```
module: 6
title: Booking / Termin-Buchung (Calendly-Style)
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-4, LP-7, LP-9, LP-10]
spec_style: requirements-pur
```

Modul 6 — Booking / Termin-Buchung

> ***Detail-Spec gemäß LP-9.*** *Calendly-Style öffentliche Buchungs-Seiten pro Tenant. Löst HubSpot-Booking-Tool ab (LP-7). Multi-Provider, Verfügbarkeits-Sync mit Google/MS-Kalender, Embed-Code für externe Websites, Auto-Bestätigungsmail, Auto-Eintrag in internen Kalender. Buchungen erzeugen Leads (neuer Kunde) oder Activities (Bestandskunde).*

Themenblöcke

```
A. Datenmodell
B. Menü-Struktur und Routes
C. Listen-View / Übersicht aller Buchungen
D. Detail-View / Einzelne Buchung
E. Booking-Page-Verwaltung (Editor)
F. Verfügbarkeits-Verwaltung (Pro-User)
G. Öffentliche Booking-Page (Endkunden-Flow)
H. Embed-Code / iFrame
I. Auto-Bestätigungsmail
J. Verknüpfungen (CRM, Kalender, Lead-Inbox, Activity-Stream)
K. Permissions
L. Klartext-Hinweise
M. Phase 2
```

Block A — Datenmodell

A.1 — BookingPage

Tabelle: `BookingPage`. Tenant-scoped (RLS).

Felder:
- `id`, `tenant_id`, `created_at`, `updated_at`, `created_by_user_id`
- `slug` (Pflicht, unique pro Tenant; URL-safe `[a-z0-9-]`, min. 3, max. 60 Zeichen — z.B. `wartungstermin`)
- `title` (Pflicht, User-sichtbarer Name z.B. „Wartungstermin Beratung")
- `description` (Markdown-Freitext, wird auf öffentlicher Page angezeigt)
- `duration_minutes` (Pflicht, Enum: `15 | 30 | 45 | 60 | 90 | 120`)
- `buffer_before_minutes` (Default 0; Pufferzeit vor Termin)
- `buffer_after_minutes` (Default 0; Pufferzeit nach Termin)
- `slot_increment_minutes` (Default 15; in welchen Schritten Slots angeboten werden)
- `min_notice_hours` (Default 4; minimaler Vorlauf für Buchung)
- `max_advance_days` (Default 60; wie weit in die Zukunft buchbar)
- `assignment_strategy` (Enum: `round_robin | specific_provider_choice | first_available`; siehe A.2)

- `confirmation_email_subject` (Pflicht, Liquid/Handlebars-Template-Strings erlaubt)
- `confirmation_email_body` (Pflicht, Markdown + Template-Strings, siehe Block I)
- `internal_calendar_event_title` (Template-String für Kalender-Eintrag-Titel; Default „Termin: {{contact.name}} — {{booking_page.title}}")
- `redirect_url_after_booking` (optional, externe URL z.B. Tenant-Website-Danke-Page)
- `intake_form_id` (FK auf Form aus Modul 5, optional — wenn gesetzt wird dieses Form als Intake genommen statt Standard-Mini-Form)
- `is_published` (Boolean — nur publizierte Pages sind öffentlich erreichbar)
- `archived_at`

Indizes:
- `(tenant_id, slug)` UNIQUE
- `(tenant_id, archived_at, is_published)` für Liste

A.2 — BookingPageProvider

M:N-Tabelle zwischen `BookingPage` und `User` (Tenant-User). Eine Booking-Page hat einen oder mehrere Provider (Mitarbeiter, deren Verfügbarkeit angezeigt wird).

Felder:
- `booking_page_id`, `tenant_user_id`, `tenant_id`
- `display_order` (für UI-Reihenfolge bei `specific_provider_choice`)
- `is_active` (Provider zeigt aktuell Verfügbarkeit; falls deaktiviert: bestehende Termine bleiben, neue Buchungen schließen ihn aus)

Assignment-Strategien:

Strategie	Verhalten
`round_robin`	System wählt Provider mit den wenigsten Buchungen in den letzten 30 Tagen aus den verfügbaren
`specific_provider_choice`	Endkunde sieht Provider-Auswahl auf öffentlicher Page, wählt selbst
`first_available`	Slot-Picker zeigt zusammengeführte Verfügbarkeit; bei Buchung wird dem Provider zugewiesen, der zuerst frei ist

A.3 — AvailabilityRule

Tabelle: `AvailabilityRule`. Pro Tenant-User. Definiert wöchentliche Standard-Verfügbarkeit.

