app.guidetranslator.com

Update_V7_test/build7b: 26: Februar, 2026, 20:03

HOTFIX

7 Kritische Fixes — Alle erledigt ✅

Die letzten 7 kritischen Sicherheits- und Stabilitätsprobleme wurden in einem Durchgang behoben — Webhook-Sicherheit, fehlende Tabellen, Tier-Mapping, Add-On-Checkout und mehr.

💡 Ulrichs Erklärung: Was ist ein 'kritischer Fix'?

Ein kritischer Fix ist wie das Reparieren einer kaputten Kasse kurz vor der Eröffnung. Die App läuft — aber ohne diese Fixes würde Stripe keine Abos aktivieren, Kunden würden falsche Pläne bekommen, und ein Angreifer könnte gefälschte Zahlungen einschleusen. Jetzt sind alle 7 Probleme behoben.

#	Was wurde gefixt	Status
1	Webhook Signatur-Validierung + Replay-Attack-Schutz	✅ Erledigt
2	Fehlende SQL-Tabellen (gt_lead_notes, gt_contact_requests, gt_calculations)	✅ Erledigt
3	Webhook setzt jetzt subscription_tier korrekt	✅ Erledigt
4	Add-On Checkout gefixt (one_time statt subscription)	✅ Erledigt
5	.env.example: von 2 auf 11 Variablen dokumentiert	✅ Erledigt
6	Hardcoded Admin-Passwort entfernt	✅ Erledigt
7	SQL-Migration sql/phase8-missing-tables.sql	⚠️ Muss noch ausgeführt werden

> Einziger offener Punkt: sql/phase8-missing-tables.sql im Supabase SQL Editor ausführen — nach phase4 und phase5-7.

SICHERHEIT

Fix #1: Webhook Signatur-Validierung & Replay-Schutz

Der Stripe-Webhook hatte keine Signatur-Prüfung — jeder hätte gefälschte Zahlungs-Events einschleusen können. Jetzt ist der Endpoint abgesichert.

💡 Ulrichs Erklärung: Was ist ein Replay-Attack?

Stell dir vor, jemand fängt eine echte Zahlungsbestätigung von Stripe ab und schickt sie 100 Mal erneut — die App denkt, es wurden 100 Zahlungen gemacht und aktiviert 100 Abos. Das nennt man Replay-Attack. Der Schutz dagegen: Jede Nachricht hat einen Zeitstempel. Ist er älter als 5 Minuten, wird sie abgelehnt. Außerdem prüft die App jetzt, ob die Nachricht wirklich von Stripe kommt (Signatur-Check).

