app.guidetranslator.com

Update_V7_build: 26: Februar, 2026, 12:07

NEUER PR

PR Summary: Admin-Upgrade + Strategischer Masterplan

Dieser PR macht zwei Dinge gleichzeitig: Er legt den strategischen Gesamtplan als PLAN.md fest — und verbessert sofort das Admin-Dashboard mit Bulk-E-Mail und smarten Aktions-Empfehlungen.

💡 Ulrichs Erklärung: Was ist ein PR?

PR steht für 'Pull Request' — das ist wie ein Änderungsantrag. Claude hat Code geschrieben und sagt damit: 'Ich habe hier Verbesserungen gebaut, schau sie dir an.' Dieser PR enthält: (1) Einen 556-Zeilen Masterplan als Datei, (2) ein verbessertes Admin-Werkzeug zum Massen-E-Mailen, und (3) schönere automatische E-Mails.

PLAN.md

556 Zeilen — vollständiger Masterplan für die SaaS-Transformation mit 6 Phasen und 17 Schritten

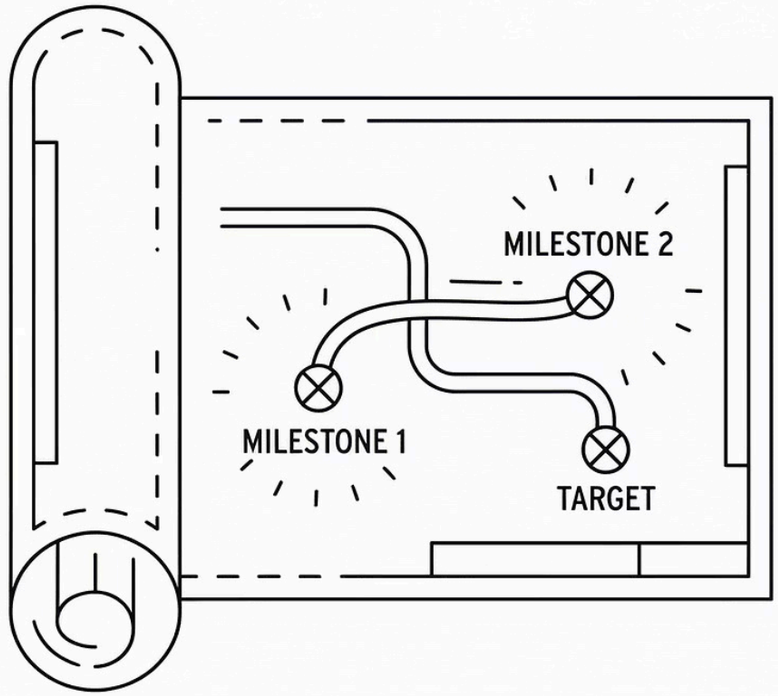
Admin Dashboard

Bulk-E-Mail Modal, Empfohlene Aktionen Widget, Pipeline-Verwaltung für mehrere Leads gleichzeitig

E-Mail System

Professionelle HTML-Templates mit Dark Theme, CTA-Buttons und automatischer Signatur

> Phase 1 (Auth/Routing) hat höchste Priorität — sie ist das Fundament für alle weiteren Phasen.



DOKUMENTATION

PLAN.md — Der 556-Zeilen Masterplan

PLAN.md ist eine neue Datei, die den kompletten Umbauplan der App dokumentiert — Ist-Zustand, Ziel-Zustand, alle 6 Phasen, 17 Schritte und offene Fragen.

💡 Ulrichs Erklärung: Was ist PLAN.md?

Stell dir vor, du willst dein Haus umbauen. Bevor der Handwerker anfängt, zeichnet der Architekt einen Plan: Was ist jetzt da? Was soll am Ende da sein? In welcher Reihenfolge wird gebaut? PLAN.md ist genau dieser Architektenplan — nur für die App. 556 Zeilen heißt: sehr detailliert, sehr durchdacht, nichts dem Zufall überlassen.

