FOCUS FAMILY

Technisches Handbuch

Server betreiben, absichern und administrieren

Version 1.0.0 · Stand September 2026 · Software Notion

Inhaltsverzeichnis

Über dieses Handbuch

Dieses Handbuch richtet sich an **IT-Administration und technische Dienstleister**, die den Focus-Family-Server eines Trägers betreiben. Es beschreibt Architektur, Installation, Konfiguration, Absicherung, Sicherung, Aktualisierung und Fehlersuche – und die technischen Bereiche des Admin-Panels.

Die fachliche Bedienung des Panels steht im **Handbuch für das Admin-Panel**, die Bedienung der Apps in den beiden App-Handbüchern.

Vorausgesetzte Kenntnisse: Linux-Grundlagen, Docker und Docker Compose, ein Reverse-Proxy (nginx, Caddy oder Traefik) sowie der Umgang mit TLS-Zertifikaten.

Grundsätze des Betriebsmodells

- **Ein Träger betreibt eine Serverinstanz** in eigener Infrastruktur. Es gibt keine Hersteller-Cloud.
- Der Stack benötigt **keine Verbindung** zu Analyse-, Crash- oder KI-Diensten des Herstellers.
- Der einzige optionale externe Kontakt ist der **Lizenzserver** (fflicense) für die Organisationslizenz.
- Die Apps funktionieren **vollständig offline**; der Server ist für Teamarbeit und Verwaltung zuständig.

Architektur

Der Stack besteht aus vier Containern:

Dienst	Aufgabe
db	PostgreSQL 17, nicht öffentlich erreichbar, benanntes Volume `pgdata`
migrator	One-Shot beim Start: legt das Schema an bzw. wendet ausstehende Schema-Migrationen an und provisioniert die App-Rolle
clamav	Malware-Scan aller Uploads, benanntes Volume `clamdb` für die Signaturen
app	der Server selbst, gebunden an `127.0.0.1:8080`, hinter dem Reverse-Proxy des Trägers

Datenhaltung und Volumes

Volume	Inhalt	Folge bei Verlust
`pgdata`	PostgreSQL-Daten	vollständiger Datenverlust
`blobs`	Anhänge, ruhend verschlüsselt	Dokumente und Medien verloren
`dpkeys`	ASP.NET-Data-Protection-Keyring	alle Sitzungen und Antiforgery-Token ungültig
`clamdb`	ClamAV-Signaturen	werden neu geladen (unkritisch)
`licstate`	Identität der Lizenz-Installation	Neu-Provisionierung beim Lizenzserver nötig
`deploy/docker/keys`	Signier- und Verschlüsselungszertifikate (read-only)	Anmeldung nicht mehr möglich

> **Merksatz:** `pgdata`, `dpkeys` und `licstate` bilden zusammen die **Identität der Installation**. Werden sie gelöscht, wird aus einer Wiederherstellung eine Neu-Provisionierung.

Rollen in der Datenbank

- **`fokus_migrator`** besitzt das Schema und darf DDL ausführen. Diese Rolle nutzt ausschließlich der Migrator-Container.
- **`fokus_app`** betreibt die Anwendung und hat **nur DML-Rechte**; `CREATE TABLE` und `DROP` sind verweigert.

Die Anwendung selbst führt **nie** DDL aus. Schemaänderungen laufen ausschließlich über den kontrollierten Migrator-Schritt.

Voraussetzungen

- **Server:** Linux (amd64/x86_64), Docker Engine mit Compose v2.
- **Empfehlung:** 4 CPU-Kerne, 8 GB RAM, 50 GB Speicher als Ausgangspunkt. ClamAV allein braucht etwa 1,5–2 GB RAM.
- **Netzwerk:** eine Domain mit DNS-Eintrag auf den Server, ein TLS-Zertifikat (institutionelle PKI oder Let's Encrypt) am Reverse-Proxy.
- **Zertifikate:** produktive **Signatur- und Verschlüsselungszertifikate** im PKCS#12-Format für die Token-Ausstellung.
- **Lizenz:** ein **Enrollment-Key** vom Betreiber des Lizenzservers, falls eine Organisationslizenz genutzt wird.

> In einer Entwicklungsumgebung startet der Server mit ephemeren Schlüsseln. In Produktion sind die Zertifikate **Pflicht** – ohne sie startet die Anwendung bewusst nicht.

Installation Schritt für Schritt

1. Dateien auf den Server bringen

Kopieren Sie das Deployment-Verzeichnis (Compose-Datei, `.env.example`, `keys/`, Proxy-Beispiele, Backup-Skripte) auf den Server, zum Beispiel nach `/opt/focusfamily`.

Das Server-Image wird **einmalig auf einem Build-Rechner** erzeugt und als Tarball mitgeliefert – auf dem Server ist kein .NET-SDK nötig:

```
docker load -i image/fokusfamilie-server-1.0.0-amd64.tar.gz
```

2. Konfiguration anlegen

```
cp .env.example .env
uuidgen                     # Wert als SERVER_INSTANCE_ID eintragen
nano .env                   # jedes CHANGE_ME ersetzen
```

3. Zertifikate erzeugen

```
mkdir -p deploy/docker/keys
openssl req -x509 -newkey rsa:2048 -nodes -keyout /tmp/k.pem -out /tmp/c.pem \
  -days 825 -subj "/CN=fokusfamilie"
openssl pkcs12 -export -inkey /tmp/k.pem -in /tmp/c.pem \
  -out deploy/docker/keys/signing.pfx    -passout pass:"$CERT_PASSWORD"
openssl pkcs12 -export -inkey /tmp/k.pem -in /tmp/c.pem \
  -out deploy/docker/keys/encryption.pfx -passout pass:"$CERT_PASSWORD"
```

Für echte Deployments verwenden Sie die institutionelle PKI statt eines selbstsignierten Zertifikats.