Felder:
- `id`, `tenant_id`, `tenant_user_id`, `created_at`, `updated_at`
- `day_of_week` (Enum: `monday | tuesday | ... | sunday`)
- `start_time` (Time, z.B. `09:00`)
- `end_time` (Time, z.B. `17:00`)
- `timezone` (IANA, z.B. `Europe/Berlin`; Default aus User-Profil)

Sonderfall: mehrere Regeln pro Tag möglich (z.B. 09:00-12:00 + 14:00-17:00 mit Mittagspause).

Indizes:
- `(tenant_user_id, day_of_week)`

A.4 — AvailabilityException

Tabelle: `AvailabilityException`. Datum-spezifische Overrides der Standard-Regeln.

Felder:
- `id`, `tenant_id`, `tenant_user_id`, `created_at`, `updated_at`
- `date` (Pflicht)

- `kind` (Enum: `unavailable | custom_window`)
- `start_time`, `end_time` (nur wenn `kind = custom_window`; nullable bei `unavailable`)
- `reason` (optional, Freitext z.B. „Urlaub", „Schulung")
- `timezone` (IANA, Default wie Regel)

Sonderfälle:
- `unavailable` ohne Zeiten: ganzer Tag blockiert
- `custom_window`: ersetzt Standard-Regel für diesen Tag (mehrere Exceptions pro Tag erlaubt für mehrere Fenster)

Indizes:
- `(tenant_user_id, date)`

A.5 — Appointment

Tabelle: `Appointment`. Tenant-scoped.

Felder:
- `id`, `tenant_id`, `created_at`, `updated_at`
- `booking_page_id` (FK BookingPage)
- `provider_user_id` (FK TenantUser — wer hält den Termin)
- `start_at` (Pflicht, UTC)
- `end_at` (Pflicht, UTC; berechnet aus `start_at + duration_minutes`)
- `timezone` (IANA, in welcher Zeitzone der Endkunde gebucht hat — für Anzeige)
- `status` (Enum: `scheduled | held | cancelled | no_show | rescheduled`)
- `contact_id` (FK Contact, nullable — gesetzt sobald gematcht/promoted)
- `lead_id` (FK Lead, nullable — bei Buchung von neuem Endkunden zunächst Lead)
- `form_submission_id` (FK FormSubmission — die Intake-Daten der Buchung)
- `external_calendar_event_ids` (JSON-Array mit Provider-Calendar-Event-IDs aus Google/MS — siehe Block J)
- `cancelled_at`, `cancelled_by_kind` (Enum: `customer | provider | system`), `cancellation_reason`
- `rescheduled_from_id` (FK Appointment, nullable — bei Reschedule)
- `notes_internal` (Markdown, nur intern sichtbar)

Status-Lifecycle:

```
scheduled    → gebucht, in Zukunft
held         → vergangener Termin, fand statt
cancelled    → storniert (Kunde oder Provider)
no_show      → Kunde nicht erschienen (Provider markiert manuell)
rescheduled  → durch neueren Termin ersetzt (alter bleibt zur Historie)
```

Indizes:
- `(tenant_id, provider_user_id, start_at)`
- `(tenant_id, contact_id, start_at)`
- `(tenant_id, status, start_at)` für Übersichten
- `(booking_page_id, start_at)`

A.6 — Slot-Berechnungs-Logik (zur Laufzeit, kein eigenes Modell)

Bei Anfrage „verfügbare Slots am Datum X für Booking-Page Y":

1. Lade alle aktiven Provider der Booking-Page
2. Pro Provider:
   - Standard-Verfügbarkeit aus `AvailabilityRule` für `day_of_week` von X
   - Überlagern mit `AvailabilityException` für Datum X (falls vorhanden)
   - Subtrahieren: bestehende `Appointment`-Records (mit Buffer-Zeiten) für Provider in Zeitraum
   - Subtrahieren: externe Kalender-Events aus `CalendarEvent`-Tabelle (Modul 25) wo `provider_user_id` der Provider ist und Zeitraum überlappt

- Subtrahieren: `min_notice_hours` (keine Slots in den nächsten N Stunden)
- Subtrahieren: alles außerhalb `max_advance_days`

3. Slot-Increment anwenden: nur Slots an Vielfachen von `slot_increment_minutes`
4. Nach `assignment_strategy` zusammenführen:
   - `round_robin` / `first_available`: Union aller Provider-Slots, jeder Slot mit „Provider X frei"-Marker
   - `specific_provider_choice`: pro Provider getrennt zurückgeben

Performance:
- Slot-Berechnung server-side, gecacht für 60 Sekunden pro `(booking_page_id, date)` mit Redis
- Cache-Invalidation bei: neue/gelöschte Appointment, neue/geänderte AvailabilityRule/Exception, neue Calendar-Event-Sync

---

Block B — Menü-Struktur und Routes

Sidebar-Hauptpunkt: *Buchungen*. Sub-Punkte: Übersicht / Booking-Pages / Meine Verfügbarkeit.