Phase 1: Auth + Rollen + Routing

Supabase Profiles, Organizations, Rollen-System (Admin/Sales/Kunde) — Fundament für alles andere

Phase 2: Segment-Differenzierung

6 Kundentypen mit eigenen Parametern: Stadtführer, Agentur, Veranstalter, Kreuzfahrt, Großveranstalter, Fintutto

Phase 3: Sales-Flow

Angebot → Test → Follow-up → Conversion — vollständig automatisiert

Phase 4: Pricing + Stripe

Free → Starter → Pro → Business → Enterprise → Custom — Kostenanalyse zuerst!

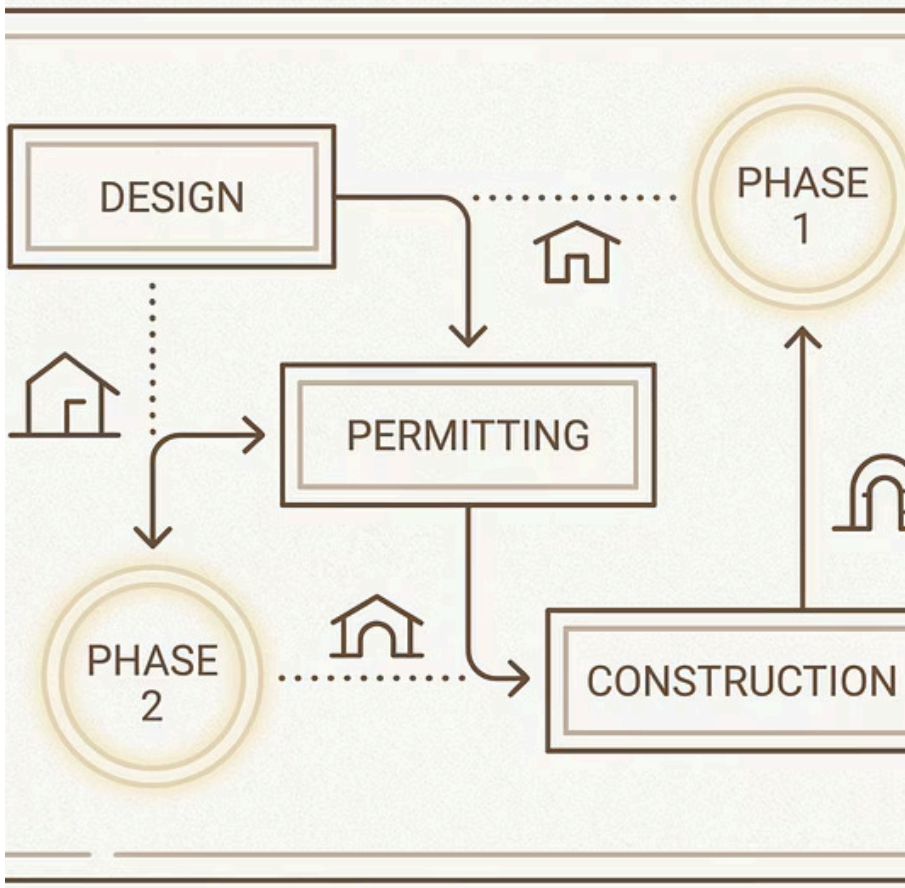
Phase 5: Kunden-Dashboard

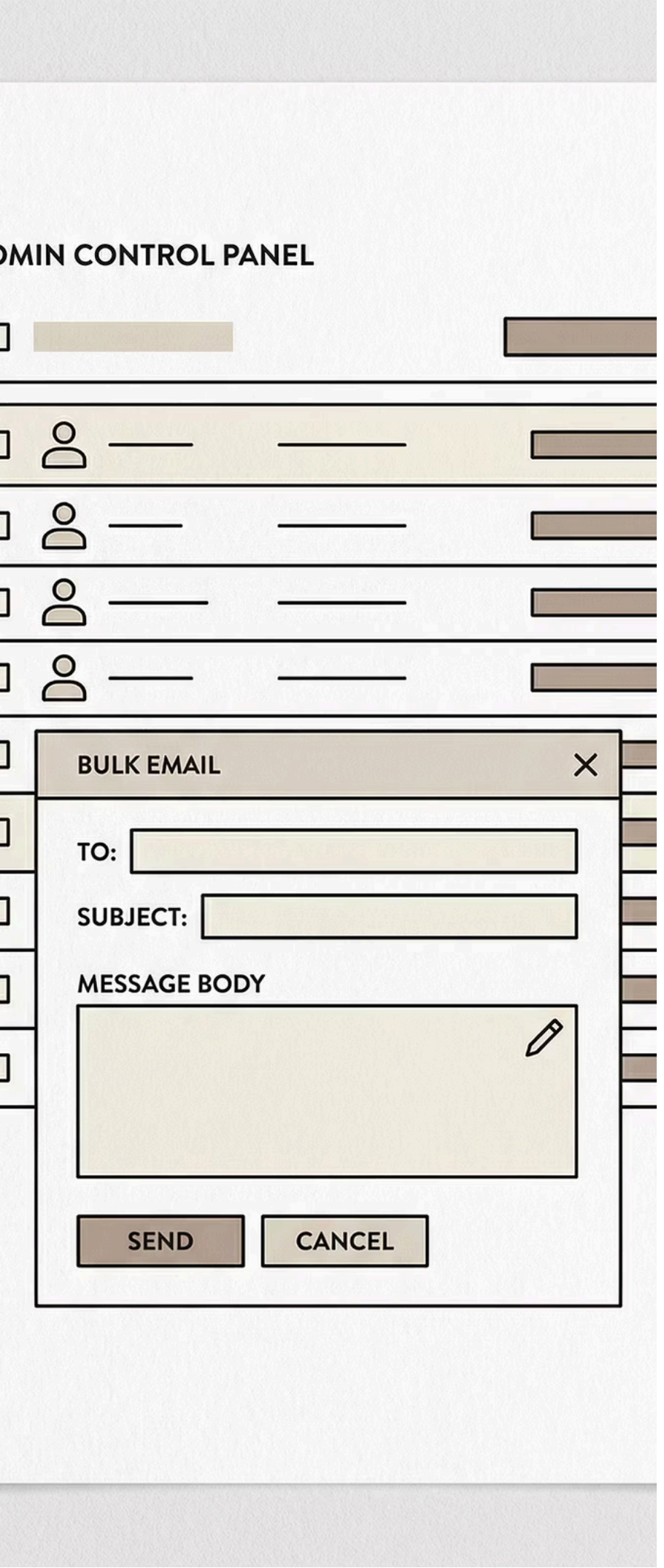
Usage-Tracking, Plan-Übersicht, Self-Service Billing Portal

Phase 6: Admin-Erweiterungen

Bulk-Aktionen, Reporting, erweiterte Pipeline-Verwaltung

> **17 priorisierte Schritte mit Abhängigkeits-Mapping — die Reihenfolge ist nicht zufällig, sie minimiert Blocker.**





ADMIN

Admin Dashboard — Bulk-E-Mail & Empfohlene Aktionen

Das Admin-Dashboard wurde massiv aufgewertet: Statt einzelner Aktionen gibt es jetzt Bulk-Werkzeuge — mehrere Leads gleichzeitig anschreiben, Pipeline-Stages in einem Schritt ändern.

💡 Ulrichs Erklärung: Was ist Bulk-E-Mail?

Stell dir vor, du hast 15 Interessenten, die alle noch keine Einladung bekommen haben. Früher: 15 Mal einzeln klicken, 15 Mal E-Mail schreiben. Jetzt: Alle 15 markieren, eine Vorlage auswählen, einmal auf 'Senden' klicken — fertig. Das nennt man Bulk-E-Mail. Die App zeigt dir dabei live an, wie viele erfolgreich gesendet wurden.

Bulk-E-Mail Modal

BulkEmailModal Komponente — Leads auswählen, Vorlage wählen, an alle gleichzeitig senden. Fortschrittsanzeige: X gesendet / Y fehlgeschlagen.

