Lokaltreu Gesamt-Roadmap 2.3.1 – final,auditierbar, resilient, Go-Live-fähig

**Version:** 2.3.1

**Status:** Final, resilient & Go-Live-fähig **Datum:** 2025-12-01

**Geltungsbereich:** MVP Lokaltreu SaaS Deutschland (fachlich), Betrieb in EU-Regionen (technisch)

**0. Einleitung & Zielbild**

Diese Roadmap beschreibt den vollständig durchdeklinierten Umsetzungsplan für dasLokaltreu-MVP von der Projektaufsetzung über Backend-/Frontend-Implementierung,Qualitätssicherung, Security- & Compliance-Nachweise bis hin zu Blue-Green-Go-Live, DR-Tests und Post-Go-Live-Hardening.

Sie übersetzt die Anforderungen aus SaaS-Beschreibung, Technischer Spezifikation v2.0,konsolidierter Spezifikation, Architektur-Empfehlung und AGENTS.md in 49 klar definierteSchritte mit jeweils messbarer Definition of Done (DoD).

Die Roadmap ist sequentiell nummeriert (1–49), aber explizit für Parallelisierung ausgelegt:

• Phase 1 (1–12) bildet das Fundament (Governance, Compliance, OpenAPI/Contracts,

UX).

• Ab Phase 2 (13–26) laufen Backend-Stream (A) und Frontend-/PWA-Stream (B, 27–

36) parallel.

• Phase 4 (37–45) schließt Qualitäts-, Performance-, Security- und Compliance-

Nachweise.

• Phase 5 (46–49) umfasst Blue-Green-Release, SLO/FinOps/Status-Page,

DR/Backups+Tombstone und Hardening.

Zentral für 2.3.1 sind:

• **Expand-Contract-Migrationspattern** (einschließlich Blue-Green-

Kompatibilitätstests),

• ein **Tombstone-basiertes DSR-/Backup-Verhalten**,

• ein konsequentes **Mock-First/Contract-Sync-Vorgehen** (Prism-Mock, Codegen, CI-

Job),

• **FinOps-Metriken** inkl. cost\_per\_tenant, • eine **öffentliche Status-Page** und

• ein formal dokumentiertes „Emergency Break**-**Glass“**-Verfahren** inkl. geprüftem

Meta-Gate (Schritt 42).

**0.1 Normative Referenzen**

Bei Widersprüchen gilt folgende Reihenfolge:

1. **SPEC** – Technische Spezifikation Lokaltreu v2.0

2. **OpenAPI** – apps/api/openapi/lokaltreu-openapi-v2.0.yaml

3. **ARCH** – Architektur-Empfehlung Lokaltreu

4. **AGENTS** – Lokaltreu AGENTS.md — Gold-Standard 5. **ROADMAP** – Dieses Dokument (2.3.1, ersetzt 2.3)

**0.2 Abhängigkeiten & Parallelisierung**

• **Strikt sequentiell:** Schritte 1–12 (Fundament, Governance, OpenAPI/Prism,

Codegen, Prototyp-UAT).

• **Backend-Stream (Phase 2):** Schritte 13–26; Start nach Abschluss 1–12.• **Frontend-/PWA-Stream (Phase 3):** Schritte 27–36; Start nach 1–12; arbeitet Mock- First gegen Prism/OpenAPI.

• **Qualität & Nachweise (Phase 4):** Schritte 37–45; schließen Test-, Resilienz- und

Compliance-Lücken.

• **Go-Live & Betrieb (Phase 5):** Schritte 46–49; Blue-Green, SLO/FinOps/Status-Page,

DR/Backups/Tombstones, Hardening.

0.3 Änderungen 2.3 → 2.3.1 (Kurzüberblick)

Gegenüber 2.3 sind u. a. folgende Punkte normativ geändert bzw. verschärft:

• **Migrationen (Schritt 13):**

o Verpflichtendes Expand-Contract-Pattern (schema-kompatible Erweiterung,

Backfill, verzögertes Entfernen),

o Blue-Green-Kompatibilitätstests: alte API-Version muss gegen neues Schema

durch Smoke-Tests laufen.

• **DSR & Backups (Schritte 2, 23, 48):**

o Einführung einer expliziten deleted\_subjects-Tombstone-Tabelle, o Backups werden nicht selektiv geändert;

o nach Restore wird die Tombstone-Liste erneut angewendet, um gelöschte

Subjekte wieder zu löschen/pseudonymisieren.

• **Mock-First & Contract-Sync (Schritte 10, 11, 27, 37):** o Prism-Mock-Server als offizielles Mock-Backend, o realistische Examples in OpenAPI,

o CI-Job contract-sync-frontend (OpenAPI → Types/Client → Frontend-

Build),

o wöchentliche Contract-Sync-Meetings Backend/Frontend. • **FinOps (Schritte 8, 25, 47, 48):**

o Metrik cost\_per\_tenant (laufend beobachtet), o FinOps-Dashboards (Infra-Kosten, Plan-Nutzung, Alerts bei Kostenanstiegen).• **Status & Kommunikation (Schritt 47):**

o Öffentliche Status-Page (status.lokaltreu.de) mit automatisierten Checks

und Incident-Templates.

• **Governance (Schritte 4, 42):**

o Dokumentiertes „Emergency Break-Glass“-Verfahren in AGENTS,o Schritt 42 verifiziert, dass Break-Glass nur einmal, auditierbar und mit automatischen Follow-Up-Tickets existiert.

Phase 1 – Fundament, Governance & UX-Discovery

**Schritt 1: Projekt-Rahmen fixieren**

**Ziel/Kontext**

Projekt fachlich und technisch sauber rahmen: Vision, Scope, Nicht-Ziele, NFRs, Sicherheits-und DSGVO-Leitplanken.

**Ergebnisse / Artefakte**

• /docs/01-Project-Canvas.md

• /docs/02-Processes.md (Onboarding, Stempel, Prämien, Referral, DSR, Betrieb) • /docs/03-NFRs.md (Latenz, SLO, RPO/RTO)

• /docs/04-Security-and-Abuse.md (Missbrauchsszenarien, Gegenmaßnahmen) • /docs/05-Compliance.md (DSGVO-Grundlagen, Logs, Lösch-/Speicherfristen)

**Definition of Done (DoD)**

• Geltungsbereich fachlich: Deutschland; technische Datenhaltung: EU-Regionen,

explizit dokumentiert. • NFRs definiert, z. B.:

o p50 ≤ 500 ms, p95 ≤ 3.000 ms, p99 ≤ 6.000 ms je Route,o SLO 99,90 %/30 Tage für /stamps/claim und /rewards/redeem,o RPO 15 Min, RTO 60 Min.

• Architekturprinzipien (modularer Monolith, PWA-First, EU-PaaS) beschrieben.• Dokumente im Repo versioniert, bauen fehlerfrei (z. B. Docusaurus/Markdown-Lint) und sind in CI verlinkt.

**Security / Compliance**

• Rechtsgrundlagen (insb. Art. 6(1)(f) DSGVO) und Missbrauchsprävention

beschrieben.

• Logs als personenbezogene Daten klassifiziert; geplante Retention 180 Tage

festgehalten.

**Umsetzungshinweise**

• Inhalte schlank, fokus auf Entscheidungen statt Prosa.

• Erstes ADR (z. B. /docs/ADR/0001-single-admin-saas.md).

**Owner**

• Primär: Product Owner / Domain-Owner • Sekundär: Tech Lead

**Voraussetzungen**

• keine

**Schritt 2: Compliance-Skeleton inkl. Backup-/DSR-Konzept**

**Ziel/Kontext**

Zentrale Compliance-Artefakte früh anlegen – inkl. Strategie für DSR und Backups mit

Tombstone-Konzept.