Routen (Tenant-intern, eingeloggt):

```
/{tenant}/bookings                          → Übersicht aller Buchungen (Liste)
/{tenant}/bookings/:id                      → Einzelne Buchung Detail-View
/{tenant}/bookings/pages/                   → Liste aller Booking-Pages
/{tenant}/bookings/pages/new                → Neue Booking-Page
/{tenant}/bookings/pages/:id                → Booking-Page editieren
/{tenant}/availability                      → Eigene Verfügbarkeit pflegen (pro User)
```

Routen (öffentlich, kein Login):

```
app.hivegrid.de/buch/{tenant}/{slug}        → Öffentliche Booking-Page
app.hivegrid.de/buch/{tenant}/{slug}/embed  → Embed-Variante (iFrame, ohne Tenant-Branding-Header)
```

Cancel-/Reschedule-Tokens (Phase 2): Phase 1 noch nicht im Scope; Cancel nur intern + via Antwort auf Bestätigungsmail.

Block C — Listen-View / Übersicht aller Buchungen (`/{tenant}/bookings`)

Zu zeigende Daten pro Eintrag:
- Datum + Uhrzeit (`start_at` in Tenant-User-Timezone)
- Dauer
- Endkunden-Name (aus Contact oder Lead)
- Endkunden-Email
- Booking-Page-Title
- Provider (Tenant-User der den Termin hält)
- Status

Benötigte UI-Elemente:
- Tabellarische Liste, sortierbar
- Default-Sortierung: `start_at ASC` für `status = scheduled`, dann historisch nach Datum
- Filter:
- Status (Multi-Select)
- Provider (Multi-Select)
- Booking-Page (Multi-Select)
- Zeitraum (Date-Range-Picker)

- Volltext-Suche über Endkunden-Name/-Email
- Tabs/Toggle: „Anstehende" / „Vergangene" / „Stornierte" / „Alle"
- Pagination (Cursor-basiert, 50 pro Page)
- Bulk-Auswahl + Bulk-Cancel
- Spalten-Konfiguration (User wählt sichtbare Spalten)

Operationen:
- Klick auf Zeile → Detail-View
- Bulk-Cancel (mit Klartext-Confirm und automatischer Cancel-Mail)
- Export als CSV (gefilterte Auswahl)

Permissions-Wirkung:
- `bookings.read_own`: nur Termine wo `provider_user_id = self`
- `bookings.read_all`: alle Termine

Performance:
- Index auf `(tenant_id, status, start_at)` für Default-Liste
- Server-side-Filter

Block D — Detail-View / Einzelne Buchung (`/{tenant}/bookings/:id`)

Zu zeigende Daten:
- Termin-Details (Datum, Uhrzeit, Dauer, Status, Booking-Page-Title)
- Provider (User-Avatar + Name)
- Endkunde (Contact- oder Lead-Verlinkung mit Stammdaten-Vorschau)
- Form-Submission-Daten (Antworten aus Intake-Form)
- Cancellation-Info (falls cancelled): wer, wann, warum
- Reschedule-Info (falls reschedules)
- Externe-Kalender-Sync-Status (synced/pending/error pro Provider-Kalender)
- Notes (intern)
- Activity-Stream-Sektion (Termin-bezogene Activities)

Benötigte UI-Elemente:
- Sektionierte Anzeige der Termin-Daten + Endkunden-Verknüpfung + Form-Antworten + Notes + Activity-Stream
- Inline-Edit für Notes
- Action-Buttons für die Operationen (siehe unten)
- Kalender-Sync-Status-Badge mit Klartext-Tooltip bei Fehler
- Direkt-Link zum verknüpften Contact/Lead

Operationen:
- Status setzen: `held` (manuell als „fand statt"), `no_show` (Kunde nicht erschienen)
- Cancel (mit Cancellation-Reason-Eingabe + Auto-Cancel-Mail an Endkunde)
- Reschedule → öffnet Slot-Picker, wählt neuen Slot, alter Termin bekommt `status=rescheduled`, neuer Termin verlinkt rückwärts mit `rescheduled_from_id`
- Notiz hinzufügen (intern)
- Provider neu zuweisen (mit Klartext-Warnung über Kalender-Resync)
- Hard-Delete (nur `bookings.admin`, mit Klartext-Warnung)
- Endkunde re-/an-mailen (öffnet Compose mit vorausgefülltem Empfänger)

Klartext-Hinweise:
- Provider-Wechsel: *„Termin wird im Kalender von [alter Provider] entfernt und bei [neuer Provider] eingetragen. Endkunde bekommt automatisch eine Update-Mail."*
- Cancel: *„Endkunde bekommt eine Storno-Mail mit Begründung."*
- Reschedule: *„Termin wird verschoben. Endkunde bekommt eine Update-Mail mit neuen Slot."*
- Kalender-Sync-Fehler: *„Konnte Termin nicht in Google-Kalender von [Provider] eintragen ({reason}). [Erneut versuchen]"*

