

CONVOYPLAN

Betriebs- und Administrationshandbuch

Installation, Betrieb, Sicherheit, Lizenz und Schnittstellen einer ConvoyPlan-Instanz

Stand 31.10.2026 · RettTech Solutions

Über dieses Handbuch

Dieses Handbuch fasst die Dokumentation von ConvoyPlan zu einem druckbaren Dokument zusammen. Quelle sind die Markdown-Dateien in den Repositories (Wiki und `docs/`); bei Abweichungen gilt der aktuelle Stand dort.

Stand	31.10.2026
Quellen	RettTechSolutions/ConvoyPlan@1517aa2
Herausgeber	RettTech Solutions

Verweise innerhalb des Handbuchs sind anklickbar. Verweise auf Dateien und Seiten außerhalb führen nach GitHub. Die Bildschirmfotos stammen aus einer Demo-Organisation; Namen und Kennungen darin sind Platzhalter.

Inhalt

TEIL 1 · INSTALLATION & ARCHITEKTUR

1 **Installation und Setup** 7
- 1.1 Voraussetzungen 7
- 1.2 Installation mit dem Installer (empfohlen) 7
- 1.3 Setup-Wizard 8
- 1.4 Manuelle Installation (ohne Installer) 9
- 1.5 Portainer 9
- 1.6 Konfiguration 10
- 1.7 Kartenregion wechseln 15
- 1.8 Checkliste für Produktion 16
- 1.9 Nützliche Docker-Befehle 16
- 1.10 Sicherheitshinweise 17

2 **Systemübersicht** 18
- 2.1 Was erfasst wird 18
- 2.2 Historie und Auflösung 19
- 2.3 Monatsberichte exportieren 19
- 2.4 Anbindung an ein Monitoring (PRTG) 20
- 2.5 Container-Zustand (Docker) 20
- 2.6 Datenschutz 20
- 2.7 Konfiguration 20
- 2.8 Fehlerbehebung 21

3 **Multi-Tenancy** 22
- 3.1 Org-Code-Slug 22
- 3.2 Org-spezifische Login-Seite und Branding 22
- 3.3 Datenisolation 22
- 3.4 Rollen im Org-Kontext 23
- 3.5 Anlegen und Zuweisen 23

4 **🚦 Verkehrsdaten einrichten — Baustellen, Sperrungen & Stau** 24
- 4.1 📊 Überblick — welche Quelle liefert was 24
- 4.2 1. ✅ Sofort aktiv (kein Setup) 24
- 4.3 2. 🟢 Live-Verkehrslage (Stau) aktivieren — HERE oder TomTom 25
- 4.4 3. 🇩🇪 Bundesweite Baustellen über die mobilithek (DATEX II) 25
- 4.5 4. 🇦🇹🇨🇭 Baustellen in Österreich und der Schweiz 27
- 4.6 5. 🛠️ Fehlerbehebung 27
- 4.7 📋 Referenz — Umgebungsvariablen 28

TEIL 2 · LAUFENDER BETRIEB

5 **Auto-Updater** 30
- 5.1 GitHub-Token (optional) 30
- 5.2 Update-Kanäle 30
- 5.3 Update-Modi 31
- 5.4 Update-Status und manueller Trigger 32
- 5.5 Selbstheilung bei fehlgeschlagenen Deployments 32
- 5.6 Routing-Graph: kein Deploy mitten im Import 32
- 5.7 Host-Watchdog 33

6 **Backup, Restore & Verschlüsselung at-rest** 34
- 6.1 1. Was gesichert werden muss 34
- 6.2 2. Backup 34
- 6.3 3. Restore 35
- 6.4 4. Verschlüsselung at-rest (T12) 35
- 6.5 5. Content-Security-Policy (CSP) scharfschalten 36

7 **Lizenz und Demo-Modus** 37
- 7.1 Demo-Modus 37
- 7.2 Der Demo-Bereich im Admin-Portal 37
- 7.3 Kontaktangabe beim Demo-Start 38
- 7.4 Nachfrage nach Ablauf der Sitzung 38
- 7.5 Offene Demo-Sitzungen verwalten (Admin) 39
- 7.6 Lizenz beantragen und aktivieren 41
- 7.7 Ablauf ankündigen 41
- 7.8 Schlüsselquelle: Env vs. Datenbank 41
- 7.9 Was ein Lizenzschlüssel erlaubt 42
- 7.10 Pläne je Organisation (Hosting) 42
- 7.11 Module je Organisation 43
- 7.12 Lizenzmodell des Quellcodes 44

8 **Sicherheit und Datenschutz** 45
- 8.1 Härtung im Überblick 45
- 8.2 Die Sitzung im Browser 46
- 8.3 Content-Security-Policy scharfschalten 46
- 8.4 Kartenkacheln über die eigene Instanz 47
- 8.5 Passkeys 47
- 8.6 Anmeldung der Begleit-App 49
- 8.7 Audit-Log 49
- 8.8 DSGVO-Werkzeuge 49
- 8.9 KI-Schnittstelle (MCP) 50
- 8.10 Datenaufbewahrung (Retention) 51
- 8.11 Backup & Restore 51
- 8.12 Verschlüsselung at-rest 52
- 8.13 Update-Integrität 52

TEIL 3 · SCHNITTSTELLEN

9 API-Dokumentation 55

9.1 Zugangswege im Überblick 55
9.2 Authentifizierung mit Bearer-Token 56
9.3 Authentifizierung mit Org-API-Key 57
9.4 Authentifizierung mit System-API-Key 57
9.5 Rollen im API-Kontext 58
9.6 Statuscodes und Fehler 58
9.7 Ersteinrichtung (Setup-Wizard) 60
9.8 Superadmin-Verwaltung 60
9.9 Sicherheit und Datenschutz 63
9.10 Lizenz 63
9.11 Fahrzeuge 64
9.12 Marschverbände (Convoys) 64
9.13 Freigabelinks und Tracking-App 66
9.14 Live-Tracking 66
9.15 Organisationen 67
9.16 Leitstellen 68
9.17 Wetter, Sperrungen und Verkehrslage 68
9.18 System 68
9.19 Systemübersicht (Superadmin) 69
9.20 Datenmodell 71

10 MCP-Server (KI-Schnittstelle) 72

10.1 Was der Server kann 72
10.2 Was der Server nicht kann 73
10.3 Einschalten 73
10.4 Verbinden 76
10.5 Berechtigungen 77
10.6 Nachvollziehbarkeit 77
10.7 Zugang entziehen 78
10.8 Datenschutz 79
10.9 Einstellungen 79
10.10 Programme ohne Selbstregistrierung (CIMD) 80
10.11 Oberflächen in ChatGPT 80
10.12 Wenn die Rechte nicht reichen 81
10.13 Technischer Hintergrund 81

11 ConvoyPlan in ChatGPT verbinden 82

11.1 Voraussetzungen 82
11.2 Schritt 1: Developer Mode einschalten 82
11.3 Schritt 2: Die Verbindung anlegen 82
11.4 Schritt 3: Zustimmen — hier entscheidet sich alles 83
11.5 Schritt 4: Im Gespräch einschalten 83
11.6 Was du dann fragen kannst 83
11.7 Schreibende Aktionen 84
11.8 Was das für die Daten bedeutet 84
11.9 Wenn es nicht funktioniert 84
11.10 Zugang wieder entziehen 85
11.11 Was (noch) nicht geht 85

12 **Auskunft für Suchmaschinen und KI-Agenten** 87
12.1 Was ausgeliefert wird 87
12.2 Markdown statt HTML 88
12.3 Was die Auskunft über die Instanz sagt 88
12.4 Die öffentliche OpenAPI-Beschreibung 88
12.5 Werkzeuge im Browser (WebMCP) 89
12.6 Ein- und abschalten 89
12.7 Was nicht im Code liegt 89
12.8 Verwandt 89

TEIL 4 · ENTWICKLUNG

13 **Entwicklung** 91
13.1 Projektstruktur 91
13.2 Backend lokal entwickeln (ohne Docker) 92
13.3 Frontend prüfen und bauen 93
13.4 Datenbankmigrationen 93
13.5 Nützliche Docker-Befehle 93
13.6 Pre-Commit-Hook 93
13.7 Native App / PWA 94
13.8 CI und Releases 94

TEIL 1

Installation & Architektur

Installation und Setup

Systemübersicht

Multi-Tenancy

🚦 Verkehrsdaten einrichten — Baustellen, Sperrungen & Stau

KAPITEL 1

Installation und Setup

Diese Seite beschreibt die Installation von ConvoyPlan auf einem Server. Für die Arbeit am Code (Backend/Frontend lokal, Tests, Migrationen) siehe **Entwicklung**.

> **Kurz gesagt:** Den Installer benutzen. Für den Betrieb ist **kein** `git clone` nötig. Die `docker-compose.yml` zieht fertige Images aus der GitHub Container Registry und baut nichts selbst, ein `--build` hat also keine Wirkung. Wer im Repository-Klon `docker compose up` aufruft, scheitert an der fehlenden `.env` (`required variable POSTGRES_PASSWORD is missing a value`).

Voraussetzungen

Komponente	Zweck
Docker Engine + Docker Compose Plugin	Alle Dienste laufen als Container
`curl`, `openssl`, `sudo` (Linux)	Werden vom Installer benutzt
Ports `80` und `443` von außen erreichbar	Caddy, Let's Encrypt
DNS-Eintrag der Domain auf den Server	Für ein öffentliches Zertifikat
Arbeitsspeicher je nach Kartenregion	Für den Import des Routing-Graphen, siehe Tabelle

Region	Download	RAM für den Import (`-Xmx`)	Erster Import ca.
DACH (Standard)	~5,5 GB	8 GB	60–120 Min.
Deutschland	~4 GB	6 GB	45–90 Min.
Bayern	~1 GB	3 GB	10–20 Min.
Berlin (zum Testen)	~30 MB	1 GB	wenige Minuten

Im laufenden Betrieb braucht GraphHopper deutlich weniger, siehe `GH_DATAACCESS` unter GraphHopper.

Installation mit dem Installer (empfohlen)

Linux:

```
curl -sSL https://convoyplan.de/install.sh | bash
```

Windows (PowerShell als Administrator):

```
irm https://convoyplan.de/install.ps1 | iex
```

Der Installer fragt interaktiv nach:

1. **Installationsverzeichnis**: Standard `~/convoyplan`, unter Windows `%USERPROFILE%\convoyplan`
2. **Domain** (FQDN, z. B. `convoy.example.com`)
3. **E-Mail für Let's Encrypt**
4. **Datenbankpasswort** (zweimal)

5. **OSM-Region**: DACH, Deutschland, Bayern, Berlin oder eine eigene Geofabrik-URL
6. **Lizenzschlüssel** (optional): Enter = Demo-Modus, später im Admin-Bereich nachtragbar

Einen GitHub-Token fragt der Installer nicht ab. Das Repository ist öffentlich, der Auto-Updater kommt ohne aus (siehe Auto-Updater).

Danach läuft er ohne weitere Rückfragen durch:

- Er lädt `docker-compose.yml` und `caddy/entrypoint.sh` aus `main` ins Installationsverzeichnis.
- Er schreibt die `.env` (Rechte `600`) mit DB-Zugang, Domain, Region und einem frisch erzeugten `JWT_SECRET`. Dazu kommen die Image-Adressen und die Pfade, die der Auto-Updater braucht (`STACK_FILE_PATH`, `CADDY_ENTRYPOINT_PATH`, `COMPOSE_PROJECT_NAME=convoyplan`).
- Er führt `docker compose pull` und `docker compose up -d` aus. Im Terminal fährt dabei ein Konvoi als Fortschrittsanzeige über den Bildschirm. Beim Laden der Images rückt er je fertigem Dienst vor. Die Docker-Ausgabe erscheint dann nur bei einem Fehler. Wer sie vollständig sehen will, startet mit `CONVOYPLAN_PLAIN=1` (Linux: `curl -sSL https://convoyplan.de/install.sh | CONVOYPLAN_PLAIN=1 bash`).
- Unter Linux mit systemd richtet er zusätzlich einen Watchdog-Timer ein (`convoyplan-updater-watchdog.timer`, alle 2 Minuten). Er holt einen hängengebliebenen Updater zurück.

Am Ende nennt der Installer die Adresse des Setup-Wizards: `https://<DOMAIN>/setup`. GraphHopper baut den Routing-Graphen im Hintergrund, und bis dahin gibt es keine Routenberechnung. Fortschritt anzeigen:

```
cd ~/convoyplan
docker compose logs -f graphhopper
docker compose ps        # Zustand und Healthchecks aller Dienste
```

> ⚠️ **Die `docker-compose.yml` im Installationsverzeichnis nicht von Hand ändern.** Der Installer ersetzt sie bei jedem Lauf und der Auto-Updater bei Updates. Eigene Einstellungen gehören in die `.env`. Dort ist praktisch jeder Wert der Compose-Datei als Variable herausgeführt (siehe Konfiguration).

Erneut ausführen: Aktualisieren oder neu konfigurieren

Updates spielt normalerweise der Auto-Updater ein. Der Installer lässt sich trotzdem jederzeit erneut starten. Findet er im gewählten Verzeichnis eine vollständige `.env`, bietet er zwei Wege an:

- **[J] Nur aktualisieren** (Standard): Die Einstellungen bleiben, fehlende Einträge werden ergänzt, Compose-Datei und Images erneuert, der Stack wird neu gestartet.
- **[n] Neu konfigurieren**: Alle Fragen kommen erneut, die bisherigen Werte sind vorausgewählt. Datenbankpasswort, Region und Lizenzschlüssel bleiben mit Enter erhalten, ein vorhandener `GITHUB_TOKEN` bleibt ohne Rückfrage. `JWT_SECRET` wird nicht abgefragt und bleibt immer, damit bestehende Anmeldungen und MFA-Secrets gültig bleiben.

Setup-Wizard

Beim ersten Aufruf leitet die Anwendung automatisch auf `/setup` weiter. Der fünfstufige Wizard führt durch:

1. **Superadmin-Account**: E-Mail-Adresse und Passwort festlegen.
2. **Erste Organisation**: Org-Name und Org-Code anlegen (URL-Slug, 4–8 Zeichen). Der Slug wird Teil aller org-spezifischen URLs (`/o/[slug]/plan/`, `/o/[slug]/admin/`).
3. **Domain und SSL**: Serverdomain (FQDN) eingeben und TLS-Modus wählen:
   - **Let's Encrypt**: automatisches öffentliches Zertifikat
   - **Eigenes Zertifikat**: PEM-Datei hochladen
   - **Intern**: selbstsigniertes Zertifikat für lokale Nutzung

4. **Branding** (optional): App-Name, Farben und Logo anpassen. Lässt sich überspringen und ist später im Admin-Bereich erreichbar.
5. **Abschluss**: Caddy wird live neu geladen, danach geht es direkt zur Anmeldung unter `https://<DOMAIN>/o/[slug]/login`.

Den Lizenzschlüssel danach im Admin-Bereich unter **System → Lizenz** eintragen. Ohne Schlüssel läuft die Instanz im Demo-Modus (siehe Lizenz und Demo-Modus).

Manuelle Installation (ohne Installer)

Nur nötig, wenn der Installer nicht in Frage kommt. Er ist ein Shell-Skript und macht nichts anderes als die Schritte hier. Ein Repository-Klon wird auch hier nicht gebraucht:

```
mkdir -p ~/convoyplan/caddy && cd ~/convoyplan
RAW=https://raw.githubusercontent.com/RettTechSolutions/ConvoyPlan/main
curl -sSfL $RAW/docker-compose.yml -o docker-compose.yml
curl -sSfL $RAW/caddy/entrypoint.sh -o caddy/entrypoint.sh
curl -sSfL $RAW/.env.example        -o .env
chmod +x caddy/entrypoint.sh
chmod 600 .env
```

In der `.env` mindestens setzen:

Variable	Wert
`POSTGRES_PASSWORD`	eigenes Passwort. Ohne Wert startet Compose gar nicht.
`JWT_SECRET`	`openssl rand -hex 32`. Mit dem Platzhalter startet das Backend nicht.
`DOMAIN`, `ACME_EMAIL`	FQDN bzw. E-Mail für Let's Encrypt
`STACK_FILE_PATH`	absoluter Pfad dieser `docker-compose.yml`, z. B. `/home/<user>/convoyplan/docker-compose.yml` (für den Auto-Updater)
`CADDY_ENTRYPOINT_PATH`	absoluter Pfad von `caddy/entrypoint.sh`
`COMPOSE_PROJECT_NAME`	`convoyplan`

Dazu nach Bedarf Region und Speicher (`OSM_DOWNLOAD_URL`, `OSM_FILENAME`, `JAVA_OPTS`, Werte siehe Voraussetzungen). Danach:

```
docker compose pull
docker compose up -d
```

und den Setup-Wizard unter `https://<DOMAIN>/setup` aufrufen.

Portainer

Portainer kann denselben Stack betreiben. Die Images liegen in der GitHub Container Registry:

Variable	Wert
`BACKEND_IMAGE`	`ghcr.io/retttechsolutions/convoyplan/backend:latest`
`FRONTEND_IMAGE`	`ghcr.io/retttechsolutions/convoyplan/frontend:latest`
`GRAPHHOPPER_IMAGE`	`ghcr.io/retttechsolutions/convoyplan/graphhopper:latest`
`UPDATER_IMAGE`	`ghcr.io/retttechsolutions/convoyplan/updater:latest`
`REGION_MERGE_IMAGE`	`ghcr.io/retttechsolutions/convoyplan/osmium:latest`
`MEDIEN_IMAGE`	`ghcr.io/retttechsolutions/convoyplan/medien:latest`
`POSTGRES_PASSWORD`	sicheres Datenbankpasswort (Pflicht)
`JWT_SECRET`	mit `openssl rand -hex 32` erzeugen (Pflicht)
`DOMAIN` / `ACME_EMAIL`	FQDN bzw. E-Mail für Let's Encrypt

Caddy bindet `caddy/entrypoint.sh` per Bind-Mount ein (`CADDY_ENTRYPOINT_PATH`, Standard `./caddy/entrypoint.sh`). Bei einem reinen Web-Editor-Stack liegt die Datei nicht neben der Compose-Datei. Dann muss sie auf dem Host abgelegt und `CADDY_ENTRYPOINT_PATH` auf diesen absoluten Pfad gesetzt werden.

> **Hinweis:** Der `updater` braucht die Compose-Datei auf dem Host (`STACK_FILE_PATH`). In Portainer übernimmt sonst Portainers eigener Stack-Update-Mechanismus das Deployment neuer Images.

Konfiguration

Eine vollständige Vorlage liegt in `.env.example`. Die wichtigsten Variablen:

Datenbank

Variable	Beschreibung
`POSTGRES_USER`	Datenbankbenutzer
`POSTGRES_PASSWORD`	Datenbankpasswort – in Produktion zwingend ändern
`POSTGRES_DB`	Datenbankname

Backend

Variable	Standard	Beschreibung
`DATABASE_URL`	*(aus POSTGRES_* zusammengesetzt)*	PostgreSQL/PostGIS-Verbindung
`APP_ENV`	`production`	`production` erzwingt einen starken `JWT_SECRET` (Fail-Closed); `development` lockert die Prüfung für lokale Arbeit
`JWT_SECRET`	*(kein sicherer Default)*	Signaturschlüssel für JWTs – in Produktion zwingend ≥ 32 Zeichen; sonst **startet das Backend nicht**
`JWT_ALGORITHM`	`HS256`	JWT-Algorithmus
`JWT_EXPIRE_MINUTES`	`10080`	Token-Ablaufzeit in Minuten (7 Tage)

Variable	Standard	Beschreibung
`GRAPHHOPPER_URL`	`http://graphhopper:8989`	URL der Routing-Engine
`APP_BASE_URL`	`https://convoyplan.example.com`	Öffentliche App-Origin (Fallback für CORS in Produktion)
`CORS_ORIGINS`	*(leer)*	Komma-getrennte Allowlist oder `*`; leer = App-Origin aus `APP_BASE_URL`. `*` nur in Entwicklung

Sicheren JWT-Secret generieren (die Installer tun das automatisch):

```
openssl rand -hex 32
```

> ⚠️ In Produktion (`APP_ENV=production`, Default) verweigert das Backend den Start, wenn `JWT_SECRET` leer, der Platzhalter oder kürzer als 32 Zeichen ist (Fail-Closed).

SSL / Caddy

Variable	Beschreibung
`DOMAIN`	Serverdomain (z. B. `convoy.example.com`), Standard: `localhost`
`ACME_EMAIL`	E-Mail für Let's Encrypt
`CADDY_TLS_CERT`	Pfad zum PEM-Zertifikat (optional, für eigene Zertifikate)
`CADDY_TLS_KEY`	Pfad zum PEM-Schlüssel (optional, für eigene Zertifikate)
`HTTP_PORT`	Externer HTTP-Port, Standard: `80`
`HTTPS_PORT`	Externer HTTPS-Port, Standard: `443`

GraphHopper

Variable	Standard	Beschreibung
`OSM_DOWNLOAD_URL`	`https://download.geofabrik.de/europe/dach-latest.osm.pbf`	Download-URL der OSM-PBF-Datei
`OSM_FILENAME`	`dach-latest.osm.pbf`	Dateiname im persistenten OSM-Volume
`JAVA_OPTS`	`-Xmx8g -Xms1g -XX:+UseG1GC`	JVM-Speicherkonfiguration für den **Graph-Import**
`GH_DATAACCESS`	`MMAP`	Speicherzugriff des laufenden Routing-Servers: `MMAP` (Graph im Seitencache des Kernels, JVM bleibt klein) oder `RAM_STORE` (Graph komplett im Java-Heap)
`GH_SERVER_JAVA_OPTS`	*(leer = Standard)*	Zusätzliche JVM-Optionen nur für den Server; Standard gibt ungenutzten Heap im Leerlauf zurück und beendet die JVM bei `OutOfMemoryError`
`GH_LM`	`true`	Landmarks (LM) für schnelleres Routing langer Strecken; `false` schaltet sie ab

Richtwerte: DACH `-Xmx8g`, Deutschland `-Xmx6g`, Bayern `-Xmx3g`, Berlin `-Xmx1g`. Sie gelten für den Import. Im Betrieb liegt der Graph mit `GH_DATAACCESS=MMAP` im Seitencache statt im Heap — der Speicher wird also nur belegt, solange er nicht anderweitig gebraucht wird, und die JVM bleibt deutlich unter `-Xmx`. Wer genug RAM hat und die Einlesezeit nach einem Neustart vermeiden will, setzt `GH_DATAACCESS=RAM_STORE`. MMAP blendet den Graphen in 1-MB-Segmenten ein; ein 14-GB-Graph braucht damit rund 14 000 Mappings, der Linux-Standard `vm.max_map_count=65530` reicht also bis weit über 50 GB Graph.

Landmarks: Zum Graphen rechnet GraphHopper Landmarks, die lange Routen bei „schnell" und „standard" gut doppelt so schnell machen (gemessen auf Österreich: 682 km in 3,7 s statt 6,5 s). Das kostet nach dem Import einen zweiten Lauf etwa derselben Dauer und 64 Byte je Knoten auf der Platte und im Seitencache — für Österreich 262 MB zu 421 MB Graph. Der Heap des Imports reicht dafür; neu gebaut wird nichts: Nach dem Update rechnet jede Installation die Landmarks einmal zu ihrem vorhandenen Graphen nach, während der Zeit pausiert das Routing (Statusseite und Admin-Panel zeigen den Aufbau). Scheitert die Rechnung, etwa am Speicher, routet GraphHopper ohne LM weiter wie bisher. Einen neuen Versuch (etwa nach mehr Heap in `JAVA_OPTS`) startet, wer `.lm_gescheitert` im Graph-Volume löscht und GraphHopper neu startet.

Regionswechsel: Die Variablen hier gelten nur für den **Erststart**. Zum Wechseln der Kartenregion im laufenden Betrieb gibt es seit `2026.4.0` die Karte „Kartenregion" im Admin-Panel unter **System** — dort mit Vorabschätzung von Speicher-, Platten- und Zeitbedarf, Live-Fortschritt und ohne nennenswerten Routing-Ausfall (der neue Graph entsteht neben dem laufenden). Seit `2026.5.0` lassen sich dort auch **mehrere** Regionen gleichzeitig wählen; sie werden zu einer Karte zusammengeführt, sodass Routen über die Ländergrenzen hinweg funktionieren.

Die aktive Region liegt danach in `/data/osm/.region` im `osm_data`-Volume und hat Vorrang vor den Variablen aus der `.env` — so überlebt sie ein `docker compose up`. Wer die Variablen hier nachträglich ändert, ändert deshalb **nichts** an einer bereits per Panel gewechselten Installation.

Der Umriss auf der Karte — die Maske, die alles außerhalb des routbaren Gebiets abdunkelt — folgt der aktiven Region automatisch: Das Backend leitet ihn unter `GET /api/region/outline` aus `.region` und den Geometrien des Geofabrik-Index ab. Bis `2026.5.2` war er eine mitgelieferte DACH-Datei und musste von Hand getauscht werden; wer das früher getan hat, braucht dafür jetzt nichts mehr zu tun.

Bei einer aus mehreren Regionen zusammengesetzten Karte nennt die Karte „Kartenregion" unter **Aktuelle Region** zusätzlich die einzelnen Bestandteile — praktisch, um nach einem Wechsel nachzuvollziehen, welche Länder/Gebiete tatsächlich geladen sind, statt das nur am kryptischen `merged-<hash>.osm.pbf`-Dateinamen abzulesen. Bei nur einer Region entfällt die Zeile, da der Dateiname (z. B. `dach-latest.osm.pbf`) die Auskunft bereits enthält.

Wer eine geladene Karte nur **ergänzen** will, nimmt „Region erweitern" statt „Region wechseln": Die Auswahl beginnt dann mit allen Bestandteilen der geladenen Karte, Suche und Schnellauswahl fügen hinzu. Gebaut wird trotzdem eine vollständig neue Karte aus allen Bestandteilen — Download und Import umfassen also auch die schon vorhandenen Länder.

Wartungsmodus und Terminierung: Seit `2026.6.0` lässt sich beim Wechsel die Option „Routing während des Imports pausieren" aktivieren. GraphHopper wird dabei angehalten, **bevor** der Graph gebaut wird, statt erst danach — sein Speicher steht damit dem Import zur Verfügung, wodurch große kombinierte Karten auf Maschinen mit knappem Arbeitsspeicher überhaupt erst baubar werden. Der Preis: keine Routenplanung, bis der Wechsel durch ist. Zusätzlich lässt sich unter „Startzeitpunkt" ein Termin für den Wechsel setzen (höchstens 30 Tage voraus, in der Vergangenheit abgelehnt) — sinnvoll in Kombination mit pausiertem Routing, wenn der Ausfall in eine nutzungsarme Zeit fallen soll. Ein geplanter Wechsel übersteht einen Neustart des Updaters und lässt sich bis zum Anlauf abbrechen; ohne Termin startet der Wechsel sofort. Die Speicherprüfung läuft dabei vor dem Download, nicht erst nach zwölf Minuten Import.

Abbruch: Ein abgebrochener Wechsel wird als **„Abgebrochen"** gemeldet, nicht als „Fehlgeschlagen" — der Unterschied ist der zwischen einer erfüllten Absicht und einer Störung. Die bisherige Region läuft in beiden Fällen unverändert weiter. Abbrechen lässt sich bis einschließlich Phase 3 (Graph-Bau); ab dem Schwenk ist kein Abbruch mehr vorgesehen.

Wenn Geofabrik hakt: Die Größenabfrage vor dem Wechsel läuft für alle Bestandteile gleichzeitig und wiederholt einen Fehlversuch zweimal (1 s, 3 s Pause). Bleibt sie erfolglos, nennt die Meldung die betroffene Region. Sie betrifft ausschließlich die Vorab-Rechnung — die geladene Karte und das Routing sind davon nicht berührt.

Sicherheit und Datenschutz

Variable	Standard	Beschreibung
`MFA_ENCRYPTION_KEY`	*(aus `JWT_SECRET` abgeleitet)*	Fernet-Schlüssel zur Verschlüsselung der TOTP-Secrets at-rest. Rotation von `JWT_SECRET` macht ohne eigenen Schlüssel bestehende MFA-Secrets unlesbar

Variable	Standard	Beschreibung
`PASSWORD_BREACH_CHECK_ENABLED`	`true`	Abgleich neuer Passwörter gegen Have-I-Been-Pwned (k-Anonymity, fail-open). Für Air-Gapped-Setups auf `false`
`CSP_ENFORCE`	`false`	Content-Security-Policy erzwingen (Default: Report-Only, bricht die UI nicht)
`QUOTA_ROUTING_PER_HOUR`	`240`	Stundenbudget für Routenberechnungen (GraphHopper) je angemeldetem Benutzer. `0` = Drossel aus
`QUOTA_ROUTING_DEMO_PER_HOUR`	`40`	Dasselbe für Demo-Sitzungen (zusätzlich pro IP gezählt)
`QUOTA_GEOCODE_PER_HOUR`	`600`	Stundenbudget für die Adresssuche (HERE/Photon) je Benutzer
`QUOTA_GEOCODE_DEMO_PER_HOUR`	`100`	Dasselbe für Demo-Sitzungen
`QUOTA_TRAFFIC_PER_HOUR`	`600`	Stundenbudget für die Live-Verkehrslage (HERE/TomTom) je Benutzer
`QUOTA_TRAFFIC_DEMO_PER_HOUR`	`100`	Dasselbe für Demo-Sitzungen
`RETENTION_ENABLED`	`true`	Periodisches Purgen alter Daten durch den `retention`-Container
`RETENTION_INTERVAL`	`3600`	Sekunden zwischen den Purge-Läufen
`RETENTION_POSITIONS_HOURS`	`24`	Live-Positionen älter als dieser Wert werden gelöscht
`RETENTION_AUDIT_DAYS`	`365`	Audit-Log-Einträge älter als dieser Wert werden gelöscht
`RETENTION_SHARE_LINKS_DAYS`	`30`	Widerrufene Share-Links älter als dieser Wert werden gelöscht
`RETENTION_POSITION_TRAIL_DAYS`	`45`	Positionsverlauf der Aktionsseiten älter als dieser Wert wird gelöscht
`RETENTION_OAUTH_TOKENS_GRACE_DAYS`	`30`	Karenz für tote MCP-Refresh-Tokens — kürzer setzen schwächt die Wiederverwendungserkennung
`RETENTION_OAUTH_CLIENTS_DAYS`	`7`	Verwaiste MCP-Registrierungen (ohne Tokens und Codes) älter als dieser Wert werden gelöscht
`BACKUP_DIR`	`./backups`	Zielverzeichnis für `scripts/backup.sh`
`BACKUP_RETENTION_DAYS`	`30`	Aufbewahrungsdauer der Backups

> Details zu Härtung, Audit-Log, DSGVO und Backup/Restore: **Sicherheit und Datenschutz**.

Lizenz und Auto-Updater

Variable	Beschreibung
`LICENSE_KEY`	Lizenzschlüssel. Ohne gültigen Schlüssel läuft die App im Demo-Modus. Alternativ über den Admin-Bereich eintragbar (wird dann in der DB gespeichert).
`GITHUB_TOKEN`	Optional. Hebt nur das Rate-Limit der GitHub-API (60 → 5000 Anfragen/Stunde je IP). Wenn, dann ein fine-grained Token mit „Public repositories (read-only)" ohne weitere Rechte. Details: Auto-Updater.