**Ergebnisse / Artefakte**

• /compliance/AVV.md • /compliance/TOMs.md • /compliance/RoPA.md • /compliance/DPIA.md

• /compliance/Infos-DE.md

• /compliance/Retention-Policy.md

• Abschnitt „Backups & DSR“ in DPIA/Retention-Policy:

o Backups werden nicht selektiv editiert,

o DSR-Löschungen werden später über eine deleted\_subjects-Tombstone-

Liste kompensiert.

• /docs/runbooks/Incident-Breach.md

• CI-Job gdpr-compliance.yml (prüft Existenz & Grundstruktur der Dokumente)

**Definition of Done (DoD)**

• Rechtsgrundlagen, Retention 180 Tage, Art.-11-Pfade grob definiert.• Tombstone-Konzept textlich beschrieben (auch wenn technisch in Schritten 23/48 umgesetzt).

• CI-Job gdpr-compliance.yml läuft und schlägt rot bei fehlenden/defekten

Dokumenten.

**Security / Compliance**

• Zielbild: WORM-Audit, signierte Exporte, 180-Tage-Log-Aufbewahrung.

• DSR-/Backup-Strategie konsistent über AVV, DPIA, RoPA, Retention-Policy.

**Umsetzungshinweise**

• Platzhalter „TODO: Konkretisierung in Phase 4/5“ zulässig, solange Struktur klar und

prüfbar ist.

• Format konsistent halten (Nummerierung, Versionierung) für spätere Audits.

**Owner**

• Primär: Audit-Officer

• Sekundär: Docs-Keeper

**Voraussetzungen**

• Schritt 1

**Schritt 3: Monorepo anlegen**

**Ziel/Kontext**

Einheitliche Codebasis für Frontend, Backend, Shared Types, IaC und Dokumentation.

**Ergebnisse / Artefakte**

• Verzeichnisstruktur, z. B.:

o apps/web (Admin + PWA) o apps/api

o packages/typeso infra/terraformo docs/ADR

• Basis-Tooling: ESLint, Prettier, TypeScript-Konfiguration, Test-Runner

**Definition of Done (DoD)**

• npm run build / pnpm build für alle Workspaces grün.• Workspaces korrekt konfiguriert (PNPM/Turborepo).• Keine Secrets im Repo; SOPS-Policy dokumentiert.

**Security / Compliance**

• .gitignore und Secret-Strategie (SOPS, CI-Secrets) definiert.

**Umsetzungshinweise**

• Früh ADR „Monorepo + TypeScript überall“ anlegen.

**Owner**

• Primär: Tech Lead

• Sekundär: Docs-Keeper

**Voraussetzungen**

• Schritte 1–2

**Schritt 4: Governance-Modell & CI-Gates + Break-Glass**

**Ziel/Kontext**

AGENTS.md als Governance-Grundlage etablieren, CI-Gates aktivieren und einen restriktiven Notfallpfad („Break-Glass“) definieren.

**Ergebnisse / Artefakte**

• AGENTS.md mit:

o Rollen (Contract-Sheriff, Test-Pilot, Device-Proof-Engineer, Audit-Officer

etc.),

o zentraler PR-Checkliste,

o Abschnitt „Emergency Break**-**Glass Deployment“ mit: ▪ zulässigen Gründen (kritische Security-Lücke, massiver Incident), ▪ berechtigten Personen/Rollen,

▪ Ablaufschema inkl. Log- und Ticketpflichten, ▪ Pflicht-Nacharbeiten mit Frist (Technical-Debt-Ticket).• .github/workflows/ci.yml (Basis-CI)

• gdpr-compliance.yml, security-gates.yml • Branch-Protection mit „required checks“

**Definition of Done (DoD)**

• CI-Gates aktiv:

o Lint, Build, Tests,

o erste Coverage-Schranke, o schema\_drift-Check,

o GDPR-/Security-Checks, o Terraform fmt/validate.

• AGENTS-PR-Checkliste in jedem PR verlinkbar.• Break-Glass-Verfahren beschrieben inkl.: o 2-Faktor-Freigabe (z. B. 2 Maintainer),

o automatisches Ticket „Break-Glass-Folgearbeit“, o klarer Ablauf zur Wiederherstellung aller Gates.

**Security / Compliance**

• Notfallpfad dokumentiert, auditierbar und bewusst restriktiv (Nachweis gelebter „least

privilege“ im Change-Management).

**Umsetzungshinweise**

• Break-Glass als klarer Ausnahme-Pfad, nicht als Shortcut für Feature-Druck. • In Incident-Runbooks referenzieren.

**Owner**

• Primär: Tech Lead

• Sekundär: Audit-Officer, Contract-Sheriff

**Voraussetzungen**

• Schritte 1–3

**Schritt 5: CI/CD-Grundgerüst**

**Ziel/Kontext**

Stabile Basis für Build, Tests und erste Deployment-Artefakte.

**Ergebnisse / Artefakte**

• CI-Pipeline, die mindestens:

o Lint, o Build,

o Unit-Tests,

o Coverage-Report erzeugt.

• Artefakt-Speicherung (Reports, Coverage, Test-Ergebnisse).

**Definition of Done (DoD)**

• CI läuft automatisch für alle PRs und auf main.

• Coverage wird reportet (Zielwert ≥ 80 % wird in Phase 4 formell durchgesetzt). • Fehlerhafte Builds sind klar nachvollziehbar (Logs, Artefakte).

**Security / Compliance**

• Keine Secrets in CI-Logs.

• Berechtigungskonzept für Pipelines (Schreibrechte nur für Maintainer).

**Umsetzungshinweise**

• AGENTS-Recipes für Test-Pilot und Contract-Sheriff verwenden.

**Owner**

• Primär: Test-Pilot

• Sekundär: Contract-Sheriff

**Voraussetzungen**

• Schritte 3–4

**Schritt 6: IaC-Bootstrap EU**

**Ziel/Kontext**

Basis-Infrastruktur deklarativ in Terraform (o. ä.) beschreiben, EU-Regionen erzwingen.

**Ergebnisse / Artefakte**

• infra/terraform mit Grundmodulen:

o Netzwerk, App-Runtime, DB, Redis, Storage, Mail, ggf. CDN • Remote-State mit SOPS-verschlüsselten Variablen

**Definition of Done (DoD)**

• terraform fmt und terraform validate grün.

• EU-Regionen in Provider-Konfiguration erzwungen (z. B. region = "eu-central-

1").

• keine Klartext-Secrets in State/Code.

**Security / Compliance**

• EU-Datenhaltung in IaC verankert, in Compliance-Doku referenziert.

**Umsetzungshinweise**

• Zunächst nur „dev“; „stage/prod“ später aus den gleichen Modulen abgeleitet.

**Owner**

• Primär: Infra-Engineer

• Sekundär: Audit-Officer

**Voraussetzungen**

• Schritte 3–5

**Schritt 7: Provider in EU**

**Ziel/Kontext**

Konkrete PaaS-/DB-/Cache-/Storage-/Mail-/CDN-Provider auswählen, dokumentieren und inIaC abbilden.

**Ergebnisse / Artefakte**

• Providerliste inkl. Regionen (EU).

• DPAs/AVVs als Links/Referenzen in Compliance-Doku.

• IaC-Konfiguration der Provider (z. B. Module, Workspaces, Tags).

**Definition of Done (DoD)**

• Provider im IaC referenziert, in Compliance-Doku erwähnt.

• Mindestens eine dev-Umgebung erfolgreich provisioniert.

**Security / Compliance**

• Verträge mit EU-Datenhaltung und DSGVO-Konformität dokumentiert.

**Umsetzungshinweise**

• Kostenmodelle früh prüfen (Grundlage für FinOps-Metriken in Schritten 8/47/48).