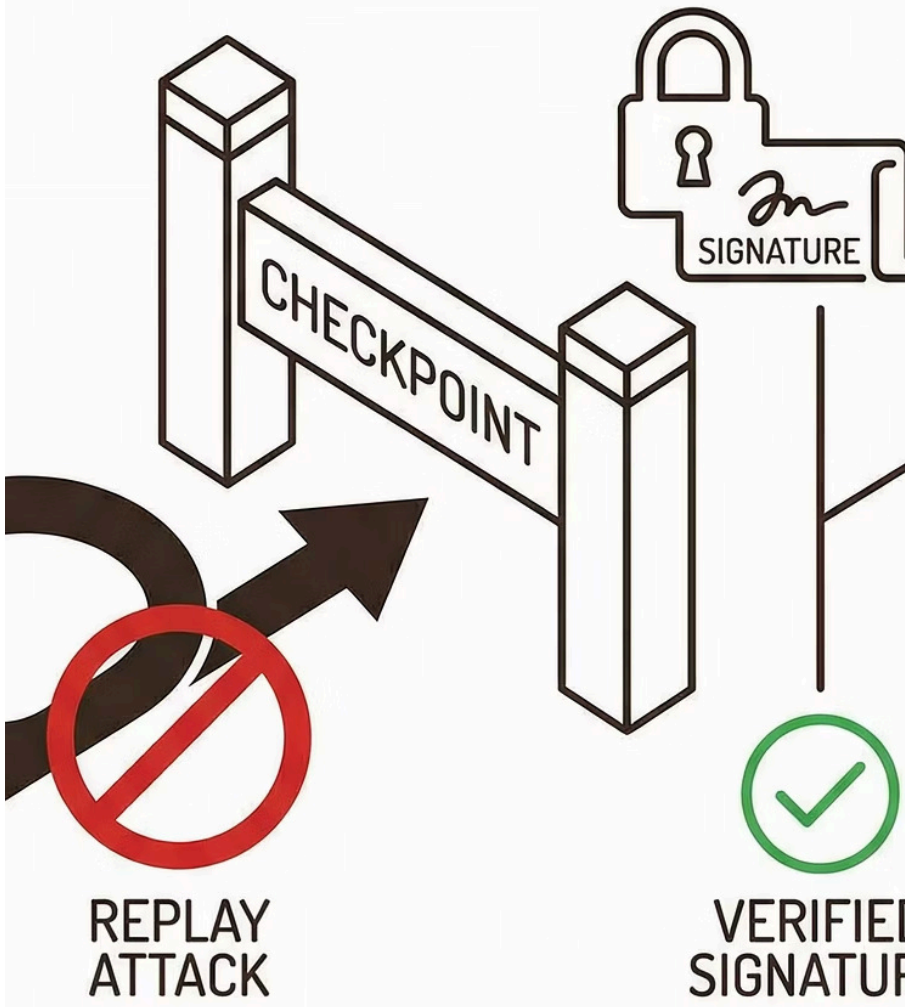
Vorher — Unsicher

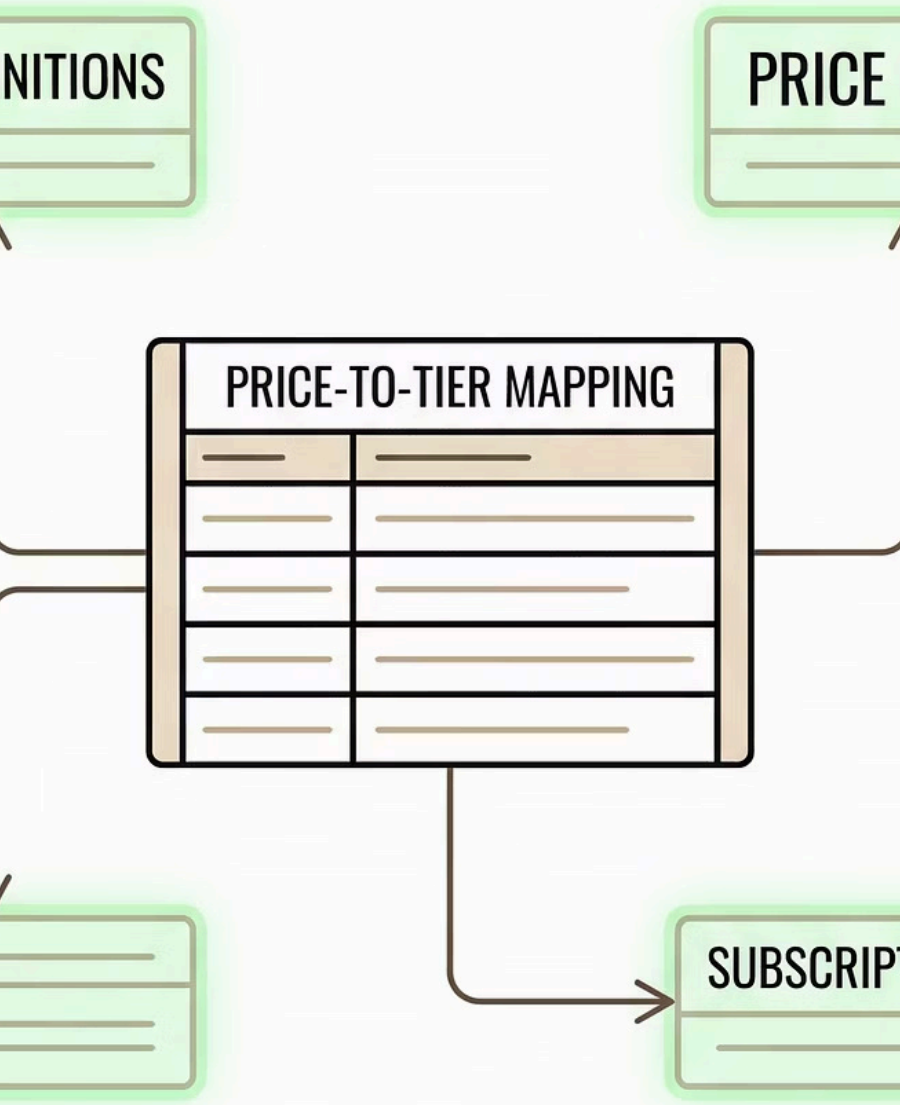
- Kein Timestamp-Check — alte Events wurden akzeptiert
- Kein Signatur-Check auf fehlende Werte
- Jeder konnte gefälschte POST-Requests senden
- Replay-Attacks möglich: gleiche Zahlung mehrfach einschleusen

Nachher ✅ — Abgesichert

- Replay-Attack-Schutz: 5-Minuten Timestamp-Toleranz
- Prüfung auf fehlende timestamp/signature Werte → sofort abgelehnt
- Stripe-Signatur wird kryptografisch verifiziert
- Gefälschte oder veraltete Events werden blockiert

> Ohne diesen Fix hätte ein Angreifer kostenlos Abos aktivieren können — indem er eine echte Zahlungsbestätigung einfach nochmal schickt.