Variable	Beschreibung
`GITHUB_REPO`	Repository, das der Auto-Updater überwacht. Standard: `RettTechSolutions/ConvoyPlan`.
`UPDATE_CHANNEL`	Fallback-Kanal: `stable` (Standard), `beta` oder `nightly`. Der Schalter im Admin-Bereich überschreibt diesen Wert.
`UPDATE_MODE`	Fallback-Modus: `auto` (Standard) oder `notify`. Der Schalter im Admin-Bereich überschreibt diesen Wert.
`UPDATE_NOTIFY_ON_AUTO`	Nur bei `auto`: `true` schickt zusätzlich eine E-Mail an Superadmins nach automatischer Installation (Standard: `false`).
`UPDATE_NOTIFY_INTERVAL`	Prüfintervall in Sekunden für fällige Update-Benachrichtigungen (Standard: `1800`).

> Details: **Auto-Updater** und **Lizenz und Demo-Modus**.

Verkehrsdaten

Variable	Beschreibung
`HERE_TRAFFIC_API_KEY` / `TOMTOM_TRAFFIC_API_KEY`	Optionale API-Keys für die Live-Verkehrslage. Ohne Key bleibt die Funktion inaktiv. Alternativ im Admin-Bereich hinterlegbar (hat Vorrang).
`OVERPASS_URLS`	Overpass-Server für Sperrungen, Tankstellen und Brücken ohne Höhenangabe, kommagetrennt, in dieser Reihenfolge versucht. Standard: `https://overpass-api.de/api/interpreter`. Die Abfragen enthalten die geplante Route. Nur Server mit weltweiten Daten eintragen, ideal ist eine eigene Instanz.
`TRAFFIC_FLOW_PROVIDER`	Anbieter erzwingen (`here` / `tomtom`). Standard: automatisch, HERE bevorzugt.
`HERE_API_KEY`	Optional. Aktiviert die Adresssuche im Plan-Editor über HERE Geocoding & Search (serverseitig proxied). Leer = `HERE_TRAFFIC_API_KEY` mitbenutzen bzw. Photon-Fallback.
`HERE_MONTHLY_LIMIT`	Kostendeckel für die Adresssuche: max. HERE-Anfragen pro Kalendermonat (Standard `25000`, `0` = kein App-Deckel). Deckel erreicht → automatischer Photon-Fallback.
`OPENDATA_TRAFFIC_ENABLED`	Offene Baustellen-/Sperrungsfeeds aktiviert lassen. Standard: `true`.
`OPENDATA_TRAFFIC_FEEDS`	Kommaseparierte Liste `format|url`. Formate: `mobidata_bw`, `berlin_viz`, `datex2`.
`OPENDATA_TRAFFIC_CLIENT_CERT`	Client-Zertifikat (PEM) für per mTLS geschützte `datex2`-Feeds (mobilithek).
`OPENDATA_TRAFFIC_CA_CERT`	Nur für Broker mit **privater** CA. Für den mobilithek-Broker **leer lassen**.

> Schritt-für-Schritt-Anleitung: **Verkehrsdaten**.

Kartenregion wechseln

Die Region wechselt man im laufenden Betrieb im Admin-Panel unter **System → Kartenregion** (siehe oben). Der neue Graph entsteht dabei neben dem laufenden.

Die Vorabschätzung rechnet den Heap für den Import als `(2 GB + 1,1 × Extract)` plus 20 % Sicherheitsaufschlag. Gesperrt wird ein Wechsel erst, wenn schon der Bedarf **ohne** Aufschlag nicht in den freien Speicher (abzüglich 1 GB Reserve) passt — dieselbe Schwelle, an der der Updater abbräche. Passt nur der Aufschlag nicht, heißt das „knapp": Der Import läuft mit gedeckeltem Heap, und „Routing während des Imports pausieren" gibt ihm den Speicher des laufenden GraphHopper dazu.

Gutgeschrieben wird dabei, was GraphHopper **tatsächlich** belegt, gemessen am Container (Panel über den `dockerproxy`, Updater über die cgroup), nicht sein `-Xmx`. Seit der Server den Graphen per MMAP einblendet, sind das meist nur einige hundert MB — der Wartungsmodus bringt also deutlich weniger als früher. Lässt sich nicht messen, nennt das Panel den `-Xmx` als Obergrenze und sagt dazu, dass er nicht gemessen ist.

Wie viel Heap ein Import wirklich gebraucht hat, steht am Ende seines Protokolls im Panel (`garbage-first heap total …K`). Auf der Platte belegt der fertige Graph erfahrungsgemäß gut die Hälfte des Extracts (gemessen: 8,5 GB Extracts, 4,9 GB Graph).

Nur wenn das nicht geht, lässt sich der Graph-Cache von Hand verwerfen. Danach ist das Routing bis zum Ende des Neuimports weg:

```
cd ~/convoyplan
docker compose down
docker volume rm convoyplan_gh_graph
docker compose up -d
```

Checkliste für Produktion

- ☐ Installation über den Installer, nicht aus einem Repository-Klon
- ☐ `JWT_SECRET` stark und nirgends versioniert (der Installer erzeugt ihn)
- ☐ Eigenes Datenbankpasswort
- ☐ `CORS_ORIGINS` leer lassen oder auf die produktive Domain einschränken, nie `*`
- ☐ Persistente Volumes (`postgres_data`, `caddy_data`, `cert_uploads`, `logo_uploads`) regelmäßig sichern, siehe `scripts/backup.sh`
- ☐ Genug RAM für den Import der gewählten Region (siehe Voraussetzungen)
- ☐ GraphHopper-Graph-Cache (`gh_graph`) auf schnellem Speicher
- ☐ Lizenzschlüssel eintragen (Admin → System → Lizenz), sonst läuft die Instanz dauerhaft im Demo-Modus

---

Nützliche Docker-Befehle

Im Installationsverzeichnis (Standard `~/convoyplan`):

```
docker compose ps                   # Zustand und Healthchecks
docker compose logs -f backend      # Backend-Logs
docker compose logs -f graphhopper  # GraphHopper-Logs (Import-Fortschritt)
docker compose pull && docker compose up -d   # Images von Hand aktualisieren (Kanal-Tags aus .env,
pflegt der Updater)
docker compose down                 # Dienste stoppen
docker compose down -v              # Dienste stoppen und ALLE Daten löschen
```

Sicherheitshinweise

- `POSTGRES_PASSWORD` ist Pflicht. Ohne Wert verweigert Compose den Start, es gibt kein Standardpasswort.
- In Produktion (`APP_ENV=production`, Default) verweigert das Backend den Start, wenn `JWT_SECRET` leer, der Platzhalter oder kürzer als 32 Zeichen ist (Fail-Closed).
- Datenbank (`5432`) und GraphHopper (`8989`) sind nur an `127.0.0.1` gebunden, das Backend ist von außen nur über Caddy erreichbar.
- Die Caddy-Admin-API läuft auf Port `:2019` und ist nur intern im Docker-Netzwerk erreichbar.
- Caddy liefert Security-Header (HSTS, `X-Content-Type-Options`, `X-Frame-Options` u. a.) und eine Content-Security-Policy aus (Report-Only, mit `CSP_ENFORCE=true` erzwingbar).
- Die persistierte Caddyfile (`/certs/Caddyfile`, vom Setup-Wizard geschrieben) wird bei jedem Backend-Start gegen die aktuelle Header-Baseline geprüft und bei Bedarf neu erzeugt und live nachgeladen.
- Endpunkte, die fremdes Kontingent kosten (Routing, Adresssuche, Verkehrslage), haben ein Stundenbudget je Aufrufer (`QUOTA_*`).
- TOTP-Secrets werden Fernet-verschlüsselt at-rest gespeichert. Passwort- und MFA-Reset entziehen über die `token_version` alle bestehenden JWTs.
- Öffentliche Share-Links sind ohne Login abrufbar. Ihre Tokens sind wie vertrauliche Links zu behandeln und bei Bedarf zu widerrufen.
- Live-Tracking verarbeitet Standortdaten. Die Aufbewahrung regelt der `retention`-Container.

> Vollständige Übersicht: **Sicherheit und Datenschutz**.

KAPITEL 2

Systemübersicht

Der Reiter **Systemübersicht** im Admin-Portal (`/admin` , Superadmin) zeigt, wie es der Maschine unter ConvoyPlan geht: Auslastung der Hardware, Zustand der Container, Belegung und Druck auf den Platten sowie die Nutzung des Portals — live, im Verlauf und als Monatsbericht zum Ablegen.

Alle Zahlen sind auch über die API abrufbar (siehe API-Dokumentation), etwa für ein eigenes Monitoring oder eine Tabellenkalkulation. Für einen Dauerauftrag lohnt sich statt eines Benutzer-Tokens ein **System-API-Key** (Admin → API-Keys, Geltungsbereich *System*): Er läuft nicht nach sieben Tagen ab und darf nur lesen — siehe API-Dokumentation.

Was erfasst wird

Bereich	Kennzahlen
CPU	Auslastung in Prozent, Kernzahl, Load Average (1/5/15 Min.)
Arbeitsspeicher	belegt/verfügbar/gesamt, Prozent, Swap
Massenspeicher	Belegung je Mountpoint, Lese-/Schreibdurchsatz, Busy-Zeit
Druck (PSI)	Wartezeit-Anteil für I/O, CPU und Speicher
Container	Zustand und Healthcheck je Container, CPU/RAM je Container
Datenbank	Größe, offene Verbindungen, Anzahl Benutzer/Organisationen/Kolonnen
Portalnutzung	gleichzeitig aktive Benutzer, eindeutige Benutzer je Tag (getrennt nach Portalnutzern, Demo-Besuchern und Administration), Anmeldungen, Requests, Serverfehler, Antwortzeit

Portalnutzer, Demo-Besucher, Administration

Die Nutzerzahlen werden nach drei Gruppen getrennt geführt, weil sie verschiedene Fragen beantworten:

Gruppe	Wer	Warum getrennt
Portalnutzer (`member`)	angemeldete Mitglieder einer Organisation	die eigentliche Kennzahl: tatsächliche Nutzung des Portals
Demo-Besucher (`demo`)	Besucher der öffentlichen Demo	jede Demo-Sitzung legt ein eigenes Wegwerfkonto an — mitgezählt sähen fünf Probeklicks aus wie fünf neue Nutzer
Administration (`admin`)	Superadmins im Admin-Portal	der Betreiber selbst, keine Portalnutzung

Die Gruppe ergibt sich aus dem Zugriffstoken und gehört zum Tagesschlüssel: Ein Superadmin, der zusätzlich in einer Organisation arbeitet, zählt für diese Arbeit als regulärer Portalnutzer — das Org-Token trägt keine Superadmin-Kennung. In Kacheln, Charts, `/api/admin/system/usage` und den Monatsberichten meint „Benutzer" bzw. `unique_users` immer nur die erste Gruppe; `demo_users` / `admin_users` stehen daneben.

Warum „Druck" und nicht nur „Auslastung"

Eine Platte kann bei 100 % Auslastung völlig entspannt sein und bei 30 % bereits der Flaschenhals — entscheidend ist, ob Prozesse *warten* müssen. Genau das misst **PSI** (Pressure Stall Information, Linux ab 4.20): der Anteil der Zeit, in dem mindestens ein Task auf die Ressource wartet. Als Faustregel:

- **unter 10 %** — unauffällig,
- **10–25 %** — spürbar, im Auge behalten,
- **über 40 %** — die Ressource ist der Engpass.

Liefert der Kernel keine PSI-Werte, bleiben die entsprechenden Anzeigen leer; alles andere funktioniert unverändert weiter.

Host oder Container?

CPU, Arbeitsspeicher und Platten-I/O liest das Backend aus dem procfs, das im Container die Werte des **Hosts** zeigt — die Übersicht beschreibt also die Maschine, auf der der Stack läuft, nicht nur den Backend-Container. Die Plattenbelegung wird über die tatsächlich eingehängten Pfade ermittelt (`/`, `/uploads`, `/update_status`); Pfade auf demselben Dateisystem werden zusammengefasst.

Historie und Auflösung

Ein Hintergrund-Job im Backend schreibt alle **5 Minuten** eine Stichprobe. Nach Mitternacht verdichtet er jeden abgeschlossenen Tag zu einer Tageszeile.

Datenart	Auflösung	Aufbewahrung (Standard)
Stichproben	5 Minuten	90 Tage
Tageswerte	1 Tag	3 Jahre
Nutzung je Benutzer und Tag	1 Tag	3 Jahre

Die Verlaufsdiagramme wählen die Auflösung automatisch passend zum Zeitraum (Rohdaten → Stundenmittel → Tageswerte), sodass auch ein Jahresverlauf zügig lädt. Monatsberichte greifen auf die Tageswerte zu und funktionieren deshalb auch dann noch, wenn die Rohdaten des Monats längst gelöscht sind.

> **Hinweis zur Deutung:** Zustandsgrößen (CPU, RAM, Belegung, aktive Benutzer) werden je Zeitfenster gemittelt, Zähler (Requests, Fehler, Anmeldungen) aufsummiert.

Monatsberichte exportieren

Im Abschnitt **Monatsbericht** den Monat wählen und exportieren:

- **CSV** — Semikolon-getrennt mit deutschem Dezimalkomma, öffnet sich in Excel ohne Nacharbeit. Enthält Kopfdaten, Zusammenfassung und alle Tageswerte.
- **PDF** — Querformat, Kennzahlen-Zusammenfassung plus Tagestabelle; gedacht zum Ablegen, z. B. als Betriebsnachweis.

Über die API geht dasselbe direkt:

```
curl -H "Authorization: Bearer $TOKEN" \
  "https://web.convoyplan.de/api/admin/system/reports/monthly?month=2026-07&format=csv" \
  -o bericht.csv
```

```
# oder ohne Login, mit einem System-API-Key:
curl -H "X-API-Key: $CVP_SYSTEM_KEY" \
  "https://web.convoyplan.de/api/admin/system/reports/monthly?month=2026-07&format=csv" \
  -o bericht.csv
```

Der laufende Monat ist jederzeit abrufbar — der heutige Tag wird beim Abruf frisch verdichtet.

Anbindung an ein Monitoring (PRTG)

Für PRTG gibt es einen fertigen Endpunkt: `GET /api/admin/system/prtg` liefert dieselben Live-Kennzahlen wie die Übersichtsseite, nur im Kanalformat des Sensortyps **HTTP Data Advanced**. Auf der Probe braucht es dadurch weder ein Skript noch ein Mapping-Template — Sensor anlegen, URL eintragen, unter *Custom HTTP Headers* die Zeile `X-API-Key: cvp_…` mit einem System-API-Key hinterlegen.

Der Sensor bringt Kanäle für CPU, Last, Arbeitsspeicher, Platte, PSI-Druck, Container, Datenbank und Portalnutzung mit, dazu Warn- und Fehlergrenzen als Startwerte und einen Kanal „Alter der letzten Stichprobe", an dem ein stehengebliebener Sampler auffällt. Die vollständige Kanalliste und die Einrichtung Schritt für Schritt stehen in der API-Dokumentation.

Andere Monitoring-Systeme lesen `/api/admin/system/overview` direkt als JSON. Für einen reinen Erreichbarkeitscheck genügt ohne Key `/health` oder `/api/status`.

Container-Zustand (Docker)

Den Zustand der Container holt das Backend über die Docker-Engine-API. Dabei bekommt das Backend **keinen Docker-Socket**: Zugriff auf `/var/run/docker.sock` ist gleichbedeutend mit Root auf dem Host, und das Backend ist von außen erreichbar. Stattdessen läuft im Stack der Sidecar `dockerproxy` (docker-socket-proxy), der ausschließlich lesende Aufrufe (`/containers`, `/info`, `/version`) durchlässt; schreibende Aufrufe sind gesperrt (`POST=0`), Container lassen sich darüber also weder starten noch stoppen noch erzeugen.

Ist der Sidecar nicht erreichbar, zeigt die Übersicht bei „Container" einen Hinweis samt Grund — alle übrigen Kennzahlen bleiben verfügbar.

Wer den Sidecar nicht möchte, setzt `DOCKER_METRICS_ENABLED=false` (dann entfällt nur die Container-Anzeige) oder lässt das Backend über `DOCKER_API_URL=unix:///var/run/docker.sock` direkt mit dem Socket reden — dann muss der Socket bewusst selbst ins Backend gemountet werden, mit der oben genannten Konsequenz.

Datenschutz

Die Nutzungshistorie speichert je Benutzer, Tag und Nutzergruppe **eine** Zeile: Benutzer-ID, Organisation, erster und letzter Zugriff sowie die Zahl der Requests. Es werden **keine** aufgerufenen Seiten, IP-Adressen oder Inhalte protokolliert — die Auswertung beantwortet „wie viele Leute waren wann im Portal", nicht „wer hat was getan" (dafür gibt es das Audit-Log). Die Daten unterliegen der oben genannten Aufbewahrungsfrist und werden danach automatisch gelöscht.

Konfiguration

Alle Werte sind optional; die Standards passen für den Normalbetrieb.

Variable	Standard	Bedeutung
`SYSTEM_METRICS_ENABLED`	`true`	Erfassung insgesamt an/aus
`SYSTEM_METRICS_INTERVAL`	`300`	Sekunden zwischen zwei Stichproben
`SYSTEM_METRICS_RETENTION_DAYS`	`90`	Aufbewahrung der Stichproben
`SYSTEM_METRICS_DAILY_RETENTION_DAYS`	`1095`	Aufbewahrung der Tageswerte
`SYSTEM_METRICS_DISK_PATHS`	`/,/uploads,/update_status`	überwachte Pfade
`DOCKER_METRICS_ENABLED`	`true`	Container-Zustand erfassen
`DOCKER_API_URL`	`http://dockerproxy:2375`	Docker-API (alternativ `unix://…`)
`DOCKER_STATS_ENABLED`	`true`	CPU/RAM je Container mitmessen

Eine kürzere Taktung erhöht die Detailtiefe und den Speicherbedarf entsprechend: 5 Minuten ergeben rund 8.600 Zeilen pro Monat, 1 Minute rund 43.000.

Fehlerbehebung

„Noch keine Messwerte für diesen Zeitraum" Nach der Installation dauert es ein Sampling-Intervall, bis der erste Punkt da ist. Mit **Jetzt messen** lässt sich sofort eine Stichprobe erzwingen.

Container zeigen „n. v." Der `dockerproxy`-Sidecar läuft nicht oder ist nicht erreichbar. Prüfen: `docker compose ps dockerproxy` und `docker compose logs dockerproxy`.

Plattendruck bleibt leer Der Kernel liefert keine PSI-Werte (Linux < 4.20 oder ohne `CONFIG_PSI`). Der Durchsatz in Bytes pro Sekunde wird trotzdem erfasst.

Monatsbericht ist leer Tageswerte entstehen erst nach Mitternacht aus den Stichproben. Für den laufenden Monat wird der heutige Tag beim Abruf verdichtet — liegen für ihn noch keine Stichproben vor, bleibt der Bericht leer.

KAPITEL 3

Multi-Tenancy

ConvoyPlan ist mandantenfähig: Mehrere Organisationen teilen sich eine Instanz, arbeiten aber vollständig voneinander isoliert.

Org-Code-Slug

Jede Organisation erhält einen kurzen **Org-Code** (4–8 Zeichen) als URL-Slug. Er wird beim Anlegen der Organisation vergeben (Setup-Wizard, Schritt 2, oder später im Admin-Bereich) und ist Teil aller org-spezifischen URLs unter dem Scope `/o/[slug]/`:

Bereich	URL
Login	`https://<DOMAIN>/o/[slug]/login`
Planung	`https://<DOMAIN>/o/[slug]/plan/`
Live-Tracking	`https://<DOMAIN>/o/[slug]/tracking/`
Org-Admin	`https://<DOMAIN>/o/[slug]/admin/`

Org-spezifische Login-Seite und Branding

Die Login-Seite unter `/o/[slug]/login` zeigt das **eigene Branding** der Organisation (Logo, Farben, App-Name). Org-Admins pflegen es im Org-Admin-Bereich unter **System / Branding**.

Technisch ist das Org-Branding ein **JSON-Override auf der Organisation** (Spalte `organizations.branding`), der über dem globalen Plattform-Branding liegt:

- Nicht gesetzte Felder erben automatisch den Plattform-Wert (`NULL` = vollständig geerbt).
- Der Override gilt **ausschließlich innerhalb von** `/o/[slug]/*` (Login, Planung, Tracking, Org-Admin) — der Superadmin-Bereich und andere Organisationen bleiben beim Plattform-Branding.
- Logos werden pro Organisation namespaced abgelegt (`org-<id>-<slot>.<ext>`) und überschreiben nie die globalen Logos.
- **„Plattform-Branding wiederherstellen"** entfernt alle Overrides der Organisation wieder — sie fällt dann vollständig auf das Plattform-Branding zurück.

Das **globale Plattform-Branding** (`/api/branding`, Superadmin-only) bleibt davon unberührt und gilt für alle Bereiche außerhalb von `/o/[slug]`.

> 📖 Vollständige Endpunkte siehe API-Dokumentation → Org-Branding.

Datenisolation

Alle fachlichen Objekte (Konvois, Fahrzeuge, Leitstellen-Vorschläge, Positionen) sind über die `organization_id` an eine Organisation gebunden. Benutzer sehen ausschließlich Daten der Organisationen, denen sie zugeordnet sind. Die Zuordnung erfolgt über `UserOrganizations`.

Rollen im Org-Kontext

Innerhalb einer Organisation gelten die üblichen Rollen (**Admin, Planer, Fahrer, Beobachter**). Details siehe Rollen & Berechtigungen.

- **Superadmin** ist instanzweit und verwaltet Organisationen, globale Leitstellen, Updates und System-Einstellungen.
- **Org-Admin** verwaltet Mitglieder, org-eigene Leitstellen, Branding und System-Einstellungen der eigenen Organisation.

Anlegen und Zuweisen

Aktion	Wo
Organisation anlegen	Setup-Wizard (erste Org) oder Superadmin → Organisationen
Benutzer einer Org zuweisen	Superadmin → Benutzer, oder Org-Admin → Mitglieder
Org-Code / Branding ändern	Org-Admin → System / Branding

Bei der Benutzeranlage im Superadmin-Bereich lässt sich optional direkt eine **Organisation samt Rolle** mitgeben — die Mitgliedschaft wird in derselben Transaktion angelegt. Wird stattdessen ein **bestehender** Benutzer nachträglich über Org-Admin → Mitglieder zu einer Organisation hinzugefügt, erhält er automatisch eine E-Mail mit Organisationsname, Rolle und dem passenden Org-Login-Link (`/o/[slug]/login`).

Relevante Endpunkte: siehe API-Dokumentation → Organisationen.

KAPITEL 4

🚦 Verkehrsdaten einrichten — Baustellen, Sperrungen & Stau

ConvoyPlan blendet auf der Karte **Baustellen, Sperrungen und Hindernisse** sowie optional die **Live-Verkehrslage (Stau)** ein. Die Daten stammen aus mehreren Quellen, die zu einer gemeinsamen Ebene zusammengeführt werden („in einen Topf"). Fällt eine Quelle aus, liefern die anderen weiter.

Diese Anleitung zeigt, was **out of the box** läuft und wie du die optionalen Quellen (Live-Stau, bundesweite Baustellen) aktivierst.

> **Abdeckung:** Die Routenplanung läuft im gesamten DACH-Raum. Ohne Setup liefern Overpass und die Live-Verkehrslage überall; die amtlichen Baustellenfeeds sind je Land einzeln einzurichten — Deutschland in Abschnitt 3, Österreich und die Schweiz in Abschnitt 4.

📊 Überblick — welche Quelle liefert was

Quelle	Abdeckung	Kosten	Setup nötig?
Overpass (OpenStreetMap)	DACH-weit	frei	nein — läuft
Autobahn-API (bund.dev)	Deutschland, alle Bundesautobahnen	frei	nein — läuft
MobiData BW	Baden-Württemberg (bis Kreisstraße)	frei	nein — läuft
Berlin VIZ	Berlin	frei	nein — läuft
mobilithek (DATEX II)	weitere Bundesländer (Bundes-/Landesstraßen)	frei	✅ Abschnitt 3
ASFINAG (DATEX II)	Österreich	frei, Registrierung	✅ Abschnitt 4
opentransportdata.swiss (DATEX II)	Schweiz	frei, Registrierung	✅ Abschnitt 4
HERE / TomTom	Live-Verkehrslage (Stau), DACH-weit	eigener API-Key (Freikontingent)	✅ Abschnitt 2

> **Farbcodierung auf der Karte:** 🟠 Baustelle · 🔴 Sperrung · 🟡 Warnung/Hindernis · Live-Verkehrslage grün→gelb→rot nach Staufaktor.

1. ✅ Sofort aktiv (kein Setup)

Autobahn-Baustellen/-Sperrungen (bundesweit) sowie die offenen Feeds für **Baden-Württemberg** und **Berlin** sind standardmäßig aktiv. Nichts zu tun.

2. 🟢 Live-Verkehrslage (Stau) aktivieren — HERE oder TomTom

Die Echtzeit-Verkehrslage (wie das rot/orange Stau-Bild bei Google) gibt es nur bei kommerziellen Anbietern — beide mit kostenlosem Kontingent.

1. **Kostenlosen API-Key holen:**
   - **HERE** — platform.here.com · 250.000 Anfragen/Monat gratis (empfohlen)
   - **TomTom** — developer.tomtom.com
2. Im ConvoyPlan **Admin-Bereich → „Live-Verkehrslage (Stau)"** den Key eintragen und speichern. Fertig — im Plan-Export-Tab erscheint der Schalter **„🚦 Verkehrslage laden"**.

> ⚠️ **Lizenzhinweis:** Die Anzeige von HERE-/TomTom-Verkehrsdaten auf der OSM-Basiskarte kann lizenzpflichtig sein. Vor produktivem Einsatz die Nutzungsbedingungen des Anbieters prüfen.

> 💡 **Ein Key, zwei Funktionen:** HERE gibt **einen** API-Key für alle Produkte aus. Ist ein HERE-Key hinterlegt (hier oder als `HERE_TRAFFIC_API_KEY`), nutzt ihn ConvoyPlan **automatisch auch für die Adresssuche** im Plan-Editor (HERE Geocoding & Search, serverseitig proxied — der Key verlässt den Server nie). Ohne Key läuft die Adresssuche wie bisher über das offene Photon (komoot). Ein eigener `HERE_API_KEY` überschreibt den Traffic-Key nur für die Adresssuche.

> 💰 **Kostendeckel für die Adresssuche:** Damit auf dem HERE-**Base-Plan** (30.000 Transaktionen/Monat gratis, darüber kostenpflichtig) **keine Kosten** entstehen, begrenzt `HERE_MONTHLY_LIMIT` die HERE-Anfragen pro Kalendermonat (Standard **25.000**, mit Puffer unter dem Freikontingent). Ist der Deckel erreicht, läuft die Adresssuche für den Rest des Monats automatisch über das kostenlose Photon weiter — kein Ausfall. `0` schaltet den App-Deckel ab (dann greift nur HEREs eigenes Kontingent). Der Client reduziert die Anfragen zusätzlich (Suche erst ab 3 Zeichen, längere Tipp-Pause, Cache identischer Eingaben).
>
> **Ganz ohne Kostenrisiko:** Wer den **Limited Plan** (ohne Kreditkarte, 1.000 Anfragen/Tag) nutzt, kann grundsätzlich nicht abgerechnet werden — dort führt ein erschöpftes Kontingent nur zum automatischen Photon-Fallback.

---

3. 🇩🇪 Bundesweite Baustellen über die mobilithek (DATEX II)

Für Baustellen/Sperrungen **abseits der Autobahn** in weiteren Bundesländern nutzt ConvoyPlan die **mobilithek** (nationaler Zugangspunkt des Bundes) im Standard **DATEX II**. Der Zugang ist kostenlos, erfordert aber ein Konto, ein Zertifikat und ein abonniertes Datenangebot.

3.1 Konto, Organisation & Maschinenkonto (in der mobilithek)

1. Auf mobilithek.info registrieren und freischalten lassen.
2. Eine **Organisation** anlegen (Zertifikate & Datennutzungen hängen an der Organisation, nicht am persönlichen Profil).
3. In der Organisation ein **Maschinenkonto** anlegen und dafür ein **Client-Zertifikat** erzeugen. Beim Download vergibst du ein Passwort für die `.p12`-Datei — merken!

3.2 Baustellen-Angebote abonnieren → Broker-URLs

1. Unter **Datenangebote** ein **DATEX-II-Baustellen-Angebot** eines Bundeslands öffnen (Filter: Kategorie „Verkehr", Lizenz „frei/kostenlos").
2. Eine **Datennutzung** als **HTTP Client Pull** anlegen und mit dem Maschinenkonto verknüpfen.