Empfohlene Aktionen

Ersetzt den alten 'Überfällige Follow-ups' Banner. Zeigt jetzt 3 Kategorien: überfällige Follow-ups, neue Leads ohne Einladung, eingeladene Leads ohne Termin.

Bulk Pipeline-Verwaltung

Mehrere Leads markieren → Pipeline-Stage für alle gleichzeitig ändern → E-Mails in Batch senden. Alles in einem Schritt.

> **Was früher 20 Minuten dauerte, geht jetzt in 30 Sekunden — Leads auswählen, Vorlage wählen, senden.**

E-MAIL

E-Mail System — Professionelle HTML-Templates

api/send-email.js wurde komplett überarbeitet. Alle ausgehenden E-Mails sehen jetzt professionell aus — automatisch, ohne dass jemand manuell formatieren muss.

💡 Ulrichs Erklärung: Was ist ein HTML-E-Mail-Template?

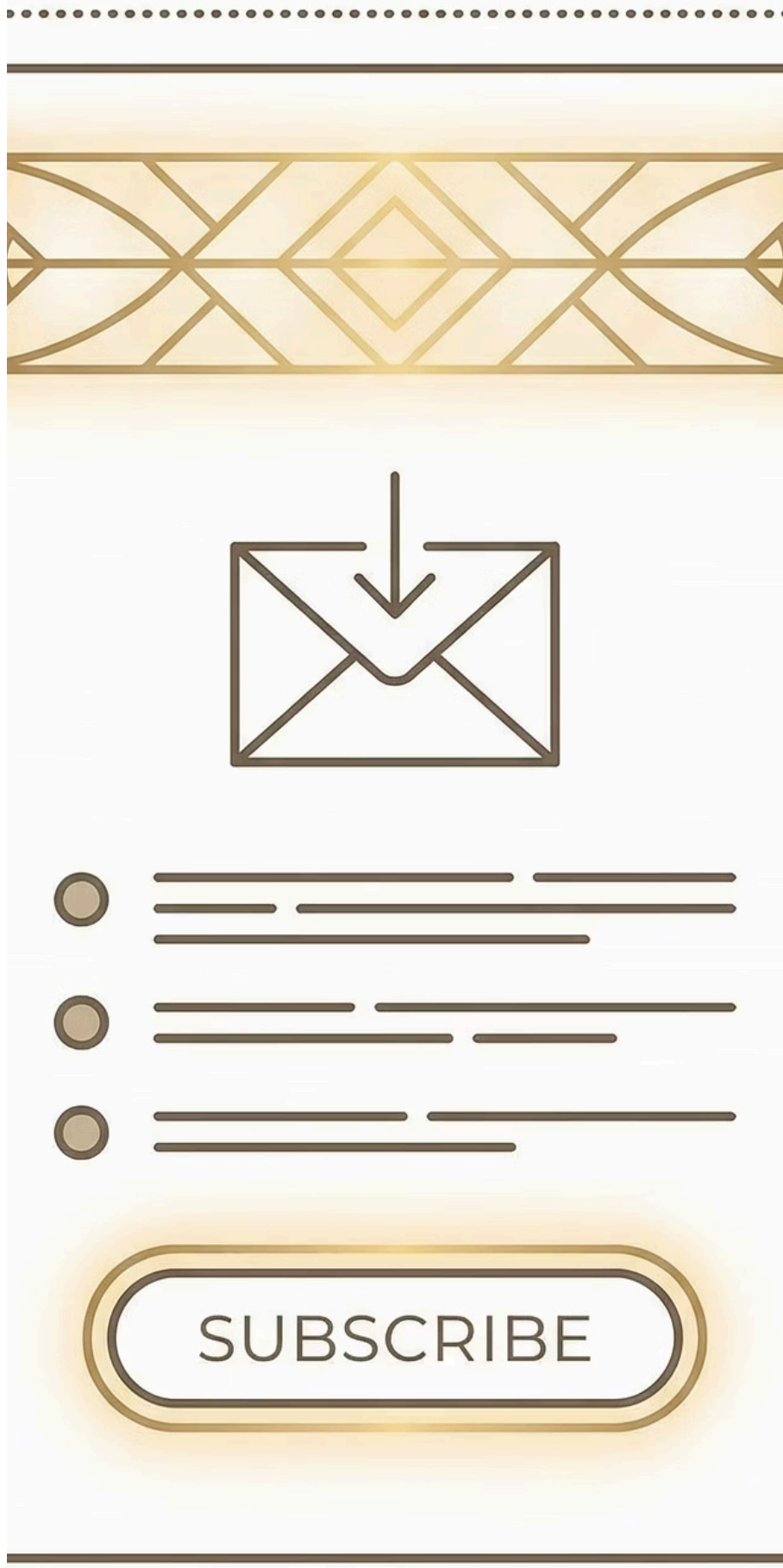
Normale E-Mails sind wie ein handgeschriebener Brief auf weißem Papier. Ein HTML-E-Mail-Template ist wie ein gedruckter Firmenbrief mit Logo, Farben, schönem Layout und einem 'Jetzt klicken' Button — aber als E-Mail. Die neue wrapHtml()-Funktion macht das vollautomatisch: Du (oder die App) schreibt den Text, und sie verpackt ihn in ein professionelles GuideTranslator-Design.