Block E — Booking-Page-Verwaltung

E.1 — Listen-View (`/{tenant}/bookings/pages/`)

Zu zeigende Daten pro Booking-Page:
- Title, Slug
- Anzahl aktive Provider
- Dauer
- Anzahl Buchungen (gesamt + letzte 30 Tage)
- Status (publiziert / nicht publiziert)
- Public-URL mit Copy-to-Clipboard

Benötigte UI-Elemente:
- Liste mit Suchfunktion + Filter (publiziert/nicht/archived)
- Action: neue Page anlegen
- Pro Page: Edit-Funktion, Duplizieren, Archivieren, Hard-Delete

E.2 — Editor (`/{tenant}/bookings/pages/:id` und `/new`)

Zu zeigende Daten:
- Alle Felder von `BookingPage`
- Provider-Auswahl (Multi-Select aus aktiven Tenant-Usern)
- Vorschau-Funktion (öffnet öffentliche Page in neuem Tab)
- Embed-Code-Generator (siehe Block H)

Benötigte UI-Elemente:
- Sektionierte Anzeige für: Grunddaten / Verfügbarkeits-Konfiguration / Provider / Bestätigungsmail / Erweitert
- Slug-Eingabe mit Live-Validation (eindeutigkeit + Reserved-Words-Check, Auto-Slug-Vorschlag aus Title)
- Markdown-Editor für Beschreibung + Bestätigungsmail-Body
- Multi-Select für Provider mit Drag&Drop für `display_order`
- Intake-Form-Auswahl (Dropdown mit allen Forms aus Modul 5, oder „Standard-Mini-Form" als Default)
- Toggle für Publish-Status
- Public-URL-Anzeige (sobald publiziert)
- Embed-Code-Generator (siehe Block H)

Operationen:
- Anlegen / Editieren / Speichern (mit Concurrent-Update-Check)
- Vorschau öffnen
- Embed-Code generieren
- Duplizieren (mit Suffix „(Kopie)")
- Archivieren / Reaktivieren
- Hard-Delete (mit Klartext über bestehende Termine)

Validierungen:
- Slug eindeutig pro Tenant + URL-safe
- Mindestens 1 aktiver Provider
- Standard-Verfügbarkeits-Regel auf mindestens 1 Provider vorhanden, sonst Klartext-Warnung *„Kein Provider hat Verfügbarkeit hinterlegt — Endkunden würden keine freien Slots sehen."*
- `min_notice_hours < max_advance_days * 24` (sonst Klartext-Fehler)

Klartext-Fehler:
- Slug-Konflikt: *„Diese URL ist schon vergeben — wähle eine andere."*
- Reserved-Slug: *„'{slug}' ist reserviert. Wähle einen anderen."*
- Booking-Page hard-löschen mit zukünftigen Terminen: *„Diese Booking-Page hat noch {n} anstehende Termine. Beim Löschen werden diese storniert. Stattdessen archivieren? [Archivieren] [Trotzdem löschen]"*

Permissions:
- Anlegen/Editieren: `bookings.admin`
- Vorschau lesen: `bookings.read_*`

Block F — Verfügbarkeits-Verwaltung (`/{tenant}/availability`)

Pro-User-Page (jeder Tenant-User pflegt seine eigene). Kein Admin-Recht zum Editieren fremder Verfügbarkeiten in Phase 1 (außer über Direct-API als `bookings.admin`).

Zu zeigende Daten:
- Aktuelle wöchentliche Standard-Regeln (`AvailabilityRule`-Liste)
- Aktuelle Datums-Ausnahmen (`AvailabilityException`-Liste, ab heute aufwärts)
- Hinweis: welche Booking-Pages aktuell auf diesen User zugreifen (read-only Information)

Benötigte UI-Elemente:
- Wöchentliche Verfügbarkeits-Editor (pro Wochentag mehrere Zeit-Fenster definierbar mit Start/End)
- Ausnahmen-Editor (Datum + Toggle „nicht verfügbar" oder Custom-Window mit Zeit-Fenster)
- Timezone-Auswahl (Default aus User-Profil, änderbar pro Regel)
- Liste der Ausnahmen mit Filter „nur künftige" und Edit/Delete pro Eintrag
- Klartext-Hinweis bei leerer Standard-Verfügbarkeit *„Du bist auf keiner Booking-Page als verfügbar markiert — du kannst keine Termine bekommen."*

Operationen:
- Standard-Regeln anlegen / editieren / löschen
- Ausnahmen anlegen / editieren / löschen
- Bulk-Aktion „Diese Woche bin ich nicht verfügbar" (erzeugt 7 Exceptions mit `kind=unavailable`)