3. Du erhältst eine **Broker-Abruf-URL** in dieser Form:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/subscription/<ID>/clientPullService?
subscriptionID=<ID>
```

Pro Bundesland ein Abo = eine URL.

3.3 Zertifikat konvertieren & auf dem Server ablegen

ConvoyPlan liest das Zertifikat als **PEM** (nicht `.p12`). Auf dem Server im Installationsverzeichnis (Standard `~/convoyplan`):

```
cd ~/convoyplan
mkdir -p secrets

# .p12 → PEM (fragt nach dem beim Download vergebenen Passwort):
openssl pkcs12 -in /pfad/zu/certificate.p12 -out secrets/mobilithek-client.pem -nodes

# WICHTIG: der Container-User (appuser, UID 1001) muss lesen dürfen:
sudo chown -R 1001:1001 secrets
sudo chmod 700 secrets
sudo chmod 600 secrets/*
```

🔒 **Geheimhaltung:** `mobilithek-client.pem` enthält den privaten Schlüssel — nur auf dem Server, **niemals ins Git**. (Der Ordner `secrets/` ist per `.gitignore` ausgeschlossen.)

3.4 `.env` konfigurieren

In `~/convoyplan/.env` ergänzen — die Feeds kommasepariert, jeweils als `datex2|<url>`:

```
OPENDATA_TRAFFIC_CLIENT_CERT=/secrets/mobilithek-client.pem
OPENDATA_TRAFFIC_FEEDS=mobidata_bw|https://api.mobidata-
bw.de/datasets/traffic/roadworks/roadworks_geojson.json,berlin_viz|https://api.viz.berlin.de/daten/
baustellen_sperrungen_viz.json,datex2|<broker-url-land-1>,datex2|<broker-url-land-2>
```

⚠️ **`OPENDATA_TRAFFIC_CA_CERT` NICHT setzen!** Der mobilithek-Broker (`mobilithek.info:8443`) nutzt ein **öffentliches Telekom-Zertifikat** — die Prüfung läuft über den System-Truststore. Setzt man die mobilithek-„CA-Kette", **bricht** die Verbindung (`self-signed certificate in chain`).

3.5 Neu starten & prüfen

```
docker compose up -d backend

# Erreichbarkeit + Anzahl der Verkehrs-Features:
docker compose exec backend python -c "import asyncio; from app.services import opendata_traffic as
o; print(len(asyncio.run(o._get_features())), 'Features')"
```

Steigt die Zahl deutlich an, fließen die Länder-Feeds. Auf der Karte erscheinen die Baustellen nach spätestens dem 5-Minuten-Cache.

---

4. 🇦🇹🇨🇭 Baustellen in Österreich und der Schweiz

Die Routenplanung deckt den gesamten DACH-Raum ab, die amtlichen Baustellenfeeds oben sind aber **deutsche** Quellen. Für den österreichischen und Schweizer Teil gibt es zwei Wege:

- **Ohne Setup:** Overpass (OpenStreetMap) liefert überall, und die Live-Verkehrslage von HERE/TomTom (Abschnitt 2) deckt DACH mit ab. Damit sieht man Stau und dauerhaft in OSM erfasste Sperrungen — aber nicht die kurzfristigen amtlichen Meldungen.
- **Mit Setup:** Sowohl die **ASFINAG** (AT) als auch **opentransportdata.swiss** (CH) veröffentlichen Verkehrsmeldungen im Standard **DATEX II** — dasselbe Format wie die mobilithek. Sie lassen sich deshalb ohne Codeänderung als weitere `datex2`-Feeds eintragen.

4.1 Zugang besorgen

Beide Angebote sind kostenlos, aber registrierungspflichtig. Der Ablauf ist bei beiden gleich: Konto anlegen, das Verkehrsmeldungs-Angebot abonnieren, und einen **API-Key bzw. Token** erzeugen. Die konkrete Abruf-URL bekommst du dabei aus dem jeweiligen Portal — sie steht bewusst nicht hier, weil beide Anbieter ihre Endpunkte gelegentlich umstellen.

Anders als der mobilithek-Broker verlangen beide **kein Client-Zertifikat**, sondern die Kennung im HTTP-Header.

4.2 Eintragen

Die Abruf-URL kommt in die Feed-Liste, die Kennung getrennt davon in `OPENDATA_TRAFFIC_API_KEYS`:

```
OPENDATA_TRAFFIC_FEEDS=...,datex2|https://<asfinag-
endpunkt>,datex2|https://api.opentransportdata.swiss/<endpunkt>
OPENDATA_TRAFFIC_API_KEYS=<asfinag-host>|X-API-Key:
<key>,api.opentransportdata.swiss|Authorization:Bearer <token>
```

Warum getrennt? Die Feed-Liste ist damit frei von Geheimnissen und kann unverändert weitergegeben oder in ein Ticket kopiert werden. Der Header geht außerdem **nur an genau den angegebenen Host** — nicht an andere Feeds und nicht über einen Redirect hinweg.

> ⚠️ Header-Werte dürfen **kein Komma** enthalten, weil die Liste kommasepariert ist.

4.3 Prüfen

Wie in Abschnitt 3.5: Feature-Zahl vor und nach der Änderung vergleichen. Kommt nichts dazu, ist meist der Header falsch benannt — der Abruf schlägt dann still fehl, damit ein kaputter Feed die anderen nicht mitreißt.

---

5. 🛠️ Fehlerbehebung

Symptom	Ursache & Lösung
Backend startet nicht, Log: `Can't locate revision '00xx'`	Das laufende Image ist **älter** als die Datenbank. Aktuelles Image ziehen: `docker compose pull backend && docker compose up -d backend`. **Nicht** die DB downgraden.
`PermissionError [Errno 13]` beim Zertifikat	Datei gehört nicht dem Container-User. `sudo chown 1001:1001 secrets/* && chmod 600 secrets/*`.

Symptom	Ursache & Lösung
`CERTIFICATE_VERIFY_FAILED: self-signed certificate in chain`	`OPENDATA_TRAFFIC_CA_CERT` ist gesetzt — **entfernen** (der Broker nutzt ein öffentliches Zertifikat).
`HTTP 400` beim `curl`-Test	Fehlender `Accept-Encoding: gzip`-Header — der Broker liefert immer gzip. Beim `curl`-Test `--compressed` nutzen; ConvoyPlan (httpx) sendet das automatisch.
`HTTP 204` / leer	Noch **kein Datenpaket** im Puffer — das Abo ist frisch oder der Anbieter hat noch nicht publiziert. Kommt automatisch.
`HTTP 403` / `404`	Datennutzung/Zertifikat nicht scharfgeschaltet bzw. Abo ungültig — in der mobilithek den Status der Datennutzung prüfen.
Feature-Zahl steigt nicht	Feeds testen (siehe unten), `.env`-Werte prüfen (`docker compose exec backend printenv

Einzelnen Feed direkt testen (auf dem Server, mit Zertifikat):

```
curl -s --compressed --cert secrets/mobilithek-client.pem \
  "https://mobilithek.info:8443/mobilithek/api/v1.0/subscription/<ID>/clientPullService?
subscriptionID=<ID>" \
  | head -c 300
```

Kommt `<?xml … <d2LogicalModel …` zurück, ist alles korrekt.

📋 Referenz — Umgebungsvariablen

Variable	Zweck
`OPENDATA_TRAFFIC_ENABLED`	Offene Baustellenfeeds an/aus (Standard `true`)
`OPENDATA_TRAFFIC_FEEDS`	Feed-Liste `format|url`, kommasepariert. Formate: `mobidata_bw`, `berlin_viz`, `datex2`
`OPENDATA_TRAFFIC_API_KEYS`	Zugangsdaten je Host: `host|Header-Name:Wert`, kommasepariert. Für Feeds mit API-Key/Token statt Zertifikat (ASFINAG, opentransportdata.swiss)
`OPENDATA_TRAFFIC_CLIENT_CERT`	Pfad zum mTLS-Client-Zertifikat (PEM) für `datex2`-Feeds
`OPENDATA_TRAFFIC_CA_CERT`	Nur für Broker mit **privater** CA. Für die mobilithek **leer lassen**.
`HERE_TRAFFIC_API_KEY` / `TOMTOM_TRAFFIC_API_KEY`	Live-Verkehrslage (alternativ im Admin-Panel)
`OVERPASS_URLS`	Overpass-Server, kommagetrennt (Standard: overpass-api.de)
`TRAFFIC_FLOW_PROVIDER`	`here` / `tomtom` erzwingen (leer = automatisch, HERE bevorzugt)
`HERE_API_KEY`	Adresssuche über HERE Geocoding & Search. Leer = HERE-Traffic-Key mitbenutzen bzw. Photon-Fallback
`HERE_MONTHLY_LIMIT`	Kostendeckel: max. HERE-Anfragen/Monat für die Adresssuche (Standard 25000, `0` = aus). Deckel erreicht → Photon-Fallback

> 💡 Die Zertifikatsdateien werden über den Host-Ordner `./secrets` read-only nach `/secrets` in den Container gemountet (siehe `docker-compose.yml`).

TEIL 2

Laufender Betrieb

Auto-Updater

Backup, Restore & Verschlüsselung at-rest

Lizenz und Demo-Modus

Sicherheit und Datenschutz

KAPITEL 5

Auto-Updater

ConvoyPlan bringt einen Updater-Container mit, der das GitHub-Repository pollt und neue Versionen automatisch deployt. Kanal und Modus lassen sich im Admin-Bereich (**Admin → Software-Update**) umschalten; die Env-Variablen dienen nur als Fallback vor dem ersten Setzen in der UI.

GitHub-Token (optional)

Der Updater fragt Releases und Build-Stände über die GitHub-API ab. Das Repository ist öffentlich, dafür braucht es **keinen Token**.

Ein `GITHUB_TOKEN` hebt nur das Rate-Limit: Ohne Token erlaubt GitHub 60 API-Anfragen pro Stunde **je IP-Adresse**, mit Token 5000. Eine einzelne Instanz kommt mit dem kleinen Kontingent aus (der Updater prüft alle 5 Minuten). Eng wird es erst, wenn mehrere Instanzen oder andere Werkzeuge hinter derselben öffentlichen IP GitHub abfragen. Dann meldet die Instanz „GitHub nicht erreichbar", bis die Stunde um ist.

Wer einen Token hinterlegt, nimmt einen fine-grained Token mit **„Public repositories (read-only)"** und **ohne** weitere Berechtigungen. Einen klassischen Token mit `repo` oder `public_repo` nicht nehmen: beide erlauben Schreibzugriff, `repo` sogar auf alle privaten Repositories des Kontos. Der Token steht im Klartext in der `.env` bzw. in der Datenbank.

Hinterlegt wird er im Admin-Bereich unter **Software-Update** (wirkt ohne Neustart) oder als Env-Variable. Die Installer fragen ihn nicht ab.

`GITHUB_REPO` legt das überwachte Repository fest (Standard `RettTechSolutions/ConvoyPlan`; bei Fork anpassen).

Update-Kanäle

Kanal	Verhalten	Docker-Tag
Stable (Standard, empfohlen)	Nur veröffentlichte GitHub-Releases	`:latest`
Beta	Nummerierte Vorabversionen / Release-Kandidaten (`vX.Y.Z-beta.N`)	`:beta`
Nightly	Jeder Commit auf `main`	`:nightly`
LTS (Wartungsvertrag)	Releases der aktuellen LTS-Linie: 12 Monate nur Sicherheits- und Fehlerkorrekturen	`:lts`

Der Beta-Kanal funktioniert auch bei image-basierten Standard-Installationen.

Fallback-Env: `UPDATE_CHANNEL=stable|beta|nightly|lts`.

Der Updater trägt die Tags des gewählten Kanals auch in die `.env` im Installationsverzeichnis ein (`BACKEND_IMAGE`, `FRONTEND_IMAGE`, `GRAPHHOPPER_IMAGE`, `UPDATER_IMAGE`, `REGION_MERGE_IMAGE`, `MEDIEN_IMAGE`) — beim Start und nach jedem Kanalwechsel. Ein `docker compose pull && docker compose up -d` von Hand zieht damit dieselben Images wie der Updater. Vorher las es die `.env` mit `:latest` und tauschte eine Nightly-Installation still auf das ältere Stable-Image zurück. Angefasst werden nur Einträge, die auf ein Image dieses Projekts mit Tag zeigen; ein eigener Mirror oder ein Digest (`@sha256:…`) bleibt stehen.

LTS-Kanal

Der LTS-Kanal gehört zum Wartungsvertrag. Wählbar ist er nur, wenn der Lizenzschlüssel eine LTS-Freigabe (`lts_until`) trägt und das Datum nicht überschritten ist. Die Freigabe steht unter **Admin → Lizenz**.

- Eine LTS-Linie ist eine Version wie `2026.7`, die 12 Monate lang nur Sicherheits- und Fehlerkorrekturen bekommt (`v2026.7.1`, `v2026.7.2`, …), keine neuen Funktionen.
- Pro Jahr beginnt eine neue Linie. Dann wechselt der Kanal von selbst auf sie — im Modus **Benachrichtigen** erst nach Klick auf „Jetzt updaten".
- Läuft die Freigabe ab, bleibt eine Installation auf dem LTS-Kanal stehen, statt ungefragt auf die neueste Stable-Version zu springen. Die Admin-Oberfläche weist darauf hin.

Update-Modi

Modus	Verhalten
Automatisch (`auto`, Standard)	Verfügbare Updates werden im gewählten Kanal automatisch installiert
Benachrichtigen (`notify`)	Es wird nicht automatisch installiert; stattdessen erhalten Superadmins eine E-Mail. Installation erfolgt manuell

Ergänzende Env-Variablen:

Variable	Beschreibung
`UPDATE_MODE`	`auto` oder `notify` (Fallback)
`UPDATE_NOTIFY_ON_AUTO`	Nur bei `auto`: `true` schickt zusätzlich eine Bestätigungs-Mail nach automatischer Installation (Standard `false`)
`UPDATE_NOTIFY_INTERVAL`	Prüfintervall in Sekunden für fällige Benachrichtigungen (Standard `1800`)

> Der `notify`-Modus und `UPDATE_NOTIFY_ON_AUTO` setzen einen konfigurierten SMTP-Dienst voraus (Admin → SMTP).

Kein automatisches Update, solange jemand angemeldet ist

Im Modus **Automatisch** wartet der Updater, solange **Nutzer oder Demo-Nutzer** im Portal aktiv sind (eine Anfrage in den letzten fünf Minuten) oder ein **Fahrer-Link** eines Konvois mit Status *aktiv* oder *läuft* verbunden ist (auch die Begleit-App; ein Fahrer-Link an einem Konvoi in Planung oder einem abgeschlossenen hält nichts auf, ebenso wenig einer aus einer Organisation, deren Plan abgelaufen ist; nach einem Verbindungsabriss noch fünf Minuten — ein Funkloch ist kein Feierabend). Superadmins zählen nicht — sonst hielte der Betreiber sein eigenes Update auf. Der Admin-Bereich zeigt neben „Update verfügbar", seit wann und wegen wie vieler Nutzer zurückgehalten wird; installiert wird beim ersten Zyklus, in dem niemand mehr aktiv ist.

- **„Jetzt updaten" wartet nicht.** Wer den Knopf drückt, hat entschieden.
- **Höchstens `UPDATE_HOLD_MAX` Sekunden am Stück** (Standard `86400` = 24 h, `0` = unbegrenzt). Danach wird trotzdem installiert — eine Instanz, auf der rund um die Uhr gearbeitet wird, bekäme sonst nie ein Update, auch keinen Sicherheitsfix.
- **Meldet das Backend nichts** (kein Lebenszeichen seit drei Minuten), wird installiert: Ein Backend, das hängt, darf das Update, das es reparieren könnte, nicht blockieren.

- **Tracker zählen**, wenn sie sich in den letzten fünf Minuten gemeldet haben und ihr Fahrzeug in einem Konvoi mit Status *aktiv* oder *läuft* steht — ein Deploy mitten in der Fahrt ließe ihre Bündel ins Leere laufen, auch wenn niemand mit Telefon dabei ist. Gesperrte Tracker zählen nicht, ebenso wenig Tracker einer gesperrten Organisation (Plan abgelaufen, Modul *Tracker* nicht freigegeben, Instanz ohne Lizenz) — sie bekommen ohnehin die Anweisung zu schweigen.
- Lese-Links zählen **nicht**: Über sie plant und fährt niemand.

Update-Status und manueller Trigger

Der Admin-Bereich zeigt den aktuellen Deploy-SHA und den GitHub-Stand. Ein Update lässt sich per Button manuell anstoßen; der Updater-Prozess wird als Live-Log (SSE) im Browser mitgeschrieben.

Relevante Endpunkte:

Methode	Endpunkt	Zweck
`POST`	`/api/admin/trigger-update`	Update manuell anstoßen
`GET`	`/api/admin/update-status`	Deploy-Stand abrufen
`GET`	`/api/admin/update-log`	Live-Update-Log (SSE)
`GET/PUT`	`/api/admin/settings/update-channel`	Kanal lesen/setzen
`GET/PUT`	`/api/admin/settings/update-mode`	Modus lesen/setzen

Selbstheilung bei fehlgeschlagenen Deployments

Ein Backend-Image, das älter als der DB-Migrationsstand ist (z. B. ein versehentliches Downgrade auf ein `:beta`-Image, das eine bereits angewendete Migration nicht kennt), konnte früher die gesamte API lautlos lahmlegen — Caddy lieferte auf alle `/api/*`-Routen 502, ohne dass jemand benachrichtigt wurde. Vier Schutzebenen verhindern das jetzt:

Ebene	Wirkung
Healthcheck	Der `backend`-Dienst hat einen Docker-Healthcheck auf `/health`; Docker und der Updater erkennen einen crashenden Container, statt „gestartet" als Erfolg zu werten
Robuster Boot-Entrypoint (`backend/docker-entrypoint.sh`)	Erkennt „Image älter als DB-Schema", protokolliert eine klare Diagnose, hinterlegt einen Alert-Marker und bricht bewusst fail-closed ab
Automatischer Rollback	Der Updater merkt sich vor jedem Deploy das laufende Backend-Image (per Image-ID), wartet nach dem Deploy auf `healthy` und stellt bei Fehlschlag automatisch die vorherige Version wieder her (beide Updater-Varianten)
Alert an Superadmins	Der wieder gesunde Backend-Container liest den Alert-Marker und benachrichtigt alle Superadmins per E-Mail über den fehlgeschlagenen Deploy bzw. Rollback — genau einmal pro Ereignis

Routing-Graph: kein Deploy mitten im Import

GraphHopper baut aus den OSM-Daten einen Routing-Graphen. Bei den größeren Kartenregionen dauert das 45–75 Minuten, und in dieser Zeit routet die Instanz nicht. Ein Deploy, der den Container mitten darin austauscht, hinterlässt Bruchstücke im Graph-Volume — der Import beginnt anschließend von vorn.

Der Updater nimmt den `graphhopper`-Dienst deshalb aus dem Deploy heraus, solange dort ein Import läuft, und zieht ihn nach, sobald der Graph fertig ist. Im Update-Log steht dann:

```
GraphHopper baut gerade den Routing-Graphen (kein /data/graph/edges) – dieser
Dienst wird aus dem Deploy herausgenommen und nach dem Import nachgezogen;
alle anderen werden aktualisiert.
```

Alle übrigen Dienste werden normal aktualisiert; das Update gilt als erfolgreich. Nach spätestens vier Stunden ohne fertigen Graphen wird der Dienst trotzdem mitgetauscht — sonst käme ein Container, der aus einem anderen Grund nie fertig wird (zu wenig Speicher, volle Platte), nie an das Update, das den Fehler behebt.

Bleibt trotzdem ein unvollständiger Graph zurück (Host-Neustart, abgeschossener Container), erkennt GraphHopper das beim nächsten Start selbst, räumt das Verzeichnis und baut neu. Das kostet den Import erneut, aber die Instanz routet danach wieder statt in einer Neustart-Schleife zu hängen.

Sichtbar ist ein laufender Bau an zwei Stellen. Die öffentliche Statusseite (`/status`) nennt ihn als Grund, warum die Routenplanung gerade nicht verfügbar ist. Im Admin-Portal steht er unter **System → Kartenregion**, mit Beginn und Dauer:

- **Kartendaten werden heruntergeladen**: Es liegt noch kein Extract vor, GraphHopper lädt es und baut danach.
- **Routing-Graph wird neu aufgebaut**: Der Import läuft, die Karte zeigt, seit wann.
- **Import ohne Abschluss**: Seit mehr als vier Stunden gibt es keinen fertigen Graphen. Das ist dieselbe Frist wie oben. Die Ursache steht im Container-Log (`docker compose logs graphhopper`), meist zu wenig Speicher oder eine volle Platte. Ein Neustart des Containers räumt den Rest weg und baut neu. Die Statusseite meldet diesen Fall nicht mehr als Neuaufbau, sondern als Störung.

Solange heruntergeladen oder importiert wird, startet kein Regionswechsel **sofort**: „Wechsel starten" ist gesperrt, und auch die API lehnt ab (409). Ein Wechsel liefe parallel zum laufenden Import, konkurrierte mit ihm um den Speicher und würfe ihn am Ende weg. **Einplanen** geht dagegen. Der Updater startet einen geplanten Wechsel erst, wenn sein Termin erreicht **und** der Aufbau fertig ist, und schreibt das Warten einmal ins Log. Die Karte zeigt dann „Fällig, aber GraphHopper baut gerade seinen Routing-Graphen". Ein hängender Import hält nichts auf: Dann kann ein Wechsel auf eine kleinere Region gerade der Ausweg sein.

Abgelesen wird das an den Dateien im Graph-Volume, nicht am Container. Einen Lebensbeweis für den Import gibt es dabei nicht: Ein Container, der mitten im Import angehalten wurde und nicht wiederkommt, sieht bis zum Ablauf der Frist aus wie ein laufender Import.

Host-Watchdog

Ein systemd-Timer (`scripts/updater-watchdog.sh`) räumt verwaiste Updater-Container auf und startet abgestürzte Updater neu. Er wird vom Linux-Installer eingerichtet.

> In Portainer übernimmt der Stack-Update-Mechanismus von Portainer selbst das Deployment neuer Images; der `updater`-Container ist nur in `docker-compose.yml` enthalten.

KAPITEL 6

Backup, Restore & Verschlüsselung at-rest

> Bezug: `docs/iso-certifications-review.md` (T11 Backup, T12 Verschlüsselung at-rest) · ISO/IEC 27001 A.8.13 (Backup), A.8.24 (Kryptografie), ISO 22301 (Business Continuity)

1. Was gesichert werden muss

Komponente	Speicherort	Inhalt
PostgreSQL	Volume `…_postgres_data`	Benutzer, Organisationen, Konvois, Fahrzeuge, Audit-Log, Einstellungen
Uploads	Volume `…_logo_uploads` (`/uploads`)	Hochgeladene Branding-Logos
TLS / Caddyfile	Volume `…_cert_uploads` (`/certs`)	Persistierte Caddyfile + ggf. eigene Zertifikate

> Nicht sicherungsbedürftig: `osm_data`, `gh_graph` (GraphHopper-Routing-Graph, wird bei Bedarf neu gebaut), `caddy_data` / `caddy_config` (ACME-Zertifikate werden automatisch neu ausgestellt).

2. Backup

```
scripts/backup.sh
```

Erzeugt unter `./backups/<YYYYMMDD-HHMMSS>/`:

- `database.sql.gz` — komprimierter `pg_dump` (mit `--clean --if-exists`, direkt per `psql` einspielbar)
- `uploads.tar.gz`, `certs.tar.gz` — die Named Volumes
- `docker-compose.yml` — Kopie der Stack-Definition
- `SHA256SUMS` — Integritäts-Prüfsummen

Automatisierung (Cron, täglich 03:00):

```
0 3 * * * /opt/convoyplan/scripts/backup.sh >> /var/log/convoyplan-backup.log 2>&1
```

Konfiguration (Env-Variablen oder `.env`):

Variable	Default	Zweck
`BACKUP_DIR`	`./backups`	Zielverzeichnis
`BACKUP_RETENTION_DAYS`	`30`	Ältere Backup-Ordner werden gelöscht
`COMPOSE_PROJECT_NAME`	`convoyplan`	Prefix der Volume-/Container-Namen
`DB_CONTAINER`	`<project>-db-1`	Name des Postgres-Containers

> **Off-site:** Backups gehören zusätzlich an einen zweiten Ort (verschlüsselt) — z. B. per `rsync` / `rclone` auf externen Speicher. Lokale Backups allein schützen nicht vor Hardware-/Standortausfall (ISO 22301).

---

3. Restore

```
scripts/restore.sh ./backups/20260603-030000
```

Das Skript ist **destruktiv** (überschreibt DB + Dateien) und verlangt zur Bestätigung die Eingabe `RESTORE`. Danach:

```
docker compose restart backend caddy
```

Restore regelmäßig testen — ein ungetestetes Backup ist kein Backup. Empfehlung: vierteljährlich auf einer separaten Instanz einspielen und Login + Kartenansicht prüfen.

---

4. Verschlüsselung at-rest (T12)

ConvoyPlan verschlüsselt **Anwendungsgeheimnisse selektiv** (MFA-TOTP-Secrets via Fernet, siehe `services/crypto.py`). Die **vollständige** Verschlüsselung der Daten im Ruhezustand erfolgt auf **Infrastruktur-Ebene** und ist deployment-abhängig:

4.1 Festplatten-/Volume-Verschlüsselung (empfohlen)

- **LUKS (Linux):** Das gesamte Docker-Daten-Verzeichnis (`/var/lib/docker`) bzw. die darunterliegende Partition auf einem LUKS-Container ablegen. Schützt alle Volumes (DB, Uploads, Backups) transparent.
- **Cloud:** Verschlüsselte Block-Volumes nutzen (z. B. „encryption at rest" des Hosters aktivieren).

4.2 Datenbank-Ebene (optional, zusätzlich)

- PostgreSQL-TDE-Varianten oder verschlüsselte Tablespaces, falls regulatorisch gefordert. Für die meisten Self-hosted-BOS-Setups ist LUKS auf Host-Ebene ausreichend und einfacher zu betreiben.

4.3 Backups

- Backup-Zielverzeichnis ebenfalls auf verschlüsseltem Volume ablegen **oder** die Archive vor dem Off-site-Transfer verschlüsseln (z. B. `age` / `gpg`):