4. Signierschlüssel für Offline-Grants

```
{ printf '\nOFFLINE_GRANT_KEY="'; openssl ecparam -genkey -name prime256v1 -noout; printf
'"\n'; } >> .env
```

Bleibt der Wert leer, erzeugt der Server bei jedem Start einen flüchtigen Schlüssel und protokolliert eine Warnung – Offline-Grants überleben dann keinen Neustart.

5. Ersten Start mit Bootstrap

Für den allerersten Start aktivieren Sie den einmaligen Bootstrap in der `.env`:

```
BOOTSTRAP_ENABLED=true
BOOTSTRAP_ADMIN_EMAIL=admin@ihr-traeger.de
BOOTSTRAP_ADMIN_PASSWORD=<Einmal-Passwort>
```

```
docker compose up -d
docker compose logs -f migrator     # Schema anlegen beobachten
curl http://127.0.0.1:8080/health/ready    # erwartet: 200
```

Der Migrator legt das Schema an, danach erzeugt die Anwendung den ersten Organisations-Admin mit erzwungenem Passwortwechsel. Das Initialpasswort wird **nie** protokolliert.

Sobald die Anmeldung am Panel funktioniert:

```
BOOTSTRAP_ENABLED=false
```

```
docker compose up -d
```

6. Reverse-Proxy einrichten

Terminieren Sie TLS am Proxy und leiten Sie auf `app:8080` weiter. Beispiele für nginx und Caddy liegen im Deployment-Verzeichnis.

Wichtig:

- `X-Forwarded-Proto`, `X-Forwarded-Host` und `X-Forwarded-For` weiterreichen.
- `REVERSE_PROXY_IP` in der `.env` auf die Adresse des Proxys setzen. Nur von dort werden Forwarded-Header akzeptiert.
- `client_max_body_size` (nginx) groß genug für Anhänge wählen.

Konfigurationsreferenz

Die Compose-Datei liest folgende Variablen aus der `.env`. Die `.env` gehört **nie** in die Versionsverwaltung; Werte können auch aus Docker-Secrets oder einem Secret-Store kommen.

Variable	Zweck
`POSTGRES_PASSWORD`	Passwort der DDL-Rolle `fokus_migrator`
`FOKUS_APP_PASSWORD`	Passwort der Least-Privilege-Rolle `fokus_app`
`SERVER_INSTANCE_ID`	stabile GUID-Identität dieser Server-Instanz
`ORGANIZATION_NAME`	Anzeigename des Trägers
`BOOTSTRAP_ENABLED`, `BOOTSTRAP_ADMIN_EMAIL`, `BOOTSTRAP_ADMIN_PASSWORD`	einmaliger Erst-Admin-Bootstrap
`BOOTSTRAP_FACHADMIN_EMAIL`, `BOOTSTRAP_FACHADMIN_PASSWORD`	optionaler Bootstrap eines Fach-Admin-Kontos
`SERVICE_CLIENT_ID`, `SERVICE_CLIENT_SECRET`	optionaler vertraulicher Service-Client
`MOBILE_REDIRECT_URI`	zusätzliche OAuth-Redirect-URI für die mobile App
`CERT_PASSWORD`	Passwort der PKCS#12-Zertifikate
`REVERSE_PROXY_IP`	einzige Adresse, deren `X-Forwarded-*`-Header vertraut wird
`BLOB_ENCRYPTION_KEY_BASE64`	**Pflicht:** AES-256-Schlüssel (32 Byte Base64) für Anhänge und Datenkopien
`OFFLINE_GRANT_KEY`	privater EC-P-256-Schlüssel (PEM) zum Signieren der Offline-Grants
`OFFLINE_GRANT_KEYSET_VERSION`	monotone Version des Offline-Grant-Keysets (Standard 1)
`OFFLINE_GRANT_ADDITIONAL_PUBLIC_KEY`	zusätzlicher vertrauenswürdiger öffentlicher Schlüssel während einer Rotation
`LICENSING_INSTALLATION_KEY`	Einmal-Enrollment-Key des Lizenzservers
`FileScanning__Enabled`, `FileScanning__Host`	Malware-Scan konfigurieren (siehe unten)

Blob-Verschlüsselung: Den Schlüssel mit `openssl rand -base64 32` erzeugen und **getrennt vom Blob-Archiv** sichern. Ohne ihn ist das Archiv unlesbar. In Entwicklungsumgebungen kann `BlobStorage:AllowGeneratedKey=true` gesetzt werden – in Produktion niemals.

Malware-Scan: `FileScanning__Enabled=true` plus `FileScanning__Host` aktiviert ClamAV. Ist in Produktion **nichts** konfiguriert, arbeitet der Server **fail-closed**: Uploads werden abgelehnt, statt ungeprüft angenommen zu werden.

Datenbank und Migrationen

Es gibt **keine EF-Migrations-Ordner**. Stattdessen arbeitet ein handgeschriebener `MigrationRunner`:

- **Frische Datenbank:** `EnsureCreated` baut das vollständige Schema auf.

- **Bestehende Datenbank:** Ausstehende Versionen aus dem `SchemaMigrations`-Ledger werden in **einer Transaktion** unter `pg_advisory_xact_lock` angewendet; der Versionsstempel wird in derselben Transaktion gesetzt. Konkurrierende Migratoren serialisieren sich damit selbst.
- Danach provisioniert der Migrator die Rolle `fokus_app` und beendet sich. Die App startet erst nach erfolgreichem Migrator.

Der Migrator läuft bei **jedem** `docker compose up` – ein zusätzlicher manueller Schritt ist nicht nötig.

Sicherung und Wiederherstellung

> Ein Backup gilt erst dann als funktionierend, wenn seine **Wiederherstellung getestet** wurde. Und ein Datenbank-Backup **ohne die kryptografischen Schlüssel** ist nicht vollständig wiederherstellbar.