Was wurde gebaut (api/send-email.js)

- wrapHtml() — wandelt plain Text automatisch in HTML-E-Mail um
- Dark Theme (Navy/Gold) — passt zum GuideTranslator Brand
- Bullet-Point-Erkennung — Listen werden automatisch formatiert
- Link-Erkennung → wird zu CTA-Button mit Klick-Styling
- Signatur-Block — automatisch an jede E-Mail angehängt
- Responsive — funktioniert auf Handy und Desktop

Vorher → Nachher

- ❌ Plain-Text, kein Design, wirkt unprofessionell
- ✅ Branded HTML mit Farben, Buttons, Signatur
- ❌ Jede E-Mail manuell formatiert
- ✅ Template-Variablen automatisch befüllt
- ❌ Links als roher Text
- ✅ Links als klickbare CTA-Buttons



> Jede E-Mail, die die Plattform sendet, sieht jetzt aus wie von einem professionellen SaaS-Unternehmen — ohne Mehraufwand.

app.guidetranslator.com

Update_V7_build-test: 26: Februar, 2026, 12:09

CODE REVIEW

Was noch fehlt — Tiefer Scan vor Go-Live

Der tiefe Scan hat 23 offene Punkte gefunden — sortiert nach Priorität. 7 davon sind kritisch und müssen vor dem Go-Live behoben werden.

💡 Ulrichs Erklärung: Was ist ein 'tiefer Scan'?

Claude hat den gesamten Code systematisch durchgelesen — wie ein Gutachter, der ein Haus vor dem Einzug prüft. Er sucht nach Dingen, die fehlen, kaputt sind oder ein Sicherheitsrisiko darstellen. Das Ergebnis: 7 kritische Probleme (muss vor Launch), 7 wichtige Probleme (bald danach) und 9 Nice-to-Haves (irgendwann).

7 Kritisch 🔴

Müssen vor Go-Live behoben werden — sonst funktioniert Stripe, Auth oder die Datenbank nicht korrekt

7 Wichtig 🟡

Sollten zeitnah nach Launch behoben werden — Sicherheit, UX und Stabilität

9 Nice-to-Have 🟢

Verbesserungen für nach dem Launch — Tests, Analytics, Mobile, GDPR

> Die kritischsten Punkte sind Stripe Webhook, SQL-Migrationen und das fehlende Tier-Mapping — ohne diese drei funktioniert kein einziges Abo.