**Owner**

• Primär: Infra-Engineer

• Sekundär: Audit-Officer

**Voraussetzungen**

• Schritte 1–6

**Schritt 8: Observability-Basis (Dev/Stage) inkl. FinOps**

**Ziel/Kontext**

Frühe technische und einfache Kosten-Metriken etablieren.

**Ergebnisse / Artefakte**

• OTel-Integration (API + PWA). • Dashboards für:

o Latenz p50/p95/p99, o Fehler-Rate,

o Replay-Failures, 429-Rates,

o Kernkostenkomponenten (DB-Storage/IO, Redis-Operationen, Object-Storage,

Traffic).

• Basis zur Kennzahl cost\_per\_tenant (z. B. geschätzte Monatskosten / aktive

Tenants).

**Definition of Done (DoD)**

• Mindestens ein Observability-Dashboard für dev/stage aktiv.• Hot-Routen /stamps/claim und /rewards/redeem sind sichtbar (Throughput, Latenz, Fehler).

• Zeitreihe für cost\_per\_tenant und Kostenkomponenten vorhanden.

**Security / Compliance**

• Logs und Metriken ohne PII; Tenant-, Geräte- und Card-IDs genügen.

**Umsetzungshinweise**

• Tags/Labels verwenden, um Lasttest-Metriken später klar zu separieren.• Identische Panel-Struktur für dev/stage, früh anschlussfähig für prod-Dashboards (Schritt 47).

**Owner**

• Primär: Test-Pilot

• Sekundär: Audit-Officer

**Voraussetzungen**

• Schritte 3–7

**Schritt 9: Runbooks initial**

**Ziel/Kontext**

Standardisierte Abläufe für kritische Betriebsereignisse definieren.

**Ergebnisse / Artefakte**

• Runbooks:

o JWKS-Rotation.md & JWKS-Rollback.md, o Restore.md,

o Replay-Suspected.md,

o Incident-Breach.md (inkl. 72h-Pfad),

o Referenz auf Break-Glass-Verfahren aus AGENTS.

**Definition of Done (DoD)**

• Runbooks im Repo, verlinkt in Compliance- und Betriebsdokumentation.• Mindestens ein „Trockenlauf“ je kritischem Runbook gedanklich durchgeführt und dokumentiert.

**Security / Compliance**

• Verantwortlichkeiten pro Runbook (Owner, Eskalationspfad) klar beschrieben. • Incident-Pfad zur Meldung an Aufsichtsbehörde (72h) dokumentiert.

**Umsetzungshinweise**

• Runbooks an SPEC/Architektur-Anhänge anlehnen; bei Provider-Änderungen

aktualisieren.

**Owner**

• Primär: Audit-Officer • Sekundär: Tech Lead

**Voraussetzungen**

• Schritte 1–2, 4, 7

**Schritt 10: OpenAPI 3.1 SSOT + Prism-Mock**

**Ziel/Kontext**

HTTP-API als Single Source of Truth definieren; realistische Examples ermöglichen Mock-First-Entwicklung.

**Ergebnisse / Artefakte**

• apps/api/openapi/lokaltreu-openapi-v2.0.yaml (Admins, Devices, Stamps,

Rewards, Referrals, DSR, Reporting).

• Fehlerformat application/problem+json nach RFC 7807 mit error\_code-Enum. • Realistische Examples für:

o Stamps/Rewards (inkl. Anti-Replay, Rate-Limit), o DSR (inkl. Art. 11-Flows),

o Reporting.

• Prism-Mock-Setup:

o apps/api/mock-server,

o Script npm run mock:api (oder npm run dev:mock-api).

**Definition of Done (DoD)**

• OpenAPI lintet ohne Fehler (Spectral grün).

• Alle 4xx/5xx-Responses referenzieren Problem+JSON.• Hot-Routen vollständig und mit Beispielen modelliert (inkl. 422 vs. 409 vs. 401/403/429).

• Prism-Mock-Server läuft lokal mit den Examples (manuell geprüft).

**Security / Compliance**

• SecuritySchemes (AdminAuth, DeviceKey + X-Device-Proof) modelliert. • Keine PII in Examples (nur pseudonyme IDs).

**Umsetzungshinweise**

• SPEC-Prozesse/Sequenzen in Pfade übersetzen, Examples aus SPEC/OpenAPI

konsistent halten.

• Mock-Server-Command in AGENTS.md dokumentieren.

**Owner**

• Primär: Contract-Sheriff

• Sekundär: ProblemJSON-Arbiter

**Voraussetzungen**

• Schritte 1–3

**Schritt 11: Codegen & Typen + Contract-Sync-Check**

**Ziel/Kontext**

Gemeinsame Typbasis und automatischer Kompatibilitäts-Check zwischen Contract undFrontend.

**Ergebnisse / Artefakte**

• packages/types aus OpenAPI generiert (z. B. openapi-typescript). • Gemeinsamer HTTP-Client für API-Zugriffe.

• CI-Job contract-sync-frontend, der:

o OpenAPI → Types/Client generiert, o Frontend-Build ausführt.

**Definition of Done (DoD)**

• Keine manuell gepflegten API-Types im Frontend (Lint-Regel no-manual-api-

types).

• CI-Job contract-sync-frontend als „required check“ aktiv.

• Build schlägt rot, wenn Frontend nicht zu aktuellem Contract passt.

**Security / Compliance**

• Generierte Files enthalten keine Secrets/Test-Credentials.

**Umsetzungshinweise**

• AGENTS-Recipe „OpenAPI → Types“ nutzen.

**Owner**

• Primär: Contract-Sheriff • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritt 10

**Schritt 12: UX/UI-Design & Prototyping + frühes UAT**

**Ziel/Kontext**

Kern-User-Flows früh mit echten Zielnutzern testen („radikale Einfachheit“), bevor das Datenmodell finalisiert wird.

**Ergebnisse / Artefakte**

• Figma/Prototyp:

o Admin-Onboarding + erste Kampagne (< 5 Minuten),o Mitarbeiter-UI mit genau zwei Hauptaktionen (Stempel vergeben, Prämie einlösen).

• UAT-Protokoll (3–5 Zielnutzer:innen).

• Liste von UX-Entscheidungen mit Auswirkungen auf Domain/Datenschema.

**Definition of Done (DoD)**

• Alle Testpersonen schaffen US-1 (Onboarding+erste Kampagne) ohne Hilfe. • Pain Points dokumentiert, Prioritätenliste erstellt.

• Prototyp optional gegen Prism-Mock verprobt (mind. ein kompletter Flow).

**Security / Compliance**

• Prototyp enthält keine echten Produktionsdaten.

**Umsetzungshinweise**

• Prototyp und Learnings in SPEC/ARCH referenzieren.

• UX-Entscheidungen in Schritt 13 explizit berücksichtigen.

**Owner**

• Primär: Product Owner / UX • Sekundär: Tech Lead

**Voraussetzungen**

• Schritte 1–11

Phase 2 – Architektur & Backend (StreamA)

*(Start nach Abschluss von 1*–*12)*

**Schritt 13: Datenmodell & Migrationen (Expand-Contract)**

**Ziel/Kontext**

Robustes Datenmodell mit migrationssicherem Expand-Contract-Pattern und Blue-Green-Kompatibilität.

**Ergebnisse / Artefakte**

• Migrationen für Kernobjekte:

o Tenant, Device, Campaign, StampToken, Stamp, RewardToken, Reward,

Referral, AuditLog, PlanCounter.

• „Schema-Compatibility-Checkliste“ pro Migration (Welche API-

Versionen/Schemastände sind kompatibel?).