Was gesichert werden muss

Nr.	Gegenstand	Häufigkeit
1	**PostgreSQL** – alle Inhalte, Benutzer, Teams, Zuweisungen, Change-Feed, Audit, Richtlinien	periodisch
2	**Blob-Speicher** – Anhänge aus dem `blobs`-Volume	periodisch
3	**Signier-/Verschlüsselungszertifikate** + `CERT_PASSWORD`	selten, im Secret-Manager
4	**Offline-Grant-Schlüssel** + `OFFLINE_GRANT_KEYSET_VERSION`	selten, im Secret-Manager
5	**Data-Protection-Keyring** (`dpkeys`)	selten
6	**Lizenz-Installationszustand** (`licstate`)	selten

Punkt 2 ist ohne den **Blob-Schlüssel** (`BLOB_ENCRYPTION_KEY_BASE64`) unlesbar – sichern Sie beides **getrennt**.

Backup

```
./deploy/docker/backup.sh      # schreibt ./backups/db-<stamp>.dump + blobs-<stamp>.tgz
```

`db-*.dump` ist ein PostgreSQL-Dump im Custom-Format, direkt aus dem `db`-Container gestreamt. Sichern Sie die Ergebnisse verschlüsselt und versioniert weiter (restic, borg, Objektspeicher). Im Produkt ist **kein** statisches Backup-Passwort eingebaut.

Wiederherstellung

In einen frischen Stack (den alten zuvor mit `docker compose down -v` entfernen):

```
DB_DUMP=./backups/db-<stamp>.dump BLOBS=./backups/blobs-<stamp>.tgz ./deploy/docker/restore.sh
curl http://127.0.0.1:8080/health/ready     # erwartet: 200
```

Das Skript startet eine leere Datenbank, spielt den Dump ein, stellt die Blobs wieder her, lässt den Migrator laufen (dadurch werden ausstehende Schemaschritte angewendet und die Rolle `fokus_app` neu provisioniert – Rollen sind clusterweit und nicht Teil eines Dumps) und startet die Anwendung.

Test-Wiederherstellung

1. Beispieldaten erzeugen (z. B. einen synchronisierten Fall).
2. `./deploy/docker/backup.sh`.
3. `docker compose down -v`.
4. `restore.sh` mit Dump und Blobs ausführen.
5. Schlüssel, Zertifikate und `licstate` aus dem Secret-Manager zurückspielen.
6. Prüfen: Health ready, Anmeldung funktioniert, der Beispielfall ist vorhanden, Anhänge öffnen sich, die Prüfsummen finalisierter Dokumente stimmen.

Versionskompatibilität

Stellen Sie **in die gleiche oder eine neuere** Serverversion wieder her. Ein neuerer Server wendet ausstehende Schemaschritte beim Start an. Spielen Sie **nie** einen neueren Dump in einen älteren Server.

Aktualisierung

Upgrades sind **Backup-first** und nie destruktiv.

```
./deploy/docker/backup.sh                  # 1. sichern
docker compose pull                        # 2. neues Image holen
docker compose up -d                       # 3. Migrator + Neustart der App
curl http://127.0.0.1:8080/health/ready    # 4. Smoke-Test
```

Hinweise

- **Schemaänderungen** laufen nur über den Migrator-Schritt, nie unter der App-Rolle. Prüfen Sie vor einem schemaändernden Upgrade die Release-Notes.
- **API-Kompatibilität:** Server N unterstützt Clients N und N-1, soweit das Schema es zulässt. Über die Richtlinie **Mindest-Clientversion** sperren Sie ältere Clients weich oder hart aus. Lokale Daten bleiben den Clients immer zugänglich.
- **Rolling Upgrade:** Es wird kein kritischer Sitzungszustand im Speicher gehalten – Schlüssel, Keyring und Blobs liegen auf geteilten Volumes. Mehrere App-Instanzen hinter dem Proxy lassen sich nacheinander ersetzen; die Migrationstransaktion serialisiert sich über den Advisory-Lock. Version 1 wird als Einzelinstanz ausgeliefert.
- **Rollback:** das Vor-Upgrade-Backup in die vorherige Image-Version einspielen.

Lizenzierung

Nutzt Ihr Träger eine Organisationslizenz, meldet sich die Installation beim **Lizenzserver** an. Das geschieht **verzögert (lazy)**: Beim Start passiert nichts; erst beim ersten Lizenzkontakt erzeugt der Server sein EC-P-

256-Schlüsselpaar, löst den **einmal verwendbaren** `LICENSING_INSTALLATION_KEY` ein und legt die entstehende Installationsidentität im Volume `licstate` ab.

> **Niemals** das Volume `licstate` auf einem bereits angemeldeten Server löschen – der Enrollment-Key lässt sich nicht erneut einlösen. Beim Umzug eines Deployments nehmen Sie `licstate` mit.

Alternativ provisionieren Sie manuell: `--licensing-keygen` gibt eine frische Installations-ID und ein Schlüsselpaar aus; den öffentlichen Schlüssel übergeben Sie dem Betreiber des Lizenzservers.

Lizenzplätze (Seats) weist ausschließlich die Administration im Panel zu. Es gibt kein automatisches Beanspruchen durch Geräte. Ohne Platz kann ein Gerät nicht synchronisieren.

Sicherheit

Transport und Absicherung

- In Produktion ausschließlich **HTTPS**; TLS terminiert am Proxy.
- **Forwarded-Header** werden nur von der konfigurierten `REVERSE_PROXY_IP` akzeptiert. Damit lässt sich das nach Client-IP partitionierte **Rate Limiting** nicht über gefälschte Header umgehen.
- **Security-Header, CSP, HSTS** und **Antiforgery** im Cookie-Bereich des Panels sind aktiv.
- **Rate Limiting** pro IP mit festen Zeitfenstern, strenger für Anmeldung und Geräteregistrierung.
- Signier- und Verschlüsselungszertifikate sind **Pflicht** (Fail-fast beim Start, wenn sie fehlen).
- In Produktion sind sensible EF-Protokollierung und die Entwickler-Fehlerseite abgeschaltet; Diagnosen geben nie Verbindungszeichenfolgen preis, Protokolle nie Tokens, Secrets oder Fallinhalte.