⚠ VOR LAUNCH

🔴 Kritisch: 7 Punkte vor Go-Live

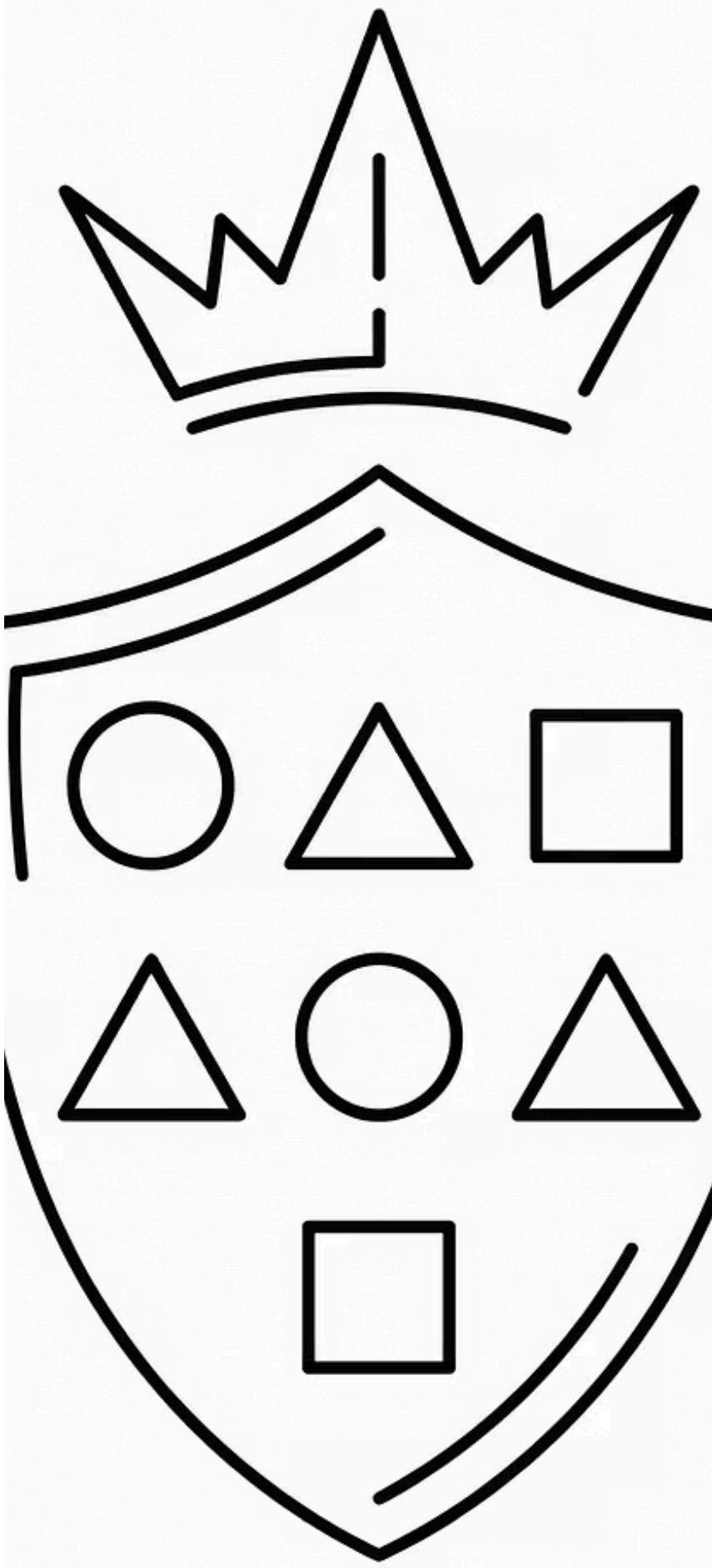
Diese 7 Punkte müssen behoben werden, bevor die Plattform live geht. Ohne sie funktioniert Stripe, Auth oder die Datenbank nicht korrekt.