**Definition of Done (DoD)**

• Alle neuen Migrationen folgen Expand-Contract:

o Phase 1 (Expand): Schema-Erweiterung kompatibel zur alten Version,o Phase 2 (Code-Rollout): neuer Code funktioniert mit altem + neuem Schema, Backfill als Hintergrundjob,

o Phase 3 (Contract): Entfernen alter Strukturen erst nach vollständiger

Umstellung (inkl. Backfill-Monitoring). • Blue-Green-Simulation in Dev/Stage:

o Alte API-Version läuft gegen neues Schema,

o Smoke-Tests grün (Hot-Routen, DSR, Reporting).

**Security / Compliance**

• Multi-Tenant-Isolation (tenant\_id Pflicht in relevanten Tabellen). • Audit-Log-Tabellen WORM-fähig (Schritt 24).

**Umsetzungshinweise**

• UX/Prototyp-Erkenntnisse aus Schritt 12 ins Modell zurückspiegeln (z. B.

Kampagnen-Varianten, Plan-Features).

• ERD aus SPEC konsolidiert übernehmen.

**Owner**

• Primär: Tech Lead

• Sekundär: Audit-Officer

**Voraussetzungen**

• Schritte 1–12

**Schritt 14: Admin-Auth**

**Ziel/Kontext**

Sichere, kurzlebige Admin-Sitzungen mit JWT + JWKS.

**Ergebnisse / Artefakte**

• Access-Token ≤ 15 Min, Refresh-Token ≤ 30 Tage, JWKS-Endpoint. • Rotation/Rollback-Runbooks (Verweis auf Schritt 9).

**Definition of Done (DoD)**

• Rotationstests grün; alte Keys werden korrekt invalidiert.

• Admin-Sessions im Audit-Log nachvollziehbar (Login, Logout, Token-Refresh).

**Security / Compliance**

• Secure Cookies, HTTP-Only, SameSite.

• Personenbezug in Logs gem. SPEC/Art. 6(1)(f) begründet.

**Umsetzungshinweise**

• JWT-Bibliothek mit JWKS-Support wählen.

• Key-Management mit IaC/Secret-Store integrieren.

**Owner**

• Primär: Security-Engineer / Device-Proof-Engineer • Sekundär: Audit-Officer

**Voraussetzungen**

• Schritte 6–7, 9–11, 13

**Schritt 15: Geräte-Auth (Ed25519 + Proof)**

**Ziel/Kontext**

Gerätegebundene Sicherheit für Mitarbeiter-UI und kritische Aktionen.

**Ergebnisse / Artefakte**

• Device-Registry mit Bindung an Mandant.

• Middleware für X-Device-Proof (Signatur über method|path|ts|jti).

**Definition of Done (DoD)**

• Tests für Positiv-/Negativfälle grün (gültige/ungültige Signaturen, Zeitdrift, Replay). • KPI proof\_failures\_caught = 100 % erfüllt (AGENTS).

**Security / Compliance**

• Keine Speicherung privater Keys serverseitig. • Zeitdrift ±30 s berücksichtigt (Monitoring).

**Umsetzungshinweise**

• libsodium/Ed25519 einsetzen.

• Device-Onboarding (Schritt 18) und Mitarbeiter-UI (Schritt 32) eng abstimmen.

**Owner**

• Primär: Device-Proof-Engineer

• Sekundär: Idempotency-Guardian, Test-Pilot

**Voraussetzungen**

• Schritte 7–8, 10–11, 13–14

**Schritt 16: QR/Token-Modul**

**Ziel/Kontext**

Einmalige, kurzlebige Tokens für QR-Stempel-Workflows mit deterministischem Replay-Block.

**Ergebnisse / Artefakte**

• Token-Modell: StampToken, RewardToken.

• Redis-basiertes Anti-Replay (SETNX + TTL).

**Definition of Done (DoD)**

• Jeder Token kann genau einmal eingelöst werden.

• TTL überschritten → Fehler mit Problem+JSON (z. B. TOKEN\_EXPIRED). • Parallel-Tests (10 parallele Versuche, 1×201, 9×409) grün.

**Security / Compliance**

• Keine PII in Tokens.

• Anti-Replay-Tests werden in CI ausgeführt.

**Umsetzungshinweise**

• Redis so konfigurieren, dass Operationen kurz und kosteneffizient bleiben.

**Owner**

• Primär: Idempotency-Guardian

• Sekundär: Device-Proof-Engineer, Test-Pilot

**Voraussetzungen**

• Schritte 7–8, 10–11, 13–15

**Schritt 17: Idempotenz & Rate-Limits**

**Ziel/Kontext**

Schutz vor Duplikaten und Missbrauch bei sensiblen Endpunkten.

**Ergebnisse / Artefakte**

• Middleware für Idempotency-Key (Gültigkeit 24 h).• Rate-Limits pro Tenant/IP/Card/Device gemäß SPEC (z. B. /stamps/claim 30 rpm/Card, /rewards/redeem 10 rpm/Device).

**Definition of Done (DoD)**

• Für alle Hot-Routen ist Idempotency-Key Pflicht.• Rate-Limit-Fehler liefern RFC-7807-konforme Antworten inkl. error\_code = RATE\_LIMITED und retry\_after.

**Security / Compliance**

• Anti-Abuse-Strategie durch Limits und Logs belegbar.

**Umsetzungshinweise**

• Globaler Throttler + per-Route-Anpassungen (Konfiguration versioniert).

**Owner**

• Primär: Idempotency-Guardian

• Sekundär: ProblemJSON-Arbiter

**Voraussetzungen**

• Schritte 10–11, 13–16

**Schritt 18: Geräte-Onboarding**

**Ziel/Kontext**

Sicheres, einfaches Hinzufügen von Mitarbeiter-Geräten.

**Ergebnisse / Artefakte**

• POST /devices/registration-links (Idempotent, TTL 15 Min).

• POST /devices/register/confirm mit Device-Key, Audit-Eintrag.

**Definition of Done (DoD)**

• Mini-k6-Lasttest (10 parallele Registrierungen) integriert; p95 < 3 s. • Alle Geräteereignisse im Audit-Log nachvollziehbar.

**Security / Compliance**

• E-Mail mit Security-Hinweis bei Gerätebindung.

• Lösch-DSR (Schritt 23) entfernt/pseudonymisiert Gerätebezüge konsistent.

**Umsetzungshinweise**

• Admin-UI-Flows (Schritt 29) und Mail-Templates (Schritt 26) konsistent halten.

**Owner**

• Primär: Device-Proof-Engineer • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 13–17

**Schritt 19: Stempelvergabe (Hot-Route)**

**Ziel/Kontext**

Kernprozess: sichere, nachvollziehbare Vergabe von Stempeln via QR.

**Ergebnisse / Artefakte**

• POST /stamps/tokens → QR-Token.

• POST /stamps/claim → ACID-Transaktion (INSERT Stamp + Audit).

**Definition of Done (DoD)**

• Anti-Replay-Tests grün (10 parallele Claims, 1×201, 9×409).• KPIs in Observability (Anzahl Claims, Fehlerraten, Rate-Limits).• Mini-k6-Lasttest (10 parallele Claims, p95 < 3 s in Stage) grün.

**Security / Compliance**

• Keine PII; nur tenant\_id/card\_id/device\_id. • Tenant-Isolation vollständig.

**Umsetzungshinweise**

• Idempotenz und Anti-Replay klar entkoppeln (Middlewares + DB-Transaktion).

**Owner**

• Primär: Idempotency-Guardian, Audit-Officer • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 13–18

**Schritt 20: Prämieneinlösung (Hot-Route)**

**Ziel/Kontext**

Sichere Einlösung von Prämien ohne Doppelbuchung, mit klaren Fehlerbildern.

**Ergebnisse / Artefakte**

• POST /rewards/redeem mit Device-Proof und ACID-Transaktion. • Audit-Events reward.redeemed.

