```
module: 23
title: Files / Dokumentenablage
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-1, LP-7, LP-9, LP-10, LP-11, LP-13]
spec_style: requirements-pur
```

Modul 23 — Files / Dokumentenablage

> ***Detail-Spec gemäß LP-9.*** *Zentrale Dokumentenablage pro Tenant mit Verknüpfung an alle relevanten Entitäten. Storage komplett auf Scaleway (EU, DSGVO; LP-13). Löst Datei-Anhänge in HubSpot und ad-hoc-Dropbox-Nutzung bei Geerkens ab.*

Themenblöcke

```
A. Datenmodell (File, FileVersion, FileChunk, FileUploadProgress, FileEntityLink)
B. Menü-Struktur und Routes
C. Listen-Views (Files-Browser mit Entitäts-Filter)
D. Detail-Views (File-Vorschau, Versionen, Verknüpfungen)
E. Upload-Workflow (Chunked Upload mit Progress)
F. Versionierung (gleicher Dateiname → V1, V2, ...)
G. Preview-Generation (PDF, Bild, Office)
H. Verknüpfungen an Entitäten (Kontakt, Deal, Projekt, User, Form-Submission, Auftrag)
I. Activity-Stream-Integration
J. Permissions-Anwendung
K. Storage-Architektur (Scaleway, Tenant-Isolation)
```

Block A — Datenmodell

File

Tenant-scoped, Soft-Delete, RLS-aktiv.

- `id`, `tenant_id`, `created_at`, `updated_at`, `created_by_user_id`
- `display_name` (User-sichtbarer Name; identisch zum Original-Dateinamen beim Upload)
- `mime_type`
- `size_bytes`
- `storage_key` (Pfad in Scaleway-Bucket; Format `tenants/{tenantId}/files/{fileId}/v{version}/original`)
- `current_version` (Integer, Default 1)
- `checksum_sha256` (für Duplikat-Erkennung und Integrität)
- `preview_status` (Enum: `pending | generating | ready | failed | not_supported`)
- `preview_storage_key` (Pfad zur Preview-Datei, NULL bei `not_supported`)
- `assigned_to_user_id` (optional — Hauptverantwortlicher für die Datei)
- `tags[]` (siehe Modul 1, Block H — wiederverwendet)
- `notes` (Markdown-Freitext, optional)

- `archived_at` (Soft-Delete)
- `source` (Enum: `manual_upload | customer_portal | form_submission | email_attachment | automation | api`)

Indizes:
- `(tenant_id, archived_at, created_at DESC)` — Default-Sortierung im Browser
- `(tenant_id, created_by_user_id)` — Owner-Filter
- `(tenant_id, checksum_sha256)` — Duplikat-Erkennung
- Trigram-Index auf `display_name` für Fuzzy-Suche

FileVersion

Append-only — Versionen werden nie gelöscht (außer beim Hard-Delete der File).

- `id`, `file_id`, `tenant_id`, `created_at`, `created_by_user_id`
- `version_number` (Integer, fortlaufend)
- `storage_key`
- `size_bytes`
- `checksum_sha256`
- `change_note` (optional Freitext — was wurde geändert)

Constraint: `(file_id, version_number)` unique.

FileChunk

Temporär während Upload, wird nach erfolgreichem Zusammenbau gelöscht.

- `id`, `tenant_id`, `upload_id` (siehe `FileUploadProgress`)
- `chunk_index`
- `storage_key` (temporärer Pfad)
- `size_bytes`
- `checksum_sha256`
- `received_at`

Constraint: `(upload_id, chunk_index)` unique.

FileUploadProgress

Steuert chunked Upload; ermöglicht Resume nach Verbindungsabbruch (mobil-relevant).

- `id`, `tenant_id`, `created_by_user_id`, `created_at`, `updated_at`
- `original_filename`
- `total_size_bytes`
- `total_chunks`
- `received_chunks` (Integer)
- `expires_at` (Default: 24h nach `created_at` — danach Cleanup-Job)
- `status` (Enum: `in_progress | completed | failed | expired | cancelled`)
- `target_file_id` (NULL bis Abschluss; bei Versions-Upload bereits gesetzt)
- `target_version_intent` (Enum: `new_file | new_version`)
- `failure_reason` (optional Klartext)

Indizes: `(tenant_id, created_by_user_id, status)`.

FileEntityLink

M:N-Verknüpfung von File zu beliebigen Entitäten.