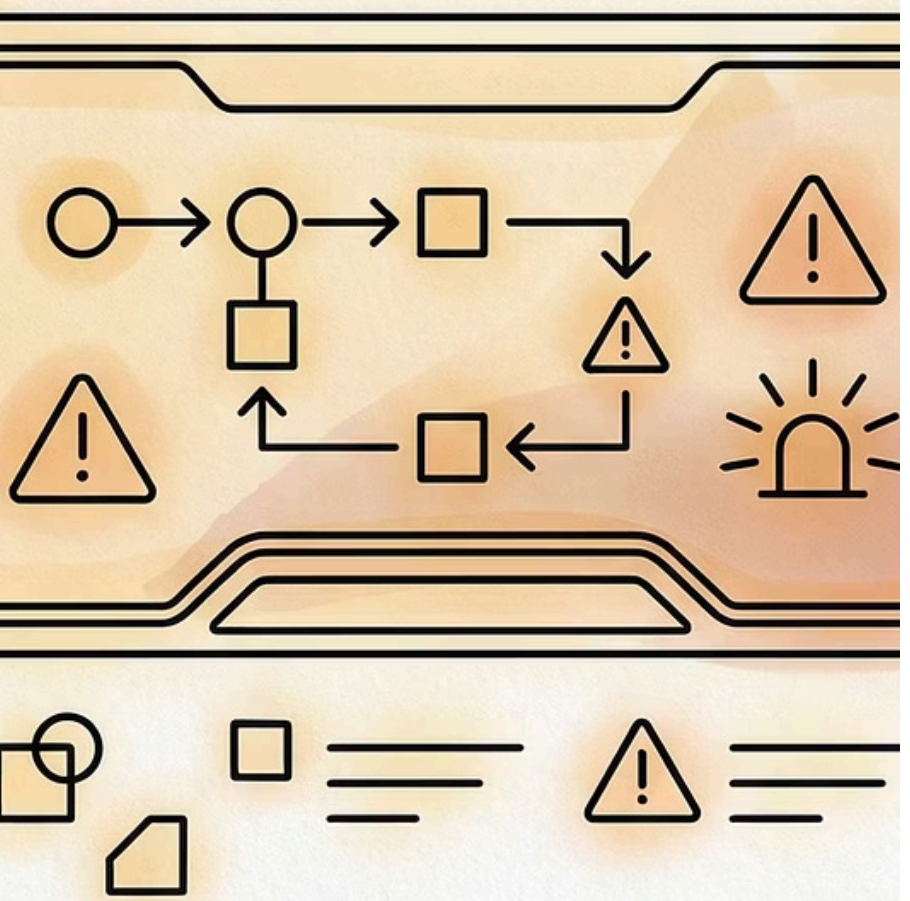
💡 Ulrichs Erklärung: Was bedeutet 'kritisch'?

Kritisch bedeutet: Wenn das nicht behoben wird, ist die App zwar online — aber kaputt. Stell dir vor, du eröffnest ein Café, aber die Kasse funktioniert nicht und die Kühlschränke sind nicht angeschlossen. Kunden kommen rein, aber du kannst nichts verkaufen. Genau das passiert hier, wenn diese 7 Punkte nicht behoben werden.

#	Was fehlt	Warum kritisch
1	Stripe Webhook registrieren	Ohne STRIPE_WEBHOOK_SECRET kommen keine Events an — Abos werden nie aktiviert
2	SQL-Migrationen ausführen	phase4-auth-roles.sql + phase5-7-stripe-usage.sql — ohne das fehlen Tabellen/Spalten
3	Fehlende DB-Tabellen	gt_lead_notes, gt_contact_requests, gt_calculations fehlen im Schema — Code crasht
4	Webhook setzt kein subscription_tier	Nach Checkout weiß das Dashboard nicht welcher Plan gekauft wurde
5	Add-Ons: kein Kauf-Button	Add-Ons sind one_time, aber create-checkout.js erzwingt Subscription
6	.env.example unvollständig	8+ Variablen fehlen: STRIPE_SECRET_KEY, WEBHOOK_SECRET, RESEND_API_KEY, CRON_SECRET
7	Hardcoded Admin-Passwort	Fallback "guidetranslator2026" — wenn ENV fehlt, kann jeder ins Admin-Panel

> Punkt 7 ist besonders heikel — ein hardcodiertes Passwort im Code ist ein Sicherheitsrisiko, das sofort behoben werden muss.