  ```
  age -r <recipient-key> -o backup.tar.gz.age backup.tar.gz
  ```

4.4 Schlüsselverwaltung

Geheimnis	Quelle	Hinweis
`JWT_SECRET`	`.env` / Secrets-Store	Stark (`openssl rand -hex 32`), nicht rotieren ohne Grund
`MFA_ENCRYPTION_KEY`	`.env` (optional)	Wird sonst aus `JWT_SECRET` abgeleitet — bei Rotation müssen MFA-Secrets neu eingerichtet werden

Geheimnis	Quelle	Hinweis
`POSTGRES_PASSWORD`	`.env`	Stark wählen
LUKS-Passphrase	Host / TPM / KMS	Nicht neben den Daten ablegen

> `.env` enthält Klartext-Geheimnisse → Dateirechte `chmod 600`, nicht ins Backup-Archiv mit Off-site-Versand ohne zusätzliche Verschlüsselung.

---

5. Content-Security-Policy (CSP) scharfschalten

Die CSP wird standardmäßig im **Report-Only**-Modus ausgeliefert (bricht die Karten-UI nicht, meldet nur Verstöße im Browser-Log). Zum Erzwingen:

1. `CSP_ENFORCE=true` in `.env` setzen.
2. Stack neu starten (`docker compose up -d caddy backend`).
3. Karte, Routing, Live-Tracking und Geocoding-Suche im Browser prüfen; bei Verstößen in der Konsole die betroffenen Quellen in der Policy ergänzen (`caddy/entrypoint.sh` bzw. `setup.py::_security_header_block`).

Die Policy erlaubt out-of-the-box `tile.openstreetmap.org` (Karten), `nominatim.openstreetmap.org` und `photon.komoot.io` (Adress-Suche/Geocoding), MapLibre-Worker (`blob:`) und Same-Origin-WebSockets.

KAPITEL 7

Lizenz und Demo-Modus

ConvoyPlan läuft ohne gültigen Lizenzschlüssel im **Demo-Modus**. Lesezugriffe sind uneingeschränkt möglich; alle schreibenden Operationen sind gesperrt.

Demo-Modus

Aspekt	Verhalten
Lesezugriffe (GET)	uneingeschränkt
Schreibzugriffe (POST/PUT/PATCH/DELETE)	mit **HTTP 402** gesperrt
Tracker	melden sich weiter, hören aber `schweigen` und senden keine Positionen; neue Tracker einrichten geht nicht
Eignung	Tests und Evaluierung — **nicht** für den Einsatzbetrieb

Der Demo-Bereich im Admin-Portal

Alles zur Demo steht in einem eigenen Reiter **Demo** — vorher lag es verstreut unter **System**, zwischen Lizenz, Updates und MFA. Der Reiter ist in fünf Abschnitte gegliedert, in der Reihenfolge, in der man sie üblicherweise braucht:

1. **Übersicht** — Kennzahlen auf einen Blick (siehe unten).
2. **Demo-Modus** — an/aus, Sitzungslaufzeit, Karenzzeit je IP, Nachfrage-Mail.
3. **Offene Demo-Sitzungen** — wer gerade testet, verlängern und beenden.
4. **Interessenten aus der Demo** — die Kontakte samt Stand der Nachfrage.
5. **Gesperrte IP-Adressen** und **Dauerhaft freigestellte Adressen** — die Zugangssteuerung.
6. **E-Mail-Vorlage: Nachfrage nach der Demo** — der Wortlaut der Mail, samt Vorschau und Testversand.

Die Kennzahlen

Gezählt wird in der Datenbank, nicht aus den darunter stehenden Listen: Die Interessentenliste zeigt die letzten 100 Einträge, die Übersicht soll aber auch bei mehr die richtige Zahl nennen.

Kachel	Bedeutung
Offene Sitzungen	Laufende Demo-Organisationen, dazu wie viele davon in den nächsten 24 Stunden ablaufen
Marschverbände	In laufenden Sitzungen angelegte Konvois — der Anhaltspunkt dafür, ob tatsächlich gearbeitet oder nur geschaut wurde
Interessenten	Alle Kontaktangaben in der Aufbewahrungsfrist, dazu die letzten 7 und 30 Tage
Nachfragen versandt	Bereits verschickte Nachfragen, dazu wie viele nach Sitzungsende noch folgen
Nachfragen ausstehend	Sitzung abgelaufen, Mail steht im nächsten Retention-Durchgang an. Bleibt die Zahl stehen, nennt die Kachel den Grund (Nachfrage abgeschaltet, kein SMTP)

Kachel	Bedeutung
Nachfragen aufgegeben	Nach drei Fehlversuchen liegen gelassen — in der Regel eine unzustellbare Adresse
Abbestellt	Adressen mit Widerspruch
Gesperrte IPs	Aktuell blockierte Adressen, dazu die Zahl der dauerhaften Ausnahmen

Die beiden Kacheln **ausstehend** und **aufgegeben** heben sich farblich hervor, sobald sie über 0 stehen. Im Normalbetrieb sind sie leer — deshalb ist eine Zahl dort ein Hinweis und kein Dauerzustand, den man nach einer Woche nicht mehr wahrnimmt.

Kontaktangabe beim Demo-Start

Der Demo-Knopf auf der Startseite führt auf `/demo`, wo vor dem Start eine **E-Mail-Adresse** abgefragt wird; **Vor- und Nachname sind freiwillig**. Ohne Adresse lässt sich weder auseinanderhalten, wer gerade in welcher Demo sitzt, noch nach Ablauf nachfragen — ein Name allein leistet beides nicht. Alles darüber hinaus kostet nur Abbrüche auf dem Weg in die Demo, deshalb bleibt es bei diesem einen Pflichtfeld.

Die Angabe landet in der Tabelle `demo_leads`, **nicht** als Kennung des Demo-Benutzers: Dort steht weiterhin die interne `demo-<hex>@demo.local`. Sonst kollidierte der Demo-Zugang mit einem bestehenden Konto derselben Adresse (Unique-Constraint), und ein Demo-Zugang wäre plötzlich unter einer echten Kennung anmeldbar.

Die Kontaktzeile liegt außerhalb der Demo-Organisation, weil sie diese überleben muss — die Nachfrage-Mail geht erst raus, wenn die Umgebung bereits gelöscht ist. Beim Aufräumen der Organisation fällt lediglich der Verweis (`org_id`) auf NULL; der Code der Sitzung (`org_slug`) bleibt als Text stehen.

Aufbewahrung: `RETENTION_DEMO_LEADS_DAYS`, Standard **180 Tage**. Danach löscht der Retention-Job die Zeile. Einzelne Kontakte lassen sich im Admin-Bereich vorzeitig löschen (Löschersuchen nach DSGVO Art. 17).

Nachfrage nach Ablauf der Sitzung

Ist eine Demo-Sitzung abgelaufen, geht **einmalig** eine E-Mail an die beim Start angegebene Adresse: Hat alles so funktioniert wie erwartet? Ist eine Frage offen geblieben? Soll man sich das Ganze gemeinsam in einer Session ansehen? Die Mail enthält einen Knopf auf die Kontaktseite (`DEMO_FOLLOWUP_CONTACT_URL`) und lässt sich schlicht beantworten.

Verschickt wird sie vom **Retention-Durchgang** (stündlich, `RETENTION_INTERVAL`), nicht von einem eigenen Dienst. Ausgelöst wird sie über den in der Kontaktzeile gespiegelten Ablaufzeitpunkt (`session_expires_at`) — der Abgleich funktioniert also unabhängig davon, ob die Organisation noch existiert. Verlängert oder beendet ein Superadmin eine Sitzung, wird dieser Zeitpunkt mitgeführt: Nach „+24 h" kommt die Nachfrage entsprechend später, nach „✕ Beenden" beim nächsten Durchgang.

Bedingung	Verhalten
Schalter **Admin** → **Demo** → „Nachfrage nach Sitzungsende" aus (bzw. `DEMO_FOLLOWUP_ENABLED=false`)	kein Versand
Kein SMTP konfiguriert	kein Versand, kein gezählter Versuch
Postfach unzustellbar	Fehler wird vermerkt, Durchgang läuft weiter; nach **3** Versuchen ruht die Adresse

Bedingung	Verhalten
Adresse hat abbestellt	kein Versand — auch nicht nach einer späteren Demo

Einwilligung und Widerspruch. Es gibt kein Opt-in-Häkchen beim Start; das Formular weist stattdessen auf die einmalige Nachfrage hin. Dafür trägt jede Mail im Fußbereich einen Abmeldelink (`/demo/abmelden?token=…`). Ein Klick darauf vermerkt den Widerspruch für **die Adresse**, nicht für die einzelne Sitzung — ein späterer Demo-Start hebt ihn nicht wieder auf.

Wortlaut ändern. Ganz unten im Reiter steht der Abschnitt **E-Mail-Vorlage: Nachfrage nach der Demo**: Betreff und HTML lassen sich dort bearbeiten, mit **Vorschau** (öffnet ein Musterexemplar im neuen Tab), **Testmail an mich** (schickt dasselbe Musterexemplar an die eigene Adresse — der Weg dahin ist derselbe wie im Betrieb, kommt sie an, kommt auch die echte an) und **Auf Standard zurücksetzen**. Welche Platzhalter erlaubt sind, steht aufgeklappt unter „Verfügbare Variablen"; alles andere in geschweiften Klammern bleibt unverändert stehen, ein Tippfehler verschluckt den Text also nicht.

Interessentenliste. Der Abschnitt **Interessenten aus der Demo** (Admin → Demo) listet die Kontakte mit Sitzungscode, Startzeitpunkt und Stand der Nachfrage (ausstehend / versandt / fehlgeschlagen / abbestellt / aufgegeben). Er überdauert die Sitzung selbst — genau dafür ist er da.

Offene Demo-Sitzungen verwalten (Admin)

Jede Demo-Nutzung läuft als eigene, befristete Organisation. Der Reiter **Admin** → **Demo** listet im Abschnitt **Offene Demo-Sitzungen** alle offenen Sitzungen mit Name, **Kontakt** (die beim Start angegebene Adresse), Ablaufzeit, Anzahl angelegter Konvois und **Herkunft**.

Die Herkunft (Stadt/Region/Land) wird beim Start der Sitzung aus der Client-IP ermittelt und per Hintergrund-Geolokation (ipapi.co) angereichert — ein langsamer oder nicht erreichbarer Geo-Dienst verzögert den Demo-Start nie, die Anfrage läuft als Background-Task nach der Antwort. Private oder ungültige IPs (z. B. lokale Entwicklung) werden nicht abgefragt und bleiben ohne Herkunftsangabe. Die Daten helfen, Demo-Sitzungen Interessenten zuzuordnen.

Sitzungen lassen sich hier verlängern oder sofort beenden. Herkunftsdaten (`created_ip` , `created_location`) werden zusammen mit der Demo-Organisation vom bestehenden Retention-Job gelöscht — es gibt keine gesonderte Aufbewahrungsfrist dafür.

Eine Demo je IP-Adresse (Karenzzeit)

Damit die Demo eine Vorführung bleibt und nicht als Dauerbetrieb genutzt wird, kann je Client-IP nur **eine** Sitzung pro Karenzzeit gestartet werden — Standard **24 Stunden**, einstellbar unter **Admin** → **Demo** → „Karenzzeit je IP" (`0` schaltet die Sperre ab) bzw. über `DEMO_IP_COOLDOWN_HOURS` . Ein weiterer Versuch innerhalb des Fensters wird mit **HTTP 429** und `Retry-After` abgewiesen.

Der Besucher sieht dabei den Grund der Absage, nicht nur die Absage: Startseite und `/demo` zeigen die Begründung des Backends unverändert an — welche Karenzzeit gilt, wann der nächste Versuch möglich ist, oder dass der Demo-Modus gerade abgeschaltet ist. Nur wenn keine Antwort des Backends vorliegt (Netzwerkfehler), erscheint der allgemeine Hinweis „Bitte später nochmal versuchen".

Dazu nennt die Absage den **genauen Zeitpunkt** der Wiederfreigabe („Wieder möglich ab Donnerstag, 14.08.2026 um 09:35 Uhr"), nicht nur die verbleibende Dauer. Das Backend liefert ihn als ISO-Zeitstempel (`retry_at` im Antwortkörper, zusätzlich zu `Retry-After`); der Browser rechnet ihn in die Zeitzone des Besuchers um, sodass die Angabe auch außerhalb Mitteleuropas stimmt.

Wiedereinstieg in die laufende Sitzung

Karenzzeit und Sitzungslaufzeit sind zwei verschiedene Fristen — die Karenzzeit (Standard 24 h) ist üblicherweise länger als die Laufzeit einer Sitzung (Standard 2 h). Wer die Demo startet und danach den Tab schließt, den Rechner wechselt oder den Browserspeicher leert, verliert sein Token und stand bislang vor der Sperre, obwohl **seine eigene** Sitzung noch lief.

Deshalb wird vor der Absage geprüft, ob zu dieser IP-Adresse noch eine gültige Demo-Sitzung existiert. Wenn ja, gibt `POST /api/auth/demo-session` sie mit einem frischen Token zurück (`resumed: true`) statt einer 429-Absage — der Besucher landet wieder in seiner Umgebung samt allem, was er dort angelegt hat, und das Banner oben sagt „Demo-Sitzung fortgesetzt". Erst wenn die Sitzung abgelaufen ist, folgt die Absage mit dem Zeitpunkt der Wiederfreigabe.

Der Wiedereinstieg hängt allein an der IP-Adresse — demselben Merkmal, an dem auch die Karenzzeit hängt. Hinter einem gemeinsamen Anschluss (Büro-NAT, Mobilfunk-CGNAT) landen mehrere Besucher damit in derselben Demo-Umgebung; bislang bekam der zweite dort gar keine. Wo das nicht erwünscht ist, gehört der Anschluss auf die Liste der **dauerhaft freigestellten Adressen** (siehe unten): dort wird jedes Mal frisch angelegt und nie fortgesetzt. Der Wiedereinstieg wird im Audit-Log als `demo.session.resumed` vermerkt; die Karenzzeit selbst verlängert er nicht.

Die Sperre wird in der Datenbank geführt (Tabelle `demo_origins`) und hat damit zwei Eigenschaften, die ein reiner Zähler im Arbeitsspeicher nicht hätte: Sie übersteht einen **Neustart des Backends** (also jedes Update), und sie gilt auch dann noch, wenn die Demo-Organisation längst abgelaufen und gelöscht ist — bei kurzer Sitzungslaufzeit ist das der Normalfall.

Hinter einem Firmenanschluss oder einem Messe-WLAN teilen sich alle Besucher eine Adresse. Für diesen Fall listet der Abschnitt **Gesperrte IP-Adressen** die aktuell blockierten Adressen mit Zeitpunkt der letzten Demo, Freigabezeitpunkt und Gesamtzahl der Sitzungen — einzelne Sperren lassen sich dort direkt aufheben. Abgelaufene Einträge löscht der Retention-Job; die Adressen werden also nicht länger gespeichert als die Sperre gilt.

Dauerhaft freigestellte Adressen

Manche Anschlüsse sollen gar nicht erst gesperrt werden: der eigene Vertrieb, ein Firmenanschluss beim Interessenten, das WLAN auf einer Messe oder in einer Schulung. Dort einzeln zu entsperren wäre nach jeder Vorführung erneut nötig. Der Abschnitt **Dauerhaft freigestellte Adressen** (Admin → Demo) nimmt solche Anschlüsse dauerhaft von der Karenzzeit aus.

Eingetragen wird entweder eine einzelne Adresse (`203.0.113.7`) oder ein ganzes Netz in CIDR-Schreibweise (`203.0.113.0/24`, ebenso IPv6: `2001:db8::/32`) — Letzteres, weil ein Firmenanschluss selten auf eine feste Adresse festgelegt ist. Dazu gehört eine Notiz, damit in einem Jahr noch nachvollziehbar ist, warum ausgerechnet dieses Netz freigestellt ist.

Zwei Dinge geschehen dabei automatisch:

- **Eine laufende Sperre für die Adresse wird mit aufgehoben.** Wer einen Anschluss freistellt, will ihn jetzt freigeschaltet haben und nicht erst nach Ablauf der Karenzzeit.
- **Freigestellte Adressen werden nicht mehr mitgeschrieben.** Sie tauchen in der Liste der gesperrten Adressen also gar nicht erst auf.

Hostbits werden beim Speichern verworfen: Aus `203.0.113.7/24` wird `203.0.113.0/24`, damit der Eintrag das bedeutet, was dort steht. Ein Netz ohne Präfixlänge (`0.0.0.0/0`) wird abgelehnt — das würde die Karenzzeit stillschweigend für alle abschalten; dafür gibt es die Einstellung „Karenzzeit 0". Anlegen und Entfernen werden im Audit-Log vermerkt.

> 📖 Endpunkte: siehe API-Dokumentation → Demo-Sitzungen.

Lizenz beantragen und aktivieren

1. **Instanz-UUID abrufen** — im Admin-Bereich unter **System** oder per API:

   ```
   GET /api/license/instance-id
   ```

   Die UUID wird während der Installation erzeugt und identifiziert die Instanz.

2. **Lizenzschlüssel anfordern** — mit der UUID beim Anbieter (ConvoyPlan-Lizenzmanager) einen Schlüssel erstellen lassen.

3. **Aktivieren** — Schlüssel im Admin-Bereich **System** eintragen oder per API:

   ```
   POST /api/license/activate
   { "license_key": "<lizenzschlüssel>" }
   ```

   Der Schlüssel wird validiert, gespeichert und der Lizenz-Cache ohne Neustart zurückgesetzt.

Ablauf ankündigen

Nach dem Ablaufdatum fällt die Instanz in den Demo-Modus — dann ist sie nur noch lesbar. Damit das nicht mitten im Betrieb passiert, kündigt die Instanz den Ablauf selbst an. Sie braucht dafür keine Verbindung nach außen: das Datum steht im Schlüssel.

Wann	Mail an alle Superadmins	Im Portal
30 Tage vorher	einmal	Hinweisleiste oben (gelb) und Plakette neben „Gültig bis" unter **Admin → System → Lizenz**
7 Tage vorher	einmal	Hinweisleiste wird orange
nach dem Ablauf	einmal (nur in den ersten 30 Tagen danach)	der bekannte Demo-Banner

- **Jede Stufe genau einmal** je Schlüssel, auch über Neustarts hinweg. Wer erst in der letzten Woche aktualisiert, bekommt nur die 7-Tage-Mail, nicht beide.
- **Ein neuer Schlüssel fängt von vorn an** — mit ihm verschwindet auch die Hinweisleiste, ohne Neuladen.
- **Ohne SMTP** geht keine Mail hinaus; Hinweisleiste und Plakette erscheinen trotzdem. Sobald SMTP eingerichtet ist, holt der nächste Durchgang die fällige Mail nach.
- **Nur Superadmins** sehen die Hinweisleiste, denn nur sie können einen Schlüssel eintragen. Auf den Fahrer-Seiten (`/track`) erscheint sie nicht.
- Die Mail nennt die **Instance ID** — die braucht der Anbieter für den neuen Schlüssel. Steht der Schlüssel in `LICENSE_KEY`, sagt die Mail, dass er dort zu ersetzen ist; ein Eintrag im Portal bliebe dann wirkungslos.

Geprüft wird alle 6 Stunden (`LICENSE_EXPIRY_CHECK_INTERVAL`, Sekunden). `GET /api/license/status` liefert dazu `expires_in_days` (0 = heute letzter gültiger Tag, negativ = abgelaufen) und `expiry_warning`.

Schlüsselquelle: Env vs. Datenbank

Der Lizenzschlüssel kann auf zwei Wegen gesetzt werden:

Quelle	Setzen über	Persistenz
`LICENSE_KEY`	Umgebungsvariable	im Container-Env
Admin-UI	Admin → System → Lizenz	in der Datenbank

`GET /api/license/status` liefert `demo_mode`, den aktuellen Status und `key_source` (woher der aktive Schlüssel stammt).

Was ein Lizenzschlüssel erlaubt

Ein Schlüssel schaltet den Demo-Modus ab. Funktionen schaltet er nicht frei — jede lizenzierte Instanz kann dasselbe. Seit dem Lizenzmanager mit Vorlagen (Payload v2) trägt er zusätzlich:

Feld	Bedeutung
`contract`	Vertragsart: `free` (Self-hosted kostenlos), `wartung_basis`, `wartung_plus`, `hosting`, `individuell` — nur zur Anzeige
`max_orgs`	Wie viele Organisationen auf der Instanz **angelegt** werden dürfen; `0` = unbegrenzt
`contract_until`	Ende des Wartungs- bzw. Hostingvertrags (optional)

Die Grenze für Organisationen greift ausschließlich beim **Anlegen** — über das Admin-Portal, über `POST /api/organizations/` und bei der Ersteinrichtung. Ist sie erreicht, antwortet das Backend mit **HTTP 402** und einer Erklärung. Bestehende Organisationen laufen in jedem Fall weiter. Demo-Organisationen zählen nicht mit.

Nach `contract_until` bleibt die Instanz lizenziert; nur für **neue** Organisationen gilt dann wieder die kostenlose Stufe (eine). Nichts wird abgeschaltet.

Ältere Schlüssel (ohne Versionsfeld) bleiben gültig und behalten, was sie erlaubt haben: beliebig viele Organisationen. Im Admin-Portal erscheinen sie als „Altschlüssel".

Im Admin-Portal unter **System → Lizenz** stehen Vertrag, Vertragsende und die Belegung („2 von 5 Organisationen"). `GET /api/license/status` liefert dafür `contract`, `contract_until`, `contract_ended`, `max_orgs`, `orgs_limit` (was fürs Anlegen gerade gilt) und `orgs_count`.

Pläne je Organisation (Hosting)

Betreibt jemand eine Instanz für mehrere Organisationen, gilt der Schlüssel für die ganze Instanz — gebuchte Pakete unterscheiden sich aber je Organisation. Dafür gibt es im Admin-Portal den Reiter **Pläne**. Dort setzt der Superadmin je Organisation ein Paket (Hosting S/M/L, Einsatz-Paket oder individuell); die Grenzen kommen aus dem Katalog und lassen sich je Organisation anpassen.

- **Ohne Plan keine Grenzen.** Das ist der Normalfall jeder eigenen Installation; ein Update ändert daran nichts.
- **Grenzen sind weich.** Gezählt werden Fahrzeuge, Planerzugänge (Rollen *Admin* und *Planer*) und Tracker; Fahrer und Beobachter zählen nie, ein gesperrter Tracker auch nicht. Tracker sind in keinem Paket des Katalogs begrenzt, eine Grenze setzt der Superadmin je Organisation. Wer mehr nutzt als gebucht, wird **nicht** abgewiesen — Anlegen klappt immer. Planer und Admins der Organisation sehen einen Hinweis, der Betreiber die Überschreitung in der Übersicht.
- **Ablauf mit Kulanz.** Hat ein Plan ein Ablaufdatum (beim Einsatz-Paket automatisch 30 Tage), laufen danach noch **14 Tage** weiter wie gehabt, mit Hinweis an alle Mitglieder. Danach ist die Organisation **nur**

noch lesbar — wie der Demo-Modus, aber nur für diese Organisation: schreibende Aufrufe (REST und KI-Schnittstelle) antworten mit **HTTP 402**, Lesen und Fahrer-Links funktionieren weiter, nichts wird gelöscht. Verlängern oder den Plan entfernen hebt die Sperre sofort auf.

Endpunkt	Zweck
`GET /api/admin/plans/catalog`	Pakete mit Ausgangswerten (Superadmin)
`GET /api/admin/plans/organizations`	Alle Organisationen mit Plan und Nutzung, Auffälliges zuerst (Superadmin)
`PUT /api/admin/plans/organizations/{id}`	Plan setzen; fehlt ein Grenzwert, gilt der Katalog, `null` heißt unbegrenzt (Superadmin)
`DELETE /api/admin/plans/organizations/{id}`	Plan entfernen (Superadmin)
`GET /api/org/plan`	Plan, Nutzung und Hinweise der eigenen Organisation (jedes Mitglied)

Setzen und Entfernen stehen im Audit-Log (`admin.org.plan_set`, `admin.org.plan_removed`).

Module je Organisation

Einige Bereiche schaltet der Superadmin je Organisation frei: **Tracker**, **Aktionsseiten**, **Branding** und **KI-Zugriff**. Das geht im Admin-Portal unter **Organisationen**, Spalte *Module* — ein Klick auf ein Modul schaltet es frei (✓) oder sperrt es (🔒). Mitglieder, Leitstellen, GPS-Freigaben und der Datenexport sind keine Module und stehen jeder Organisation zu.

- **Standard nach Betriebsart.** Auf einer selbst betriebenen Instanz (`INSTANCE_MODE=selfhost`) sind alle Module frei, auf dem Hosting-Server (`INSTANCE_MODE=hosting`) bekommt eine **neue** Organisation zunächst keines. Demo-Organisationen haben immer alle.
- **Ein Update nimmt nichts weg.** Jede Organisation, die es vor der Einführung schon gab, hat alle Module ausdrücklich freigeschaltet — auch auf dem Hosting-Server.
- **Im Org-Admin** bleibt ein gesperrtes Modul als ausgegrauter Reiter mit 🔒 sichtbar; wer ihn öffnet, liest, dass der Betreiber es freischalten kann.
- **Gesperrt heißt gesperrt, nicht versteckt.** Die Verwaltung antwortet mit HTTP 403. Tracker bekommen *schweigen* und senden nicht; die Konvoi-Ansicht zeigt keine Tracker. Aktionsseiten sind nicht mehr abrufbar (404) und zeichnen keinen Verlauf auf. Statt des eigenen Brandings erscheint das der Plattform. Die KI-Schnittstelle gilt für die Organisation als abgeschaltet, bestehende Verbindungen kommen nicht mehr weiter.
- **Gelöscht wird nichts.** Geräte, Seiten, Logos und die KI-Richtlinie bleiben gespeichert und gelten wieder, sobald das Modul freigeschaltet ist.

Endpunkt	Zweck
`GET /api/admin/modules/catalog`	Module, Standard und Betriebsart (Superadmin)
`GET /api/admin/modules/organizations`	Alle Organisationen mit ihren Modulen (Superadmin)
`PUT /api/admin/modules/organizations/{id}`	Module setzen, z. B. `{"tracker": true}`; nur die genannten ändern sich (Superadmin)
`GET /api/org/modules`	Module der eigenen Organisation (jedes Mitglied)

Jede Änderung steht im Audit-Log (`admin.org.modules_set`).

Lizenzmodell des Quellcodes

Der Quellcode selbst steht unter einem **Dual-Lizenz-Modell** (AGPL-3.0 bzw. kommerzielle Lizenz) — das betrifft die Nutzung/Weitergabe des Codes und ist unabhängig vom obigen Laufzeit-Lizenzschlüssel. Siehe `LICENSE`, `COMMERCIAL_LICENSE.md` und `CLA.md` im Repository.

KAPITEL 8

Sicherheit und Datenschutz

Diese Seite fasst die Sicherheits-Härtung, das Audit-Log, die DSGVO-Werkzeuge sowie Backup/Restore und Datenaufbewahrung zusammen.

Härtung im Überblick

Bereich	Umsetzung
Fail-Closed JWT	In Produktion (`APP_ENV=production`) startet das Backend nicht, wenn `JWT_SECRET` leer, Platzhalter oder < 32 Zeichen ist
Brute-Force-Schutz	Rate-Limiting auf Login, Passkey-Anmeldung, MFA-Verify und Passwort-Reset (HTTP 429)
Kontingent-Drosselung	Stundenbudget je Aufrufer auf Routing, Adresssuche und Verkehrslage (HTTP 429 mit `Retry-After`); Demo-Sitzungen mit kleinerem Budget und zusätzlich pro IP gezählt
Passwort-Policy	Mind. 10 Zeichen mit Buchstaben + Ziffern, Abgleich gegen Have-I-Been-Pwned (k-Anonymity, fail-open)
E-Mail-Normalisierung	Login-Adressen werden getrimmt und klein geschrieben gespeichert/verglichen — Login ist case-insensitive, doppelte Konten mit nur abweichender Groß-/Kleinschreibung sind ausgeschlossen (`lower(email)`-Unique-Index)
Sitzung im HttpOnly-Cookie	Das Zugriffstoken liegt nicht mehr im `localStorage`, sondern in einem `HttpOnly`-Cookie — JavaScript im Browser kommt nicht mehr an den Wert (siehe unten)
CSRF-Schutz	Cookie-Anmeldungen brauchen bei ändernden Methoden den Kopf `X-Requested-With: ConvoyPlan`; zusätzlich `SameSite=Lax`
JWT-Revocation	`token_version` entzieht alle Tokens bei Passwort-/MFA-Reset
MFA at-rest	TOTP-Secrets mit Fernet verschlüsselt gespeichert
Passkeys	Anmeldung per WebAuthn ohne Passwort, mit Pflicht zur Benutzerverifikation (siehe unten)
CORS-Lockdown	In Produktion auf die eigene App-Origin beschränkt
CSP & Security-Header	Content-Security-Policy (Report-Only/Enforce) plus HSTS, X-Content-Type-Options u. a. über Caddy
security.txt	Vulnerability-Disclosure-Kontakt unter `/.well-known/security.txt`
Dependency-Scanning	Dependabot + CI-Job (`pip-audit`, `npm audit`)
Update-Integrität	Alle Images sind mit Sigstore signiert; der Updater prüft die Signatur, bevor er zieht (siehe unten)

Sicherheitslücken bitte gemäß `SECURITY.md` bzw. `/.well-known/security.txt` melden.

Die Sitzung im Browser

Das Zugriffstoken lag früher im `localStorage` und wurde von der Oberfläche in jeden API-Aufruf geschrieben. Das ist bequem und hat einen Preis: **jedes** Stück JavaScript auf der Seite konnte es lesen. Ein einziges Cross-Site-Scripting — in einer Abhängigkeit, in einem eingebetteten Namen, in einer Vorschau — hätte genügt, und das Token wäre abgeflossen. Es ist sieben Tage gültig (`JWT_EXPIRE_MINUTES`) und von jedem Rechner der Welt einlösbar; gestohlen ist es eine Woche lang eine vollwertige Anmeldung.

Jetzt liegt es in einem **`HttpOnly`-Cookie**. Der Browser schickt es von sich aus mit, Skripte kommen nicht mehr an den Wert heran.

> **Was das nicht leistet.** Läuft fremder Code auf der Seite, kann er weiterhin im Namen des Angemeldeten Anfragen stellen — das Cookie geht ja automatisch mit. Was er nicht mehr kann, ist das Token *mitnehmen*. Der Unterschied ist erheblich: ein Angriff endet mit der Sitzung im Browser, statt eine Woche lang von einem fremden Rechner aus weiterzulaufen. Er ist aber kein Freibrief, und deshalb steht er hier statt im Kleingedruckten.

Eine Sitzung je Organisation. In ConvoyPlan meldet man sich pro Organisation getrennt an, und man kann in zwei Tabs in zwei Organisationen arbeiten. Entsprechend gibt es ein Cookie je Organisation (`cp_session__<slug>`) plus eines für die organisationslose Superadmin-Sitzung (`cp_session`). Welche gemeint ist, sagt die Oberfläche über den Kopf `X-Org-Slug`.

CSRF. Ein Cookie schickt der Browser auch dann mit, wenn eine fremde Seite die Anfrage auslöst — das Problem, das es mit einem `Authorization`-Header nicht gab. Dagegen zwei Dinge: `SameSite=Lax` hält das Cookie bei Unterseiten-Anfragen fremder Ursprünge zu Hause, und alle ändernden Methoden verlangen den festen Kopf `X-Requested-With: ConvoyPlan`. Einen eigenen Header kann fremdes JavaScript nur nach einem CORS-Preflight setzen, und den beantwortet die Instanz nur für den eigenen Ursprung; ein Formular-POST von außen kann ihn gar nicht setzen.

`Secure`-Flag. Ob das Cookie nur über HTTPS gesendet wird, leitet sich aus `APP_BASE_URL` ab und **nicht** aus dem Schema der eingehenden Anfrage. Hinter dem Reverse Proxy spricht das Backend unverschlüsselt; eine Ableitung aus der Anfrage ergäbe in Produktion immer „http" und das Cookie nie `Secure`.

Eine Quelle für die Anmeldung. Header *oder* Cookie wird an genau einer Stelle entschieden (`deps._credential`, bzw. `deps.credential_from_request` für Aufrufer ohne Dependency-Injection). Das ist keine Formalie: zweimal hat eine Stelle, die sich das Token selbst aus dem `Authorization`-Header zog, nach der Umstellung still aufgehört zu funktionieren — einmal die lesenden Endpunkte der Systemübersicht, einmal die Zählung der aktiven Nutzer. Ein Test durchsucht deshalb den Backend-Code und schlägt fehl, sobald wieder jemand am Header vorbeiliest.

Für API-Clients ändert sich nichts. Der Weg über `Authorization: Bearer` bleibt bestehen, inklusive `access_token` in der Login-Antwort. Er war nie das Problem — das Problem war, das Token dafür im Browser zu lagern. Skripte, API-Keys und der MCP-Server sind unberührt.

> Wer schon angemeldet war, wird beim ersten Laden nach dem Update einmal zur Anmeldung geschickt: die alte Ablage wird nicht mehr gelesen, sondern aufgeräumt.

Content-Security-Policy scharfschalten

Die CSP wird standardmäßig im **Report-Only**-Modus ausgeliefert (bricht die Karten-UI nicht, meldet nur Verstöße im Browser-Log). Zum Erzwingen:

1. `CSP_ENFORCE=true` in `.env` setzen.

2. Stack neu starten (`docker compose up -d caddy backend`).
3. Karte, Routing, Live-Tracking und Geocoding-Suche prüfen; bei Verstößen die betroffenen Quellen in der Policy ergänzen.

Die Policy erlaubt out-of-the-box dem eigenen Ursprung für Karten (Kachel-Proxy, siehe unten), `nominatim.openstreetmap.org` und `photon.komoot.io` (Geocoding), MapLibre-Worker (`blob:`) und Same-Origin-WebSockets.

Kartenkacheln über die eigene Instanz

Die Karte lädt ihre Kacheln nicht direkt beim Kachelserver, sondern über den Proxy der Instanz (`/api/tiles/{z}/{x}/{y}.png`). Der Kachelserver sieht damit nur die IP-Adresse der Instanz, nicht die der Nutzer — auch nicht die der Fahrer am Tracking-Link. Die CSP lässt keinen fremden Kachelserver mehr zu.

Einstellung	Standard	Wirkung
`TILE_UPSTREAM_URL`	`https://tile.openstreetmap.org/{z}/{x}/{y}.png`	Kachelserver hinter dem Proxy
`TILE_CACHE_MAX_MB`	`2048`	Obergrenze des Plattencaches (Volume `tile_cache`); älteste Kacheln gehen zuerst
`TILE_CACHE_MAX_AGE_DAYS`	`7`	Danach wird eine Kachel neu geholt; bei Ausfall des Kachelservers kommt die alte
`TILE_RATE_LIMIT_PER_MINUTE`	`1200`	Abrufe je Client-IP und Minute — der Endpunkt ist ohne Anmeldung erreichbar

Vorabladen entlang der Route. Damit die Karte im Funkloch nicht leer wird, lädt die Tracking-Ansicht den Korridor der Route vorab. Beim öffentlichen OSM-Server nur Zoom 12–14 und höchstens 1.500 Kacheln — dessen Nutzungsregeln sehen Massenabrufe nicht vor. Mit eigenem oder kommerziellem Kachelserver in `TILE_UPSTREAM_URL` das volle Profil (Zoom 12–16, bis 8.000 Kacheln). Das Profil liefert `GET /api/tiles/config`.

Der Cache ist nicht im Backup; er füllt sich beim Kartenaufruf von selbst wieder.

Passkeys

Statt mit E-Mail und Passwort kann man sich mit einem **Passkey** anmelden — Fingerabdruck, Gesichtserkennung oder Geräte-PIN auf dem Telefon, Rechner oder Sicherheitsschlüssel. Eingerichtet wird er im eigenen Konto: in der Planung unter **Konto → Passkeys**, im Adminportal unter **System → Passkeys**. Auf der Anmeldeseite steht dann **Mit Passkey anmelden**; eine E-Mail-Adresse muss man dafür nicht eingeben.

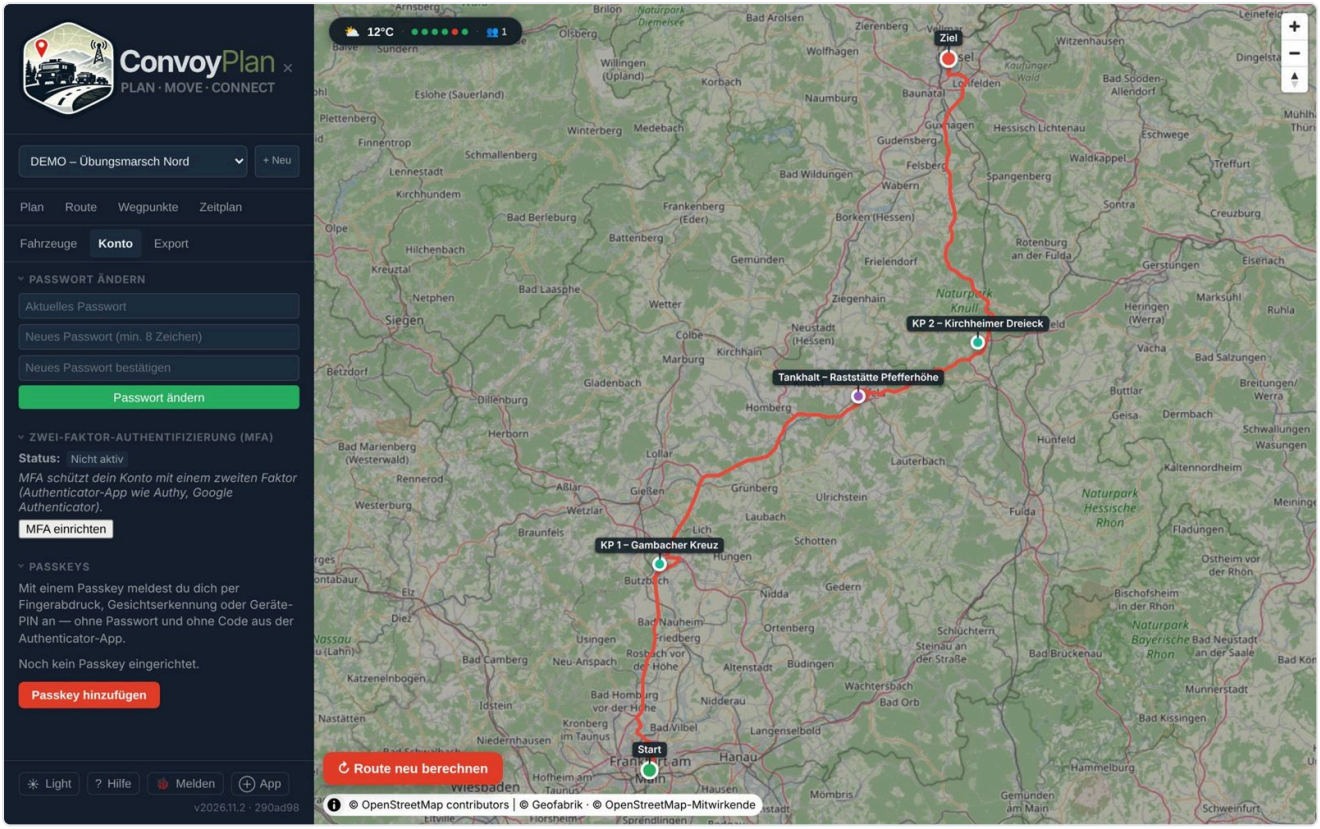


Abbildung: Reiter „Konto“: Passwort, Zwei-Faktor-Authentifizierung und Passkeys

Einfacher geht es über das **E-Mail-Feld**: wer hineinklickt, bekommt den Passkey im Vorschlagsmenü des Browsers angeboten, wie ein gespeichertes Passwort. Auswählen, Fingerabdruck oder PIN — angemeldet, ohne Klick auf *Anmelden*. Das Menü zeigen Chrome, Edge und Safari; wo es fehlt, bleibt der Knopf der Weg, ebenso für einen Passkey auf einem anderen Gerät (QR-Code am Telefon).

- **Ersetzt Passwort und Zweitfaktor.** Ein Passkey belegt Besitz (das Gerät) und Wissen bzw. Biometrie (PIN, Fingerabdruck) in einem Schritt. Die Instanz verlangt diese Benutzerverifikation; ein Gerät, das nur „anwesend" meldet, wird abgewiesen. Deshalb folgt auf eine Anmeldung per Passkey **kein** TOTP-Code, auch wenn MFA eingeschaltet ist.
- **Einrichten verlangt das Passwort.** Ein Passkey bleibt gültig, wenn das Passwort geändert oder zurückgesetzt wird. Wer eine offene Sitzung kurz in die Hand bekommt, soll sich darüber keinen eigenen dauerhaften Zugang anlegen können.
- **Gilt für die Domain der Instanz.** Maßgeblich ist `APP_BASE_URL`: ihr Hostname ist die Relying-Party-ID. Zieht die Instanz auf eine andere Domain um, passen bestehende Passkeys dort nicht mehr — sie müssen neu eingerichtet werden. Das ist eine Eigenschaft von WebAuthn, keine Einstellung.
- **Dieselben Regeln wie beim Passwort.** Anmelden kann sich nur ein aktives Konto, in einer Organisation nur ein Mitglied, ohne Organisation nur ein Superadmin. Wer ein Konto deaktiviert, sperrt damit auch dessen Passkeys.
- **Gespeichert wird nur der öffentliche Schlüssel.** Der private verlässt das Gerät nie; ein Abzug der Datenbank ist damit, anders als einer der Passwort-Hashes, für niemanden eine Anmeldung.
- **Verloren oder gestohlen?** Den Passkey im Konto unter **Entfernen** löschen — vom Gerät verschwindet er dadurch nicht, anmelden kann man sich damit aber nicht mehr. Ist kein anderer Zugang mehr da, setzt ein Superadmin das Passwort zurück; danach lässt sich der Passkey selbst entfernen.

Anlegen und Entfernen landen im Audit-Log (`auth.passkey.added`, `auth.passkey.removed`), Anmeldungen als `auth.login.success` mit `"passkey": true`. Die Auskunft nach Art. 15 nennt Name und Zeitpunkte der Passkeys, nicht aber Kennung und Schlüssel.

Anmeldung der Begleit-App

Die ConvoyPlan-App meldet sich — wo die Instanz es anbietet — **über den Browser des Telefons** an statt mit einer eigenen Passwortmaske: Die App öffnet die Anmeldeseite der Organisation, man meldet sich dort an wie im Web (Passwort, MFA oder Passkey), und der Browser gibt die App wieder frei. Technisch ist das OAuth 2.1 mit PKCE nach RFC 8252; der Passkey bleibt bei der Webseite, die App bekommt nur Zugangstokens.

- **Keine Freigabeliste wie bei der KI-Schnittstelle.** Die App darf, was die Rolle in der Organisation hergibt — dieselbe Prüfung wie im Web, bei jedem Aufruf neu. Gilt unabhängig davon, ob MCP eingeschaltet ist.
- **Eine Rückfrage bei bestehender Sitzung.** Wer sich in diesem Ablauf frisch angemeldet hat, ist sofort zurück in der App. Ist der Browser noch von früher angemeldet, fragt die Seite einmal nach („In der ConvoyPlan-App als … anmelden?"). Nur bestätigen, wenn man die Anmeldung gerade selbst in der App gestartet hat.
- **Nur die eigene Organisation.** Wer nicht Mitglied der Organisation ist, für die die App fragt, wird abgewiesen; eine andere wird nicht angeboten.
- **Laufzeiten.** Der Zugang der App gilt 15 Minuten und wird im Hintergrund erneuert. Wird die App **90 Tage** nicht benutzt, muss man sich neu anmelden (`APP_OAUTH_REFRESH_TOKEN_TTL_DAYS`).
- **Beenden.** Passwort ändern oder zurücksetzen, das Konto deaktivieren oder es aus der Organisation entfernen beendet auch die App-Anmeldung. Ganz abschalten lässt sich der Weg mit `APP_OAUTH_ENABLED=false`; die App bleibt dann bei ihrer Passwortmaske.

Ob eine Instanz den Weg anbietet, liest die App aus `/.well-known/oauth-authorization-server` (Feld `convoyplan_app_client_id`). Anmeldungen stehen im Audit-Log als `auth.app.granted` bzw. `auth.app.denied`.

Audit-Log

Ein **append-only** Protokoll erfasst sicherheitsrelevante Ereignisse (Logins, MFA, Passkeys, Passwortänderungen, Benutzer-/Org-Anlage, Lizenzaktivierung) inklusive Akteur, Ziel, IP und User-Agent. Superadmins rufen es über `GET /api/admin/audit-log` (filterbar nach Aktion) ab.

DSGVO-Werkzeuge

Recht	Endpunkt	Wirkung
Auskunft (Art. 15)	`GET /api/admin/users/{id}/export`	Liefert alle personenbezogenen Daten als JSON
Löschung (Art. 17)	`DELETE /api/admin/users/{id}/data`	Löscht den Benutzer und pseudonymisiert den Audit-Trail
Rückgabe / Umzug einer Organisation	`GET /api/organizations/{id}/export` (Org-Admin), `GET /api/admin/organizations/{id}/export` (Superadmin)	ZIP mit allen Daten der Organisation, siehe unten
Löschung einer Organisation	`DELETE /api/admin/organizations/{id}`	Löscht die Organisation samt Konvois, Wegpunkten, Routen, Positionen und Tracking-Links

Organisations-Export

Gedacht für die Rückgabe der Daten bei Vertragsende (Art. 28 Abs. 3 lit. g DSGVO) und für einen Umzug, etwa vom Hosting auf die eigene Installation. Admins der Organisation finden ihn unter **Organisation → Admin → Datenexport**, der Superadmin mit ⬇ in der Organisationsliste.

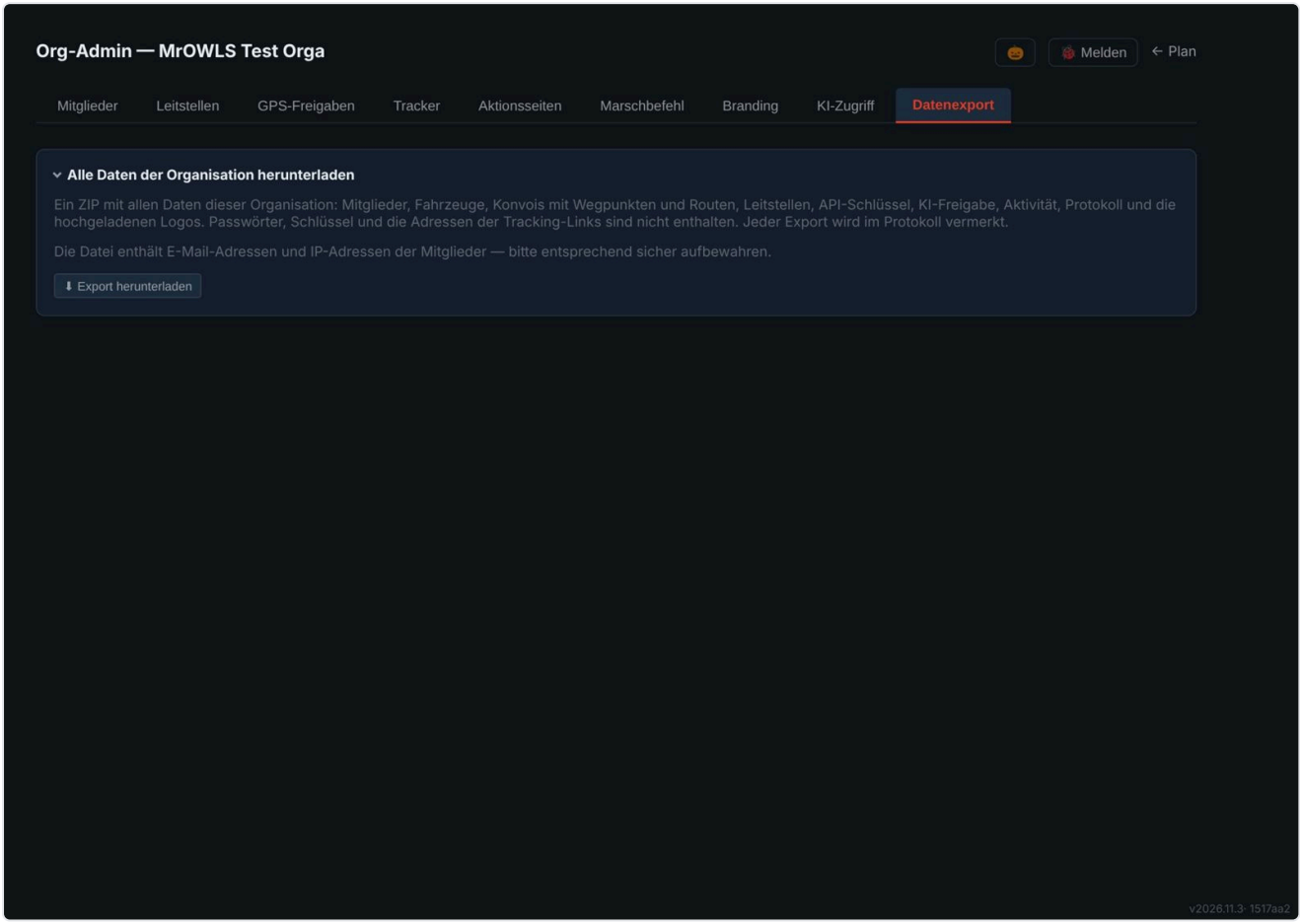


Abbildung: Org-Admin, Reiter „Datenexport“: alle Daten der Organisation als ZIP

- **Inhalt:** ein ZIP mit `organisation.json`, den hochgeladenen Logos unter `logos/` und den Fotos der Aktionsseiten unter `beitraege/` (WebP, so wie sie hinausgehen). Das JSON enthält Organisation, Mitgliedschaften, Mitglieder, Fahrzeuge, Konvois mit Fahrzeugzuordnungen, Wegpunkten, Routen (GeoJSON), letzten Positionen und Tracking-Links, Leitstellen, API-Schlüssel, KI-Freigabe, Aktivitätstage, Aktionsseiten samt Beiträgen (`beitraege`, `beitrag_bilder`) und das Audit-Log der Organisation.
- **Jede Spalte:** Die Felder werden aus dem Datenmodell gelesen, nicht von Hand aufgezählt. Ein Test prüft außerdem, dass jede Tabelle mit Bezug zu einer Organisation entweder im Export steht oder bewusst ausgenommen ist.
- **Nicht enthalten:** Passwort- und Schlüssel-Hashes, MFA-Geheimnisse, die Adressen (Slugs) der Tracking-Links, OAuth-Tokens sowie Rückmeldungen an den Betreiber und Demo-Kontaktdaten. Von den Beiträgen fehlen Absender und alle Telegram-Kennungen (Gruppe, Nachrichten, Fahrer, Presse) — sie gehören Dritten; ebenso Bot-Token, Codes, die Liste der Presse und die verbundenen Gruppen.
- **Nur mit Anmeldung:** Ein API-Key bekommt keinen Export, auch nicht mit Admin-Rolle. Jeder Export steht als `org.exported` im Audit-Log.

KI-Schnittstelle (MCP)

Standardmäßig **abgeschaltet**. Ohne `MCP_ENABLED=true` existiert weder `/mcp` noch ein Discovery-Dokument.

Eingeschaltet gilt: Zugriff entsteht erst durch die ausdrückliche Zustimmung eines angemeldeten Benutzers (inklusive MFA), gilt für genau eine Organisation und ist durch dessen Rolle gedeckelt. Jeder schreibende Aufruf landet im Audit-Log mit Quelle `mcp` und dem auslösenden Programm. Gelöscht werden kann über die Schnittstelle nichts.

Was ein Modell liest, verlässt dabei die Instanz und geht an den Anbieter des KI-Programms — das ist der Zweck der Schnittstelle. Die datenschutzrechtliche Einordnung, das Zeitfenster beim Widerruf und die Einstellungen stehen in MCP-Server.

Datenaufbewahrung (Retention)

Der `retention`-Container purgt periodisch abgelaufene Daten:

Variable	Standard	Wirkung
`RETENTION_ENABLED`	`true`	Retention-Läufe aktiv
`RETENTION_INTERVAL`	`3600`	Sekunden zwischen den Läufen
`RETENTION_POSITIONS_HOURS`	`24`	Live-Positionen älter als … löschen
`RETENTION_AUDIT_DAYS`	`365`	Audit-Log-Einträge älter als … löschen
`RETENTION_SHARE_LINKS_DAYS`	`30`	Widerrufene Share-Links älter als … löschen
`RETENTION_POSITION_TRAIL_DAYS`	`45`	Positionsverlauf der Aktionsseiten älter als … löschen
`RETENTION_DEMO_LEADS_DAYS`	`180`	Kontaktangaben aus dem Demo-Start älter als … löschen
`RETENTION_OAUTH_TOKENS_GRACE_DAYS`	`30`	Rotierte und widerrufene MCP-Refresh-Tokens älter als … löschen
`RETENTION_OAUTH_CLIENTS_DAYS`	`7`	Verwaiste MCP-Registrierungen älter als … löschen

Im selben Durchgang laufen zwei Aufgaben ohne eigene Variable: Abgelaufene Demo-Sitzungen samt ihrer Daten werden gelöscht (Frist im Admin-Portal), und für abgelaufene Sitzungen geht die einmalige Nachfrage-Mail raus (siehe Lizenz und Demo-Modus). Die Kontaktangabe hält bewusst länger als die Sitzung — daran hängen Nachfrage und ein etwaiges Vertriebsgespräch —, aber nicht dauerhaft; einzelne Kontakte lassen sich im Admin-Bereich sofort löschen.

Backup & Restore

Was gesichert werden muss

Komponente	Volume	Inhalt
PostgreSQL	`…_postgres_data`	Benutzer, Orgs, Konvois, Fahrzeuge, Audit-Log, Einstellungen
Uploads	`…_logo_uploads`	Hochgeladene Branding-Logos
TLS / Caddyfile	`…_cert_uploads`	Persistierte Caddyfile + ggf. eigene Zertifikate

> Nicht sicherungsbedürftig: `osm_data`, `gh_graph` (wird neu gebaut), `caddy_data` (ACME-Zertifikate werden neu ausgestellt).

Backup