Permissions:
- Eigene: jeder eingeloggte Tenant-User mit `bookings.write` oder `bookings.read_own`
- Fremde: nur `bookings.admin` (für Backoffice-Korrektur)

Klartext-Hinweise:
- Überlappende Zeit-Fenster: *„Diese Fenster überschneiden sich — wir haben sie zusammengeführt: 09:00-17:00."* (auto-merge mit Hinweis)
- Zeit-Fenster außerhalb des Tages: *„Endzeit muss nach Startzeit liegen."*
- Exception in Vergangenheit: *„Datum liegt in der Vergangenheit — Eintrag wird gespeichert, hat aber keine Wirkung mehr."*

Block G — Öffentliche Booking-Page (`app.hivegrid.de/buch/{tenant}/{slug}`)

G.1 — Endkunden-Flow

Schritt 1 — Page-Anzeige:
- Title + Beschreibung der Booking-Page
- Dauer
- Bei `assignment_strategy = specific_provider_choice`: Provider-Auswahl mit Avatar + Name (+ optional Kurz-Bio in Phase 2)
- Bei anderen Strategien: nur Booking-Page-Titel ohne Provider-Auswahl
- Slot-Picker:
- Datum-Auswahl (Kalender-Komponente, Tage ohne Verfügbarkeit ausgegraut)
- Zeitslots am gewählten Tag (Liste mit Uhrzeiten in Endkunden-Timezone)
- Timezone-Anzeige + Wechsler (Default aus Browser-Locale)

Schritt 2 — Intake-Form:
- Pflicht-Felder Standard-Mini-Form: Vor-/Nachname, Email, Telefon (optional)
- Falls `intake_form_id` gesetzt: das verknüpfte Form aus Modul 5 wird angezeigt mit allen Custom-Feldern
- DSGVO-Hinweis und Daten-Verarbeitungs-Akzeptanz (Pflicht-Checkbox)

Schritt 3 — Bestätigung:
- Klartext-Bestätigung *„Termin {{title}} am {{date}} um {{time}} mit {{provider}}"*
- Hinweis dass Bestätigungsmail unterwegs ist
- Optional: Redirect auf `redirect_url_after_booking` falls gesetzt
- Add-to-Calendar-Funktion (.ics-Download für Endkunde)

G.2 — Endkunden-UI-Anforderungen

Benötigte UI-Elemente:
- Multi-Step-Flow (Slot wählen → Daten eingeben → Bestätigung)
- Kalender-Komponente mit Verfügbarkeits-Indikator pro Tag
- Timezone-Wechsler
- Form-Renderer für Intake-Form (analog Modul 5 Renderer)
- DSGVO-Checkbox + verlinkte Datenschutzerklärung
- Loading-States während Slot-Refresh und Submit
- Error-States mit Klartext (siehe G.4)

G.3 — Konkurrenz-/Race-Condition

- Beim finalen Submit prüft Server erneut Slot-Verfügbarkeit (Slots können zwischen Anzeige und Submit von anderen gebucht werden)
- Bei Konflikt: Klartext *„Ohje, dieser Slot wurde gerade gebucht. Hier sind aktuelle Alternativen: ..."* mit aktualisierter Slot-Liste
- DB-Level: Unique-Constraint via `EXCLUDE USING gist`-Range-Constraint pro Provider verhindert echte Doppelbuchung

G.4 — Klartext-Hinweise

- Booking-Page nicht publiziert: 404 mit Klartext *„Diese Buchungsseite gibt's nicht (mehr)."*
- Tenant inaktiv: 404 mit gleichem Klartext (keine Tenant-Existenz preisgeben)
- Keine freien Slots: *„Aktuell sind alle Termine belegt — schreib uns direkt an [Tenant-Email] oder probier später nochmal."*
- Submit-Fehler 5xx: *„Buchung fehlgeschlagen — bitte erneut versuchen oder direkt anrufen."*
- DSGVO-Checkbox nicht gesetzt: *„Bitte akzeptiere die Datenverarbeitungs-Hinweise um zu buchen."*

Block H — Embed-Code

H.1 — iFrame-Variante

Im Booking-Page-Editor generierter Code:

```
<iframe
  src="https://app.hivegrid.de/buch/{tenant}/{slug}/embed"
  width="100%"
  height="700"
  frameborder="0"
  style="border:none">
</iframe>
```

Embed-spezifisch:
- Ohne Tenant-Branding-Header (kein Logo, kein Footer)
- Responsive
- `X-Frame-Options: ALLOWALL` für `/embed`-Route (CSP-konfiguriert)

H.2 — JS-Snippet-Variante (nice-to-have Phase 1, sonst Phase 2)

Lightweight-Loader für Pop-up-Buchung:

```
<script src="https://app.hivegrid.de/embed/booking.js"></script>
<button data-hivegrid-booking="{tenant}/{slug}">Termin buchen</button>
```

Phase-1-Empfehlung: nur iFrame. JS-Snippet als Phase 2 oder als optionales Feature falls Aufwand klein.