Autorisierung

Jeder Lese- und Schreibzugriff wird **serverseitig** gegen den aktuellen Zustand autorisiert, nie gegen Behauptungen des Clients. Die Sichtbarkeit wird bereits in SQL gefiltert. Als **privat** markierte Einträge sind für andere App-Nutzer unsichtbar – auch für Teamleitungen und alle Admin-Rollen, und zwar bei Abfrage, Suche, Export und direktem Zugriff über eine ID.

Gerätebindung

Geräte registrieren sich über einen Challenge- und Besitznachweis-Ablauf. Jede API-Anfrage eines gebundenen Geräts trägt einen signierten **Device-Proof**, dessen Nonce persistiert wird (Replay-Schutz); Access-Tokens sind an das Gerät gebunden. Für Neuinstallationen stellt die Administration **Re-Enrollment-Codes** aus.

Anmeldung

- **OpenIddict** mit Authorization Code + PKCE, kurzlebige und rotierte Tokens.
- **MFA** per TOTP als globale Richtlinie, mit Challenge beim Login – für Apps und Panel.
- **Passkeys** (WebAuthn) als passwortlose Alternative; ist ein Passkey aktiv, ist die Passwortanmeldung für dieses Konto abgeschaltet.
- **Technische Admin-Konten** können sich grundsätzlich **nicht in den Apps** anmelden.

- **Break-Glass** ist eine zeitlich begrenzte (höchstens 120 Minuten), MFA-pflichtige und auditierte Ausnahmesitzung – und nur dann möglich, wenn die Organisationsrichtlinie es erlaubt.

Uploads

Uploads sind größen- und typbeschränkt und werden über ClamAV geprüft. In Produktion gilt **fail-closed**: Ist der Scanner nicht konfiguriert, nicht erreichbar oder läuft er in einen Timeout, werden Uploads abgelehnt. Blob-Inhalte sind ruhend verschlüsselt.

Offline-Grant-Keyset rotieren

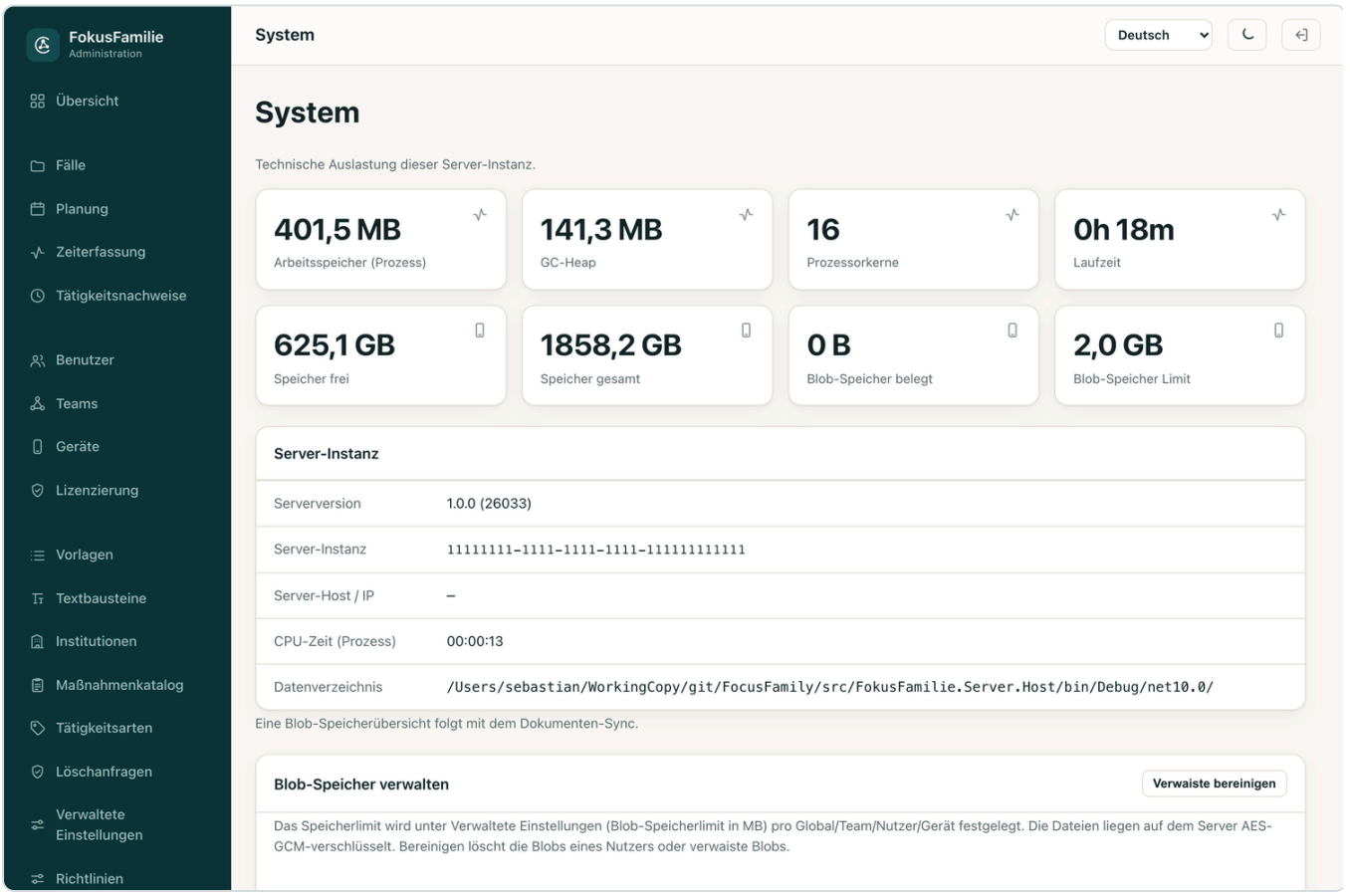
Clients vertrauen Offline-Grant-Schlüsseln nur aus einem **signierten Keyset**, das an die `ServerInstanceId` gebunden ist und eine monotone Version trägt. Die Rotation läuft in **zwei Phasen**:

1. Alten privaten Schlüssel als Signierer behalten, den **neuen öffentlichen** Schlüssel in `OFFLINE_GRANT_ADDITIONAL_PUBLIC_KEY` eintragen und `OFFLINE_GRANT_KEYSET_VERSION` erhöhen. Warten, bis alle Clients mindestens einmal synchronisiert haben.
2. `OFFLINE_GRANT_KEY` auf den neuen privaten Schlüssel umstellen, den **alten öffentlichen** Schlüssel übergangsweise behalten und die Version erneut erhöhen. Den alten Schlüssel entfernen (und die Version wieder erhöhen), sobald die längste Offline-Grant-Laufzeit verstrichen ist.

> Wird die Version beim Ändern der Schlüsselliste **nicht** erhöht, behalten die Clients ihr bisheriges Keyset (fail-closed). Die Version niemals verringern.

Technische Bereiche des Admin-Panels

System-Status

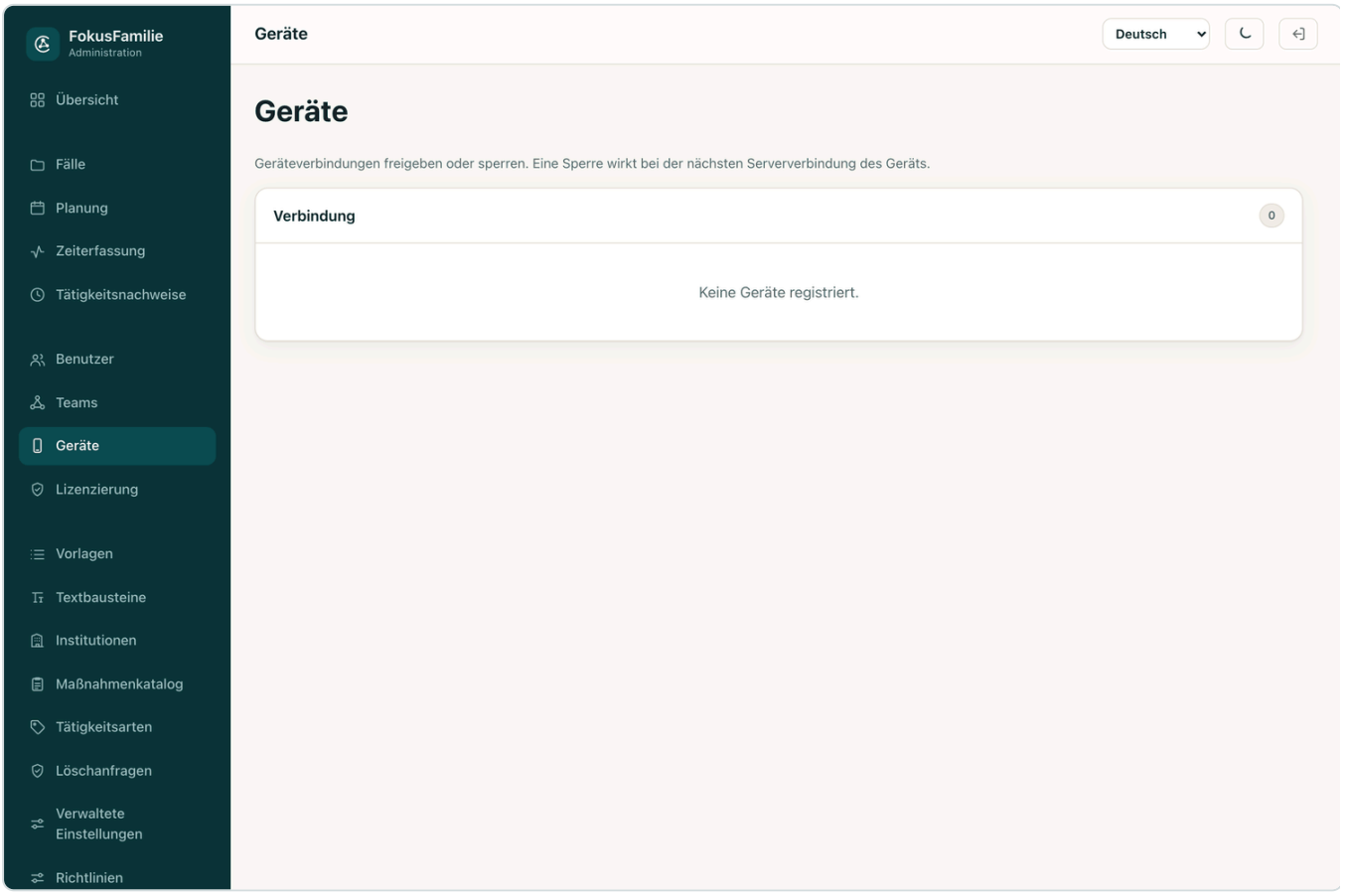


Serverversion, Instanz, Speicherbelegung und die Verwaltung des Blob-Speichers.

Der Menüpunkt **System** (Recht *System-Status ansehen*) zeigt Serverversion, API-Version, Organisation, Server-Instanz-ID, Change-Sequenz und die Auslastung. Hier liegt auch die Verwaltung des **Blob-Speichers**.