**Definition of Done (DoD)**

• Ungültiger Proof → 403; abgelaufene Tokens → 400; Doppelredeem → 409.• Mini-k6-Lasttest für Redeem (10 parallele Requests, p95 < 3 s).• Parallel-Tests zeigen korrekte Sperrung bei Doppelredeem.

**Security / Compliance**

• Rate-Limits pro Device/Card.

• Kein monetärer Wert, nur Stempel/Prämien (vermeidet Zahlungsdienste-Regime).

**Umsetzungshinweise**

• Businessregeln mit Mitarbeiter-UI (Schritt 32) synchronisieren.

**Owner**

• Primär: Idempotency-Guardian

• Sekundär: Device-Proof-Engineer, Test-Pilot

**Voraussetzungen**

• Schritte 13–19

**Schritt 21: Referral-Modul**

**Ziel/Kontext**

Stempelbasiertes „Kunden-werben-Kunden“ gemäß SaaS-Modell ohne Auszahlungen.

**Ergebnisse / Artefakte**

• GET /referrals/link generiert Referral-Links (Plan-Gate aktiv). • Logik:

o Qualifizierung beim ersten Stempel des Geworbenen,

o Bonus-Stempel für Werber,o Self-Referral-Block,o Velocity-Limits.

**Definition of Done (DoD)**

• Self-Referral technisch verhindert (422 SELF\_REFERRAL\_BLOCKED).• Limits pro Zeitraum und Card in Tests verifiziert.• Referral-Events im Audit-Log.

**Security / Compliance**

• Kein Geldfluss, rein stempelbasiert.

• Anti-Sybil-Mechanismen dokumentiert.

**Umsetzungshinweise**

• Enge Kopplung mit Plan-Gates (Starter ohne Referral; s. Schritt 22).

**Owner**

• Primär: ProblemJSON-Arbiter • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 13–20

**Schritt 22: Plan-Enforcement-Modul**

**Ziel/Kontext**

Plan-Limits (Stempelkontingent, Geräteanzahl) und Feature-Gates (Referral, Angebote)serverseitig erzwingen.

**Ergebnisse / Artefakte**

• PlanCounter-Tabelle.

• Middleware für Plan-Prüfungen.

• Plan-Warnmails (80 %, 100 %) und UI-Banner.

**Definition of Done (DoD)**

• Kein Hard-Block bei 100 % der Stempel; nur Soft-Limits + Upgrade-Option. • Referral bei Starter → 403 PLAN\_NOT\_ALLOWED in Backend und UI.

**Security / Compliance**

• Transparente Kommunikation der Limits; keine versteckten Sperren.

**Umsetzungshinweise**

• Plan-Logik in SPEC/Preismodell spiegeln.

**Owner**

• Primär: ProblemJSON-Arbiter • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 13–21

**Schritt 23: DSR-Workflow inkl. Tombstone-Tabelle**

**Ziel/Kontext**

DSR gemäß Art. 11 DSGVO implementieren und Backups via Tombstone-Strategieabdecken.

**Ergebnisse / Artefakte**

• /dsr/\*-Endpoints (Anfrage, Status, Erfüllung).

• Tabelle deleted\_subjects (Tombstone-Liste, inkl. Subjekt-Identifier, Löschgrund,

Zeitpunkt).

• Prozess-Doku für DSR inkl. Verhalten gegenüber Backups und Restore-Szenarien.

**Definition of Done (DoD)**

• Jede Lösch-DSR erzeugt einen Tombstone-Eintrag. • DSR-Prozess beschreibt explizit:

o Backups werden nicht selektiv geändert,

o im Restore-Fall wird die Tombstone-Liste erneut angewendet (Subjekte werden nach Restore erneut gelöscht/pseudonymisiert).• DPIA/Retention-Policy/AVV/RoPA aktualisiert, Tombstone-Verfahren konsistent beschrieben.

**Security / Compliance**

• Datenminimierung, Art.-11-Pfad ohne zusätzliche Identifizierung. • Konsistenter Umgang mit Subjekten in Logs/Audit/Backups.

**Umsetzungshinweise**

• Saubere Trennung zwischen technischer Unveränderlichkeit des Backups und

logischer Löschung nach Restore.

• Standard-Antworttexte für Fälle fehlender Identifizierbarkeit und für Informationen

zur Backup-Behandlung.

**Owner**

• Primär: Audit-Officer

• Sekundär: Docs-Keeper

**Voraussetzungen**

• Schritte 2, 13–22

**Schritt 24: Audit-Log WORM 180 Tage**

**Ziel/Kontext**

Unveränderliche Nachweise für kritische Ereignisse mit 180-Tage-Retention.

**Ergebnisse / Artefakte**

• WORM-Audit-Tabelle.

• Export-Prozess nach Object-Storage (z. B. R2/S3) als signierte Snapshots alle ≤ 15

Min.

**Definition of Done (DoD)**

• Alert bei Export-Lücke > 15 Min. • audit\_gaps = 0 (AGENTS-KPI).

**Security / Compliance**

• Zugriff nur nach „least privilege“.

• 180-Tage-Aufbewahrung in Compliance-Dokumenten spezifiziert.

**Umsetzungshinweise**

• Hash-Kette über Audit-Einträge als Integritätsnachweis.

**Owner**

• Primär: Audit-Officer

• Sekundär: Infra-Engineer

**Voraussetzungen**

• Schritte 2, 7–8, 13–23

**Schritt 25: Reporting-APIs**

**Ziel/Kontext**

Zeitreihen und KPIs für Admin-Dashboard bereitstellen.

**Ergebnisse / Artefakte**

• Endpunkte für Stempelzahlen, Prämien, aktive Kampagnen, Plan-Nutzung, ggf.

Referral-KPIs.

• Konsistenz mit Observability-Metriken (< 1 % Abweichung).

**Definition of Done (DoD)**

• Alle Kennzahlen im Admin-Dashboard (Schritt 30) werden durch Reporting-API

abgedeckt.

• Performance der Reporting-Endpunkte im Rahmen der NFRs.

**Security / Compliance**

• Aggregierte Daten, keine PII.

**Umsetzungshinweise**

• Ggf. Hintergrundjobs für aggregierte Reports planen; FinOps-KPIs für

cost\_per\_tenant vorbereiten.

**Owner**

• Primär: Test-Pilot

• Sekundär: Audit-Officer

**Voraussetzungen**

• Schritte 13–24

**Schritt 26: Mail-Integration**

**Ziel/Kontext**

Transaktionale Mails (Security-Alerts, Plan-Warnungen, Einladungen) über EU-Mail-Provider versenden.

**Ergebnisse / Artefakte**

• SMTP/API-Integration mit EU-Mail-Provider (z. B. Mailjet/Brevo). • Templates für:

o Security-Alerts (Gerätebindung),o Plan-Warnungen (80 %, 100 %),o Einladungsmails.

**Definition of Done (DoD)**

• Mails werden in dev/stage erfolgreich gesendet.

• Bounces/Fehler werden geloggt und sind im Audit nachvollziehbar.

**Security / Compliance**

• DPA/AVV mit Mail-Provider dokumentiert. • Keine sensiblen Inhalte in Betreff/Headers.

**Umsetzungshinweise**

• Mails in Runbooks verlinken (z. B. Plan-Warnung → Upgrade-Prozess, Incident-

Meldung).

**Owner**

• Primär: Infra-Engineer • Sekundär: Docs-Keeper

**Voraussetzungen**

• Schritte 2, 7, 13–22

Phase 3 – Frontend & PWA (Stream B)

*(Start nach 1*–*12)*

**Schritt 27: UI-Fundament Web/PWA (inkl. Mock-Mode)**

**Ziel/Kontext**

Solider technischer Unterbau für Admin-Dashboard und PWA mit Mock-First-Option.