H.3 — UI-Anforderungen Embed-Code-Generator

- Auswahl iFrame vs. JS-Snippet (falls beides angeboten)
- Vorschau-Funktion
- Copy-to-Clipboard-Funktion
- Klartext-Hinweis: *„Copy-Paste in deine Website. Bei WordPress: HTML-Block. Bei Webflow: Embed-Element."*

Block I — Auto-Bestätigungsmail

I.1 — Template

Tenant-Admin definiert Subject + Body pro Booking-Page. Defaults werden geseedet:

```
Subject: Bestätigung: {{booking_page.title}} am {{appointment.date}}

Hallo {{contact.first_name}},

dein Termin ist bestätigt:

  📅 {{appointment.date}} um {{appointment.time}}
 ⏱ {{booking_page.duration_minutes}} Minuten
  👤 mit {{provider.full_name}}
  📍 {{appointment.location}}

{{booking_page.description}}

Falls du den Termin verschieben oder absagen musst,
antworte einfach auf diese Mail.

Bis bald,
{{tenant.company_name}}
```

I.2 — Verfügbare Template-Variablen

- `{{appointment.date}}`, `{{appointment.time}}`, `{{appointment.duration}}`, `{{appointment.location}}`
- `{{booking_page.title}}`, `{{booking_page.description}}`, `{{booking_page.duration_minutes}}`
- `{{provider.full_name}}`, `{{provider.email}}`, `{{provider.phone}}`
- `{{contact.first_name}}`, `{{contact.last_name}}`, `{{contact.email}}`
- `{{tenant.company_name}}`

I.3 — Sende-Mechanik

- Auslösung: direkt nach erfolgreichem Submit (synchron)
- Versand über Tenant-Sending-Domain (Modul 9)
- Fallback wenn Sending-Domain nicht konfiguriert: System-Domain `noreply@hivegrid.de` mit Klartext-Hinweis im Editor *„Konfiguriere eine Absender-Domain für professionellere Mails."*
- Fehler beim Versand: Termin bleibt gebucht, Internal-Note an Provider („Bestätigungsmail an {email} fehlgeschlagen — bitte manuell kontaktieren"), zudem Activity `email_send_failed`

I.4 — Reminder-Mails (Phase 2)

Nicht in Phase 1 — Erinnerungen 24h/1h vor Termin sind Phase 2.

Block J — Verknüpfungen

J.1 — Lead-Inbox / CRM

Beim Submit:
1. Email-Match-Check gegen `Contact.email`-Liste:
- **Match gefunden** → `Appointment.contact_id` direkt gesetzt, kein Lead erzeugt; Activity `appointment_scheduled` an Contact
- **Kein Match** → neuer `Lead`-Record erzeugt mit `source = booking`, `Appointment.lead_id` gesetzt; Lead-Inbox zeigt diesen mit Booking-Hinweis
2. Form-Submission-Daten werden an `FormSubmission` gespeichert und mit `Appointment.form_submission_id` verknüpft

J.2 — Internal Calendar

- Termin wird automatisch in interne `CalendarEvent`-Tabelle (Modul 25) für `provider_user_id` eingetragen
- Title aus `BookingPage.internal_calendar_event_title` mit Variablen-Substitution

J.3 — External Calendar Sync (Modul 25)

- Falls Provider Google/MS-Kalender verbunden hat: bidirektionale Sync
- Beim Erstellen: `CalendarEvent` wird in externen Kalender geschoben, externe Event-ID in `Appointment.external_calendar_event_ids` gespeichert
- Beim Cancel: externes Event gelöscht
- Beim Reschedule: externes Event aktualisiert
- Bei Sync-Fehler (z.B. Token abgelaufen): Klartext-Status im Termin-Detail + automatische Re-Try-Queue
- Wichtig: externe Calendar-Events des Providers werden auch **gegen** Slot-Berechnung gerechnet (siehe A.6) — d.h. wenn der Provider in seinem Google-Kalender einen privaten Termin anlegt, wird dieser Slot automatisch in der Booking-Page ausgegraut

J.4 — Activity-Stream-Integration

Activity-Typen produziert von Modul 6:

Activity-Typ	Wann	Subject
`appointment_scheduled`	bei Buchung	Contact (oder Lead falls neu)
`appointment_held`	bei manueller Status-Änderung auf `held`	Contact
`appointment_cancelled`	bei Stornierung	Contact / Lead
`appointment_rescheduled`	bei Verschiebung	Contact
`appointment_no_show`	bei Status `no_show`	Contact

