Sicherheitsdienst■Tool — Entwicklungsleitfaden (Codex■optimiert)

> Ziel: Dieses Dokument ist **die Quelle der Wahrheit** für MVP, Architektur und Arbeitsweise. Es ist so geschrieben, dass **Codex (Terminal)** es direkt versteht und schrittweise umsetzen kann. Änderungen erfolgen nur per Pull Request.

1. Zweck & Rahmen

- Branchenfokus: **deutsche Sicherheitsdienste** (Objektschutz, Veranstaltungssicherheit).
- Primärziele MVP: Einsatz■/Schichtplanung, Mitarbeiter■Stammdaten, Benachrichtigungen, einfache AZG■Konformität, Basis■Mobile■App (Anzeige/Zeiterfassung/Vorfall).
- Sekundär: Stabiler Developer■Workflow (Tests, Lint, CI/CD, Docker).
Nicht■Ziele im MVP: komplexe Lohnabrechnung, anspruchsvolles OWKS, Kundenportal, KI■Optimierung (Post■MVP).

2. Leitprinzipien

- **Konzepttreue:** Dieses Dokument hat Vorrang. Bei Unklarheiten **Rückfragen** stellen, nicht raten.
- **Kleine Schritte:** Jeder Task = kleiner, überprüfbarer Diff + Tests + Doku.
- **Sicherheit vor Features:** Auth/RBAC, Validierung, Audit■Logs zuerst stabil.
- **Automatisierung:** Lint/Format/Test/Build in CI; reproducible Docker■Setup.
- **Lesbarkeit > Cleverness:** saubere Services, klare Fehlerwege, strukturierte Logs.

3. Architektur & Stack (MVP)

- **Backend:** Node.js (>=20), TypeScript, Express, Prisma ORM, PostgreSQL.
- **Validierung:** Zod (DTOs), zentrale Fehlerbehandlung.
- **Auth:** JWT (Access/Refresh), RBAC (admin/dispatcher/guard), Passwort■Hash (bcrypt).
- **Tests:** Jest (unit/integration), minimale E2E■Smoke■Tests.
- **Infra:** Docker Compose (api + db), Healthchecks, Migrations beim Start.
- **Logging:** strukturierte Logs (JSON), Korrelations■IDs; später Metrics.
- **Ordnerstruktur (Backend):**
backend/
src/
controllers/ # I/O (HTTP)
routes/ # Express-Router
services/ # Anwendungslogik
models/ # Prisma schema & mappers
middlewares/ # auth, validation, error
utils/ # helpers
app.ts
tests/ # Jest
prisma/ # schema.prisma, migrations, seed

4. Domain■Modell (MVP)

Kern■Entities:
Employee, Site, Shift, Assignment, TimeTracking, Notification, Incident (basic).
Beziehungen (vereinfacht):

- Site 1■n Shift
- Employee m■n Assignment (über Shift)
- Employee 1■n TimeTracking
- Incident 1■n zu Shift (oder Site), createdBy: Employee
Minimal■Felder (Beispiele):
- Employee: id, name, email, phone, role, qualifications[], active
- Site: id, name, address, contact, notes
- Shift: id, siteId, startsAt, endsAt, requiredRoles[], notes
- Assignment: id, shiftId, employeeId, role, status
- TimeTracking: id, employeeId, shiftId, startedAt, endedAt, pauseMinutes
- Notification: id, type, recipient, subject, payload, sentAt, status
- Incident: id, shiftId?, siteId?, createdBy, title, description, createdAt, severity

5. API v1 — Blueprint

OpenAPI: docs/openapi.yaml (Single■Source).
Grundregeln:
- REST, konsistente Pfade, klare 4xx/5xx, ProblemDetails■Stil.
- Payloads per Zod validiert; Controller schlank, Service enthält Logik.
- RBAC: admin (alle), dispatcher (Dispo), guard (eigene Infos).
Endpoints (Auszug):
- **Auth:** POST /auth/login, POST /auth/refresh, GET /me
- **Employees:** CRUD /employees
- **Sites:** CRUD /sites
- **Shifts:** CRUD /sites/{siteId}/shifts, /shifts
- **Assignments:** POST/DELETE /shifts/{id}/assignments
- **TimeTracking:** POST /shifts/{id}/clock-in, POST /shifts/{id}/clock-out
- **Incidents:** CRUD /incidents
- **Notifications:** POST /notifications/test
Akzeptanzkriterien Beispiele:
- Ein **Site■CRUD** gilt als „fertig“, wenn:
1) OpenAPI definiert + DTOs (Zod) vorhanden,
2) Controller/Service + Prisma■Model + Migration,
3) Jest■Tests (Positiv/Negativ) grün,
4) README Abschnitt mit Beispiel■Requests vorhanden.