**Ergebnisse / Artefakte**

• Next.js/React/TS/Tailwind-Setup.

• Component-Library (Buttons, Inputs, Modals, Layout-Shell). • Script npm run dev:mock, das:

o apps/api/mock-server (Prism) startet, o das Frontend gegen den Mock betreibt.

**Definition of Done (DoD)**

• npm run dev für „real“ + npm run dev:mock für Mock-Mode funktionieren. • UI-Komponenten bau- und testbar, Linter & Tests in CI.

• Frontend ist nicht von laufendem Backend abhängig, um Flows zu entwickeln.

**Security / Compliance**

• Keine realen Produktionsdaten im Mock-Mode.

• CSP-Basisregeln, sichere Defaults (z. B. rel="noopener").

**Umsetzungshinweise**

• Mock-Daten eng an OpenAPI-Examples koppeln. • Design-System an Prototyp (Schritt 12) anlehnen.

**Owner**

• Primär: Frontend-Lead

• Sekundär: Contract-Sheriff

**Voraussetzungen**

• Schritte 3, 10–12

**Schritt 28: Admin-Onboarding & erste Kampagne (US-1)**

**Ziel/Kontext**

End-to-End-Flow für Registrierung und erste Kampagne in < 5 Min.

**Ergebnisse / Artefakte**

• Onboarding-Wizard (Registrierung, Stammdaten, erste Kampagne). • Verbindung zu /admins/register und Campaign-APIs.

**Definition of Done (DoD)**

• In UAT schaffen Admin-Testnutzer den Flow ohne Hilfe. • Alle Backend-Calls laufen über generierte Types.

**Security / Compliance**

• Passwort-Policy im UI validiert. • Datenschutzhinweise verlinkt.

**Umsetzungshinweise**

• Zugeschnitten auf Einzelunternehmer, keine komplexen Rollen.

**Owner**

• Primär: Frontend-Lead

• Sekundär: Product Owner

**Voraussetzungen**

• Schritte 10–13, 19 (für Kampagnen-Preview optional)

**Schritt 29: Admin** – **Geräteverwaltung (US-2/US-3)**

**Ziel/Kontext**

Einfache Verwaltung von Mitarbeiter-Geräten (Hinzufügen, Sperren, Löschen).

**Ergebnisse / Artefakte**

• Geräte-Liste, Einladungs-Flow (Registration-Link), Sperr/Remove-Aktionen. • Integration mit /devices/registration-links und /devices/register/\*.

**Definition of Done (DoD)**

• Ein Gerät anlegen, sperren, reaktivieren ist für Admin in 1–2 Klicks möglich. • Fehler (TTL abgelaufen etc.) werden im UI klar angezeigt.

**Security / Compliance**

• Warnhinweis vor Entfernen aktiver Geräte. • Geräteänderungen im Audit-Log sichtbar.

**Umsetzungshinweise**

• UAT-Feedback aus Schritt 12 nutzen.

**Owner**

• Primär: Frontend-Lead

• Sekundär: Device-Proof-Engineer

**Voraussetzungen**

• Schritte 13–18, 27–28

**Schritt 30: Admin** – **Reporting & Dashboard (US-4)**

**Ziel/Kontext**

Kompakter Überblick über wichtigste KPIs und Trends.

**Ergebnisse / Artefakte**

• Dashboard mit Kennzahlen (Stempel/Monat, Redeems, aktive Kampagnen, Plan-

Nutzung, ggf. Referral-KPIs).

• Charts (Zeitreihen) basierend auf Reporting-APIs (Schritt 25).

**Definition of Done (DoD)**

• Daten stimmen mit Observability-Metriken überein (Spot-Checks). • PWA-Performance innerhalb LCP-Zielwerte.

**Security / Compliance**

• Nur aggregierte Werte, keine PII.

**Umsetzungshinweise**

• Fokus auf Klarheit, keine überladenen Reports.

• FinOps-relevante Werte hervorheben (Plan-Nutzung, ggf. Kosten-Proxies).

**Owner**

• Primär: Frontend-Lead • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 13, 24–25, 27–28

**Schritt 31: Admin** – **Plan-Management & Angebote (US-5)**

**Ziel/Kontext**

Pläne verwalten, Upgrade-Optionen und „aktuelles Angebot“ steuern.

**Ergebnisse / Artefakte**

• Plan-Übersicht mit Limits und Features.

• UI für Upgrade-Flow und Angebots-Feature.

**Definition of Done (DoD)**

• Plan-Limits werden verständlich visualisiert.

• Referral-Features nur bei berechtigtem Plan aktiv (Backend-Gates sichtbar).

**Security / Compliance**

• Transparenz zu möglichen Mehrkosten bei Upgrade.

**Umsetzungshinweise**

• E-Mail-Warnungen aus Schritt 26 konsistent verlinken.

**Owner**

• Primär: Frontend-Lead

• Sekundär: Product Owner

**Voraussetzungen**

• Schritte 22, 25–26, 27–28

**Schritt 32: Mitarbeiter-UI (2 Kernaktionen)**

**Ziel/Kontext**

Ultrasimple UI für Mitarbeiter mit genau zwei Aktionen: Stempel vergeben, Prämie einlösen.

**Ergebnisse / Artefakte**

• Vollbild-UI mit zwei Buttons.

• Integration mit Stempel-/Prämien-APIs.

**Definition of Done (DoD)**

• In UAT verstehen Mitarbeiter die UI ohne Erklärung.

• Fehler (Expired Token, Rate-Limit etc.) werden klar angezeigt.

**Security / Compliance**

• Device-Proof technisch zwingend.

• Kein Zugriff auf Admin-Funktionen.

**Umsetzungshinweise**

• Fokus auf Geschwindigkeit, Offline-Robustheit und Klarheit.

**Owner**

• Primär: Frontend-Lead

• Sekundär: Device-Proof-Engineer

**Voraussetzungen**

• Schritte 15–20, 27–29

**Schritt 33: Endkunden-PWA**

**Ziel/Kontext**

Einfache PWA für Endkunden: Kampagnensicht, QR-Scan, Referral-Links.

**Ergebnisse / Artefakte**

• /app/pwa/\* mit Manifest, Icons, Service-Worker. • Client-Flows für Scan, Redeem, Referral.

**Definition of Done (DoD)**

• Installierbar, offline read-only nutzbar. • NFR-Ziele für PWA eingehalten.

**Security / Compliance**

• Anonyme Endkunden, pseudonyme Card-IDs.

**Umsetzungshinweise**

• UX am Smartphone-Erlebnis ausrichten (One-hand-Use).

**Owner**

• Primär: Frontend-Lead • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 19–21, 27–32

**Schritt 34: UI für DSR-Anfragen (Art. 11)**

**Ziel/Kontext**

Endkunden-UI für DSR-Anfragen über pseudonyme Card-IDs.

**Ergebnisse / Artefakte**

• /app/pwa/dsr mit Formular, Status-Ansicht, Bestätigung. • Link aus Datenschutzhinweisen.

**Definition of Done (DoD)**

• E2E-Tests: UI → /dsr/\* → Audit-Trails grün. • Missbrauchsschutz (Rate-Limits, ggf. Captcha).

**Security / Compliance**

• Art.-11-Pfad ohne zusätzliche Identifizierung.

**Umsetzungshinweise**

• Texte mit Compliance abstimmen; Tombstone-Modell in FAQs/DS-Hinweisen

erklären.

**Owner**

• Primär: Frontend-Lead

• Sekundär: Audit-Officer

**Voraussetzungen**

• Schritte 23, 27–33

**Schritt 35: Service-Worker**

**Ziel/Kontext**

PWA-Caching-Strategie definieren und implementieren.

**Ergebnisse / Artefakte**

• SWR-Strategie für statische Assets. • network-first für API-Calls.