Verknüpfung mit anderen Modulen:
- Activity am Contact (Modul 1)
- Activity am Lead (Modul 2)
- Activity am Provider-Pro-User-Dashboard
- Activity in zugeordnetem Deal (falls Contact in einem Deal ist) — automatisch via `subject_type=contact` plus `related_entity_refs`

J.5 — Form-Submission-Integration

- Bei Buchung wird ein Form-Submission erzeugt (entweder Standard-Mini-Form oder Custom-Form aus `intake_form_id`)
- FormSubmission ist vollwertig in Modul 5 sichtbar mit allen Antworten
- Activity `form_submitted` wird zusätzlich erzeugt (für CRM-Stream-Konsistenz)

Block K — Permissions-Anwendung

K.1 — Permission-IDs

Permission	Erlaubt
`bookings.read_own`	Termine sehen wo `provider_user_id = self`; eigene Booking-Pages und Verfügbarkeit sehen
`bookings.read_all`	Alle Termine, alle Booking-Pages, alle Verfügbarkeiten lesen
`bookings.write`	Eigene Verfügbarkeit pflegen, eigene Termine als `held` / `no_show` markieren, eigene Termine canceln/reschedule, Booking-Page anlegen wenn auch `bookings.admin`
`bookings.admin`	Booking-Pages anlegen/editieren/löschen, fremde Verfügbarkeiten pflegen, alle Termine canceln/hard-löschen

K.2 — Bundle-Mapping (Variante 2)

Bundle	Atomare Permissions
`bookings.none`	(keine)
`bookings.read_own`	`bookings.read_own`
`bookings.read_all`	`bookings.read_own` + `bookings.read_all`
`bookings.write`	`bookings.read_all` + `bookings.write`
`bookings.admin`	`bookings.read_all` + `bookings.write` + `bookings.admin`

K.3 — Anwendung in Routes

- `/{tenant}/bookings` → `bookings.read_own` oder `bookings.read_all`
- `/{tenant}/bookings/:id` → `bookings.read_own` (Owner-Check) oder `bookings.read_all`
- `/{tenant}/bookings/pages/` → `bookings.read_all` (Liste sehen) — Edit erfordert `bookings.admin`
- `/{tenant}/bookings/pages/new` → `bookings.admin`
- `/{tenant}/bookings/pages/:id` → Lesen `bookings.read_all`, Editieren `bookings.admin`
- `/{tenant}/availability` → `bookings.write` (eigene), für fremde User-Verfügbarkeit zu sehen reicht `bookings.read_all`, zu editieren braucht `bookings.admin`

K.4 — Anwendung in Action-Buttons

- Termin als `held` / `no_show` markieren: `bookings.write` (Owner-Check) oder `bookings.admin`
- Termin canceln: `bookings.write` (Owner-Check) oder `bookings.admin`
- Termin reschedule: `bookings.write` (Owner-Check) oder `bookings.admin`
- Provider eines Termins ändern: `bookings.admin`
- Termin hard-löschen: `bookings.admin`
- Booking-Page anlegen/editieren: `bookings.admin`
- Embed-Code generieren: `bookings.admin`

K.5 — Owner-Check-Logik für `read_own`

- `bookings.read_own`-User sehen nur Appointments mit `provider_user_id = self`
- Booking-Pages sieht User mit `read_own` nur, wenn er als Provider in `BookingPageProvider` für diese Page eingetragen ist
- Filter wird server-side angewendet (AV #5 — Permissions-aware-Default)

K.6 — Öffentliche Booking-Page

- Keine Permission-Prüfung — `app.hivegrid.de/buch/{tenant}/{slug}` ist öffentlich
- Tenant muss `active` oder `trial` sein (nicht `frozen` / `suspended` / `deleted`); sonst Klartext-404
- Booking-Page muss `is_published = true` und `archived_at = null` sein

Block L — Klartext-Hinweise / Fehler-Cases

- Slug-Konflikt im Editor: *„Diese URL ist schon vergeben — wähle eine andere."*
- Reserved-Slug: *„'{slug}' ist reserviert. Wähle einen anderen."*
- Booking-Page ohne Provider speichern: *„Mindestens 1 Provider muss zugewiesen sein, sonst können Endkunden nicht buchen."*
- Provider ohne Verfügbarkeits-Regeln: *„{Provider} hat keine Verfügbarkeit hinterlegt — er wird auf der Booking-Page nicht angezeigt."*
- Slot beim Submit nicht mehr frei: *„Ohje, dieser Slot wurde gerade gebucht. Hier sind aktuelle Alternativen."*
- Calendar-Sync-Fehler: *„Konnte Termin nicht in {Provider}-Kalender eintragen ({reason}). [Erneut versuchen]"*
- Bestätigungsmail-Fehler: *„Bestätigungsmail an {email} ist nicht durchgekommen ({reason}). Bitte den Endkunden direkt informieren."*
- Booking-Page archivieren mit zukünftigen Terminen: *„Diese Page hat noch {n} anstehende Termine. Bei Archivierung bleiben sie bestehen, aber neue Buchungen sind nicht mehr möglich."*
- Provider-Wechsel im Termin: *„Termin wird im Kalender von [alter Provider] entfernt und bei [neuer Provider] eingetragen. Endkunde bekommt automatisch eine Update-Mail."*