> ⚠ **Bereinigen löscht unwiderruflich.** Die Funktion entfernt hochgeladene Dokumente und Medien eines Nutzers oder verwaiste Dateien **endgültig** vom Server. Inhalte, die auf keinem Gerät mehr liegen, sind damit verloren. Vor dem Löschen erscheint eine Sicherheitsabfrage.

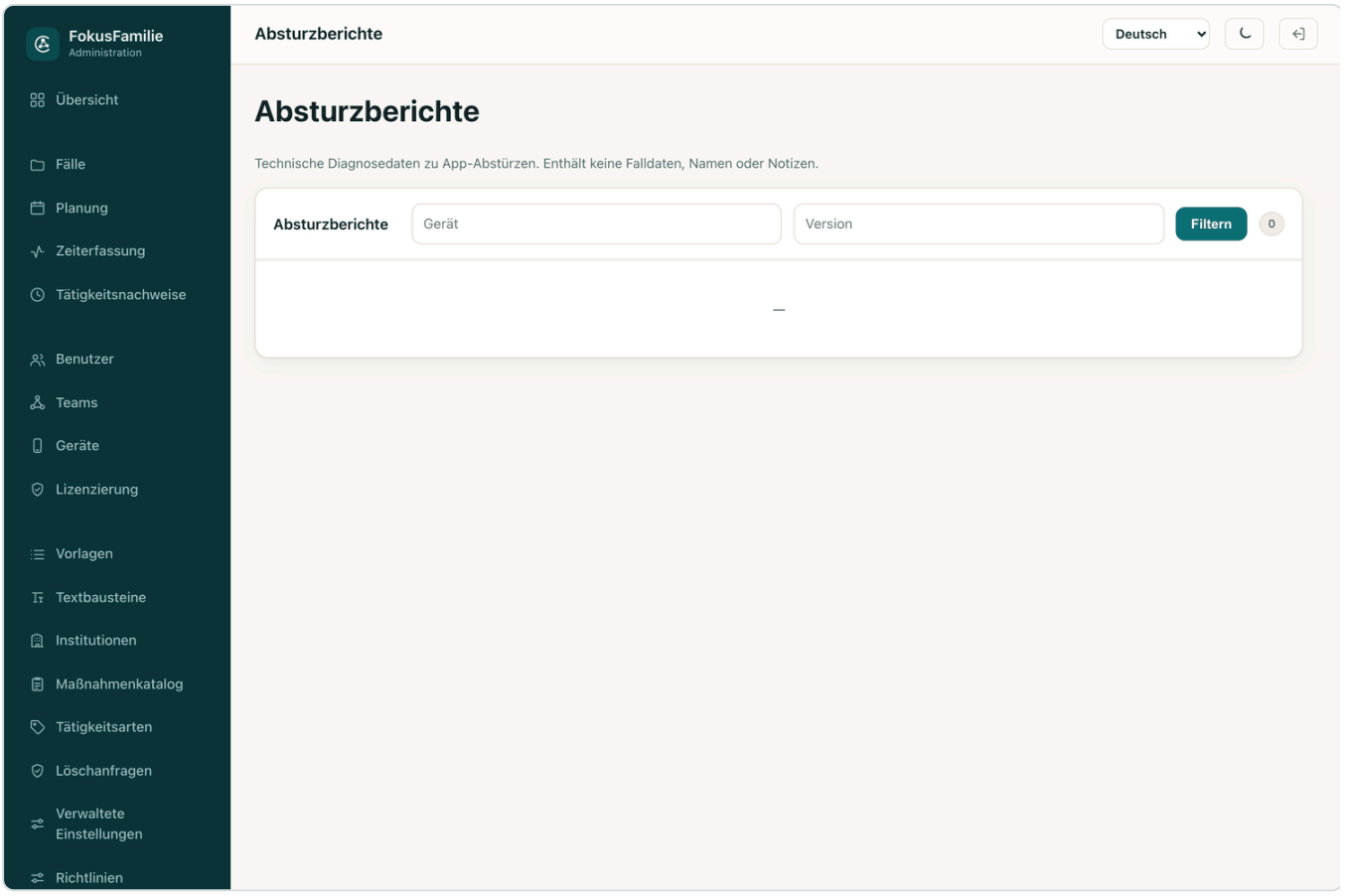
Geräte



Freigeben, sperren, PIN-Sperre zurücksetzen, Gerät entfernen und Re-Enrollment-Codes ausstellen.

Sperren wirken, sobald das Gerät das nächste Mal Kontakt aufnimmt. **Entfernen** löst die Bindung, widerruft die Geräteregistrierung der Lizenz und gibt den Platz frei.