**Definition of Done (DoD)**

• installability: yes laut Lighthouse.

• Offline-Szenarien für Kernflows verifiziert.

**Security / Compliance**

• HTTPS-only, enger Scope, keine unsicheren Defaults.

**Umsetzungshinweise**

• Update-Flow klar definieren („Neue Version verfügbar“).

**Owner**

• Primär: Frontend-Lead • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 27–33

**Schritt 36: Fehler-Handling (Problem+JSON-Mapping)**

**Ziel/Kontext**

UI-weit konsistente Fehleranzeige auf Basis von Problem+JSON.

**Ergebnisse / Artefakte**

• Mapper von error\_code → UI-Messages. • Einheitskomponenten für Modals/Toasts.

**Definition of Done (DoD)**

• Alle relevanten Error-Codes haben UI-Texte. • correlation\_id sichtbar für Support.

**Security / Compliance**

• Keine sensiblen Detailinformationen in UI-Fehlern.

**Umsetzungshinweise**

• Mapping aus SPEC/AGENTS-Fehlerkatalog ableiten.

**Owner**

• Primär: Frontend-Lead • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 25, 27–35

Phase 4 – Qualität, Performance &Nachweise

**Schritt 37: Unit- & Contract-Tests (Coverage- & Contract-Sync-Review)**

**Ziel/Kontext**

Testabdeckung und Contract-Konformität sicherstellen, inkl. organisatorischem Contract-Sync-Ritual.

**Ergebnisse / Artefakte**

• Coverage-Report (lines/functions/branches/statements ≥ 80 %). • Contract-Tests (100 % Pass-Rate).

• Protokolle der letzten 3 wöchentlichen Contract-Sync-Meetings (Backend +

Frontend).

**Definition of Done (DoD)**

• Alle kritischen Pfade (Hot-Routen, DSR, Referral, Plan-Gates) mit Unit-/Contract-

Tests abgedeckt.

• contract-sync-frontend-Job grün.

• Keine offenen, ungeklärten Punkte aus Contract-Sync-Meetings.

**Security / Compliance**

• Testdaten sind anonym/pseudonym.

**Umsetzungshinweise**

• Contract-Sync-Meeting (15–30 Min) fest im Kalender. • Schema-Drift-KPIs aus AGENTS überwachen.

**Owner**

• Primär: Test-Pilot

• Sekundär: Contract-Sheriff

**Voraussetzungen**

• Schritte 13–26, 27–36

**Schritt 38: Integrations-/Paralleltests**

**Ziel/Kontext**

Replay-Schutz, TTL und Idempotenz unter realistischen Parallelbedingungen beweisen.

**Ergebnisse / Artefakte**

• Parallel-Suites für /stamps/claim und /rewards/redeem. • Tests für TTL-Expiry, Rate-Limits und Idempotenz.

**Definition of Done (DoD)**

• Parallel-Anti-Replay grün (1×201, 9×409). • Idempotency-Cases sauber dokumentiert.

**Security / Compliance**

• Test-Tenants isoliert.

**Umsetzungshinweise**

• Worker-Pools in CI nutzen.

**Owner**

• Primär: Idempotency-Guardian • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 16–20, 37

**Schritt 39: E2E-Tests**

**Ziel/Kontext**

User-Stories Ende-zu-Ende verifizieren (Admin, Mitarbeiter, Endkunde).

**Ergebnisse / Artefakte**

• E2E-Suites für US-1 bis US-n (Onboarding, Gerätebindung, Stempel, Redeem,

Referral, DSR-UI).

**Definition of Done (DoD)**

• Keine kritischen User-Flows ohne E2E-Test. • Smoke-Suite läuft bei jedem PR.

**Security / Compliance**

• Testaccounts klar von produktiven Accounts getrennt.

**Umsetzungshinweise**

• Tests entlang realer Customer-Journeys modellieren.

**Owner**

• Primär: Test-Pilot

• Sekundär: Product Owner

**Voraussetzungen**

• Schritte 28–36, 37–38

**Schritt 40: UAT mit Zielgruppe (finale Runde)**

**Ziel/Kontext**

Gesamtprodukt mit Zielgruppe testen, Feinschliff vor Go-Live.

**Ergebnisse / Artefakte**

• UAT-Protokolle (5–10 Zielkunden).

• Liste priorisierter Findings und UI-Verbesserungen.

**Definition of Done (DoD)**

• Keine fundamental negativen Rückmeldungen zur Bedienbarkeit. • Kritische Findings haben Tickets; Blocker gelöst.

**Security / Compliance**

• Test auf Stage mit Testdaten.

**Umsetzungshinweise**

• Ergebnisse mit Prototype-UAT (Schritt 12) vergleichen.

**Owner**

• Primär: Product Owner

• Sekundär: Frontend-Lead

**Voraussetzungen**

• Schritte 12, 28–36, 39

**Schritt 41: Lasttests Hot-Routen (mit synthetischenMassendaten)**

**Ziel/Kontext**

Performance unter realistischer Datenmenge und Last nachweisen.

**Ergebnisse / Artefakte**

• Script scripts/seed-perf-data zur Erzeugung großer Datensätze (z. B. 10.000

Tenants, 1.000.000 Stempel etc.).

• Stage-Umgebung mit diesem Seed.

• Lasttest-Reports (p50/p95/p99, Fehler, Limits) inkl. PDF/HTML-Export.

**Definition of Done (DoD)**

• Lasttests laufen auf Stage mit Massendaten-Seed, nicht mit wenigen Records. • NFR-Ziele erfüllt (p50/p95/p99).

• Engpässe dokumentiert, ggf. in Backlog-Tickets überführt.

**Security / Compliance**

• Seed-Daten rein synthetisch (keine echten Nutzer).

**Umsetzungshinweise**

• Seed-/Lasttest-Läufe klar in Observability getaggt.

• Skripte aus Schritten 19/20 wiederverwenden, Lastfaktor erhöhen.

**Owner**

• Primär: Test-Pilot

• Sekundär: Infra-Engineer

**Voraussetzungen**

• Schritte 19–20, 24–25, 37–39

**Schritt 42: Automatisierte Gates (Meta-Review & Break-Glass-Check)**

**Ziel/Kontext**

Sicherstellen, dass CI-Gates konsistent durchgesetzt werden und der Break-Glass-Pfadkontrolliert ist.

**Ergebnisse / Artefakte**

• Checkliste „CI-Gates vollständig“.

• Dokumentation eines testweise durchlaufenen Break-Glass-Flows inkl. Audit-Eintrag

und Ticket.

**Definition of Done (DoD)**

• Alle in AGENTS definierten Muss-Gates (Coverage, Contract, Error-Format, Anti- Replay, Device-Proof, Plan-Gates, Terraform, GDPR) sind als „required“ gesetzt.• Merge-Block aktiv bei Fehlerszenarien.

• Break-Glass-Prozess einmal testweise in dev/stage durchlaufen:

o erzeugt Audit-Eintrag,

o erzeugt automatisches Follow-Up-Ticket,

o alle Gates werden nach Einsatz wiederhergestellt. • Es existiert nur ein dokumentierter Break-Glass-Weg.

**Security / Compliance**

• Minimierung des Risikos „inoffizielles Bypassen von Gates“.

**Umsetzungshinweise**

• Test-Run mit fiktivem Security-Fix durchführen.

**Owner**

• Primär: Tech Lead

• Sekundär: Audit-Officer, Test-Pilot

**Voraussetzungen**

• Schritte 4–5, 13–26, 37–41

**Schritt 43: Security-Nachweise (DR/Resilienz/Plan-Gates)**

**Ziel/Kontext**

Formal nachweisen, dass Security-, DR- und Plan-Mechanismen wie spezifiziertfunktionieren.

