Appointmentix – Entwickler OnePager

Stand: 2026-02-23 | Ziel: White-Label MedSpa App + Klinik-Dashboard (DACH)

1) Systemarchitektur

Mobile App (Expo React Native) und Web-Dashboard (Vanilla JS) sprechen per REST/JSON mit einem Flask-Backend. Daten liegen in PostgreSQL (oder lokal SQLite).
Integrationen: Stripe (Checkout + Webhooks), Twilio (OTP SMS), optionaler Admin-Bereich für Multi-Klinik-Management.

2) Frontends

Mobile App (/mobile/App.js): Kliniksuche, QR/Code-Resolve, OTP-Login/Gastmodus, Shop/Cart/Checkout, Membership, Rewards, Scan-Flow.
Web Dashboard (/dashboard.html + /dashboard.js): Login, Klinik-Settings, Katalog-Editor, Kampagnen, Audit-Logs, Billing, Analytics-Charts.

3) Wichtige APIs (Backend)

```
Mobile:   GET /api/mobile/clinic-bundle   | GET /api/mobile/clinics/search
          POST /api/mobile/clinics/resolve-code
          POST /api/mobile/auth/otp/request|resend|verify
          GET /api/mobile/membership/status | POST activate|cancel
          POST /api/mobile/cart/add | POST /api/mobile/checkout/complete
Dashboard: POST /api/auth/login|register|logout | GET /api/auth/me
           GET/PUT /api/clinic/settings | GET/PUT /api/clinic/catalog
           GET/POST/PUT /api/clinic/campaigns | GET /api/clinic/audit-logs
           GET /api/analytics/summary | GET /api/billing/status|history
           POST /api/billing/create-checkout-session
Webhook:  POST /api/payments/webhook
```

4) Auth & Kommunikation

Dashboard nutzt Session-Cookie (same-origin). Backend unterstützt zusätzlich Bearer-Token (api_tokens).
Mobile nutzt aktuell payload-basierte APIs + OTP-Flow. Alle Requests laufen als HTTP/HTTPS JSON mit Retry+Timeout-Handling.

5) Billing- und Status-Sync

Stripe Checkout erzeugt Subscription. Webhook-Events (checkout.session.*, customer.subscription.*, invoice.*) mappen auf interne Subscription- und Klinik/User-Statusfelder (active/past_due/canceled/...).
Payment-Method-Config über ENV: STRIPE_CHECKOUT_PAYMENT_METHOD_TYPES (z. B. card, klarna, paypal). Apple Pay läuft über card in Stripe Checkout.

6) Go-Live Mindestanforderungen

1) HTTPS-Domain + produktiver Server (kein LAN/localhost) | 2) Postgres + tägliche Backups
3) Stripe Live Keys + Webhook Secret + Live-Testzahlung | 4) Twilio produktiv für OTP SMS
5) Monitoring/Alerting + Error Tracking | 6) Impressum, Datenschutz, AGB, AVV (DACH)

7) Aktueller Fokus

Nächster sinnvoller Ausbau: stabiles Onboarding ohne technische URL-Eingabe, OTP UX finalisieren, produktiver Checkout-Event-Funnel, Membership-Status-Ende-zu-Ende validieren.