Block M — Phase 2 / spätere Features

- **Conditional Slots:** „Slot nur freigeben wenn Frage X mit Y beantwortet" (z.B. „nur Wartungstermine wenn Anlage-Typ = Gas")
- **Group-Bookings:** mehrere Teilnehmer pro Slot (Webinar-Style oder Beratungsrunde)
- **Reschedule-/Cancel-Link mit Token:** Endkunde kann ohne Login über Magic-Link-Token verschieben/stornieren (Phase 1 nur via Antwort auf Bestätigungsmail manuell durch Tenant-Mitarbeiter)
- **Reminder-Mails:** 24h/1h vor Termin automatisch
- **SMS-Reminder:** via Twilio/Sipgate (kostenpflichtige Add-on-Funktion)
- **Custom-Domain für Booking-Pages:** `termine.{tenant-domain}` statt `app.hivegrid.de/buch/{tenant}/...` (nur Enterprise)
- **Pre-Payment / Deposit:** Stripe-Vorauszahlung beim Buchen
- **Round-Robin mit Gewichtung:** Provider mit höherem Faktor bekommt mehr Termine
- **Workload-Cap pro Provider:** „max. 5 Termine pro Tag"
- **Tenant-Branding-Customization** auf öffentlicher Page (Logo, Farben — kommt aus Modul 31 in Phase 2)
- **Collective-Bookings:** mehrere Provider gemeinsam für einen Termin
- **CRM-Auto-Tag bei Buchung:** Tag X wird automatisch dem Contact gesetzt (Phase-2-Form-Action-Erweiterung)
- **JS-Snippet-Embed** mit Pop-up-Modal-Variante (falls in Phase 1 nicht geliefert)
- **Buchungs-Limit pro Endkunde:** „max. 1 offener Termin pro Email" zur Spam-Verhinderung

Block N — Architektur-Vorschriften (Querschnitt)

- **LP-7:** ist Pflicht-Modul (HubSpot-Booking-Ablösung)
- **LP-9:** Spec-Tiefe — alle Felder, Validierungen, Verknüpfungen explizit
- **LP-10:** Klartext-Diagnostik überall (Slot-Konflikt, Sync-Fehler, Mail-Fehler)
- **AV #5 — Permissions-aware-Default:** alle Listen/Slots filtern automatisch nach User-Permissions
- **AV #6 — Audit-Trail-Default:** alle Termin-Änderungen, Booking-Page-Änderungen geloggt (Modul 21)
- **LP-11 — Adapter-Pattern:** Calendar-Sync läuft über Modul-25-Calendar-Adapter, nicht direkt gegen Google/MS
- **AV #20 — Action-Boundary für KI:** KI darf Termine vorschlagen aber nicht direkt anlegen — User-Bestätigung Pflicht (relevant für Phase-2-KI-Booking-Assistent)

Detail-Spec-Status: DRAFT

Erste vollständige Detail-Spec für Modul 6. **Vor Bau:**
- Slot-Berechnungs-Algorithmus mit echten Geerkens-Anwendungsfällen testen (Mehrere Provider, mehrere parallele Booking-Pages, KWP-Termine in Google-Kalender als Blocker)
- Doppelbuchungs-Schutz auf DB-Ebene (PostgreSQL `EXCLUDE USING gist` mit `tstzrange`) verifizieren
- Bestätigungsmail-Template-Engine festlegen (Liquid vs. Handlebars vs. Mustache — sollte konsistent mit Modul 10 Email-Marketing sein)
- Embed-Route Sicherheits-Setup (X-Frame-Options, CSP, CORS)

Offene Detail-Punkte (nicht-blockierend):
- Standard-Mini-Form-Definition (welche Felder genau Pflicht sind, welche optional) — mit Designer/Noah finalisieren
- Konkrete Default-Bestätigungsmail-Texte für Branchen-Template SHK
- Workload-Cap als optionales Feld im BookingPage-Modell auch schon Phase 1 mitnehmen?
- JS-Snippet-Embed Phase 1 oder Phase 2 (Aufwand-Schätzung beim Bau)
- Add-to-Calendar-.ics-Generierung am Endpoint des Endkunden-Flows
- Reschedule-Token mit kurzer Lebenszeit als Phase-1-„Lite"-Variante prüfen (statt voller Phase 2)