Designer-First Playbook — HiveGrid 2.0

Ausführungsplan für den Fall: **Designer baut Screens zuerst, Backend wird dagegen gebaut.** Erstellt: 2026-06-08. Basiert auf ROADMAP.md (21 Phasen) + der Hot-Swap-Architektur (Phase 1, M013–M017).

---

0. Das Grundprinzip

- **Backend-Abhängigkeit** läuft 1→21 (Datenmodell vor Features). Das ist DEINE Reihenfolge.
- **Design-Abhängigkeit** läuft über **wiederverwendbare Archetypen**, NICHT über die Phasen-Nummern. Das ist die Reihenfolge des **Designers**.
- Beide Tracks brauchen **je ein Fundament zuerst**:
  - Backend-Fundament = **Phase 1** (VM/Adapter/ui-mock/Visual-Reg/Smoke-Harness). Blockt alles.
  - Design-Fundament = **Sprint D0** (Design-System + App-Shell + Archetypen). Blockt ~70% aller Screens.
- Danach läuft der **Designer 1–2 Phasen vor dir**. Du baust Backend gegen fertige Screens.

---

1. Die zwei Fundamente (Block 0 — parallel)

Track	Wer	Inhalt	Fertig wenn
Backend-Fundament = Phase 1	Du	Boilerplate, EU-Sterilisation, VM-Konvention, Action-Callback, Subframe-Adapter, ui-mock, Visual-Reg-Gate, Hot-Swap-Drill	Hot-Swap-Drill grün; 1 Beispiel-Route lädt VM gegen Mock
Design-Fundament = Sprint D0	Designer	Subframe-Design-System (Tokens, Farben, Typo, Spacing) + App-Shell (Sidebar/Breadcrumb/Topbar) + die 5 Archetypen + Zustands-Kit	Alle Archetypen + Zustände in Subframe vorhanden

Diese beiden laufen GLEICHZEITIG. Solange beide nicht stehen, gibt es kein sauberes designer-first.

---

2. Die Archetypen (= die Design-Dependency-Map)

Jeder Archetyp wird **einmal** designt und von vielen Phasen konsumiert. Reihenfolge = nach Hebel.

#	Archetyp	Einmal designen → diese Screens werden „compose"
A1	**App-Shell** (Sidebar, Breadcrumb, Topbar, Nav)	ALLE Phasen
A2	**List-View** (Filterbar, Tabelle/Cards, Bulk-Bar, Pagination, Spalten-Config, CSV, Leerzustand)	P6, P7, P8(Liste), P4(Tasks/Files), P3(Audit-Viewer/Notif-Center), P14(Segmente), P19(Tenant-Liste + Flag-Admin), P21(Help-Artikel)
A3	**Detail-View** (Header+Quick-Actions, Inline-Edit-Feld, Related-Panels, Activity-Sidebar, Custom-Felder)	P6(Contact/Company/Lead), P8(Deal), P9(Projekt), P17(HR-Profil), P18(Portal-Projekt)
A4	**Form/Input-Kit** (alle 12 Feldtypen, Validierung, Datei-Feld, Select/Multiselect)	P6(Custom-Felder), P16(Form-Builder), P20(Onboarding), P2(Auth-Forms), jedes Create/Edit
A5	**Modal/Wizard** (mehrstufig, Bestätigungs-Dialog, Side-Panel)	P6(Lead-Promotion), P20(Onboarding-Wizard), Bestätigungen überall
A6	**Settings/Connection-Cards** (Toggle, Connection-Status, Key-Eingabe)	P5(Apps-Engine), P3(Feature-Flags), P20(Billing/Branding), P19(Admin)
A7	**Zustands-Kit** (Leer / Laden-Skeleton / Fehler / keine-Berechtigung)	ALLE — Pflicht laut VM-Konvention (Loading/Error sind VM-Zustände)

Konsequenz: Sind A1–A7 fertig (= Sprint D0), kann der Designer ~70% der Screens in Stunden statt Tagen zusammenstecken.