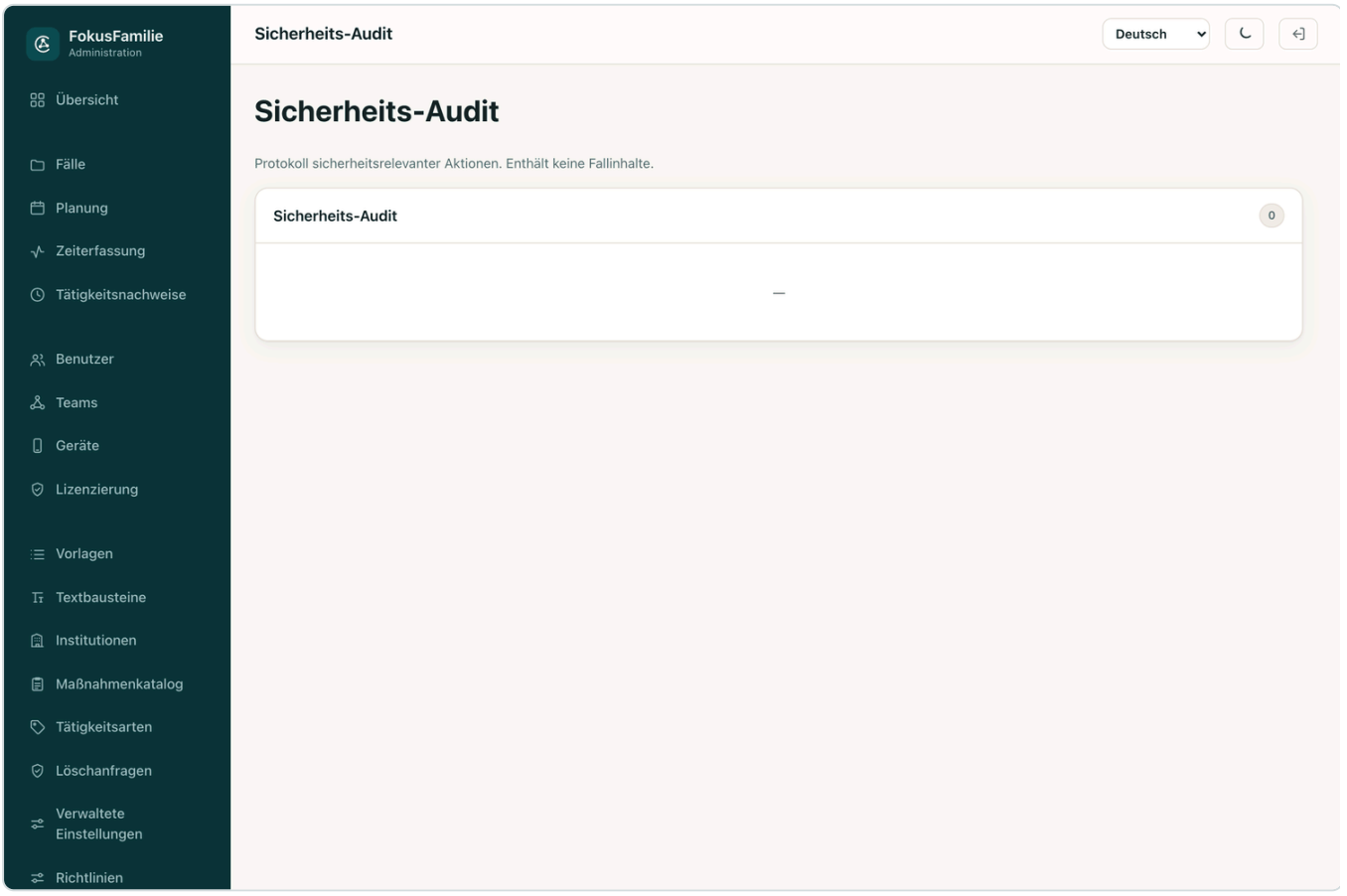
Absturzberichte



Technische Fehlerberichte der Apps – ohne Falldaten, Namen oder Notizen.

Apps können nach einem Absturz einen technischen Bericht senden (Recht *Absturzberichte ansehen*). Die Berichte enthalten Zeitpunkt, Plattform und technische Details – **keine** Inhalte.

Sicherheits-Audit



Wer hat wann was getan – mit Ergebnis (Erfolg, Abgelehnt, Fehler).

Das Audit ist Ihre erste Anlaufstelle bei Verdachtsfällen: Anmeldungen, Rechteänderungen, Gerätefreigaben, Richtlinienänderungen, Notfallzugriffe, Speicherbereinigungen und Leitungskorrekturen an Zeiteinträgen. Fallinhalte enthält es nicht.

Lizenzierung

Der Menüpunkt **Lizenzierung** zeigt den Zustand der Organisationslizenz, die belegten und freien Plätze sowie die registrierten Geräte. Ist die Lizenzintegration nicht konfiguriert, weist die Seite darauf hin, dass `Licensing:InstallationId` und `Licensing:PrivateKeyPem` fehlen (Schlüsselpaar mit `--licensing-keygen` erzeugen).

Betrieb und Überwachung

Health-Endpunkte

```
curl http://127.0.0.1:8080/health/live     # Prozess lebt
curl http://127.0.0.1:8080/health/ready    # Prozess + Datenbank bereit
```

Binden Sie `/health/ready` in Ihre Überwachung ein. Der Container bringt außerdem einen eigenen `HEALTHCHECK` mit.

Protokolle

```
docker compose logs -f app
docker compose logs migrator
docker compose logs clamav
```

Protokolle enthalten **niemals** Tokens, Secrets oder Fallinhalte.

Monitoring

Monitoring ist Sache des Trägers (z. B. Prometheus, Grafana, Loki oder OpenTelemetry) und sendet standardmäßig nichts an externe Anbieter. Sinnvolle Signale: Erreichbarkeit von `/health/ready`, Antwortzeiten, Fehlerraten, Plattenbelegung von `pgdata` und `blobs`, Alter des letzten Backups und ClamAV-Zustand.

Störungen und Vorfälle

Verlorenes oder gestohlenes Gerät

1. Im Panel unter **Geräte** das Gerät **entfernen** – die Bindung wird gelöst und der Lizenzplatz frei.
2. Bei Verdacht auf Kompromittierung zusätzlich das **Benutzerkonto deaktivieren** oder das Passwort zurücksetzen, MFA beziehungsweise Passkeys zurücksetzen.
3. Vorgang im Audit nachvollziehen.

Die Daten auf dem Gerät bleiben durch App-Sperre und lokale Verschlüsselung geschützt.

Fachkraft aus der App-Sperre ausgesperrt

Geräte → PIN-Sperre zurücksetzen. Die Sperre löst sich beim nächsten Sync; die Person vergibt eine neue PIN.