6. Security, Compliance, Datenschutz (MVP)

- **RBAC** strikt, sensible Endpunkte nur für Rollen.
- **Input■Validation** überall; keine ungeprüften Daten an DB/API.
- **Passwörter** gehasht (bcrypt), Tokens mit sinnvollen TTLs.
- **Audit■Log (basic):** login, role changes, critical actions.
- **DSGVO■Basics:** minimale Daten, Lösch-/Sperrkonzept skizziert, Log■Rotation.
- **AZG■Konformität (Basis):** Mindestpause prüfen, offensichtliche Verstöße melden (Warnung, kein Hart■Stop im MVP).

7. Dev■Workflow

- **Scripts (package.json):** dev, build, start, test, test:watch, lint, format, migrate, seed.
- **Lint/Format:** ESLint + Prettier, .editorconfig, .gitattributes (LF).
- **CI (GitHub Actions):** Install → Lint → Test → Build. Branch■Protection für main mit Status■Checks.
- **Branching:** main (schutz) + feature/* + PR Review (2 Augen). SemVer + CHANGELOG.

8. Docker & ENV

- **Compose Services:** api, postgres (Volume, Healthcheck).
- **Start:** docker compose up -d → API nach Healthcheck startklar.
- **Migrations:** automatisch prisma migrate deploy beim API■Start.
- **.env.example (Auszug):**

```
NODE_ENV=development
DATABASE_URL=postgresql://user:pass@postgres:5432/sdtool
JWT_SECRET=
SMTP_HOST=
SMTP_USER=
SMTP_PASS=
```

9. Definition of Done (DoD)

Ein Task ist **fertig**, wenn:
1) Plan, UNIFIED DIFF, **Freigabe** erfolgt,
2) Code + Tests + Docs fertig,
3) Lint/Tests lokal grün **und** CI grün,
4) README/CHANGELOG aktualisiert,
5) Docker■Compose läuft (smoke■testbar).

10. Roadmap (nächste 2–3 Wochen)

Milestone M1 (MVP■Basis)
- [ ] OpenAPI v1 (docs/openapi.yaml) + DTO■Schemata
- [ ] Auth & RBAC Grundgerüst + Seeds + Tests
- [ ] Site■CRUD (Referenz) + Tests + README Beispiele
- [ ] Docker■Compose Healthchecks + Migrations beim Start
- [ ] CI Workflow + Branch■Protection Vorschlag

Milestone M2
- [ ] Shift + Assignment + TimeTracking Endpoints
- [ ] Benachrichtigungen (E■Mail) + Tests
- [ ] Incident basic + Tests

Milestone M3
- [ ] Mobile App Basis (Anzeige/Clock■in/out, Vorfall)
- [ ] OWKS Grundstein (QR/NFC – Platzhalter API)
- [ ] Reporting erste KPIs

11. Codex■Arbeitsmodus (Terminal)

> **Immer im Repo starten:** cd ~/project && codex
> **Approval Mode:** Änderungen erst nach Freigabe schreiben/ausführen.
Session■Start■Prompt:
Lies docs/KONZEPT.md. Bestätige die Ziele in 5 Punkten.
Erzeuge/aktualisiere docs/ROADMAP.md mit 3 Tasks für heute (≤90 Min), inkl. Akzeptanzkriterien.
Nur Diff zeigen, auf Freigabe warten.
Implementierungs■Prompt (Template):
Implementiere ROADMAP Task im backend/ gemäß docs/openapi.yaml.
Vorgehen:
1) Plan (Dateien+Befehle) anzeigen

2) UNIFIED DIFF aller Änderungen
3) Auf meine Freigabe warten
4) Befehle ausführen (lint/test/build/migrate)
5) README/docs aktualisieren
Commit-Message: "feat: "
PR■Workflow:
Analysiere pr-2..pr-6 gegen docs/KONZEPT.md (MVP/Post-MVP/irrelevant).
Empfehlung je PR. Danach integrative Branches "feature/integrate-pr-#" erzeugen,
Abweichungen korrigieren, Tests/Doku ergänzen, in main mergen.
Diff & Befehle immer zuerst zeigen.

12. Offene Fragen & Risiken

- Benachrichtigungsprovider (SMTP vs. Dienst) — Entscheidung?
- Rollenfeinheiten (dispatcher/guard) — Zugriffstiefe definieren.
- Mobile Tech■Stack (React Native?) — MVP Umfang klären.
- Datenschutz & Aufbewahrungsfristen — konkretisieren.
- OWKS/NFC/QR — Hardware/Libs und Minimalumfang planen.

13. Glossar

- **MVP:** kleinstes sinnvolles Produktinkrement für Feedback.
- **RBAC:** Rollen■/Rechte■Modell.
- **OpenAPI:** maschinenlesbare API■Spezifikation (docs/openapi.yaml).

Ende. Dieses Dokument wird versioniert. Änderungen nur via PR auf main.