**Ergebnisse / Artefakte**

• Berichte zu:

o JWKS-Rotation/Rollback-Übungen,o Restore-Tests vs. RPO/RTO,o Plan-Gate-Szenarien (79/80/100 %),o Referral-Gate,

o Replay/Idempotenz-Tests.

**Definition of Done (DoD)**

• Alle Nachweise in Compliance-Anhang verlinkt. • audit\_gaps = 0.

**Security / Compliance**

• Material kann externen Auditoren vorgelegt werden.

**Umsetzungshinweise**

• vorhandene Loadtest/Chaos-Reports aus SPEC referenzieren.

**Owner**

• Primär: Audit-Officer • Sekundär: Test-Pilot

**Voraussetzungen**

• Schritte 7–9, 24, 37–41

**Schritt 44: Penetration Test (Light)**

**Ziel/Kontext**

Sicherheits-„Blick von außen“ gewährleisten (automatisiert + manuell).

**Ergebnisse / Artefakte**

• OWASP-ZAP-Report (o. ä.) gegen Stage.

• Kurzbericht manueller Tests (Auth, Device-Proof, Rate-Limits, DSR-UI).

**Definition of Done (DoD)**

• Keine kritischen Findings offen; mittlere/niedrige Findings sind entweder behoben

oder begründet akzeptiert.

• Tickets zu Findings im Tracker referenziert.

**Security / Compliance**

• Vorbereitung auf externe Audits, Nachweis gelebter Security-Praxis.

**Umsetzungshinweise**

• Pen-Test gegen Stage mit produktionsähnlicher Config, aber Testdaten.

**Owner**

• Primär: Security-Engineer • Sekundär: Audit-Officer

**Voraussetzungen**

• Schritte 13–26, 27–36, 37–43

**Schritt 45: Finale Compliance-Abnahme & Audit**

**Ziel/Kontext**

Formale Freigabe durch Compliance/Datenschutz (intern oder extern).

**Ergebnisse / Artefakte**

• aktualisierte AVV, TOMs, RoPA, DPIA, Infos-DE.

• Audit-Report inkl. DSR-E2E inkl. Backup-/Tombstone-Regelung.

**Definition of Done (DoD)**

• Kein offenes Major-Finding.

• Alle DSR-Flows (inkl. Backups & Tombstone-Verhalten) dokumentiert und geprüft.

**Security / Compliance**

• System gilt als Go-Live-fähig.

**Umsetzungshinweise**

• Alle Nachweise und Reports aus vorherigen Schritten bündeln.

**Owner**

• Primär: Audit-Officer

• Sekundär: Product Owner

**Voraussetzungen**

• Schritte 2, 23–24, 37–44

Phase 5 – Go-Live & Betrieb

**Schritt 46: Blue-Green-Release**

**Ziel/Kontext**

Sichere, reversible Inbetriebnahme der MVP-Version.

**Ergebnisse / Artefakte**

• Deployment-Pipeline mit Blue-Green/Canary-Strategie. • Rollback-Plan (inkl. getesteten Runbooks).

**Definition of Done (DoD)**

• Test-Traffic erfolgreich auf neue Version geroutet; anschließend Produktions-Traffic

ohne Zwischenfälle umgelegt. • Rollback in Stage geübt.

**Security / Compliance**

• Logs/Audit auch während Deployment-Wechsel vollständig.

**Umsetzungshinweise**

• Feature-Flags für riskante Features nutzen.

**Owner**

• Primär: Infra-Engineer • Sekundär: Tech Lead

**Voraussetzungen**

• Schritte 6–7, 37–45

**Schritt 47: SLO-Dashboards & Alerts aktiv (Prod) +Status-Page + FinOps**

**Ziel/Kontext**

Betriebstransparenz für technisches Team und Kunden herstellen.

**Ergebnisse / Artefakte**

• Prod-Dashboards:

o SLO & Latenzen (p50/p95/p99), o Fehler-Rate,

o Kosten-Kennzahlen, o cost\_per\_tenant.

• Öffentliche Status-Page (z. B. status.lokaltreu.de) mit: o Komponenten (API, PWA, Admin), o automatisierten Checks,

o Templates für Incidents.

**Definition of Done (DoD)**

• Alerts aktiv für SLO-Verletzungen und auffällige Kostenanstiege.• Status-Page bleibt bei Ausfall der Hauptsysteme erreichbar (separater Provider).• Mindestens ein „Trocken-Incident“ geübt:

o Status-Page-Eintrag, o internes Review, o Lessons Learned.

**Security / Compliance**

• Status-Page enthält keine sensiblen Details, nur hochaggregierte Informationen.

**Umsetzungshinweise**

• Status-Page-Provider getrennt von Haupt-Infrastruktur wählen.

• On-Call-Plan und Bereitschaftsdienst definieren und dokumentieren.

**Owner**

• Primär: Test-Pilot

• Sekundär: Infra-Engineer

**Voraussetzungen**

• Schritte 8, 41, 46

**Schritt 48: Backups/Restore, Multi-AZ, Plan-Monitoring**

**Ziel/Kontext**

Resilienz und DSR-Konformität auch im Restore-Fall sicherstellen.

**Ergebnisse / Artefakte**

• Produktive Backup-Zeitpläne (DB, Storage). • Restore-Runbooks.

• Skript/Prozess, der nach Restore:

o Tombstone-Liste (deleted\_subjects) auf restaurierte Daten anwendet, o betroffene Subjekte erneut löscht/pseudonymisiert.

• Monitoring für Plan-Kennzahlen (Limits, Upgrades, Kosten, FinOps-KPIs).

**Definition of Done (DoD)**

• Restore-Tests zeigen Einhaltung von RPO/RTO. • „Restore + Tombstone“-Test dokumentiert:

o gelöschte Nutzer tauchen nach Restore nicht wieder auf.

• Plan-/Kosten-Metriken werden für FinOps überwacht (Alerts auf Anomalien).

**Security / Compliance**

• Zugriff auf Backups streng limitiert.

• DSR-/Backup-Verhalten entspricht dokumentierter DPIA/Retention-Policy (inkl.

Tombstone).

**Umsetzungshinweise**

• Mindestens ein vollständiger End-to-End-DR-Test vor Go-Live. • Szenarien: DB-Ausfall, AZ-Ausfall, Konfigurationsfehler.

**Owner**

• Primär: Infra-Engineer

• Sekundär: Audit-Officer

**Voraussetzungen**

• Schritte 2, 7–9, 23–24, 41–47

**Schritt 49: Post-Go-Live-Review & Hardening**

**Ziel/Kontext**

Betrieb stabilisieren, Kosten und Risiken weiter senken, nächste Iteration vorbereiten.

**Ergebnisse / Artefakte**

• /docs/postmortems/MVP-Post-Go-Live.md.• Priorisierte Hardening-Maßnahmen: o Caching,

o Reporting-Batching,

o weitere Kostenoptimierung, o Refactorings.

• aktualisiertes Operating Manual (inkl. On-Call-Handbuch).

**Definition of Done (DoD)**

• Mindestens ein strukturiertes Review-Meeting durchgeführt.• Hardening-Backlog im Ticketsystem erfasst und priorisiert.• Dokumentation (Betrieb, Compliance) an reale Betriebsprozesse angepasst.

**Security / Compliance**

• Überprüfung, ob alle Compliance-Artefakte die tatsächliche Betriebsrealität abbilden.

**Umsetzungshinweise**

• Fokus auf „Low-Hanging-Fruits“ mit hohem Risiko-/Kostenhebel. • Lessons Learned in SPEC/ARCH/AGENTS zurückführen.

**Owner**

• Primär: Tech Lead

• Sekundär: Product Owner, Audit-Officer

**Voraussetzungen**

• Schritte 46–48