```
scripts/backup.sh
```

Erzeugt unter `./backups/<YYYYMMDD-HHMMSS>/` einen komprimierten `pg_dump`, die Volumes als `.tar.gz`, eine Kopie der Compose-Datei und `SHA256SUMS`. Konfiguration über `BACKUP_DIR` und `BACKUP_RETENTION_DAYS`.

Cron (täglich 03:00):

```
0 3 * * * /opt/convoyplan/scripts/backup.sh >> /var/log/convoyplan-backup.log 2>&1
```

> **Off-site:** Backups gehören zusätzlich verschlüsselt an einen zweiten Ort (`rsync` / `rclone`).

Restore

```
scripts/restore.sh ./backups/20260603-030000
docker compose restart backend caddy
```

Das Skript ist **destruktiv** und verlangt zur Bestätigung die Eingabe `RESTORE`. Restore regelmäßig (z. B. vierteljährlich) auf einer separaten Instanz testen.

Verschlüsselung at-rest

ConvoyPlan verschlüsselt Anwendungsgeheimnisse selektiv (MFA-TOTP-Secrets via Fernet). Die **vollständige** Verschlüsselung im Ruhezustand erfolgt auf Infrastruktur-Ebene:

- **LUKS (Linux):** `/var/lib/docker` bzw. die darunterliegende Partition auf einem LUKS-Container ablegen — schützt alle Volumes transparent.
- **Cloud:** verschlüsselte Block-Volumes des Hosters nutzen.
- **Backups:** Zielverzeichnis auf verschlüsseltem Volume ablegen oder Archive vor Off-site-Transfer verschlüsseln (`age` / `gpg`).

Schlüsselverwaltung: `.env` enthält Klartext-Geheimnisse → Dateirechte `chmod 600`, LUKS-Passphrase nicht neben den Daten ablegen. `MFA_ENCRYPTION_KEY` wird aus `JWT_SECRET` abgeleitet, falls nicht gesetzt — bei Rotation von `JWT_SECRET` ohne eigenen Schlüssel werden bestehende MFA-Secrets unlesbar.

Update-Integrität

Der Updater-Container ist die mächtigste Komponente der Installation: Er hält den **Docker-Socket mit Schreibrecht** — anders lässt sich ein Container nicht austauschen. Was er zieht und startet, läuft damit als root auf dem Host.

Bis zur Signaturprüfung entschied darüber allein ein **Tag** auf ghcr.io. Ein Tag sagt nur: irgendjemand mit Schreibrecht auf die Registry hat ihn dorthin gesetzt. Ein kompromittierter GitHub-Account oder ein gestohlenes Write-Token hätte damit gereicht, um auf jeder Installation mit Auto-Update root zu bekommen.

Wie es abgesichert ist

1. **Signiert in der CI.** `release.yml` und `nightly-images.yml` signieren jedes der sechs Images (`backend`, `frontend`, `graphhopper`, `updater`, `osmium`, `medien`) **keyless** mit Sigstore/cosign. Die Identität ist der Workflow selbst — GitHub stellt ein OIDC-Token aus, Fulcio tauscht es gegen ein kurzlebiges Zertifikat. **Es gibt**

keinen privaten Schlüssel, weder im Repository noch in einem Secret noch auf einem Rechner. Signiert wird der Digest, nicht der Tag.

2. **Geprüft vor dem Pull.** Der Updater ruft für jedes Ziel-Image `cosign verify` auf, bevor `docker compose pull` überhaupt startet. Akzeptiert wird nur eine Signatur, deren Zertifikat auf einen Workflow dieses Repositories und auf GitHubs OIDC-Aussteller zeigt.
3. **Fehlschlag heißt Stillstand, nicht Warnung.** Lässt sich auch nur ein Image nicht verifizieren, wird **nichts** gezogen und **nichts** neu gestartet. Die Installation läuft unverändert auf dem alten Stand weiter, und die Superadmins bekommen eine E-Mail („Update abgebrochen: Image-Signatur nicht verifizierbar"). Dasselbe gilt, wenn `cosign` im Updater-Image fehlt — es gibt keinen stillen Rückfall auf ungeprüftes Ziehen.

Selbst nachprüfen

```
cosign verify \
  --certificate-identity-regexp '^https://github.com/RettTechSolutions/ConvoyPlan/' \
  --certificate-oidc-issuer https://token.actions.githubusercontent.com \
  ghcr.io/retttechsolutions/convoyplan/backend:latest
```

Air-Gapped-Installationen

`cosign verify` braucht Zugang zu **Rekor** und **Fulcio** (sigstore.dev). Ohne Internet kann die Signatur nicht geprüft werden. Dafür gibt es den Schalter:

```
UPDATE_VERIFY_SIGNATURES=false
```

Das ist eine bewusste Entscheidung des Betreibers, kein automatischer Fallback: Der Updater protokolliert bei jedem Lauf, dass er ungeprüft zieht. In einer abgeschotteten Umgebung ist das vertretbar, weil die Images dort ohnehin nicht von ghcr.io kommen — überall sonst gehört der Schalter auf `true`.

Ein Fork mit eigener Registry setzt stattdessen `UPDATE_SIGNATURE_IDENTITY=^https://github.com/<org>/<repo>/`.

Was das **nicht** abdeckt

Die Signatur beweist, dass ein Workflow *dieses Repositories* das Image gebaut hat — nicht, dass der Inhalt gutartig ist. Wer Code nach `main` bringen kann, bekommt weiterhin ein gültig signiertes Image. Die Signatur schließt den Weg „Image direkt in die Registry schieben", nicht den Weg „bösartigen Code mergen". Dagegen helfen Branch-Protection, Review-Pflicht für `.github/workflows/` und `docker/updater/` sowie 2FA mit Hardware-Key auf der Organisation.

TEIL 3

Schnittstellen

API-Dokumentation

MCP-Server (KI-Schnittstelle)

ConvoyPlan in ChatGPT verbinden

Auskunft für Suchmaschinen und KI-Agenten

KAPITEL 9

API-Dokumentation

Die vollständige OpenAPI-Dokumentation wird automatisch von FastAPI bereitgestellt:

- **Swagger UI:** `/docs`
- **ReDoc:** `/redoc`
- **OpenAPI JSON:** `/openapi.json`

Lokal (Dev) sind sie unter `http://localhost:8000/docs` erreichbar.

> **Produktion:** `/docs`, `/redoc` und `/openapi.json` sind standardmäßig **deaktiviert (404)**. Mit `DOCS_API_KEY=<geheim>` lassen sie sich per API-Key absichern; mit `ENABLE_DOCS=true` werden sie offen freigegeben (nur für Dev/intern).
>
> **Anmeldung im Browser:** `/docs/login` aufrufen, Key eingeben, absenden. Das Backend setzt ein HttpOnly-Cookie (8 Stunden), danach laden `/docs`, `/redoc` und `/openapi.json` nahtlos. Wer `/docs` ohne gültiges Cookie aufruft, bekommt `401` mit demselben Formular im Rumpf.
>
> **Programmatisch:** Header `X-API-Key: <geheim>`.
>
> Ein `?key=…` im Query-String wird **nicht** akzeptiert (401). Query-Parameter landen im Access-Log des Reverse Proxy, in der Browser-History und ggf. im `Referer` — für ein Geheimnis der falsche Transportweg.

Zugangswege im Überblick

ConvoyPlan kennt **vier voneinander unabhängige Credentials**. Welches du brauchst, hängt vom Endpunkt ab — die Tabelle nennt jeweils den kürzesten Weg.

Zugangsweg	Wie mitgeben	Gilt für	Woher
Bearer-Token (JWT)	`Authorization: Bearer <token>`	alle geschützten Endpunkte	`POST /api/auth/login`
Org-API-Key	`X-API-Key: cvp_…`	organisationsbezogene Endpunkte (siehe Matrix)	Superadmin-Portal → Reiter **API-Keys**
System-API-Key	`X-API-Key: cvp_…`	nur lesende Endpunkte der Systemübersicht	Superadmin-Portal → **API-Keys** → Geltungsbereich *System*
Freigabe-Token	Teil der URL	`/api/convoys/share/{token}`, `/api/track/{slug}`	Freigabelink je Konvoi
Docs-Key	`X-API-Key: <DOCS_API_KEY>` oder Anmeldung über `/docs/login`	nur `/docs`, `/redoc`, `/openapi.json`	Umgebungsvariable `DOCS_API_KEY`
MCP-Zugang (OAuth 2.1)	`Authorization: Bearer <token>`	ausschließlich `/mcp`	Zustimmung eines Benutzers, siehe MCP-Server

> ⚠️ **MCP-Zugang ≠ Bearer-Token.** Beide reisen als `Authorization: Bearer …`, sind aber strikt getrennt: Ein MCP-Token trägt einen eigenen Typ und wird an jedem REST-Endpunkt abgewiesen — und umgekehrt öffnet ein Login-Token `/mcp` nicht. Die Schnittstelle ist standardmäßig abgeschaltet; siehe MCP-Server.

> ⚠️ **Docs-Key ≠ Org-API-Key.** Beide reisen im Header `X-API-Key`, haben aber nichts miteinander zu tun: Der Docs-Key ist eine Instanz-Umgebungsvariable und öffnet ausschließlich die Swagger-/ReDoc-Oberfläche. Ein Org-API-Key funktioniert dort nicht — und umgekehrt öffnet der Docs-Key keinen einzigen Datenendpunkt.

Ohne Credential erreichbar sind nur: `GET /health`, `GET /api/version`, `GET /api/status`, `GET /api/status/public`, `GET /api/setup/status`, `GET /api/license/mode`, `GET /api/branding/org/{slug}` sowie die Freigabe-/Track-Routen.

Authentifizierung mit Bearer-Token

Token erhalten

```
POST /api/auth/login
Content-Type: application/json

{
  "email": "benutzer@example.com",
  "password": "passwort",
  "org_slug": "jpbg"
}
```

`org_slug` steuert den Geltungsbereich des Tokens:

- **mit `org_slug`** → Token gilt *innerhalb dieser Organisation*, mit der Rolle aus der Mitgliedschaft. Das ist der Normalfall für Kolonnen, Fahrzeuge und Tracking.
- **ohne `org_slug`** → Superadmin-Token ohne Org-Bindung. Nur damit erreichst du `/api/admin/**` inklusive der Systemübersicht.

Antwort:

```
{
  "access_token": "eyJ...",
  "token_type": "bearer"
}
```

Der Token läuft nach `JWT_EXPIRE_MINUTES` ab (Standard: **7 Tage**). Er wird ungültig, sobald das Passwort geändert oder das Konto zurückgesetzt wird (Token-Versionierung).

Mit aktivierter MFA

Ist für das Konto TOTP aktiv, liefert `/api/auth/login` **kein** `access_token`, sondern eine Zwischenstufe:

```
{ "mfa_required": true, "mfa_token": "eyJ..." }
```

Diese gegen `POST /api/auth/mfa/verify` mit `{"mfa_token": "…", "code": "123456"}` eintauschen — erst diese Antwort enthält das `access_token`. Der `mfa_token` allein öffnet keinen einzigen Endpunkt.

```
TOKEN=$(curl -s -X POST https://web.convoyplan.de/api/auth/login \
  -H 'Content-Type: application/json' \
  -d '{"email":"admin@example.com","password":"…"}' | jq -r .access_token)

curl -H "Authorization: Bearer $TOKEN" https://web.convoyplan.de/api/admin/system/overview
```

Authentifizierung mit Org-API-Key

API-Keys sind für **Fremdsysteme** gedacht, die dauerhaft und ohne Benutzerkonto auf die Daten *einer* Organisation zugreifen — Leitstellen-Anbindung, Dashboard, Skript.

Anlegen im Superadmin-Portal unter **Admin → API-Keys**: Organisation wählen, Name, Rolle und optionales Ablaufdatum vergeben. Der Klartext wird **genau einmal** angezeigt (gespeichert wird nur ein bcrypt-Hash) — geht er verloren, muss ein neuer Key her.

Format: `cvp_<prefix>_<secret>` — der `prefix` (8 Hex-Zeichen) ist der öffentliche Nachschlage-Teil und taucht in Portal und Audit-Log auf, das `secret` nie.

```
curl -H "X-API-Key: cvp_a1b2c3d4_xxxxxxxx" https://web.convoyplan.de/api/convoys/
```

Eigenschaften:

- Der Key handelt **im Namen des Org-Eigentümers**, aber **immer mit seiner konfigurierten Rolle** — auch auf Objekten, die dem Eigentümer selbst gehören. Ein `beobachter`-Key kann also nichts ändern, egal wem die Kolonne gehört.
- Wird gleichzeitig ein Bearer-Token gesendet, **gewinnt der API-Key**.
- `last_used_at` wird höchstens alle 5 Minuten fortgeschrieben (ein Schreibzugriff pro Request wäre zu teuer) — der Wert ist also grob, nicht sekundengenau.
- Widerrufen wirkt sofort; abgelaufene und widerrufene Keys liefern `401`.

Welche Endpunkte akzeptieren einen Org-API-Key?

Bereich	`X-API-Key`	Anmerkung
`/api/convoys/` — Kolonnen, Fahrzeugzuordnung, Wegpunkte, Teilverbände	✅	Rolle des Keys wird durchgesetzt
`/api/vehicles/**`	✅	ab `planer` schreibend
`/api/org/leitstellen/**`	✅	
`/api/org/branding`	✅	
Routing, Export/Import, Tankstellen (`/calculate-route`, `/export/*`, …)	❌	nur Bearer-Token
Tracking und Positionen (`/positions`, `/status`)	❌	nur Bearer-Token
Freigabelinks (`/share-links`)	❌	nur Bearer-Token
`/api/organizations/**`, `/api/users/**`	❌	nur Bearer-Token
`/api/admin/**` inkl. Systemübersicht	❌	nur Superadmin-Token

Ein Org-API-Key auf einem Endpunkt, der keinen akzeptiert, führt zu `401` — nicht zu einem stillen Fallback auf anonymen Zugriff.

Authentifizierung mit System-API-Key

Ein **System-API-Key** gehört keiner Organisation. Er öffnet ausschließlich die **lesenden** Endpunkte der Systemübersicht (`GET /api/admin/system/…`) und sonst nichts — gedacht für Monitoring, Dashboards oder ein Skript, das die Kennzahlen dauerhaft abholen soll, ohne dass ein Benutzer-Token nach sieben Tagen

abläuft.

Anlegen im Superadmin-Portal unter **Admin → API-Keys**, im Auswahlfeld **Geltungsbereich** den Eintrag *System (instanzweit, nur lesen)* wählen. Format, einmalige Klartext-Anzeige, Ablauf und Widerruf verhalten sich wie bei Org-Keys; eine Rolle gibt es nicht, weil es nichts zu schreiben gibt.

