

### Bundle-Mapping

| Bundle | Atomare Permissions |
|---|---|
| `calendar.none` | (keine — Calendar-Sync nicht verfügbar) |
| `calendar.connect_own` | `calendar.connect_own` |
| `calendar.admin` | `calendar.connect_own` + `calendar.admin` |

**Default:** `calendar.connect_own` für alle Tenant-User. Calendar-Sync ist user-spezifisch — keine Tenant-weite Zentral-Connection in Phase 1.

### Anwendung in Routes

- `/{tenant}/settings/integrations/calendar/` → `calendar.connect_own`
- `/{tenant}/settings/integrations/calendar/:connection_id` → `calendar.connect_own` (Owner-Check: `connection.user_id = self` ) ODER `calendar.admin`
- `/{tenant}/admin/integrations/calendar/` (Phase 2 Tenant-Übersicht) → `calendar.admin`

- Webhook-Endpoints `/webhooks/calendar/{provider}` → keine User-Auth (Provider-Signature-Verifikation)

### Anwendung in Action-Funktionen

- Connect / Disconnect: nur eigene Connections (`calendar.connect_own`); Tenant-Admin kann fremde Connections bei User-Offboarding zwangs-trennen (`calendar.admin`)
- Sync-Settings ändern (Direction, ausgewählte Kalender): `calendar.connect_own` mit Owner-Check
- Sync-Logs einsehen: nur eigene; Tenant-Admin sieht aggregierte Statistiken (kein Event-Inhalt-Zugriff)

### Off-Boarding (Modul 27, D.2)

- Bei User-Deaktivierung → automatisch alle `CalendarConnection` des Users auf `paused`, Webhooks abgemeldet
- Bei User-Hard-Delete → Connections und alle externen Event-Caches hart löschen

---

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

- **Verbindung abgelaufen:** *„Verbindung zu {provider} ist abgelaufen. Bitte neu verbinden, damit deine Termine weiter synchronisiert werden. [Neu verbinden]"*
- **Scope reduziert:** *„Du hast HiveGrid die Kalender-Berechtigung im {provider}-Konto entzogen. Synchronisation pausiert. [Neu verbinden]"*
- **Provider-Outage:** *„{provider} ist gerade nicht erreichbar — wir holen die letzten Termine sobald die Verbindung wieder steht. Letzte erfolgreiche Synchronisation: {time}."*
- **Rate-Limit:** *„{provider} bremst unsere Anfragen — Synchronisation läuft langsamer. Termine erscheinen mit etwas Verzögerung."*
- **Mehrfach-Account-Hinweis:** falls User dasselbe Provider-Konto bereits verbunden hat: *„{email} ist bereits verbunden. Stattdessen die existierende Verbindung anpassen?"*
- **Konflikt:** *„Termin ‚{titel}' wurde gleichzeitig in {provider} geändert — die externe Version wurde übernommen. [Original-Stand anzeigen]"*
- **Webhook gestört:** *„Push-Benachrichtigungen von {provider} sind gerade nicht aktiv. Synchronisation läuft alle 30 Minuten als Fallback."* (kein automatisches Polling normal — nur als kurzfristiger Fallback)
- **MS-Admin-Consent fehlt:** *„Dein IT-Admin muss diese Verbindung im {provider}-Tenant zuerst genehmigen. [Admin-Consent-Link kopieren]"*
- **Disconnect-Bestätigung:** *„Verbindung zu {email} wird getrennt. {n} synchronisierte Termine bleiben in HiveGrid. Auch in {provider} entfernen? [Nur trennen] [Trennen + extern entfernen]"*
- **Initial-Sync läuft:** *„Synchronisation läuft — kann beim ersten Mal ein paar Minuten dauern."*

---

## Activity-Stream-Integration

Calendar-Sync erzeugt selektiv Activities (Modul 1, Block E):

- `calendar_connected` (subject = User)
- `calendar_disconnected` (subject = User)
- `calendar_sync_error` (subject = User; bei wiederholten Fehlern)
- `appointment_synced_external` (subject = Booking/Project; metadata: `provider`, `external_calendar`)