---

3. Die Bespoke-Screens (Sonderfälle — früh starten!)

Diese nutzen kaum Archetypen, brauchen die meiste Iteration → **so früh wie möglich starten, auch wenn das Backend noch fern ist.** Sortiert nach Vorlaufzeit (längste zuerst):

Rang	Bespoke-Screen	Phase	Aufwand	Warum früh
1	**Automation Flow-/Node-Editor**	P12	🔴 sehr hoch	Schwierigste UI der ganzen App; viele Iterationen

2	**Form-Builder-Canvas** (Drag-Drop)	P16	🔴 sehr hoch	Komplexe Interaktion + Conditional-Logic-Editor
3	**Plantafel / Scheduling-Board**	P9	🟠 hoch	Custom-Grid, Drag pro Monteur/Tag, offline
4	**Dashboard + Widget-Grid + Cmd+K**	P11	🟠 hoch	Widget-System + Command-Palette = 2 Sonder-UIs
5	**Customer-Portal-Shell**	P18	🟠 hoch	**Extern sichtbar** — eigene Design-Sprache, höchster Polish
6	**Mobile-Shell** (Capacitor)	P21	🟠 hoch	Komplett eigenes UI-Paradigma (Touch, Tab-Bar, offline)
7	**Kalender-View**	P16	🟡 mittel	Standard-aber-eigen (Monats/Wochen-Grid)
8	**Team-Chat / HiveMind-Chat**	P15/P13	🟡 mittel	Message-Liste + Composer + Presence
9	**Sales-Kanban**	P8	🟡 mittel	Spalten + Draggable-Cards
10	**Öffentliche Booking-Seiten**	P16	🟡 mittel	**Extern sichtbar** — Slot-Picker, Branding
11	**Help-Center** (public)	P21	🟡 mittel	Content-Site, SEO, Suche

---

4. Designer-Reihenfolge (die Antwort auf „welche Screens zuerst")

```
D0  Design-Fundament   System + App-Shell + Archetypen A1–A7 + Zustände   ◄── ZUERST, höchster Hebel
        │
D1  CRM-Cluster (P6)   Contact/Company/Lead-Liste + Detail + Lead-Promo-Modal + Custom-Feld-UI
        │              → etabliert das dominante Muster für den halben Rest
        │
D2  Bespoke früh starten (PARALLEL, lange Vorlaufzeit):
        ├─ Flow-Editor (P12)      ◄── längste Vorlaufzeit, JETZT anfangen
        ├─ Form-Builder (P16)
        ├─ Plantafel (P9)
        └─ Dashboard + Cmd+K (P11)
        │
D3  CRUD-Feature-Screens (schnell, reuse Archetypen):
        Lead-Inbox (P7), Sales-Liste+Kanban (P8), Tasks/Files (P4),
        Email/Marketing (P14), HR (P17), Auth-Screens (P2)
        │
D4  Extern/Brand-kritisch (eigene Politur-Runde):
        Customer-Portal (P18), Booking-public (P16), Onboarding-Wizard (P20),
        Help-Center + Mobile (P21)
        │
D5  Niedriger Polish-Druck (intern), dazwischen:
        Plattform-Admin (P19), Settings (P5/P3), KI-Chat (P13)
```

Kurzantwort: Design-Fundament → CRM → parallel die 4 Bespoke-Langläufer (Flow-Editor, Form-Builder, Plantafel, Dashboard).

---

5. Backend-Track (deine Reihenfolge) mit Parallelisierung

Du baust sequenziell, aber der Designer liegt vor dir. Pro Block: was du baust + was der Designer geliefert haben muss.