```
curl -H "X-API-Key: cvp_a1b2c3d4_xxxxxxxx" \
  https://web.convoyplan.de/api/admin/system/overview
```

Die Trennung gilt in beide Richtungen und ist bewusst hart:

- Ein **Org-Key** auf `/api/admin/system/…` → `403`. Die Kennzahlen umfassen die ganze Instanz, also alle Mandanten — ein Key, der einer Organisation gehört, darf sie nicht sehen.
- Ein **System-Key** auf Kolonnen, Fahrzeuge oder anderen Organisationsdaten → `403`.
- `POST /api/admin/system/sample` verändert den Zustand und bleibt Superadmins vorbehalten — ein System-Key erhält dort `401`.
- Alle übrigen `/api/admin/**`-Endpunkte bleiben ebenfalls Superadmins vorbehalten.

Verwaltet werden die Keys über diese Endpunkte (Superadmin-Token erforderlich):

Methode	Endpunkt	Beschreibung
`GET`	`/api/admin/system-api-keys`	System-Keys auflisten
`POST`	`/api/admin/system-api-keys`	System-Key anlegen (`name`, optional `expires_at`)
`DELETE`	`/api/admin/system-api-keys/{key_id}`	System-Key widerrufen

Rollen im API-Kontext

Dieselbe Hierarchie gilt für Benutzer-Mitgliedschaften und für API-Keys:

Rolle	Lesen	Position/Status melden	Anlegen & Ändern	Löschen
`beobachter`	✅	❌	❌	❌
`fahrer`	✅	✅	❌	❌
`planer`	✅	✅	✅	❌
`admin`	✅	✅	✅	✅

Reicht die Rolle nicht, antwortet die API mit `403` und dem Text `Insufficient role: requires <rolle>`. Ausführlich: **Rollen & Berechtigungen**.

Statuscodes und Fehler

Code	Bedeutung	Typische Ursache
`401`	nicht authentifiziert	Token fehlt/abgelaufen, Key ungültig, widerrufen oder abgelaufen; auch: Credential am falschen Endpunkt
`402`	Demo-Modus	schreibender Zugriff ohne gültige Lizenz (siehe unten)
`403`	authentifiziert, aber nicht berechtigt	Rolle zu niedrig, keine Org-Mitgliedschaft, Superadmin nötig, Konto deaktiviert

Code	Bedeutung	Typische Ursache
`404`	nicht gefunden	Objekt existiert nicht — oder gehört einer anderen Organisation
`422`	Validierungsfehler	fehlende/ungültige Felder (FastAPI-Standardformat)
`429`	Drosselung	Rate-Limit oder Stundenkontingent erschöpft — `Retry-After` beachten

Fehlerantworten haben durchgängig die Form `{"detail": "…"}`; im Demo-Fall zusätzlich `"demo_mode": true`.

Drosselung im Detail

Rate-Limits (pro IP, gleitendes Fenster) schützen die Anmeldung: Login und alle MFA-Endpunkte 10 Versuche / 5 Minuten, Passwort-Reset 5 / 15 Minuten, Tracking-Auth 5 / 5 Minuten, Demo-Sitzungen 10 / Stunde.

Stundenkontingente (pro Benutzer) begrenzen Endpunkte, die fremde Dienste oder CPU kosten — Standardwerte, über `QUOTA_*` konfigurierbar:

Bereich	Regulär	Demo-Sitzung
Routenberechnung	240/h	40/h
Adresssuche (Geocoding)	600/h	100/h
Live-Verkehrslage	600/h	100/h

Beide Zähler liegen **im Prozessspeicher**: Sie gelten je Worker und werden bei einem Neustart zurückgesetzt. Als erste Verteidigungslinie gedacht, nicht als harte Abrechnung.

Demo-Modus (HTTP 402)

Ohne gültigen Lizenzschlüssel läuft die Instanz im Demo-Modus: **GET immer erlaubt**, `POST` / `PUT` / `PATCH` / `DELETE` auf geschützten Pfaden mit `402` abgewiesen. Ausgenommen sind `/health`, `/api/auth/login`, `/api/license/*`, `/api/setup` und `/api/track/*`. Lesende Integrationen funktionieren also auch unlizenziert.

Endpunkte

Methode	Endpunkt	Beschreibung	Auth
`POST`	`/api/auth/register`	Account erstellen	Nein
`POST`	`/api/auth/login`	Login, JWT erhalten (optional `org_slug`)	Nein
`GET`	`/api/auth/org-lookup`	Organisation zu einem Org-Code auflösen (Login-Maske)	Nein
`POST`	`/api/auth/password`	Eigenes Passwort ändern (liefert frisches Token)	Ja
`POST`	`/api/auth/password-reset`	Passwort-Reset anstoßen	Nein
`POST`	`/api/auth/stream-ticket`	Kurzlebiges Ticket für SSE/WebSocket lösen	Ja

> SSE- und WebSocket-Verbindungen können keinen `Authorization`-Header setzen. Statt den Zugangstoken in die URL zu schreiben, holt man sich über `/api/auth/stream-ticket` ein kurzlebiges Ticket und hängt dieses als Query-Parameter an.

MFA / TOTP

Methode	Endpunkt	Beschreibung
`GET`	`/api/auth/mfa/status`	Ist MFA für das eigene Konto aktiv?
`POST`	`/api/auth/mfa/setup`	TOTP-Einrichtung starten (QR-Code generieren)
`POST`	`/api/auth/mfa/confirm`	TOTP-Einrichtung bestätigen und aktivieren
`POST`	`/api/auth/mfa/verify`	`mfa_token` + TOTP-Code gegen vollwertiges JWT tauschen
`POST`	`/api/auth/mfa/disable`	MFA deaktivieren

Demo-Sitzungen (öffentlich)

Methode	Endpunkt	Beschreibung
`GET`	`/api/auth/demo-status`	Ist der öffentliche Demo-Modus aktiviert?
`GET`	`/api/auth/demo-session/info`	Rahmendaten einer Demo-Sitzung (Laufzeit)
`POST`	`/api/auth/demo-session`	Befristete Demo-Organisation samt Token erzeugen — oder die noch laufende Sitzung dieser IP fortsetzen (`resumed: true`)
`POST`	`/api/auth/demo-followup/unsubscribe`	Nachfrage-Mails für eine Adresse abbestellen (Token aus dem Abmeldelink)

Kontaktangabe: `POST /api/auth/demo-session` erwartet einen Körper mit `email` (Pflicht, wird normalisiert) sowie optional `first_name` und `last_name` ; ohne gültige Adresse antwortet der Endpunkt mit `422` . Die Angabe landet in `demo_leads` und **nicht** als Kennung des Demo-Benutzers — dort steht weiterhin die interne `demo-<hex>@demo.local` . Beim Fortsetzen einer Sitzung ersetzt die neue Angabe die alte.

```
{"email": "name@organisation.de", "first_name": "Max", "last_name": "Mustermann"}
```

Ersteinrichtung (Setup-Wizard)

Methode	Endpunkt	Beschreibung	Auth
`GET`	`/api/setup/status`	Prüft ob Setup noch erforderlich ist	Nein
`POST`	`/api/setup`	Superadmin anlegen, Domain und TLS konfigurieren	Nein

Superadmin-Verwaltung

Nur für Benutzer mit Superadmin-Rolle zugänglich.

Methode	Endpunkt	Beschreibung
`GET/POST`	`/api/admin/users`	Benutzer auflisten oder anlegen (`password` optional — ohne Angabe wird eines generiert; optionale `org_id` / `org_role` legen die Org-Mitgliedschaft in derselben Transaktion mit an)

Methode	Endpunkt	Beschreibung
`PATCH/DELETE`	`/api/admin/users/{user_id}`	Benutzer aktivieren/deaktivieren, Rolle setzen, löschen
`GET/POST`	`/api/admin/organizations`	Organisationen auflisten oder anlegen
`POST/DELETE`	`/api/admin/users/{user_id}/orgs`	Benutzer einer Organisation zuweisen oder entfernen
`GET/POST/DELETE`	`/api/admin/organizations/{org_id}/api-keys`	Org-gebundene API-Schlüssel verwalten
`GET/POST/DELETE`	`/api/admin/system-api-keys`	Instanzweite System-API-Schlüssel verwalten (nur lesend, siehe oben)
`GET`	`/api/branding`	Aktuelles (globales) Branding abrufen
`PUT`	`/api/branding`	Globales Branding (Farben, App-Name) aktualisieren
`POST`	`/api/branding/logo/{slot}`	Globales Logo hochladen (`slot`: `main` oder `horizontal`)
`POST`	`/api/admin/trigger-update`	Auto-Update manuell anstoßen
`GET`	`/api/admin/update-status` / `/api/admin/update-log`	Deploy-Stand bzw. Live-Update-Log (SSE)
`GET/PUT`	`/api/admin/settings/update-channel`	Update-Kanal (Stable/Beta/Nightly) lesen/setzen
`GET/PUT`	`/api/admin/settings/update-mode`	Update-Modus (Automatisch/Benachrichtigen) lesen/setzen
`GET/PUT`	`/api/admin/settings/smtp`	SMTP-Konfiguration lesen/setzen
`GET/PUT`	`/api/admin/settings/traffic-keys`	HERE-/TomTom-API-Keys lesen (ohne Klartext) und setzen

E-Mail-Vorlagen

Betreff und HTML der verschickten Mails sind vom Superadmin anpassbar und liegen in `system_settings`. `{kind}` ist derzeit `password` (Zugangsdaten) oder `demo_followup` (Nachfrage nach der Demo).

Methode	Endpunkt	Beschreibung
`GET`	`/api/admin/email-templates`	Verzeichnis: welche Vorlagen es gibt und welche angepasst wurden
`GET`	`/api/admin/email-templates/{kind}`	Betreff, HTML, `is_custom` und die erlaubten Platzhalter
`PUT`	`/api/admin/email-templates/{kind}`	Eigene Fassung hinterlegen
`POST`	`/api/admin/email-templates/{kind}/reset`	Eigene Fassung verwerfen — es gilt wieder die mitgelieferte
`GET`	`/api/admin/email-templates/{kind}/preview`	Musterexemplar als HTML

Methode	Endpunkt	Beschreibung
`POST`	`/api/admin/email-templates/{kind}/test`	Musterexemplar an die Adresse des Aufrufers schicken

> Platzhalter werden als bloßes `{wort}` ersetzt, ohne die Mächtigkeit von Python-Formatstrings (kein `{x.attr}`) — auf einer vom Superadmin editierbaren Vorlage wäre das eine serverseitige Template-Injection (CWE-1336). Ein unbekannter Name bleibt wörtlich stehen, statt den Text zu verschlucken. Welche Namen eine Vorlage kennt, liefert `placeholders` mit.

> Der Testversand geht **ausschließlich** an die Adresse des angemeldeten Superadmins. Ein frei wählbarer Empfänger machte aus dem Admin-Portal einen Versender für beliebige Adressen — mit dem Absender und dem Ruf der Installation im Rücken. Ohne konfiguriertes SMTP antwortet der Endpunkt mit `400`, ein ablehnender Mailserver mit `502` samt dessen eigener Begründung.

Demo-Sitzungen

Jede Demo-Nutzung läuft als eigene, befristete Organisation (`is_demo=true`). Diese Endpunkte verwalten offene Sitzungen im Admin-Bereich.

Methode	Endpunkt	Beschreibung
`GET/PUT`	`/api/admin/settings/demo`	Demo-Modus an/aus, Sitzungsdauer, Karenzzeit je IP (Stunden) und Nachfrage-Mail konfigurieren
`GET`	`/api/admin/demo-sessions`	Offene Demo-Sitzungen auflisten (Ablaufzeit, Konvoi-Anzahl, Herkunft, Kontakt)
`POST`	`/api/admin/demo-sessions/{org_id}/extend`	Ablaufzeit einer Demo-Sitzung verlängern
`DELETE`	`/api/admin/demo-sessions/{org_id}`	Demo-Sitzung sofort beenden (Konvois, Org und Demo-Nutzer löschen)
`GET`	`/api/admin/demo-ip-locks`	Aktuell gesperrte IP-Adressen mit Restlaufzeit auflisten
`DELETE`	`/api/admin/demo-ip-locks/{ip}`	Sperre einer IP-Adresse aufheben
`GET`	`/api/admin/demo-ip-allowlist`	Dauerhaft freigestellte Adressen und Netze auflisten
`POST`	`/api/admin/demo-ip-allowlist`	Adresse (`203.0.113.7`) oder Netz (`203.0.113.0/24`) dauerhaft freistellen; hebt eine laufende Sperre mit auf
`DELETE`	`/api/admin/demo-ip-allowlist/{entry_id}`	Ausnahme zurücknehmen — ab dann gilt wieder die Karenzzeit
`GET`	`/api/admin/demo-stats`	Kennzahlen für die Übersicht im Demo-Reiter (Sitzungen, Interessenten, Stand der Nachfrage, Sperren)
`GET`	`/api/admin/demo-leads`	Interessenten aus dem Demo-Start auflisten (Kontakt, Sitzung, Stand der Nachfrage)
`DELETE`	`/api/admin/demo-leads/{lead_id}`	Kontaktangabe vorzeitig löschen (Löschersuchen nach DSGVO Art. 17)

Karenzzeit je IP: `POST /api/auth/demo-session` erlaubt je Client-IP eine Sitzung pro Karenzzeit (Standard 24 h, `DEMO_IP_COOLDOWN_HOURS` bzw. Admin-Portal; `0` schaltet sie ab). Läuft die Karenzzeit noch, **die Sitzung dieser Adresse aber ebenfalls**, wird sie mit einem frischen Token fortgesetzt: Antwort `200` mit `resumed: true` (Audit-Eintrag `demo.session.resumed`) — sonst sperrte die Karenzzeit den Besucher aus seiner eigenen Demo aus, sobald er den Tab schließt. Ist keine Sitzung mehr da, antwortet der Endpunkt mit `429`, `Retry-After` und einem strukturierten `detail`:

```
{"detail": {"message": "Pro IP-Adresse ist alle 24 Stunden …", "reason": "ip_cooldown",
            "retry_at": "2026-08-14T07:35:12+00:00", "retry_after": 79512, "cooldown_hours":
24}}
```

`retry_at` ist der genaue Zeitpunkt der Wiederfreigabe (ISO-8601, UTC) — die Oberfläche zeigt ihn in der Zeitzone des Besuchers an. Die Sperre liegt in der Datenbank (`demo_origins`), übersteht damit einen Neustart und gilt auch dann noch, wenn die Demo-Organisation längst abgelaufen und gelöscht ist. Der Retention-Job entfernt abgelaufene Einträge.

Nachfrage nach Sitzungsende: Der Retention-Durchgang verschickt einmalig eine E-Mail an die beim Start angegebene Adresse, sobald `session_expires_at` der Kontaktzeile erreicht ist — hat alles gepasst, sind Fragen offen, wird eine gemeinsame Session gebraucht. Abschaltbar über den Schalter im Admin-Portal bzw. `DEMO_FOLLOWUP_ENABLED`; ohne konfiguriertes SMTP unterbleibt der Versand, ohne einen Fehlversuch zu verbrauchen. Nach drei Fehlversuchen ruht eine Adresse. Der Abmeldelink im Fußbereich der Mail vermerkt den Widerspruch für die Adresse, nicht für die einzelne Sitzung. Siehe Lizenz und Demo-Modus.

Die Herkunftsfelder (`created_ip`, `created_location`) werden beim Start der Sitzung aus der Client-IP ermittelt und per Hintergrund-Geolokation (ipapi.co) um Stadt/Region/Land ergänzt — siehe Lizenz und Demo-Modus.

Sicherheit und Datenschutz

Methode	Endpunkt	Beschreibung
`POST`	`/api/admin/users/{user_id}/reset-password`	Passwort als Superadmin zurücksetzen
`POST`	`/api/admin/users/{user_id}/reset-mfa`	MFA eines Benutzers zurücksetzen
`GET`	`/api/admin/audit-log`	Security-Audit-Log (Filter nach Aktion)
`GET`	`/api/admin/users/{user_id}/export`	Personenbezogene Daten exportieren (DSGVO Art. 15)
`DELETE`	`/api/admin/users/{user_id}/data`	Benutzerdaten löschen, Audit-Trail pseudonymisieren (Art. 17)
`GET`	`/api/admin/organizations/{org_id}/export`	Alle Daten einer Organisation als ZIP

Lizenz

Methode	Endpunkt	Beschreibung
`GET`	`/api/license/instance-id`	Instanz-UUID abfragen (für Lizenzbeantragung)
`GET`	`/api/license/status`	Lizenzstatus, `demo_mode` und `key_source`
`POST`	`/api/license/activate`	Lizenzschlüssel validieren, speichern und Cache zurücksetzen

> Im Demo-Modus (kein gültiger Lizenzschlüssel) sind alle schreibenden Operationen (POST/PUT/PATCH/DELETE) mit **HTTP 402** gesperrt.

Fahrzeuge

Methode	Endpunkt	Beschreibung
`GET`	`/api/vehicles/`	Alle Fahrzeuge auflisten
`POST`	`/api/vehicles/`	Neues Fahrzeug anlegen
`GET`	`/api/vehicles/{vehicle_id}`	Fahrzeug abrufen
`PUT`	`/api/vehicles/{vehicle_id}`	Fahrzeug aktualisieren
`DELETE`	`/api/vehicles/{vehicle_id}`	Fahrzeug löschen

Fahrzeug-Felder:

Feld	Typ	Beschreibung
`callsign`	string	Funkrufname
`plate`	string	Kennzeichen
`length_m`	float	Fahrzeuglänge in Metern
`width_m`	float	Fahrzeugbreite in Metern
`weight_kg`	float	Gewicht in Kilogramm
`fuel_type`	string	Kraftstoffart
`fuel_capacity_l`	float	Tankvolumen in Litern
`fuel_consumption_per_100km`	float	Verbrauch auf 100 km

Marschverbände (Convoys)

Grundlegende Verwaltung

Methode	Endpunkt	Beschreibung
`GET`	`/api/convoys/`	Alle Marschverbände auflisten
`POST`	`/api/convoys/`	Neuen Marschverband anlegen
`GET`	`/api/convoys/{convoy_id}`	Marschverband abrufen
`PUT`	`/api/convoys/{convoy_id}`	Marschverband bearbeiten

Methode	Endpunkt	Beschreibung
`DELETE`	`/api/convoys/{convoy_id}`	Marschverband löschen

Fahrzeuge zuordnen

Methode	Endpunkt	Beschreibung
`POST`	`/api/convoys/{convoy_id}/vehicles`	Fahrzeug dem Konvoi zuordnen
`DELETE`	`/api/convoys/{convoy_id}/vehicles/{vehicle_id}`	Fahrzeug aus Konvoi entfernen

Wegpunkte

Methode	Endpunkt	Beschreibung
`GET`	`/api/convoys/{convoy_id}/waypoints`	Wegpunkte abrufen
`POST`	`/api/convoys/{convoy_id}/waypoints`	Wegpunkt hinzufügen
`PUT`	`/api/convoys/{convoy_id}/waypoints/{waypoint_id}`	Wegpunkt aktualisieren
`DELETE`	`/api/convoys/{convoy_id}/waypoints/{waypoint_id}`	Wegpunkt entfernen

Wegpunkt-Typen: `start`, `stop`, `checkpoint`, `fuel_stop`, `technical_halt`

Routing

Methode	Endpunkt	Beschreibung
`POST`	`/api/convoys/{convoy_id}/calculate-route`	Route und Zeitplan berechnen

> Die Fahrzeit je Wegpunkt wird streckenproportional zur Gesamtroute berechnet; die Antwort enthält zusätzlich `planned_departure` (Abmarsch) und `planned_arrival` des Ziels — siehe Konvoi-Planung → Automatische Zeitplanung.

Teilverbände

Methode	Endpunkt	Beschreibung
`GET`	`/api/convoys/{convoy_id}/sub-convoys`	Teilverbände abrufen
`POST`	`/api/convoys/{convoy_id}/sub-convoys`	Neuen Teilverband erstellen

Export und Import

Methode	Endpunkt	Beschreibung
`GET`	`/api/convoys/{convoy_id}/export/gpx`	Route als GPX exportieren
`GET`	`/api/convoys/{convoy_id}/export/json`	Route als JSON exportieren
`GET`	`/api/convoys/{convoy_id}/export/pdf`	Marschbefehl als PDF exportieren
`GET`	`/api/convoys/{convoy_id}/export/roadbook`	Roadbook (Karte + Navigationsanweisungen) als PDF
`POST`	`/api/convoys/{convoy_id}/import/gpx`	GPX-Track importieren
`POST`	`/api/convoys/{convoy_id}/import/geojson`	GeoJSON-Route importieren

Zusatzfunktionen

Methode	Endpunkt	Beschreibung
`GET`	`/api/convoys/{convoy_id}/fuel-stations`	Tankstellen entlang der Route
`GET`	`/api/convoys/share/{token}`	Öffentliche Routenansicht (kein Login)

Freigabelinks und Tracking-App

Methode	Endpunkt	Beschreibung
`GET/POST/DELETE`	`/api/convoys/{convoy_id}/share-links`	Widerrufbare Freigabelinks pro Konvoi
`GET/POST`	`/api/track/{slug}`	Öffentliche Tracking-App: Status abrufen / Position senden
`WS`	`/api/ws/track/{slug}`	WebSocket der Tracking-App

Fahrhinweise: `GET /api/track/{slug}` liefert neben Route und Kanalwechseln `route_steps` — die Abbiegehinweise der Route, nach `m` aufsteigend. Je Eintrag: `m` (Meter ab Start entlang der ausgelieferten `geojson`-Linie, Haversine über die Stützpunkte), `sign` (GraphHopper-Vorzeichen: −3…3 abbiegen, 0 geradeaus, 4 Ziel, 5 Zwischenziel, 6 Kreisverkehr, ±7 halten, ±8/−98 wenden), `text`, `street_name` (ersatzweise die Straßennummer) und `exit_number`. Leer bei importierten Routen. Wer den eigenen Standort auf dieselbe Linie projiziert, kann `m` direkt vergleichen — so nutzt es die ConvoyPlan Companion-App.

Live-Tracking

REST

Methode	Endpunkt	Beschreibung
`GET`	`/api/convoys/{convoy_id}/positions`	Aktuelle Positionen aller Fahrzeuge
`POST`	`/api/convoys/{convoy_id}/positions`	Eigene Position aktualisieren
`PATCH`	`/api/convoys/{convoy_id}/vehicles/{vehicle_id}/status`	Fahrzeugstatus setzen

Fahrzeugstatus-Werte: `planned`, `en_route`, `arrived`, `delayed`

WebSocket

```
WS /ws/tracking/{convoy_id}?token=<jwt>
```

Position senden:

```
{
  "type": "position_update",
  "vehicle_id": 42,
  "lat": 48.1374,
  "lon": 11.5755
}
```

Eingehende Updates vom Server:

```
{
  "type": "positions",
  "data": [
    {
      "vehicle_id": 42,
      "callsign": "RTW 1-1",
      "lat": 48.1374,
      "lon": 11.5755,
      "status": "en_route",
      "timestamp": "2026-05-18T14:30:00Z"
    }
  ]
}
```

Organisationen

Methode	Endpunkt	Beschreibung
`GET`	`/api/organizations/`	Eigene Organisationen auflisten
`POST`	`/api/organizations/`	Neue Organisation anlegen
`DELETE`	`/api/organizations/{org_id}`	Organisation löschen
`GET`	`/api/organizations/{org_id}/export`	Alle Daten der Organisation als ZIP (Org-Admin, nur mit Anmeldung, nicht per API-Key)
`POST`	`/api/organizations/{org_id}/members`	Bestehenden Benutzer per E-Mail zur Org hinzufügen (Org-Admin-Rolle erforderlich) — löst eine Benachrichtigungs-Mail mit Org-Login-Link an den neuen Mitglied aus

> Siehe auch **Multi-Tenancy**.

Org-Branding

Org-Admin-Rolle erforderlich (außer dem öffentlichen Slug-Endpunkt). Die Overrides gelten nur innerhalb `/o/[slug]`; das globale `/api/branding` (Superadmin, siehe oben) bleibt davon unberührt.

Methode	Endpunkt	Beschreibung
`GET`	`/api/org/branding`	Effektives Branding der eigenen Org abrufen (Plattform-Branding + Org-Overrides zusammengeführt)
`PUT`	`/api/org/branding`	Branding-Overrides (Farben, App-Name) der eigenen Org setzen
`POST`	`/api/org/branding/logo/{slot}`	Logo hochladen (`slot`: `main` oder `horizontal`), PNG/JPG/SVG, max. 2 MB
`DELETE`	`/api/org/branding`	Alle Overrides entfernen — Org fällt zurück auf das Plattform-Branding
`GET`	`/api/branding/org/{slug}`	**Öffentlich:** effektives Branding einer Org per Slug (für die Org-Login-Seite, kein Login nötig)

Methode	Endpunkt	Beschreibung
`GET`	`/api/branding/logos/{datei}`	**Öffentlich:** ein hochgeladenes Logo. Die URL samt `?v=` steht in `logo_main_url` / `logo_horizontal_url` der Branding-Antworten

Leitstellen

Methode	Endpunkt	Beschreibung
`GET/POST/PUT/DELETE`	`/api/leitstellen/`	Globale (superadmin-gepflegte) Leitstellen verwalten
`GET/POST/PUT/DELETE`	`/api/org/leitstellen/`	Org-eigene Leitstellen verwalten
`POST`	`/api/org/leitstellen/{id}/submit`	Org-Leitstelle als globalen Vorschlag einreichen
`GET`	`/api/org/leitstellen/geojson`	Sichtbare Leitstellengebiete als GeoJSON (Übersichtskarte)

Wetter, Sperrungen und Verkehrslage

Methode	Endpunkt	Beschreibung
`GET`	`/api/weather/?lat=48.13&lon=11.57`	Wetterdaten (Open-Meteo)
`GET`	`/api/overpass/closures?lat=..&lon=..`	Sperrungen/Baustellen im Radius um einen Punkt (Overpass, Autobahn-API, offene Feeds, DATEX-II)
`POST`	`/api/overpass/closures/route`	Sperrungen/Baustellen im Korridor entlang einer Routen-Geometrie
`GET`	`/api/traffic/flow/status`	Konfigurierten Verkehrslage-Anbieter (HERE/TomTom) abfragen
`GET`	`/api/traffic/flow`	Live-Verkehrslage im Radius um einen Punkt
`POST`	`/api/traffic/flow/route`	Live-Verkehrslage entlang einer Routen-Geometrie

> Einrichtung der optionalen Quellen: **Verkehrsdaten**.

System

Methode	Endpunkt	Beschreibung
`GET`	`/api/version`	Build-Version, Commit-SHA und Update-Hinweis
`GET`	`/api/version/changelog`	Release-Notes der laufenden Version (gecacht)
`GET`	`/health`	Health-Check mit Versionsangabe
`GET`	`/api/status`	Technischer Dienststatus inkl. Latenzen und Kartenausschnitt

Methode	Endpunkt	Beschreibung
`GET`	`/api/status/public`	Funktionsstatus für die öffentliche Statusseite `/status`

`GET /api/status/public` ist die Datenquelle der öffentlichen Statusseite und bewusst grobkörnig: je Funktion (Portal, Konvoi-Daten, Routenplanung, Live-Tracking, Verkehr & Sperrungen, Wetter) nur `operational`, `degraded`, `down` oder `unknown` — **keine** Latenzen, Anbieternamen, Kartenausschnitte oder Versionsangaben. Ist eine Funktion nicht betriebsbereit, nennt `reason` den Grund in Anwendersprache (etwa „Die Kartenregion wird gerade gewechselt"); sonst ist das Feld `null`. Das Ergebnis ist 15 Sekunden gecacht, damit häufiges Neuladen der Seite nicht auf die geprüften Dienste durchschlägt. Wer die technischen Details braucht, nutzt `GET /api/status`.

Systemübersicht (Superadmin)

Hardware-, Container- und Nutzungskennzahlen der Instanz — dieselbe Datenbasis, die der Reiter **Systemübersicht** im Admin-Portal anzeigt. Details siehe Systemübersicht.

Die lesenden Endpunkte akzeptieren **entweder** einen Superadmin-Token **oder** einen System-API-Key. Das Erfassen einer Stichprobe verändert den Zustand und erfordert einen Superadmin-Token.

Methode	Endpunkt	Zugang	Beschreibung
`GET`	`/api/admin/system/overview`	Token **oder** System-Key	Live-Zustand: CPU, RAM, Platten, PSI-Druck, Container, Datenbank, aktive Benutzer
`GET`	`/api/admin/system/prtg`	Token **oder** System-Key	Derselbe Live-Zustand im Kanalformat von PRTG
`GET`	`/api/admin/system/containers`	Token **oder** System-Key	Container-Zustände inkl. Healthcheck und CPU/RAM je Container (`with_stats=true|false`)
`GET`	`/api/admin/system/history`	Token **oder** System-Key	Verlauf (`range=1h…365d` oder `from`/`to`, `resolution=auto|raw|hour|day`, `format=json|csv`)
`GET`	`/api/admin/system/usage`	Token **oder** System-Key	Portalnutzung je Tag (`days=1…1095`)
`GET`	`/api/admin/system/reports/months`	Token **oder** System-Key	Monate mit verfügbarem Bericht
`GET`	`/api/admin/system/reports/monthly`	Token **oder** System-Key	Monatsbericht (`month=JJJJ-MM`, `format=json|csv|pdf`)
`POST`	`/api/admin/system/sample`	nur Superadmin-Token	Stichprobe sofort erfassen

Beispiel — Monatsbericht als PDF holen:

```
curl -H "Authorization: Bearer $TOKEN" \
  "https://web.convoyplan.de/api/admin/system/reports/monthly?month=2026-07&format=pdf" \
  -o systembericht-2026-07.pdf
```

Beispiel — Auslastung der letzten 30 Tage für ein Monitoring-System, ohne Login:

```
curl -H "X-API-Key: cvp_a1b2c3d4_xxxxxxxx" \
  "https://web.convoyplan.de/api/admin/system/history?range=30d&resolution=hour"
```

Beispiel — CPU-Auslastung als einzelner Wert für einen Uptime-/Metrik-Check:

```
curl -sH "X-API-Key: $CVP_SYSTEM_KEY" \
  https://web.convoyplan.de/api/admin/system/overview | jq '.host.cpu_percent'
```

Verlaufsdaten reichen so weit zurück, wie die Aufbewahrung erlaubt: Rohstichproben 90 Tage, verdichtete Tageswerte standardmäßig 3 Jahre. `resolution=auto` wählt selbstständig zwischen Roh-, Stunden- und Tageswerten, damit die Antwort handlich bleibt.

Anbindung an PRTG

`GET /api/admin/system/prtg` liefert dieselbe Momentaufnahme wie `/overview`, nur im Kanalformat des PRTG-Sensortyps **HTTP Data Advanced**. Ein Skript oder ein Mapping-Template auf der Probe ist dadurch nicht nötig.

Einrichtung in PRTG:

1. Im Superadmin-Portal unter **Admin → API-Keys** einen Key mit Geltungsbereich *System* anlegen. Der Klartext wird nur einmal angezeigt.
2. Neuen Sensor **HTTP Data Advanced** an dem Gerät anlegen, das die Instanz vertritt.
3. **URL:** `https://web.convoyplan.de/api/admin/system/prtg`
4. **Custom HTTP Headers:** `X-API-Key: cvp_a1b2c3d4_xxxxxxxx`
5. Abfrageintervall auf 60 s oder mehr stellen — die Momentaufnahme misst die CPU-Last über ein kurzes Messfenster und kostet dabei etwas Zeit.

Die Antwort lässt sich vorab prüfen:

```
curl -sH "X-API-Key: $CVP_SYSTEM_KEY" \
  https://web.convoyplan.de/api/admin/system/prtg | jq '.prtg.result[].channel'
```

Kanäle der Antwort:

Bereich	Kanäle
Host	CPU-Auslastung, Load 1 min, Load 5 min, Arbeitsspeicher belegt, Arbeitsspeicher verfügbar, Swap belegt, Plattenbelegung, Platte frei, Lese-/Schreibrate, Laufzeit
Druck (PSI)	CPU-Druck, Speicher-Druck, I/O-Druck
Container	Container gesamt, Container laufend, Container gestört
Datenbank	Datenbankgröße, Datenbankverbindungen
Nutzung	Aktive Benutzer, Aktive Demo-Besucher, Eindeutige Benutzer heute, Anmeldungen 24 h, Antwortzeit Ø
Erfassung	Alter der letzten Stichprobe

Zu Auslastung, Druck und gestörten Containern liefert die Antwort Warn- und Fehlergrenzen mit. PRTG übernimmt sie beim Anlegen der Kanäle; wer sie danach in den Kanaleinstellungen ändert, behält seine eigenen Werte.

Nicht jeder Kanal erscheint auf jeder Instanz:

- **PSI-Kanäle** fehlen, wenn der Kernel keine Druckwerte liefert.
- **Container-Kanäle** fehlen, solange die Docker-API nicht erreichbar ist — statt dreier Nullen, die wie ein toter Stack aussähen, steht der Grund im Sensortext.

- **Alter der letzten Stichprobe** fehlt, solange der Collector abgeschaltet ist (`SYSTEM_METRICS_ENABLED=false`) oder noch nie gemessen hat.

Ein Kennwert, den der Host grundsätzlich nicht messen kann, fehlt dauerhaft — das ist Absicht: PRTG erkennt Kanäle am Namen, und Kanäle, die kommen und gehen, zerreißen den Verlauf.

Lässt sich überhaupt nichts messen, antwortet der Endpunkt mit dem Fehlerformat des Sensors (`{"prtg": {"error": 1, "text": "…"}}`) statt mit HTTP 500, damit in PRTG der Grund und nicht nur „Server Error" am Sensor steht. Ein falscher oder fehlender Key bleibt ein `401`, ein Org-Key ein `403`.

Für einen reinen Erreichbarkeitscheck genügt ohne Key ein HTTP-Sensor auf `/health` oder `/api/status`.

Datenmodell

```
User
├── Vehicles
├── Convoys
└── UserOrganizations

Organization
├── org_code                # Kurzer URL-Slug (4–8 Zeichen), eindeutig
├── branding                # JSON-Override, überlagert das globale Plattform-Branding
└── UserOrganizations

Convoy
├── parent_convoy_id        # Teilverband / Sub-Convoy
├── organization_id         # Mandant / Organisation
├── ConvoyVehicles          # Fahrzeugzuordnung inkl. Status
├── Waypoints               # Wegpunkte, Kontrollpunkte, Halte
├── Route                   # Geometrie, Distanz, Dauer, GPX, Kanalwechsel
└── VehiclePositions        # Live-Tracking-Positionen