DATENBANK

Fix #2 + #3: Fehlende Tabellen & Tier-Mapping

Zwei Datenbankprobleme auf einmal gelöst: Drei fehlende Tabellen wurden angelegt — und der Webhook weiß jetzt, welcher Plan nach einer Zahlung aktiviert werden soll.

💡 Ulrichs Erklärung: Was ist Tier-Mapping?

Wenn jemand bei Stripe bezahlt, schickt Stripe nur eine kryptische Price-ID wie 'price_1ABC123'. Die App muss wissen: Welcher Plan steckt dahinter — Starter, Pro oder Business? Das Tier-Mapping ist wie eine Übersetzungstabelle: price_1ABC123 = 'starter'. Ohne diese Tabelle wusste das Dashboard nach dem Kauf nicht, welchen Plan der Kunde hat — und zeigte 'Kein aktives Abo'.

Fix #2: sql/phase8-missing-tables.sql

- gt_lead_notes — Admin-Notizen, Cron-Logs, Webhook-Events
- gt_contact_requests — Kontaktanfragen aus dem Formular
- gt_calculations — gespeicherte Kalkulationen
- Inkl. RLS Policies, Indexes, service_role Grants
- ⚠️ Muss noch im Supabase SQL Editor ausgeführt werden!

Fix #3: Webhook → subscription_tier

- PRICE_TO_TIER Map: Price-ID → Tier-Name (starter/pro/business)
- checkout.session.completed: Tier aus metadata.tier_id oder Stripe API
- customer.subscription.updated: Tier bei Upgrade/Downgrade aktualisiert
- subscription_status: "active" wird korrekt gesetzt
- Dashboard zeigt jetzt den richtigen Plan-Namen

> ⚠️ Aktion erforderlich: sql/phase8-missing-tables.sql im Supabase SQL Editor ausführen — Reihenfolge: phase4 → phase5-7 → phase8.

ABSCHLUSS-FIXES

Fix #4 + #5 + #6: Add-Ons, .env & Passwort

Drei weitere kritische Punkte in einem Durchgang: Add-On-Checkout funktioniert jetzt korrekt, alle Env-Variablen sind dokumentiert, und das hardcodierte Admin-Passwort ist entfernt.

💡 Ulrichs Erklärung: Was ist der Unterschied zwischen Abo und Einmalkauf?

Ein Abo (subscription) wird monatlich abgebucht — wie Netflix. Ein Einmalkauf (one_time / payment) wird einmal bezahlt — wie ein Buch kaufen. Add-Ons sind Einmalkäufe: man kauft z.B. 'Extra-Sprachen' einmalig dazu. Der Code hatte aber für alles 'subscription' eingestellt — das hätte bedeutet, Add-Ons werden jeden Monat neu abgebucht. Jetzt wird der richtige Modus übergeben.

Fix #4: Add-On Checkout

create-checkout.js akzeptiert jetzt mode: 'payment' (Einmalkauf) oder 'subscription'. Pricing.jsx: Add-Ons haben Kauf-Buttons mit mode: 'payment'. tier_id + segment als Stripe Metadata. Loading-States + Fehler-Feedback.

Fix #5: .env.example

Von 2 auf 11 dokumentierte Variablen: SUPABASE_SERVICE_KEY, STRIPE_SECRET_KEY, STRIPE_WEBHOOK_SECRET, RESEND_API_KEY, EMAIL_FROM, APP_URL, CRON_SECRET, SEED_SECRET, VITE_ADMIN_PASSWORD

Fix #6: Hardcoded Passwort entfernt

Fallback 'guidetranslator2026' → null. Login zeigt klare Fehlermeldung wenn ENV-Variable nicht gesetzt ist. Kein Sicherheitsrisiko mehr durch Code-Leaks.

> Mit diesen 6 Fixes ist die Plattform produktionsreif — alle kritischen Sicherheits- und Funktionsprobleme sind behoben.