Block	Deine Phasen (Backend)	Designer muss fertig haben	Charakter
0	**P1** Fundament	D0 läuft parallel	Infra, kein Design nötig
1	P2, P3, P4, P5	D0 fertig; D1(CRM) + D2(Bespoke) laufen	Auth/Plumbing/Files/Apps-Engine — backend-lastig
2	**P6, P7, P8, P9** (SHK-Kernpfad)	CRM, Lead, Sales-Kanban, Plantafel	erste echte Screens — Noah-Validierung startet
3	**P10** KWP	— (kein UI)	reines Backend
4	P11, P12, P13	Dashboard/Cmd+K, Flow-Editor, KI-Chat	USP-Säulen — Bespoke muss bereitliegen

5	P14, P15, P16	Email, Chat, Form-Builder, Kalender, Booking	Kommunikation
6	P17, P18	HR, **Customer-Portal (extern)**	Menschen/extern
7	P19, P20, P21	Admin, Onboarding, **Mobile**, Help	Plattform & Launch

Reorder-Spielraum (echte Deps, nicht 1→21-Zwang): P14/P15 hängen nur an P6+P5, P16 an P7+P5, P17 an P4, P19 an P3. Heißt: sobald CRM (P6) steht, könntest du Email (P14) oder Telefonie (P15) vorziehen, falls ein Screen früher fertig ist. P10 (KWP) kann jederzeit nach P9 laufen.

---

6. Verifikations-Gates (was du wo prüfst)

6a. Pro Screen — die Designer→Backend-Übergabe (Checkliste)

```
[ ] 1. UI-SPEC existiert      /gsd-ui-phase N → Element-Liste + Zustände + Interaktionen
[ ] 2. Screen designt         Designer liefert Subframe-Screen (aus der Element-Liste!)
[ ] 3. RECONCILIATION ◄── kritischster Check:
         [ ] jedes angezeigte Feld hat eine Datenquelle (Modell/Spec)?  (keine „unmöglichen Daten")
         [ ] jede Aktion/Button mappt auf eine Domänen-Operation?
         [ ] Zustände designt: leer / laden / Fehler / keine-Berechtigung?
[ ] 4. VM ableiten            VM-Typ + Action-Callbacks aus dem Screen lesen
[ ] 5. Backend bauen          /gsd-plan-phase N (Planner baut Loader/Actions gegen das VM) → execute
[ ] 6. Integration            Adapter bindet Screen an VM → Smoke-Test grün
[ ] 7. Visual-Reg-Baseline    Playwright-Baseline für den Screen setzen
[ ] 8. Hot-Swap-Drill         CI tauscht gegen ui-mock → läuft noch? (beweist saubere Trennung)
```

6b. Pro Phase — GSD-Gates (automatisch)

- **Source-Coverage-Audit** (Planner): jedes Phase-Requirement in einem Plan.
- **must_haves** (goal-backward): Truths/Artefakte/Key-Links erfüllt.
- **plan-checker**: Pläne erreichen das Ziel (vor dem Bauen, Loop ≤3×).
- **execute-phase Verifier**: Phasen-Ziel goal-backward erfüllt.

6c. Projektweit — Dauergates (CI, jeder Merge)

- **ESLint**: kein React-Import im VM-Paket (erzwingt Trennung).
- **Smoke-Test pro Route** (gegen ui-mock).
- **Visual-Regression**: Diff > Threshold → Build bricht mit Klartext-Liste betroffener Screens.
- **Hot-Swap-Drill**: ui-mock-Tausch muss grün bleiben.

6d. Stakeholder — Noah/Geerkens (AD-19)

Nach jeder UI-Phase: **PROGRESS-TRACKER-Klick-Pfad** → Noah validiert den echten Screen. Designer-first ist hier im Vorteil: Noah sieht reale Screens früh, nicht erst beim Big-Bang.

---

7. Die drei kritischen Erfolgsregeln

1. **Designer designt aus der UI-SPEC-Element-Liste**, nicht frei — sonst Reconciliation-Hölle (Screens mit Daten, die's nicht gibt).
2. **Phase 1 zuerst, ohne Kompromiss** — die Hot-Swap-Maschine ist die Voraussetzung, dass ein Screen überhaupt andocken kann.
3. **Bespoke-Langläufer (Flow-Editor, Form-Builder) sofort starten** — sie sind der kritische Pfad des Design-Tracks, auch wenn ihr Backend (P12/P16) erst spät kommt.

Erstellt 2026-06-08 als Ergänzung zu ROADMAP.md / PHASE-SOURCES.md für den designer-first-Modus.