**Bewusst NICHT** als Activity: jedes einzelne externe Event-Update (Spam-Vermeidung).

---

## Phase 2 / spätere Features

- **iCal-Feed** (read-only Feed-URL für Kalender-Apps, die keine OAuth-Anbindung haben)
- **Resource-Calendar** (Kran, Werkstatt-Heber, Pool-Fahrzeuge als buchbare Resource — eigener Kalender-Typ)
- **Tenant-Zentral-Connection** für Resource-Kalender (statt User-spezifisch)
- **Apple/iCloud-Calendar** (CalDAV-Adapter) — Trade-off Aufwand vs. Nachfrage
- **FastMail** (CalDAV)
- **Konfigurierbares Sync-Fenster** (statt fest -7/+60 Tage)
- **Multi-Calendar pro User** (User wählt mehrere externe Kalender pro Verbindung — Phase 1 erlaubt nur Default-Setup, Phase 2 macht pro-Calendar-Settings sichtbar)
- **Calendar-Color-Sync**
- **Smart-Conflict-Resolution mit KI-Vorschlag** (statt nur Last-Write-Wins)
- **Time-Zone-Awareness erweitert** (User in unterschiedlichen Zeitzonen)
- **Recurring-Event-Edits** (Edit nur dieser einen Instanz vs. Edit der Serie) — Komplex; Phase 2 ausgereifter

---

## Architektur-Vorschriften (Querschnitt)

- **LP-7:** Calendar-Sync löst teilweise Google Calendar Standalone-Nutzung ab — Geerkens-Kontext
- **LP-9, AV #1:** Klartext-Diagnostik pro Provider-Fehler (Block J)
- **LP-11, AV #2:** Adapter-Pattern für Google + Microsoft (Block G)
- **LP-13, AV #12:** EU-Stack — Tokens und Caches in EU-Region; Provider-APIs sind extern (Google/MS), das ist akzeptiert (DSGVO-konforme DPAs liegen vor)
- **AV #5:** Permissions-aware-Default — externe Event-Inhalte nie an andere User leaken
- **AV #6:** Audit-Trail-Default für Connect/Disconnect/Konflikte
- **AV #15:** Optimistic Locking + Last-Write-Wins-mit-Sichtbarkeit für Sync-Konflikte (Block F)
- **LP-1:** User-Connection mit `tenant_id` für Trennung; OAuth-State signiert mit Tenant-Kontext

---

## Abhängige Module

- **Modul 6 (Booking):** liest Verfügbarkeit aus `CalendarEvent`; pusht neue Bookings als interne Events
- **Modul 4 (Projekt-/Plantafel):** pusht Plantafel-Einträge in externen Kalender (User-Setting)
- **Modul 26 (Tasks):** optionale Synchronisation (User-Setting)
- **Modul 22 (Notifications):** zeigt Konflikte und Verbindungsfehler
- **Modul 27 (Auth):** Off-Boarding-Hook (Pause/Trennung bei User-Deaktivierung)
- **Modul 33a (Apps Engine):** stellt Adapter-Pattern-Infrastruktur bereit (LP-11)
- **Modul 21 (Audit-Log):** Connect/Disconnect/Konflikte werden geloggt

---

## Detail-Spec-Status: DRAFT

Erste vollständige Detail-Spec für Modul 25. **Vor Bau:** Verifizierung mit Noah und Abgleich mit Modul 6 (Booking-Verfügbarkeits-Spec) und Modul 33a (Apps-Engine-Adapter-Pattern).

**Offene Detail-Punkte (nicht-blockierend):**

- Konkrete Sync-Fenster-Defaults (-7/+60 Tage) mit Geerkens-Workflow validieren
- Mehrfach-Verbindung pro User (Geschäftlich + Privat?) — Default-Verhalten

- Recurring-Event-Updates: Edit-Single-Instance vs. Edit-Series — minimaler Phase-1-Scope
- All-Day-Events Behandlung über Zeitzonen
- Konkretes Webhook-Renewal-Timing (Google: max 1 Woche; MS: max ~3 Tage) — Cron-Frequenz
- Microsoft-Multi-Tenant-App-Setup mit Azure AD (Admin-Consent-Flow im Detail)
- Klartext-Wording-Review für Konflikt-Notifications mit Noah/Sandra (Verständlichkeit für Endkunden-Bezug)
- Encryption-Strategie für Refresh-Tokens (KMS-Provider, Tenant-Schlüssel-Setup)