Leitstelle
└── boundary                # GeoJSON/KML-Zuständigkeitsgebiet
```

KAPITEL 10

MCP-Server (KI-Schnittstelle)

ConvoyPlan kann seine Fachdaten über einen **Model-Context-Protocol-Server** bereitstellen. KI-Programme wie Claude Desktop, Claude Code oder claude.ai tragen die Instanz als *Remote-MCP-Server* ein und arbeiten danach mit Konvois, Fahrzeugen, Wegpunkten, Routen und Marschstatus — im Rahmen dessen, was der zustimmende Benutzer selbst darf.

> **Standardmäßig ausgeschaltet.** Ohne `MCP_ENABLED=true` existiert weder `/mcp` noch ein Discovery-Dokument. Das Einschalten öffnet Einsatzdaten für einen externen Modellanbieter — diese Entscheidung gehört dem Betreiber, nicht dem Auslieferungszustand.

Was der Server kann

Lesen (neun Werkzeuge): Konvois und Unterkonvois auflisten, Konvoi-Details samt aller sieben Abschnitte des Marschbefehls, Fahrzeugbestand und Einzelfahrzeug, Wegpunkte in Marschreihenfolge, die gespeicherte Route, zuletzt gemeldete Positionen, Marschstatus, Mannschaftsstärke und Betriebsstofflage je Fahrzeug.

Schreiben (vierzehn Werkzeuge, nur mit gültiger Lizenz): Konvoi und Fahrzeug anlegen und ändern, Fahrzeuge zuordnen und wieder lösen, Marschfolge setzen, Wegpunkte anlegen, ändern und umsortieren, Route berechnen, Fahrzeugstatus, Mannschaftsstärke und Betriebsstoff melden.

Die Stärke meldet `fahrzeugstaerke_melden` in der Notation Führer/Unterführer/Mannschaften; die Gesamtzahl rechnet der Server und nimmt sie nicht entgegen. `konvoi_status` gibt sie je Fahrzeug und als Verbandsstärke zurück. Ein Fahrzeug **ohne** Meldung steht dort als `null` — nicht als `0/0/0`, das eine Meldung wäre und „unbesetzt" hieße. Melden braucht den Scope `fleet:status`, Lesen `convoy:read`; beides zusätzlich den freigegebenen Bereich *Live-Positionen, Marschstatus, Mannschaftsstärke und Betriebsstoff*.

Den Betriebsstoff meldet `fahrzeug_betriebsstoff_melden` — meist genügt `fuellstand_prozent`, denn Tankvolumen und Verbrauch kennt ConvoyPlan aus den Stammdaten des Fahrzeugs. Die Meldung ersetzt die vorige ganz und ändert die Stammdaten nicht. `konvoi_status` gibt je Fahrzeug Füllstand, Liter im Tank und Reichweite zurück (mit dem Hinweis, wo die Stammdaten eingesprungen sind) und für den Verband, wer knapp ist (ab 25 %) und wer noch nichts gemeldet hat. Bei E-Fahrzeugen liefert `konvoi_status` dieselben Angaben unter eigenen kWh-Schlüsseln (`akku_kapazitaet_kwh`, `verbrauch_kwh_100km`, `kwh_im_akku`, `reichweite_km`) statt `tank_l` und `liter_im_tank` — auch sie zählen in der Liste der knappen Fahrzeuge. Auch hier heißt `null` „nicht gemeldet", ein Füllstand `0` dagegen „leer". Scope und Bereich wie bei der Stärke.

Dokumente (Resources): Marschbefehl als PDF, Route als GPX, Konvoi als JSON — dieselben Exporte wie im Portal.

Live mitlaufen (Abonnement): Ein Programm kann einen Konvoi *abonnieren* und wird benachrichtigt, sobald sich dort etwas bewegt, statt im Sekundentakt nachzufragen. Übertragen wird dabei nur der Anstoß „hier gibt es Neues" — die Daten holt das Programm anschließend selbst, und dabei greift dieselbe Prüfung wie bei jedem anderen Zugriff. Voraussetzung ist ein Programm, das die MCP-Revision 2026-07-28 spricht.

Was der Server nicht kann

Löschen	Es gibt kein Werkzeug, das einen Konvoi, ein Fahrzeug, einen Wegpunkt, eine Route oder einen Benutzer löscht. Die einzige Ausnahme von der Anlegen-und-Ändern-Regel ist das **Lösen einer Fahrzeug-Zuordnung** — dabei bleiben Fahrzeug und Konvoi bestehen.
Administration	Keine Benutzerverwaltung, keine Lizenz, keine Systemkennzahlen, kein Regionswechsel, kein Branding, keine Leitstellen-Konfiguration.
Mehrere Organisationen	Ein Zugang gilt für genau **eine** Organisation, auch wenn der Benutzer in mehreren Mitglied ist.

Einschalten

Im Admin-Portal unter **System → KI-Schnittstelle**: ein Klick auf *Schnittstelle aktivieren*. Das wirkt sofort, ein Neustart ist nicht nötig.

Alternativ als Ausgangswert in der `.env`:

```
MCP_ENABLED=true
```

Die Einstellung aus dem Portal hat Vorrang; die Umgebungsvariable gilt, solange im Portal nichts eingestellt wurde. Der Reiter **MCP** zeigt anschließend den Zustand, die Verbindungsadresse und die erteilten Zugänge.

Und dann noch je Organisation

Der Schalter oben sagt, ob es die Schnittstelle auf dieser Instanz **gibt**. Ob eine einzelne Organisation daran teilnimmt, entscheidet deren eigener Admin — im **Org-Adminbereich unter „KI-Zugriff"**, und **standardmäßig ist er aus**.

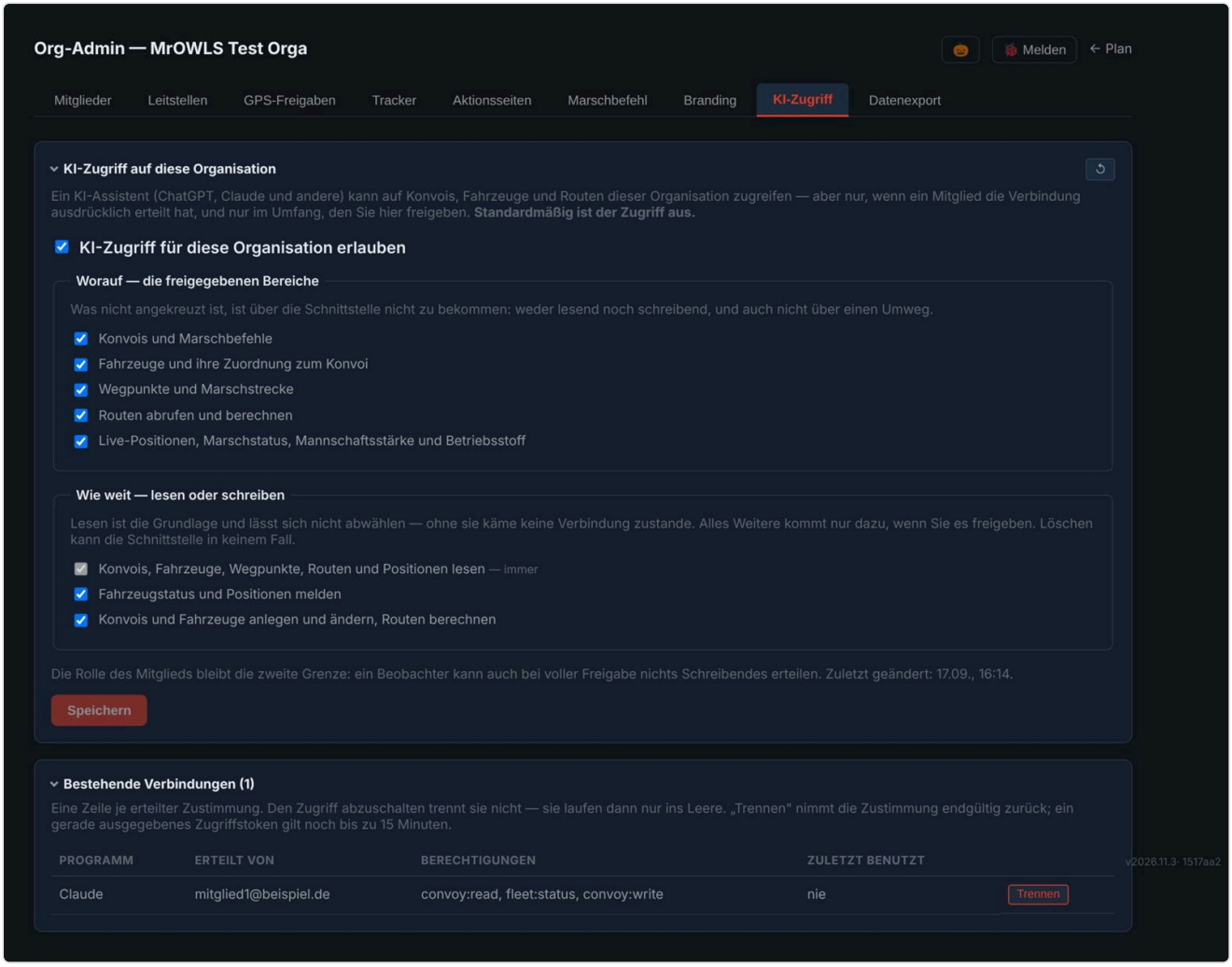


Abbildung: Org-Admin, Reiter „KI-Zugriff“: Bereiche, Lese- und Schreibrechte, bestehende Verbindungen

Das ist keine doppelte Absicherung aus Vorsicht, sondern eine Trennung: wer eine Instanz für mehrere BOS-Organisationen betreibt, entscheidet nicht über deren Einsatzdaten. Eine frisch angelegte Organisation nimmt deshalb nicht teil, auch auf einer Instanz, auf der die Schnittstelle längst läuft.

Der Org-Admin stellt dort drei Dinge ein:

Einstellung	Bedeutung
KI-Zugriff erlauben	Der Schalter der Organisation. Aus heißt: kein Werkzeug, keine Resource, kein Abo — auch für bestehende Verbindungen, ab dem nächsten Aufruf.
Bereiche	*Worauf.* Konvois und Marschbefehle, Fahrzeuge, Wegpunkte, Routen, Live-Positionen, Marschstatus und Mannschaftsstärke — einzeln an- und abwählbar. Was nicht angekreuzt ist, ist über die Schnittstelle nicht zu bekommen, auch nicht über ein anderes Werkzeug.
Lesen / Status / Schreiben	*Wie weit.* Lesen ist die Grundlage und lässt sich nicht abwählen — ohne sie käme keine Verbindung zustande. Statusmeldungen und Schreiben kommen nur dazu, wenn sie freigegeben sind. Gelöscht wird in keinem Fall.

Die beiden Achsen wirken als **Und**: ein Werkzeug steht zur Verfügung, wenn sein Bereich freigegeben *und* sein Zugriff erlaubt ist. Die Rolle des Mitglieds bleibt die zweite Grenze darüber — ein Beobachter kann auch bei voller Freigabe nichts Schreibendes erteilen.

Ein Assistent sieht dabei nur, was übrig bleibt: was die Organisation nicht freigibt, steht gar nicht erst in seiner Werkzeugliste. Das ist Absicht — ein Modell, das ein Werkzeug sieht, probiert es aus, und eine Absage nach dem Versuch ist eine schlechtere Auskunft als ein Werkzeug, das es nicht gibt.

> **Beim Update auf diese Version** bleiben Organisationen eingeschaltet, an denen bereits eine aktive Verbindung hängt — mit genau den Rechten, die dort in Gebrauch sind. Eine laufende Anbindung soll nicht wortlos abreißen. Alle übrigen Organisationen stehen auf aus.

Im selben Tab steht die Liste der **bestehenden Verbindungen** dieser Organisation. Den Zugriff abzuschalten trennt sie nicht — sie laufen dann ins Leere. Wer sie wirklich los sein will, trennt sie dort einzeln.

Was der Betreiber davon sieht

Im Admin-Portal unter **MCP** steht eine Übersicht aller Organisationen: ob sie den KI-Zugriff freigegeben haben, welche Bereiche und Rechte, und wie viele Verbindungen daran hängen. Abgeschaltete Organisationen sind eingeklappt und über *Alle anzeigen* zu erreichen — dort steht dann auch die Antwort auf die häufigste Rückfrage, warum eine Anbindung nicht zustande kommt.

Die Übersicht ist **nur zum Ansehen**. Der Betreiber der Instanz schaltet die Schnittstelle als Ganzes ein oder aus; über die Einsatzdaten einer einzelnen Organisation entscheidet deren Admin. Eine Zeile mit `0` Verbindungen und „frei" heißt: es wäre möglich, es hat nur noch niemand verbunden.

Wenn der Schalter auf „An" steht und trotzdem nichts geht

Zwei Dinge können es sein: der Reverse Proxy (unten) oder der Zustimmungsbildschirm. Kommt der Client bis zur Anmeldung und scheitert dort, liegt es nicht am Proxy.

Zustimmung: wer darf sie erteilen?

Jedes Mitglied einer Organisation, nicht nur Superadmins. Der Bildschirm erkennt die **Person** und bietet danach alle Organisationen zur Auswahl an, in denen sie Mitglied ist; die Rolle dort deckelt, welche Zugriffe erteilbar sind.

Sind im selben Browser **verschiedene Personen** angemeldet, verweigert der Bildschirm die Auskunft, statt sich eine auszusuchen. Dann in den anderen Konten abmelden und im Client neu beginnen.

Wer nicht angemeldet ist, wird zur Anmeldung geschickt und danach auf den Zustimmungsbildschirm zurückgebracht. Dauert das zu lange, verfällt die Anfrage des Clients — dann im Client erneut starten.

Der Schalter mountet die Routen **im Backend**. Ob der Reverse Proxy sie von außen auch dorthin leitet, ist eine zweite Frage — und bei einer Installation, die von vor dieser Schnittstelle stammt, lautet die Antwort oft nein. Das Portal zeigte dann „An" samt Verbindungsadresse, und ein Aufruf von außen landete beim Frontend.

Der Reiter **System → KI-Schnittstelle** prüft das jetzt mit und sagt es: steht dort eine Zeile **Proxy** mit einem Hinweis, leitet der Proxy die Pfade nicht weiter. Der Knopf **Proxy reparieren** stellt sie her — er schreibt eine dauerhafte Proxy-Konfiguration und lädt sie sofort nach, ohne Neustart und ohne Zugriff auf den Server.

Geprüft wird dabei die **laufende** Konfiguration über Caddys Admin-API, nicht eine Datei auf der Platte. Das ist der Unterschied, der zählt: die Datei kann längst stimmen, während der Caddy-Container noch mit der alten Konfiguration läuft.

„Die Schnittstelle ist erreichbar, aber nicht dauerhaft hinterlegt"

Auf Installationen, die aus der Zeit stammen, als das Backend noch als `root` lief, gehört das gemeinsame Verzeichnis `/certs` weiterhin `root` — Docker überträgt die Besitzrechte aus dem Abbild nur beim allerersten Mount eines leeren Volumes. Das Backend läuft seitdem als eigener Benutzer und darf dort nicht mehr schreiben.

Der Knopf bringt die Routen in diesem Fall trotzdem in den laufenden Proxy — das Nachladen braucht keine Datei — und sagt dazu, dass ein Neustart des Proxy-Containers sie wieder verliert. Der Proxy zieht die Besitzrechte beim nächsten Start selbst gerade, das passiert also spätestens mit dem nächsten Update von

allein. Danach genügt ein erneuter Klick, und die Konfiguration ist dauerhaft hinterlegt. **Kein SSH-Zugriff nötig.**

Solange das aussteht, bleibt der Knopf sichtbar, obwohl die Schnittstelle erreichbar ist — der Hinweis daneben sagt, warum.

Zwei Fälle kann der Knopf nicht lösen:

- **Die Setup-Werte fehlen** (Domain, TLS-Modus stehen nicht in der Datenbank). Daraus lässt sich keine Konfiguration erzeugen — dann hilft nur der Setup-Assistent. Das Portal bietet den Knopf in diesem Fall gar nicht erst an.
- **Caddys Admin-API antwortet nicht.** Dann ist der Zustand schlicht unbekannt; das Portal sagt das so, statt einen Fehler zu behaupten. Prüfen lässt es sich mit einem Aufruf von außen:

```
curl -i -X POST https://<domain>/mcp -H 'Content-Type: application/json' -d '{}'
```

Antwort	Bedeutung
401 mit `WWW-Authenticate`	Alles richtig — ein Client kann sich jetzt anmelden.
404 als JSON (`{"detail":"Not Found"}`)	Der Proxy leitet weiter, die Schnittstelle ist nur abgeschaltet.
404 als HTML	Der Proxy leitet **nicht** weiter — die Anfrage ist beim Frontend gelandet.

Ausgeschaltet heißt wirklich ausgeschaltet. Der Schalter entfernt die Routen, statt sie mit einer Fehlerseite zu bedecken: es gibt dann weder `/mcp` noch die Discovery-Dokumente — nicht als 404 eines Handlers, sondern weil keine Route passt. Genau das war der Grund, warum es lange nur eine Umgebungsvariable gab; die Zusage gilt mit dem Knopf unverändert weiter und wird von einem Test festgehalten.

Was nach dem Ausschalten bleibt: `/authorize`, `/token`, `/revoke` und `/.well-known/oauth-authorization-server` — sie dienen auch der Anmeldung der Begleit-App (siehe Sicherheit und Datenschutz, „Anmeldung der Begleit-App"). MCP-Clients kommen dort trotzdem nicht weiter, und `/mcp`, `/register` sowie die Protected-Resource-Metadaten gibt es nicht mehr. Mit `APP_OAUTH_ENABLED=false` verschwinden auch die vier.

Was das Ausschalten nicht tut: bereits ausgestellte Zugriffstoken ungültig machen. Die laufen ins Leere, weil der Endpunkt fehlt, bleiben aber Tokens. Wer sie wirklich entziehen will, trennt die Verbindungen im Reiter **MCP**.

Der Reverse Proxy braucht Routen für `/mcp` und die OAuth-Pfade an der Wurzel der Site. **Bestehende Installationen rüsten das beim nächsten Backend-Start automatisch nach**; bei Neuinstallationen ist es von vornherein enthalten.

Verbinden

1. Im Admin-Portal unter **MCP** die **Adresse** kopieren (`https://<domain>/mcp`).
2. Im KI-Programm als *Remote-MCP-Server* eintragen.
3. Das Programm öffnet den Browser auf der ConvoyPlan-Anmeldung.
4. **Anmelden** — mit dem eigenen Konto, inklusive MFA, falls eingerichtet.
5. **Organisation wählen.** Der Zugang gilt nur für diese.

6. **Rechte ankreuzen.** Vorausgewählt ist, was das Programm verlangt hat. Was die eigene Rolle darüber hinaus hergibt, steht darunter zum Ankreuzen — und was sie nicht hergibt, ist durchgestrichen und wird nicht erteilt.
7. **Zustimmen.** Erst damit entsteht ein Zugang.

Auf dem Zustimmungsbildschirm stehen zwei Angaben nebeneinander, und der Unterschied ist wichtig:

Angabe	Bedeutung
Name des Programms — als *ungeprüft* gekennzeichnet	Selbstauskunft. Jedes Programm, das die Instanz erreicht, kann sich registrieren und sich dabei nennen, wie es will — auch „ConvoyPlan Desktop".
Zieladresse — als *geprüft* gekennzeichnet	Wurde bei der Registrierung hinterlegt und wird überprüft. Dorthin geht der Zugang.

Zustimmen nur, wenn du dieses Programm gerade selbst verbunden hast und die Zieladresse dazu passt.

> **ChatGPT** braucht ein paar Schritte mehr — Developer Mode, Einschalten je Unterhaltung, und den Haken bei den Schreibrechten, den es später nicht nachholen kann. Dafür gibt es eine eigene Seite: ConvoyPlan in ChatGPT verbinden.

Berechtigungen

Die Rechte des Zugangs sind eine Projektion der Rolle in der gewählten Organisation — nicht mehr, als der Benutzer selbst darf:

Berechtigung	ab Rolle	erlaubt
`convoy:read`	beobachter	Konvois, Fahrzeuge, Wegpunkte, Routen und Positionen lesen
`fleet:status`	fahrer	zusätzlich Fahrzeugstatus, Positionen und Mannschaftsstärke melden
`convoy:write`	planer	zusätzlich anlegen, ändern und Routen berechnen

Darüber liegt die Freigabe der Organisation (siehe *Und dann noch je Organisation*). Sie deckelt beides: die Bereiche und die Berechtigungen. Was die Organisation nicht freigibt, bekommt auch ein Admin nicht erteilt — und was sie später zurücknimmt, ist mit dem nächsten Aufruf zu.

Geprüft wird bei **jedem** Aufruf frisch gegen die Datenbank. Wird eine Mitgliedschaft entzogen, eine Rolle herabgestuft oder eine Freigabe zurückgenommen, wirkt das sofort — nicht erst, wenn der Zugang abläuft.

Nachvollziehbarkeit

Jeder **schreibende** Aufruf erzeugt einen Eintrag im Audit-Log mit Benutzer, Organisation, Werkzeugname, Parametern und dem Programm, das ihn ausgelöst hat. Die Quelle ist als `mcp` gekennzeichnet — im Log ist damit unterscheidbar, was ein Mensch im Portal getan hat und was ein Modell über die Schnittstelle.

Lesende Aufrufe werden bewusst **nicht** protokolliert. Ein Modell liest im Minutentakt; das Log wäre sonst nach einer Woche unbrauchbar.

Zugang entziehen

Im **Org-Adminbereich unter „KI-Zugriff"** (für die eigene Organisation):

- **KI-Zugriff abschalten** — wirkt sofort auf alle Verbindungen dieser Organisation. Sie bleiben stehen und laufen ins Leere; wieder einschalten stellt sie ohne Zutun her.
- **Einzelne Bereiche oder Berechtigungen zurücknehmen** — wirkt ebenso sofort, ohne die Verbindung zu beenden.
- **Verbindung trennen** — nimmt eine erteilte Zustimmung endgültig zurück.

Im Admin-Portal unter **MCP** (für die ganze Instanz):

- **Verbindung trennen** — entzieht einem Benutzer den Zugang für ein bestimmtes Programm.
- **Programm sperren** — trennt alle seine Verbindungen und verhindert eine neue Autorisierung.

Zusätzlich entzieht jede Maßnahme, die ohnehin alle Sitzungen beendet (Passwortwechsel, „überall abmelden", Deaktivieren des Kontos, Entzug der Mitgliedschaft), auch die MCP-Zugänge.

Verwaiste Registrierungen aufräumen

Die Liste **Registrierte Programme** wächst schneller, als man erwartet: bei Selbstregistrierung legt **jeder** Verbindungsversuch eine Zeile an — auch der abgebrochene, und manche Programme registrieren sich bei jedem Anlauf neu. Nach ein paar Versuchen stehen dort ein Dutzend gleich aussehender Einträge, von denen einer etwas trägt.

Als **verwaist** gilt eine Zeile, an der nichts mehr hängt: keine Verbindung, kein laufender Verbindungsversuch, und älter als eine Stunde. Solche Zeilen sind gekennzeichnet, und der Knopf **„… verwaiste entfernen"** löscht genau sie.

Zwei Punkte dazu:

- **Das ist kein Zugangsentzug.** Eine Registrierung *ist* kein Zugang — der entsteht erst durch die Zustimmung eines Benutzers. Was eine Verbindung trägt, bleibt deshalb stehen, auch wenn es alt ist und auch wenn es gesperrt ist.
- **Es passiert ohnehin von selbst.** Der nächtliche Aufräumdurchgang macht dasselbe, nur mit größerer Frist (`RETENTION_OAUTH_CLIENTS_DAYS`, Standard 7 Tage). Der Knopf nimmt das vorweg, wenn man die Liste jetzt lesbar haben will.

Eine gelöschte Registrierung hält niemanden fern: das Programm registriert sich beim nächsten Versuch neu und bekommt eine neue Kennung. Mit einer Ausnahme, die man kennen sollte: **ChatGPT** registriert sich nicht von selbst neu, sondern bleibt bei seiner gespeicherten Kennung und meldet einen Fehler — dort muss die Verbindung neu angelegt werden (siehe ChatGPT verbinden). Wer ein Programm *fernhalten* will, sperrt es — oder schaltet die Selbstregistrierung ab (`MCP_ALLOW_DCR=false`).

> **Das Zeitfenster beim Widerruf**
>
> Ein Widerruf wirkt auf die Verbindung **sofort**: das Programm kann sich keinen neuen Zugang mehr holen. Ein bereits ausgestelltes Zugriffstoken bleibt aber noch bis zu seinem Ablauf gültig — standardmäßig **15 Minuten** (`MCP_ACCESS_TOKEN_TTL_MINUTES`).
>
> Das ist der Preis zustandsloser Tokens, und es wird hier genannt statt verschwiegen. Wer das Fenster kleiner haben will, setzt den Wert herunter; das kostet häufigere Erneuerungen, sonst nichts. Wer den Zugang **augenblicklich** beenden muss, deaktiviert das Benutzerkonto oder entzieht die Mitgliedschaft — beides wirkt ohne Verzögerung, weil es bei jedem Aufruf frisch geprüft wird.

Datenschutz

Der entscheidende Punkt zuerst: **Was ein Modell über diese Schnittstelle liest, verlässt die Instanz und geht an den Anbieter des KI-Programms.** Das ist keine Nebenwirkung, sondern der Zweck — nur deshalb kann das Modell damit arbeiten.

Für den Betrieb heißt das:

- **Verantwortlich bleibt der Betreiber der Instanz.** Der Modellanbieter ist Auftragsverarbeiter; ein entsprechender Vertrag ist Sache des Betreibers, nicht von ConvoyPlan.
- **Personenbezug ist realistisch.** Fahrzeugbesatzungen mit Mobilnummer, Positionen mit Zeitstempel, Funkrufnamen — das sind personenbeziehbare Daten, sobald sie einer Person zugeordnet werden können.
- **Welche Daten fließen, entscheidet die Rolle.** Ein Zugang mit `convoy:read` für eine Organisation sieht genau das, was ein *beobachter* dort sieht — nicht weniger, aber auch nicht mehr.
- **Die Einwilligung ist dokumentiert.** Jede Zustimmung landet im Audit-Log mit Benutzer, Organisation, Programm und erteilten Rechten.

Wer das nicht will, lässt `MCP_ENABLED` auf `false` — dann existiert die Schnittstelle nicht.

Siehe auch: Sicherheit und Datenschutz, Rollen & Berechtigungen, API-Dokumentation.

Einstellungen

Variable	Standard	Bedeutung
`MCP_ENABLED`	`false`	Ausgangswert für die Schnittstelle. Der Schalter im Portal (System → KI-Schnittstelle) hat Vorrang und wirkt ohne Neustart.
`MCP_PUBLIC_URL`	aus `APP_BASE_URL`	Die Adresse, unter der Clients den Server erreichen. Ohne abschließenden Schrägstrich.
`MCP_ACCESS_TOKEN_TTL_MINUTES`	`15`	Gültigkeit eines Zugriffstokens — siehe Zeitfenster oben.
`MCP_REFRESH_TOKEN_TTL_DAYS`	`30`	Wie lange eine Verbindung ohne erneute Zustimmung hält.
`MCP_ALLOW_DCR`	`true`	Ob sich Programme selbst registrieren dürfen. Aus bedeutet: Clients von Hand eintragen.
`MCP_ALLOW_CIMD`	`false`	Ob Programme sich über eine hinterlegte Steckbrief-URL ausweisen dürfen — siehe unten.
`MCP_CIMD_CACHE_MINUTES`	`60`	Wie lange ein abgeholter Steckbrief gilt.
`MCP_TOOL_CALLS_PER_MINUTE`	`120`	Obergrenze je Verbindung. Ein Modell in einer Schleife ist ein realistisches Lastprofil.

Routenberechnungen zählen zusätzlich gegen `QUOTA_ROUTING_PER_HOUR` — dieselbe Einstellung wie an der REST-API.

Programme ohne Selbstregistrierung (CIMD)

Die Selbstregistrierung (`MCP_ALLOW_DCR`) hat einen bekannten Haken: **jedes** Programm, das die Instanz erreicht, kann sich eintragen und sich dabei nennen, wie es will. Deshalb steht der Name auf dem Zustimmungsbildschirm als *ungeprüft*.

Der Nachfolger heißt **Client ID Metadata Documents**. Statt sich einzutragen, nennt ein Programm als Kennung eine HTTPS-Adresse, unter der sein Steckbrief liegt — der Server holt ihn dort ab. Die Kennung ist damit selbst überprüfbar: sie sagt, wem das Programm gehört.

ChatGPT bevorzugt diesen Weg. Ohne ihn fällt es auf die Selbstregistrierung zurück, die weiterhin funktioniert — die Anbindung scheitert daran also nicht, sie läuft nur über die schwächere Kennung.

Das ist standardmäßig **aus**, und zwar bewusst. Eingeschaltet ruft der Server eine Adresse ab, **die der Client bestimmt** — die klassische Zutat für einen Angriff auf das interne Netz. Abgesichert ist das: nur HTTPS, keine internen, privaten oder Link-Local-Adressen (jede aufgelöste, nicht nur die erste), keine Weiterleitungen, harte Zeit- und Größengrenzen. Trotzdem bleibt es eine Entscheidung des Betreibers.

Der Schalter sitzt im Admin-Portal unter **System → KI-Schnittstelle**, direkt unter dem Hauptschalter, und wirkt ohne Neustart. `MCP_ALLOW_CIMD` in der `.env` ist nur noch der Ausgangswert — was im Portal eingestellt ist, schlägt ihn.

Zwei Dinge dazu:

- **Zudrehen wirkt sofort.** Die Prüfung steht vor dem Zwischenspeicher; ein bereits abgeholter Steckbrief hilft danach niemandem mehr.
- **Aufdrehen sieht ein Client unter Umständen verzögert.** Das Discovery-Dokument, in dem die Instanz CIMD ankündigt, darf eine Stunde zwischengespeichert werden. Wer gerade verbindet und nichts davon merkt, versucht es nach einer Stunde erneut.

Bestehende Verbindungen sind von beidem nicht betroffen — sie hängen an ihren Tokens, nicht am Ausweisweg.

Oberflächen in ChatGPT

Manche Programme können zu einer Antwort mehr zeigen als Text. ConvoyPlan liefert dafür drei Ansichten mit, die ChatGPT im Gespräch einblendet:

Ansicht	erscheint bei
Konvoi-Liste — Abmarschzeit, Umfang, Status je Konvoi	„Welche Konvois habe ich?"
Konvoi-Übersicht — Marschbefehl und Fahrzeuge in Marschordnung	„Zeig mir den Konvoi nach München"
Marschstatus — Zusammenfassung, Status und Mannschaftsstärke je Fahrzeug, Ausfälle zuerst	„Wie ist der Stand?"

Wie die Verbindung dorthin entsteht: ConvoyPlan in ChatGPT verbinden.

Drei Dinge, die dazugehören:

- **Der Server funktioniert ohne sie vollständig.** Ein Programm, das keine Oberflächen kennt — Claude Desktop etwa —, bekommt unverändert dieselbe Antwort wie bisher. Es gibt nichts, was nur in einer Ansicht steht.
- **Die Ansichten laden nichts nach.** Kein Skript, kein Zeichensatz, kein Bild von einer fremden Adresse. Das Fenster spannt ChatGPT auf, die Daten darin gehören der Organisation; was dort nachgeladen würde, säße als dritte Partei in genau dieser Sichtlinie.

- **Sie tragen keine eigenen Rechte.** Eine Ansicht ist eine leere Vorlage; die Daten kommen aus dem Werkzeugaufruf, und der geht durch dieselbe Prüfung wie jeder andere.

Wenn die Rechte nicht reichen

Versucht ein Programm etwas, wofür sein Zugang nicht genügt, bekommt es keine nichtssagende Fehlermeldung, sondern die Auskunft, **welches** Recht fehlt. Programme, die das beherrschen, fordern es daraufhin von sich aus nach — für den Benutzer heißt das: eine Nachfrage im Browser, kein Neu-Einrichten der Verbindung.

Nicht jedes Programm kann das. **ChatGPT etwa fragt einmalig beim Verbinden** und später nicht mehr. Deshalb lassen sich die Rechte schon auf dem Zustimmungsbildschirm ankreuzen: Wer weiß, dass er im Chat auch etwas anlegen oder ändern will, setzt dort gleich den Haken bei *Konvois und Fahrzeuge anlegen und ändern*. Ohne diesen Haken bleibt die Verbindung lesend, und der Versuch endet mit dem Hinweis, dass sie neu erteilt werden muss.

Was erteilt werden kann, deckelt in jedem Fall die Rolle. Ein *beobachter* bekommt `convoy:write` nicht — weder durch Ankreuzen noch durch Nachfordern, und auch dann nicht, wenn das Programm ausdrücklich danach fragt.

Technischer Hintergrund

Die Instanz ist zugleich **Resource Server und Authorization Server**: eine selbst gehostete Installation hat keinen externen Identitätsanbieter, und Benutzer, MFA, Organisationen und Rollen liegen ohnehin in der eigenen Datenbank.

Umgesetzt nach der MCP-Revision **2026-07-28**: Protected Resource Metadata nach RFC 9728 an der Wurzel der Site, PKCE ausschließlich mit `S256`, Resource Indicators nach RFC 8707 (ein Zugang für Instanz A ist an Instanz B wertlos) und der `iss`-Parameter nach RFC 9207 in jeder Antwort — auch in einer Absage, damit ein Programm den Absender prüfen kann.

Zugänge erneuern sich rollierend. Taucht ein bereits erneuertes Token noch einmal auf, hat es jemand mitgelesen — dann wird die **gesamte** Verbindung entzogen, nicht nur das vorgelegte Token.

Die Live-Abonnements laufen über `subscriptions/listen`. Die Adresse einer abonnierbaren Resource trägt die Organisation in sich (`convoyplan://org/<org>/konvoi/<id>/live`) — daran entscheidet der Server, wer eine Benachrichtigung bekommt. Das ist nicht kosmetisch: das MCP-SDK nimmt jede angefragte Adresse ohne Rückfrage an, die Trennung nach Organisationen muss also aus der Form der Adresse selbst kommen. Ein Programm, das die Adresse eines fremden Konvois errät, erfährt davon nichts.

KAPITEL 11

ConvoyPlan in ChatGPT verbinden

Diese Seite beschreibt, wie eine ConvoyPlan-Instanz an ChatGPT angebunden wird, sodass man im Gespräch nach Konvois fragen und sie auch ändern kann.

Der Weg heißt Developer Mode. Das ist keine Notlösung: ConvoyPlan wird selbst betrieben, jede Organisation hat ihre eigene Adresse. Ein Eintrag im öffentlichen App-Verzeichnis von ChatGPT könnte immer nur auf **eine** Adresse zeigen und wäre für alle anderen Instanzen wertlos. Der Developer Mode verbindet ChatGPT mit **deiner** Instanz.

> Für Claude Desktop, Claude Code und andere Programme siehe MCP-Server — dort steht der allgemeine Weg.

Voraussetzungen

Auf der ChatGPT-Seite:

- Ein Konto mit **Pro, Plus, Business, Enterprise oder Education**. Mit einem kostenlosen Konto lässt sich kein eigener Server verbinden.
- Die Einrichtung läuft **im Browser**, nicht in der App.

Auf der ConvoyPlan-Seite:

- Die Instanz ist aus dem Internet über **HTTPS** erreichbar. Eine Instanz, die nur im internen Netz läuft, erreicht ChatGPT nicht — es verbindet sich von seinen eigenen Servern aus, nicht vom Rechner des Benutzers.
- Die **KI-Schnittstelle ist eingeschaltet**: Adminportal → System → KI-Schnittstelle. Siehe MCP-Server.
- Du bist Mitglied einer Organisation. Welche Rolle du dort hast, entscheidet, was die Verbindung darf — siehe Rollen & Berechtigungen.

Schritt 1: Developer Mode einschalten

In ChatGPT: **Einstellungen → Sicherheit und Anmeldung → Developer mode** einschalten.

Je nach Oberflächenstand liegt der Schalter auch unter *Apps & Connectors → Erweitert*. Gesucht ist in beiden Fällen dasselbe: die Erlaubnis, eigene Verbindungen zu Servern anzulegen, die nicht aus dem Verzeichnis stammen.

Schritt 2: Die Verbindung anlegen

1. Im Adminportal von ConvoyPlan unter **MCP** die **Adresse kopieren**. Sie sieht so aus:

   ```
   https://<deine-domain>/mcp
   ```

   Die Adresse aus dem Portal zu kopieren statt sie zu tippen erspart den häufigsten Fehler: **das `/mcp` am Ende fehlt**. Ohne diesen Pfad findet ChatGPT nichts und meldet einen Verbindungsfehler.
2. In ChatGPT unter **Apps & Connectors** eine neue Verbindung anlegen (Plus-Knopf).

3. Ausfüllen:

Feld	Wert
Name	ConvoyPlan (oder der Name deiner Organisation)
Beschreibung	frei — was du im Auswahlmenü lesen willst
Server-URL	die kopierte Adresse mit `/mcp`
Authentifizierung	**OAuth**

4. Speichern. ChatGPT ruft die Schnittstelle ab und liest die verfügbaren Werkzeuge ein.

Schritt 3: Zustimmen — hier entscheidet sich alles

ChatGPT schickt dich zur Anmeldung auf deiner ConvoyPlan-Instanz. Nach dem Anmelden (inklusive MFA, falls eingerichtet) erscheint der Zustimmungsbildschirm. **Drei Dinge dort sind wichtig:**

Die Organisation

Der Zugang gilt für **genau eine** Organisation. Gehörst du mehreren an, wähle die richtige — ein Wechsel später bedeutet, die Verbindung neu zu erteilen.

Die Rechte — Haken setzen, sonst bleibt es beim Lesen

Vorausgewählt ist nur, was ChatGPT von sich aus angefragt hat, und das ist **nur Lesen**. Wer im Chat auch etwas anlegen oder ändern will, setzt hier den Haken bei **„Konvois und Fahrzeuge anlegen und ändern"**.

> **Das lässt sich später nicht nachholen.** Andere Programme fragen bei Bedarf nach; ChatGPT fragt einmal beim Verbinden und danach nie wieder. Ohne den Haken endet jeder Änderungsversuch mit einer Fehlermeldung, und der einzige Ausweg ist, die Verbindung zu trennen und neu zu erteilen.

Angeboten wird nur, was deine Rolle hergibt. Als *Beobachter* gibt es kein Schreibrecht zum Ankreuzen — das ist kein Fehler der Anbindung.

Die Zieladresse

Auf dem Bildschirm stehen der **Name des Programms** (als *ungeprüft* gekennzeichnet — den hat es sich selbst gegeben) und die **Zieladresse** (als *geprüft*). Verlässlich ist nur die Zieladresse. Sie muss zu ChatGPT passen, und du musst die Verbindung gerade selbst gestartet haben.

Schritt 4: Im Gespräch einschalten

Die Verbindung ist nicht automatisch in jeder Unterhaltung aktiv. Im Eingabefeld über das Plus-Menü **Developer mode** wählen und ConvoyPlan ankreuzen. Das gilt je Unterhaltung — wer das vergisst, bekommt eine Antwort aus dem Allgemeinwissen des Modells statt aus seinen Konvoidaten.

Das ist die zweithäufigste Ursache für „es funktioniert nicht".

Was du dann fragen kannst

Lesend:

- „Welche Konvois stehen nächste Woche an?"
- „Zeig mir den Konvoi nach München."
- „Welche Fahrzeuge sind dem Marschverband Nord zugeordnet?"
- „Wie ist der Stand? Wer ist noch nicht da?"
- „Wann ist Abmarsch und wo ist der Ablaufpunkt?"

Schreibend (nur mit gesetztem Haken aus Schritt 3):

- „Lege einen Konvoi ‚Übung Nord' für Samstag 06:30 an."
- „Füge das ELW 1 hinzu."
- „Setz Florian 4 auf technischen Halt, Reifenschaden."
- „Berechne die Route."

ChatGPT blendet zu manchen Antworten eine **eigene Ansicht** ein: die Konvoi-Liste, die Konvoi-Übersicht mit Marschbefehl und Fahrzeugen in Marschordnung, oder den Marschstatus mit den Ausfällen zuoberst.

Schreibende Aktionen

ChatGPT fragt vor jeder schreibenden Aktion nach und zeigt, was es senden will. **Diese Anzeige lohnt sich zu lesen** — ein Modell kann eine Anweisung missverstehen, und was es dann anlegt oder ändert, steht genau dort.

Die Option, eine Zustimmung dauerhaft zu merken, sollte man sparsam benutzen. Sie ist bequem für „Status setzen" und riskant für alles, was einen Marschbefehl verändert.

Was nicht passieren kann: Kein Werkzeug der Schnittstelle löscht Daten. Es gibt keines, das einen Konvoi, ein Fahrzeug, einen Wegpunkt oder eine Route entfernt. „Fahrzeug aus Konvoi entfernen" löst nur die Zuordnung — Fahrzeug und Konvoi bleiben bestehen, und der Schritt lässt sich rückgängig machen.

Was das für die Daten bedeutet

Was ein Modell liest, verlässt die Instanz und geht an OpenAI. Das gilt für jeden Konvoi, jedes Fahrzeug und jede Position, nach der im Gespräch gefragt wird. Für eine BOS-Organisation ist das die Entscheidung, die vor der technischen Einrichtung steht — nicht danach.

Wer sie trifft, sollte wissen:

- Der Zugang ist auf **eine** Organisation und auf die Rolle des zustimmenden Benutzers begrenzt.
- Jeder **schreibende** Aufruf steht im Audit-Log, mit Benutzer, Werkzeug und dem Programm, das ihn ausgelöst hat.
- Lesende Aufrufe werden nicht protokolliert — ein Modell liest im Minutentakt.
- Der Betreiber kann die Schnittstelle jederzeit abschalten oder einzelne Verbindungen trennen.

Einzelheiten unter Sicherheit und Datenschutz und MCP-Server → Datenschutz.

Wenn es nicht funktioniert

Symptom	Ursache	Abhilfe
ChatGPT findet den Server nicht	`/mcp` fehlt in der Adresse	Adresse aus dem Adminportal kopieren

Symptom	Ursache	Abhilfe
„Verbindung fehlgeschlagen", Instanz läuft aber	Instanz nicht aus dem Internet erreichbar	ChatGPT verbindet sich von eigenen Servern aus, nicht vom Browser
Seite meldet 404 beim Anmelden	KI-Schnittstelle ist abgeschaltet	Adminportal → System → KI-Schnittstelle
Antworten kommen ohne Konvoidaten	Verbindung in dieser Unterhaltung nicht eingeschaltet	Plus-Menü → Developer mode → ConvoyPlan ankreuzen
Jede Änderung scheitert an fehlenden Rechten	Haken beim Zustimmen nicht gesetzt	Verbindung trennen und neu erteilen, diesmal mit Haken
Kein Schreibrecht zum Ankreuzen	Rolle gibt es nicht her	Rolle in der Organisation prüfen (Rollen)
Schreibende Werkzeuge fehlen ganz	Instanz ohne gültige Lizenz	Lizenz und Demo-Modus
„Zu viele Werkzeugaufrufe"	Modell in einer Schleife	kurz warten; die Grenze steht im Adminportal
„Client ID … not found" nach der Weiterleitung	Die Registrierung ist im Adminportal gesperrt oder als verwaist entfernt worden	Verbindung in ChatGPT **löschen und neu anlegen**
Fehler bleibt, obwohl die Verbindung bearbeitet wurde	ChatGPT behält beim Bearbeiten seine gespeicherte Kennung	Löschen und neu anlegen — nur dabei registriert es sich neu

Hilft nichts davon, sagt die Instanz selbst, was sie kann: frag im Gespräch „Mit welcher Organisation bist du verbunden und was darfst du?" — die Antwort nennt Organisation, Rolle, erteilte Rechte und die verfügbaren Werkzeuge.

Zugang wieder entziehen

- **In ChatGPT:** die Verbindung unter Apps & Connectors löschen.
- **In ConvoyPlan:** Adminportal → Reiter **MCP** → Verbindung trennen. Das entwertet die Zugriffstoken sofort und wirkt auch dann, wenn in ChatGPT noch etwas eingetragen ist.

Der zweite Weg ist der verlässliche: Er liegt in der Hand des Betreibers.

> **Der Papierkorb im Portal sperrt, er löscht nicht.** ChatGPT merkt sich seine Kennung aus der Registrierung und benutzt sie stur weiter — auf eine gesperrte oder inzwischen aufgeräumte Kennung antwortet die Instanz mit „Client ID … not found". Anders als andere Programme registriert ChatGPT sich dann **nicht** von selbst neu. Wer die Verbindung wiederhaben will, legt sie in ChatGPT neu an, statt die bestehende zu bearbeiten.

Was (noch) nicht geht

- **ConvoyPlan im öffentlichen App-Verzeichnis von ChatGPT.** Ein Verzeichniseintrag nennt genau eine Serveradresse; bei einer selbst gehosteten Software nützt das niemandem außer dem Betreiber genau dieser Adresse. Der Developer Mode bleibt deshalb der Weg.
- **Fahrer oder Teilnehmer zuordnen.** ConvoyPlan kennt keine Personen, die einem Fahrzeug zugeordnet wären — „Fahrer" ist eine Rolle in der Organisation, keine Zuordnung. Solche Fragen kann die Schnittstelle nicht beantworten.

- **Etwas löschen.** Bewusst nicht vorgesehen, siehe oben.

KAPITEL 12

Auskunft für Suchmaschinen und KI-Agenten

Eine ConvoyPlan-Instanz liefert unter ihrer eigenen Domain aus, **was sie ist** und **wie man sie programmatisch anspricht**. Das ist kein Marketing-Anhängsel: ein Sprachmodell, das nach einer Konvoiplanung gefragt wird, und ein Agent, der auf einer Instanz landet, brauchen beide dieselbe Auskunft, und beide lesen sie lieber als Text als aus einer Single-Page-App, die erst Javascript starten muss.

Alles hier beschreibt die **Software**, nicht ihren Inhalt. Es gibt keinen Pfad in dieser Liste, der Daten einer Organisation zeigt. Wer trotzdem nichts über die Instanz verraten will, schaltet sie mit `AGENT_DISCOVERY=false` ab — siehe unten. Auf einer selbst gehosteten Instanz (`INSTANCE_MODE=selfhost`, der Standard) ist sie von vornherein aus.

Was ausgeliefert wird

Alle Pfade sind relativ zur Adresse der Instanz (`https://<DOMAIN>/…`).

Einstiege

Pfad	Inhalt
`/llms.txt`	Der Navigationsindex. Beginnt mit dem, was ConvoyPlan ist, und sagt dann ausdrücklich, **wann ein Agent ConvoyPlan aufrufen soll und wann nicht**
`/llms-full.txt`	Dieselbe Auskunft als Langfassung, alle Dokumente hintereinander
`/agents.md`	Kurzanleitung für KI-Agenten: Fähigkeiten, nötige Scopes, Regeln
`/auth.md`	Wie ein Agent an Zugangsdaten kommt, nach der auth.md-Spezifikation gegliedert
`/api.md`	Die REST-Oberfläche in Prosa: Endpunkte, Fehlercodes, Grenzen
`/openapi.json`	OpenAPI 3.1 der **anmeldefreien** Endpunkte plus der Sicherheitsschemata
`/ask`	Fragen über ConvoyPlan, beantwortet mit passenden Seiten (NLWeb, `GET` und `POST`, auf Wunsch als SSE-Stream)

Abschnittsweise gibt es zusätzlich `/api/llms.txt`, `/docs/llms.txt` und `/developers/llms.txt` — für einen Agenten, der nur einen Teilbereich braucht.

Well-Known-Dokumente

Pfad	Standard
`/.well-known/agent-card.json`	A2A Agent Card
`/.well-known/agent-skills/index.json`	Agent-Skills-Index
`/.well-known/ard.json`	Agentic Resource Discovery
`/.well-known/mcp/server-card.json`	Kurzbeschreibung des MCP-Servers
`/.well-known/api-catalog`	API-Katalog nach RFC 9727
`/.well-known/oauth-protected-resource`	RFC 9728 für die REST-API

Pfad	Standard
`/.well-known/oauth-protected-resource/mcp`	RFC 9728 für den MCP-Server — **nur bei eingeschalteter KI-Schnittstelle**

Seiten

`/about`, `/pricing`, `/developers`, `/docs`, `/contact` und `/privacy` beschreiben das Produkt, den Betrieb, die Lizenzmodelle und die Schnittstellen. Sie stehen hinter keiner Anmeldung, weil sie genau denen etwas sagen sollen, die noch keine haben.

Dazu kommen `/robots.txt` (mit `Sitemap:` - und `Schemamap:` -Verweis und einer ausdrücklichen Erlaubnis für die großen KI-Crawler), `/sitemap.xml`, `/schema-map.xml` und die Feeds unter `/feeds/`.

Markdown statt HTML

Jede Seite hat eine Markdown-Fassung, und beide entstehen aus **derselben Quelle** (`frontend/src/lib/server/agent/documents.ts`). Sie können deshalb nicht auseinanderlaufen.

Drei Wege führen dorthin:

```
curl https://<DOMAIN>/about.md                          # direkt
curl -H 'Accept: text/markdown' https://<DOMAIN>/about  # ausgehandelt
curl https://<DOMAIN>/about?mode=agent                  # maschinenlesbare Übersicht
```

Die HTML-Fassung kündigt die Markdown-Fassung über `<link rel="alternate" type="text/markdown">` und über einen `Link`-Kopf (RFC 8288) an, der außerdem auf Sitemap, `llms.txt`, OpenAPI und den API-Katalog zeigt.

Nicht nach User-Agent unterschieden: ein Crawler bekommt dieselbe Seite wie ein Mensch. Unterschiedliche Inhalte je nach Kennung wären Cloaking, und seit die Seiten ihren Inhalt serverseitig ausliefern, hätte ein Bot davon auch nichts.

Ein Pfad, den es nicht gibt, antwortet mit `404` — auf Wunsch als Markdown mit Wegweiser statt als leere App-Hülle.

Was die Auskunft über die Instanz sagt

Die Dokumente werden **je Anfrage** aus der tatsächlichen Lage der Instanz gebaut: unter welcher Domain sie läuft, ob der MCP-Server montiert ist, ob ein Demo-Zugang freigeschaltet ist. Grundlage ist `GET /api/status/capabilities` (ohne Anmeldung abfragbar).

Das ist der Grund für den Aufwand: eine Instanz unter `feuerwehr.example` darf nicht auf `convoyplan.de` verweisen, und ein `server-card.json`, das einen MCP-Endpunkt ankündigt, den der Betreiber abgeschaltet hat, wäre schlechter als gar keines. Ist die KI-Schnittstelle aus, sagen die Dokumente das — und nennen den REST-Weg.

Die öffentliche OpenAPI-Beschreibung

`/openapi.json` ist **nicht** die vollständige Beschreibung. Die bleibt in Produktion abgeschaltet und wird per `DOCS_API_KEY` geschützt (siehe API-Dokumentation). Veröffentlicht wird eine kuratierte Teilmenge:

- nur Endpunkte, die ohnehin ohne Anmeldung erreichbar sind,
- dazu die Sicherheitsschemata (`bearerAuth`, `apiKeyAuth`, `oauth2`), damit ein Agent erfährt, wie er an ein Token kommt,

- mit eigenen Beschreibungen statt der Docstrings aus dem Quelltext, und
- nur mit den Schemata, die diese Endpunkte wirklich brauchen.

Struktur (Parameter, Antwortschemata) kommt aus der laufenden App und kann deshalb nicht veralten; die Liste der öffentlichen Operationen steht ausdrücklich in `backend/app/api/routes/public_meta.py` und wird von `tests/test_public_openapi.py` in beide Richtungen geprüft.

Werkzeuge im Browser (WebMCP)

Die Startseite meldet vier **lesende** Werkzeuge an, wenn der Browser das Modell kennt: Produktauskunft, Seitenverzeichnis, Instanzstatus und die Prüfung eines Organisations-Codes. Schreibende Werkzeuge gibt es dort bewusst nicht — dafür ist der MCP-Server da, mit ausdrücklicher Zustimmung und Audit-Log.

Ein- und abschalten

Ohne Angabe folgt die Auskunft der **Betriebsart**: an bei `INSTANCE_MODE=hosting`, aus bei `INSTANCE_MODE=selfhost` (Standard). Eine Feuerwehr, die ConvoyPlan für sich betreibt, braucht keine `/pricing`-Seite und keine Agent-Card, die Suchmaschinen auf ihre Einsatzinstanz hinweist. Ausdrücklich gesetzt, übersteuert der Schalter das:

```
AGENT_DISCOVERY=false   # oder true
```

Abgeschaltet gibt es keinen der oben genannten Pfade (404), und die Startseite zeigt nur die Anmeldung. Der Rest der Anwendung ist davon unberührt.

> **Bis 2026-10** war die Auskunft überall an. Wer sie auf einer selbst gehosteten Instanz behalten will, setzt nach dem Update `AGENT_DISCOVERY=true` in die `.env`.

Was nicht im Code liegt

Ein Teil dessen, was KI-Suchen über ein Produkt finden, entsteht außerhalb der Software: ein Wikidata-Eintrag, Presseerwähnungen, Einträge in App-Verzeichnissen, veröffentlichte SDK-Pakete. Ebenso entscheidet ein vorgelagertes CDN oder eine WAF darüber, ob ein KI-Crawler die Instanz überhaupt erreicht — die `robots.txt` erlaubt es, die Firewall kann es trotzdem verhindern.

Verwandt

- MCP-Server (KI-Schnittstelle)
- API-Dokumentation
- Sicherheit und Datenschutz

TEIL 4

Entwicklung

Entwicklung

KAPITEL 13

Entwicklung

Diese Seite beschreibt die Projektstruktur und den lokalen Entwicklungs-Workflow. Installation auf einem Server und alle Umgebungsvariablen stehen unter Installation und Setup dokumentiert.

Projektstruktur

```
ConvoyPlan/
├── backend/
│   ├── app/
│   │   ├── api/routes/       # REST- und WebSocket-Endpunkte
│   │   │   ├── auth.py       # Registrierung, Login, MFA, Passwort-Reset
│   │   │   ├── setup.py      # Ersteinrichtungs-Wizard (Status + Ausführen)
│   │   │   ├── admin.py      # Superadmin: Benutzer, Orgs, API-Keys, Audit-Log, DSGVO
│   │   │   ├── leitstellen.py  # Globale Leitstellen und Kanalwechsel
│   │   │   ├── org_leitstellen.py  # Org-eigene Leitstellen + Vorschlags-Workflow
│   │   │   ├── branding.py   # Branding-Konfiguration und Logo-Upload
│   │   │   ├── email_template.py  # E-Mail-Vorlagen (SMTP-Versand)
│   │   │   ├── license.py    # Lizenzstatus, Aktivierung, Instanz-UUID
│   │   │   ├── convoys.py    # Marschverbände, Waypoints, Export
│   │   │   ├── share_links.py  # Widerrufbare Freigabelinks pro Konvoi
│   │   │   ├── track.py      # Eigenständige Tracking-App (REST + WebSocket)
│   │   │   ├── vehicles.py   # Fahrzeugverwaltung
│   │   │   ├── organizations.py  # Organisationen und Mitglieder
│   │   │   ├── tracking.py   # Live-Positionen + WebSocket
│   │   │   ├── routing.py    # GraphHopper-Routing
│   │   │   ├── weather.py    # Open-Meteo-Integration
│   │   │   ├── overpass.py   # Sperrungsabfragen (Overpass, Autobahn-API, offene Feeds, DATEX-II),
Punkt und Routenkorridor
│   │   │   ├── traffic.py    # Live-Verkehrslage (HERE/TomTom), Punkt und Routenkorridor
│   │   │   ├── users.py      # Eigenes Benutzerprofil
│   │   │   ├── version.py    # Versions- und Build-Informationen
│   │   │   └── status.py     # Systemstatus
│   │   ├── models/           # SQLAlchemy-Modelle
│   │   ├── schemas/          # Pydantic-Schemas
│   │   ├── services/         # Routing, Zeitplan, Export, Wetter, Overpass/Autobahn/Traffic,
Tracking, Retention
│   │   ├── jobs/             # Hintergrundjobs (z. B. Retention-Purge)
│   │   ├── config.py         # Backend-Konfiguration über Umgebungsvariablen
│   │   ├── database.py       # Async-Datenbankanbindung
│   │   └── main.py           # FastAPI-App und Router-Registrierung
│   ├── alembic/              # Datenbankmigrationen (0001–0026)
│   ├── tests/                # pytest-Tests
│   ├── Dockerfile
│   └── requirements.txt
├── frontend/
│   ├── src/lib/api/          # API-Client
│   ├── src/lib/components/   # Karte, Wetter, Leitstellen-Tabelle/-Karte, UI-Bausteine
│   ├── src/lib/stores/       # Auth-, Karten-, Konvoi-, Tracking-, Org-, Branding-Stores
│   ├── src/routes/
│   │   ├── setup/            # Ersteinrichtungs-Wizard (5 Schritte inkl. Org-Anlage)
│   │   ├── admin/            # Superadmin (self-gated): Benutzer, Orgs, API-Keys, Leitstellen,
Branding, System
│   │   ├── o/[slug]/         # Org-Scope (alle org-spezifischen Routen)
```

```
│   │   │   ├── login/        # Org-spezifische Anmeldung mit eigenem Branding
│   │   │   ├── plan/         # Planungsansicht mit Karte
│   │   │   ├── tracking/     # Live-Tracking-Ansicht (Übersicht + je Konvoi)
│   │   │   └── admin/        # Org-Admin: Mitglieder, Leitstellen, Branding, System
│   │   ├── track/[slug]/     # Eigenständige Fahrer-Tracking-PWA („Convoy Tracking")
│   │   └── share/[token]/    # Öffentliche Routenansicht ohne Login
│   ├── static/geo/           # Offline-Landkreisgrenzen (GeoJSON) für Leitstellengebiete
│   ├── capacitor.config.ts
│   ├── package.json
│   └── vite.config.ts
├── docker/updater/           # Git-Polling-Container (update.sh, Dockerfile)
├── caddy/
│   └── entrypoint.sh         # Caddyfile-Generierung aus Env-Variablen
├── graphhopper/
│   ├── Dockerfile
│   ├── entrypoint.sh         # OSM-Download und GraphHopper-Start
│   └── config.yml
├── .github/
│   ├── workflows/
│   │   ├── ci.yml            # Tests + Typecheck + Docker-Build + Dependency-Audit
│   │   ├── release.yml       # Docker-Images zu GHCR + GitHub Release bei Tag
│   │   └── sync-wiki.yml     # Spiegelt wiki/ ins GitHub Wiki bei Push auf main
│   └── dependabot.yml        # Dependency-Updates (pip/npm/Actions/Docker)
├── docs/                     # Backup/Restore, Retention- und ISO-Dokumentation
├── wiki/                     # Wiki-Quellen (Markdown) — Sync ins GitHub Wiki via sync-wiki.yml
├── .hooks/pre-commit         # Lokaler Pre-Commit-Hook (ruff + svelte-check)
├── scripts/
│   ├── install.sh            # Linux-Installer (+ systemd-Watchdog)
│   ├── install.ps1           # Windows-Installer
│   ├── backup.sh             # PostgreSQL-Dump + Volumes + Prüfsummen + Retention
│   ├── restore.sh            # Wiederherstellung aus Backup
│   ├── updater-watchdog.sh   # Räumt verwaiste Updater-Container auf
│   ├── deploy.sh             # Einmalige Server-Migration / Notfall-Deploy
│   └── install-hooks.sh      # Installiert .hooks/ in .git/hooks/
├── logo/                     # Logo-, Favicon- und Design-Assets
├── CHANGELOG.md              # Versionshistorie
├── RELEASING.md              # Anleitung zum Schneiden eines Releases
├── SECURITY.md               # Vulnerability-Disclosure-Richtlinie
├── .env.example              # Alle Umgebungsvariablen mit Erklärungen
└── docker-compose.yml        # Compose-Datei für Entwicklung und Produktion (wird vom Installer
genutzt)
```

Backend lokal entwickeln (ohne Docker)

```
cd backend
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
alembic upgrade head
uvicorn app.main:app --reload
```

> PostgreSQL/PostGIS und GraphHopper müssen erreichbar sein — am einfachsten weiterhin über Docker Compose.

Tests ausführen

```
cd backend
pytest
```

Frontend prüfen und bauen

```
cd frontend
npm install
npm run check
npm run build
```

Datenbankmigrationen

```
# Aktuelle Migrationen ausführen
cd backend && alembic upgrade head

# Neue Migration erzeugen
cd backend && alembic revision --autogenerate -m "beschreibung"
```

Nützliche Docker-Befehle

Die `docker-compose.yml` hat keine `build:`-Abschnitte, sondern zieht die Images aus GHCR. `--build` hat deshalb keine Wirkung. Wer einen geänderten Stand im Stack ausprobieren will, baut das Image selbst und zeigt in der `.env` darauf:

```
docker build -t convoyplan-backend:dev backend
echo "BACKEND_IMAGE=convoyplan-backend:dev" >> .env   # analog FRONTEND_IMAGE, GRAPHHOPPER_IMAGE
docker compose up -d backend
```

Die `.env` vorher aus `.env.example` anlegen und mindestens `POSTGRES_PASSWORD` und `JWT_SECRET` setzen (oder `APP_ENV=development`).

```
docker compose up -d                # Stack starten
docker compose logs -f backend      # Backend-Logs anzeigen
docker compose logs -f graphhopper  # GraphHopper-Logs anzeigen
docker compose down                 # Services stoppen
docker compose down -v              # inklusive persistenter Daten
```

Pre-Commit-Hook

Der lokale Pre-Commit-Hook (`ruff` + `svelte-check`) lässt sich einmalig installieren:

```
./scripts/install-hooks.sh
```

Native App / PWA

Das Frontend ist als Progressive Web App konfiguriert und kann im Browser installiert werden. Für native Apps ist Capacitor vorbereitet:

```
cd frontend
npm run build
npx cap add android   # einmalig, alternativ: ios
npx cap sync
npx cap open android
```

> Für iOS wird eine macOS-Umgebung mit Xcode benötigt.

CI und Releases

CI-Checks (Backend-Tests, Frontend-Typecheck, Docker-Build, Dependency-Audit) laufen automatisch auf Push und Pull Requests gegen `main`.

Die Versionierung folgt dem kalenderbasierten Schema **`YYYY.MASTER.FIX`** (z. B. `2026.1.1`): `YYYY` ist die Jahreszahl, `MASTER` das Master-Release (größere Feature-Veröffentlichung innerhalb des Jahres) und `FIX` das Fix-/Beta-Release (Bugfixes, Sicherheits-Patches, Dependabot-Wellen). Das Schema löst die frühere semantische Versionierung ab; das erste Release nach `1.0.2` ist `2026.1.1`.

Für ein neues Release den Tag `vYYYY.MASTER.FIX` (z. B. `v2026.1.1`) setzen — Docker-Images werden dann automatisch zu GHCR gebaut und gepusht; ein GitHub Release wird erstellt. Prerelease-Tags (`vX.Y.Z-beta.N`) bauen zusätzlich `:beta`-Images und ein GitHub-Prerelease, ohne `:latest` zu berühren; jeder Push auf `main` baut außerdem `:nightly`-Images. Vollständige Anleitung in `RELEASING.md`.