NACH LAUNCH

🟡 Wichtig: 7 Punkte zeitnah nach Launch

Diese 7 Punkte sind nicht launch-blockierend, sollten aber in den ersten Wochen nach Go-Live behoben werden — sie betreffen Sicherheit, UX und Stabilität.

💡 Ulrichs Erklärung: Was ist CORS und Rate-Limiting?

CORS (*): Stell dir vor, dein Café hat eine Hintertür, die für jeden offen ist — auch für Fremde. CORS * bedeutet, jede Website der Welt kann deine API aufrufen. Das sollte auf deine eigene Domain eingeschränkt werden. Rate-Limiting: Ohne das kann jemand dein Kontaktformular 10.000 Mal pro Minute absenden und deinen Server überlasten — wie jemand, der immer wieder auf die Türklingel drückt.

#	Was fehlt	Impact
8	Checkout → Dashboard Disconnect	User bezahlt, wird zu /dashboard geleitet, ist aber nicht eingeloggt → sieht "Bitte anmelden"
9	Kein customerEmail beim Checkout	Webhook matcht per Email — wenn Stripe-Email ≠ Lead-Email, wird nichts aktualisiert
10	CORS zu offen (* auf allen APIs)	Jede Website kann die APIs aufrufen — Sicherheitsrisiko
11	Kein Rate-Limiting	Formulare können unbegrenzt abgeschickt werden — Spam/Überlastung möglich
12	Keine E-Mail-Verifizierung	Leads registrieren sich mit beliebiger E-Mail-Adresse
13	check-followups.js crasht	Versucht in gt_lead_notes zu inserieren — Tabelle nicht in SQL-Files definiert
14	Dependabot: 2 Vulnerabilities	1 high, 1 moderate — Sicherheitslücken in Dependencies

> Punkt 13 ist besonders tückisch — der Cron-Job läuft täglich und crasht still. Kein Fehler sichtbar, aber auch keine Follow-up E-Mails.

✨ ZUKUNFT

🟢 Nice-to-Have: 9 Punkte nach Launch

Diese 9 Punkte sind keine Blocker — die App funktioniert ohne sie. Sie machen die Plattform aber professioneller, sicherer und wartbarer. Ideal für die Wochen nach dem Launch.

💡 Ulrichs Erklärung: Was ist ein Error Boundary und warum braucht man Tests?

Error Boundary: Wenn in der App ein Fehler passiert, sieht der User normalerweise eine komplett weiße Seite — wie wenn das Licht ausgeht. Ein Error Boundary fängt den Fehler ab und zeigt stattdessen eine freundliche Fehlermeldung. Tests: Stell dir vor, du baust eine Brücke und testest nie, ob sie trägt. Tests sind automatische Prüfungen, die sicherstellen, dass die App nach jeder Änderung noch funktioniert.

- **#15: Kein Test-Framework**

  Vitest/Jest fehlt, besonders für Webhook-Testing wichtig

- **#16: Kein Linting/Formatting**

  ESLint/Prettier nicht konfiguriert

- **#17: Kein React Error Boundary**

  Fehler zeigen weiße Seite statt Fehlermeldung

- **#18: Kein Mobile Responsive**

  Grid-Layouts brechen auf kleinen Screens

- **#19: Kein Error-Tracking**

  Sentry o.ä. fehlt

- **#20: Kein Analytics**

  Plausible/Google Analytics nicht eingebunden

- **#21: Keine GDPR Daten-Export Funktion**

  für EU-Kunden rechtlich relevant

- **#22: Keine API-Dokumentation**

  erschwert zukünftige Entwicklung

- **#23: Kein Passwort-Reset**

  "Passwort vergessen"-Flow fehlt komplett

> Priorität nach Launch: zuerst Error Boundary (#17) und Mobile (#18) — das sehen Kunden sofort. Tests (#15) danach, sobald das System stabil läuft.