Kompromittiertes Nutzerkonto

Konto deaktivieren, Passwort zurücksetzen, MFA und Passkeys zurücksetzen, Geräte prüfen und bei Bedarf entfernen. Alle Schritte stehen im Audit.

Kompromittierte Serverschlüssel

Signier- und Verschlüsselungszertifikate austauschen und die Anwendung neu starten; alle Tokens werden dadurch ungültig und die Clients melden sich neu an. Den Offline-Grant-Schlüssel über die beschriebene **Zwei-Phasen-Rotation** wechseln.

Datenbank- oder Backup-Diebstahl

Blob-Inhalte sind ruhend verschlüsselt; ohne den Blob-Schlüssel bleiben Anhänge unlesbar. Wechseln Sie Datenbankpasswörter und prüfen Sie im Audit, welche Zugänge betroffen sein könnten. Dokumentieren Sie den Vorfall für die Meldepflichten nach DSGVO.

Verdacht auf bösartigen Upload

ClamAV verweigert betroffene Dateien bereits beim Hochladen. Prüfen Sie die Protokolle des Scanners und informieren Sie die betroffene Fachkraft.

Fehlerbehebung

Symptom	Ursache und Abhilfe
`/health/ready` liefert 503	Datenbank nicht erreichbar oder Migrator nicht gelaufen. `docker compose logs migrator` prüfen; stimmen `POSTGRES_PASSWORD` und `FOKUS_APP_PASSWORD` ?
App beendet sich beim Start (Produktion)	`deploy/docker/keys/*.pfx` oder `CERT_PASSWORD` fehlen
401- oder Weiterleitungsschleifen über den Proxy	`X-Forwarded-Proto=https` wird nicht weitergereicht oder `REVERSE_PROXY_IP` deckt den Proxy nicht ab
Lizenzierung meldet den Enrollment-Key als bereits verwendet	Volume `licstate` wurde nach dem Enrollment gelöscht – aus dem Backup zurückspielen oder neuen Key anfordern
Uploads bleiben in Quarantäne	ClamAV ist noch nicht bereit (Signatur-Download beim ersten Start dauert Minuten). In Produktion ist der Scan absichtlich fail-closed
Geräte synchronisieren nicht	Gerät nicht freigegeben, Lizenzplatz fehlt oder die Mindest-Clientversion sperrt die App aus
Panel meldet „Lizenzintegration nicht konfiguriert"	`Licensing:InstallationId` und `Licensing:PrivateKeyPem` setzen
Build schlägt mit `**/*.resx` fehl (Docker Desktop für macOS)	bekannte MSBuild-Eigenheit unter virtiofs – auf einem Linux-Host bauen oder das vorbereitete Image verwenden

---

Checklisten

Inbetriebnahme

- [ ] Domain, DNS und TLS-Zertifikat vorhanden
- [ ] Image geladen, `.env` vollständig ausgefüllt, keine `CHANGE_ME` -Werte mehr
- [ ] Signier- und Verschlüsselungszertifikate erzeugt, `CERT_PASSWORD` gesetzt
- [ ] `BLOB_ENCRYPTION_KEY_BASE64` erzeugt und **getrennt** gesichert
- [ ] `OFFLINE_GRANT_KEY` persistiert
- [ ] `REVERSE_PROXY_IP` gesetzt, Proxy reicht Forwarded-Header weiter
- [ ] Erst-Admin per Bootstrap angelegt, danach `BOOTSTRAP_ENABLED=false`
- [ ] `/health/ready` liefert 200, Anmeldung am Panel erfolgreich
- [ ] Lizenz-Enrollment durchgeführt, `licstate` gesichert
- [ ] Backup-Skript eingerichtet und ein **Test-Restore** durchgeführt
- [ ] Monitoring auf `/health/ready` und Plattenbelegung eingerichtet

Monatliche Routine

- [ ] Backups vorhanden und nicht älter als vereinbart
- [ ] Test-Restore in eine Staging-Umgebung
- [ ] Serverversion und Sicherheitsupdates prüfen
- [ ] Audit auf Auffälligkeiten durchsehen
- [ ] Geräte- und Lizenzplatzliste aufräumen
- [ ] Plattenbelegung von `pgdata` und `blobs` prüfen

Anhang

Wichtige Pfade und Ports

Gegenstand	Wert
App im Container	Port 8080, gebunden an `127.0.0.1`
PostgreSQL	nur im internen Compose-Netz
Blob-Speicher	`/var/lib/fokusfamilie/blobs` (Volume `blobs`)
Data-Protection-Keys	`/home/app/.aspnet/DataProtection-Keys` (Volume `dpkeys`)
Lizenz-Identität	`/var/lib/fokusfamilie/licensing/installation.json` (Volume `licstate`)
Zertifikate	`deploy/docker/keys/*.pfx` (read-only eingebunden)

Nützliche Befehle

```
docker compose ps                       # Zustand aller Dienste
docker compose logs -f app              # Anwendungsprotokoll
docker compose restart app              # nur die Anwendung neu starten
docker compose down                     # Stack stoppen (Volumes bleiben)
docker compose down -v                  # Stack inklusive Volumes entfernen (Datenverlust!)
./deploy/docker/backup.sh               # Sicherung erstellen
```

Weiterführende Dokumentation im Repository

`docs/server/DEPLOYMENT.de.md`, `docs/server/DATABASE.de.md`, `docs/server/BACKUP-RESTORE.de.md`, `docs/server/UPGRADE.de.md`, `docs/server/SECURITY.de.md`, `docs/server/AUTHENTICATION.de.md`, `docs/server/AUTHORIZATION.de.md`, `docs/server/SYNC-PROTOCOL.de.md`, `docs/server/INCIDENT-RESPONSE.de.md`, `docs/server/THREAT-MODEL.de.md` sowie `deploy/docker/README.de.md`.

Focus Family · Technisches Handbuch · Version 1.0.0 · Stand September 2026