- `id`, `tenant_id`, `file_id`, `created_at`, `created_by_user_id`

- `entity_type` (Enum: `contact | company | lead | deal | project | user | form_submission | order | task | appointment`)
- `entity_id`
- `link_role` (optional Freitext — z.B. „Auftragsbestätigung", „Foto vor Montage", „Acceptance-Beleg")

Constraints:
- `(file_id, entity_type, entity_id)` unique
- Index `(tenant_id, entity_type, entity_id)` für „alle Files an Entity X"-Query

Begründung M:N: Eine Datei kann gleichzeitig zu Kontakt, Deal, Projekt, Auftrag gehören (z.B. Angebots-PDF: Kontakt + Deal + ggf. Projekt nach Won).

Block B — Menü-Struktur und Routes

Sidebar-Hauptpunkt: *Files*.

Routen:

```
/{tenant}/files                          → Files-Browser (alle Files des Tenants)
/{tenant}/files/upload                   → Upload-View (Drag-and-Drop oder File-Picker)
/{tenant}/files/:id                      → File-Detail-View (Vorschau, Versionen,
Verknüpfungen)
/{tenant}/files/:id/versions             → Versionsverlauf (Sub-View)

/{tenant}/admin/files/                   → Admin-Übersicht (Storage-Verbrauch pro Tenant,
Phase 2)
```

Embedded-Anwendung: Der Files-Browser erscheint zusätzlich als Sub-Sektion in Detail-Views anderer Module:
- `/{tenant}/contacts/:id` (Tab/Sektion „Files")
- `/{tenant}/contacts/companies/:id`
- `/{tenant}/deals/:id`
- `/{tenant}/projects/:id`
- `/{tenant}/leads/:id`
- usw.

In diesen Kontexten ist der Browser **vorgefiltert** auf die jeweilige Entity.

Block C — Listen-Views (Files-Browser)

Zu zeigende Daten pro Eintrag

- Datei-Identität (Display-Name, Dateityp-Indikator)
- Größe
- Erstellt-Datum + Ersteller
- Aktuelle Version (z.B. „V3")
- Tags
- Verantwortlicher
- Verknüpfte Entitäten (Anzahl + erste 1-2 als Klartext, Rest als „+N")
- Quelle (manuell / Form / Customer-Portal / Email / Automation)
- Vorschau-Thumbnail (falls verfügbar)

Benötigte UI-Elemente

- Tabellarische oder Grid-Anzeige (Designer wählt; beide Varianten umschaltbar als User-Setting)
- Spalten-Header klickbar für Sortierung
- Such-Funktion (Volltext über Display-Name, Notes, Tags, später OCR-Text in Phase 2)
- Filter-Komponenten:
- Verknüpfte Entität (Auto-Complete: Kontakt / Firma / Deal / Projekt / ...)
- Entitäts-Typ (Multi-Select)
- Verantwortlicher (Multi-Select)
- Ersteller (Multi-Select)
- Tags (Multi-Select mit AND/OR-Toggle)
- Erstellt-Zeitraum (Date-Range-Picker)
- Dateityp / MIME-Gruppe (Multi-Select: Bild / PDF / Office / Video / Audio / Sonstige)
- Quelle (Multi-Select)
- Größe (Range-Filter)
- Toggle „Archivierte anzeigen"
- Bulk-Auswahl (Checkboxes)
- Pagination (Cursor-basiert, „Mehr laden", 50 pro Page)
- Spalten-Konfiguration pro User (sichtbare Spalten + Reihenfolge persistiert)
- Ansichts-Umschalter (Tabelle / Grid mit Thumbnails)

Operationen

- Sortierung pro Spalte (Default: Erstellt-Datum absteigend)
- Filter anwenden / kombinieren / zurücksetzen
- Bulk-Actions:
- Tag hinzufügen / entfernen
- Verantwortlicher zuweisen
- An Entity verknüpfen / Verknüpfung lösen
- Archivieren / Reaktivieren
- Download als ZIP
- Hard-Delete (mit Klartext-Warnung)
- Klick auf File-Zeile → Detail-View
- Inline-Schnell-Vorschau (Hover oder Single-Klick öffnet Quick-Preview, ohne Detail-View zu laden)

Performance-Anforderungen

- Server-side-Filter (kein Client-side-Filtern)
- Index siehe Datenmodell-Sektion
- Thumbnail-Lazy-Loading (Viewport-basiert)
- Bei Filter „verknüpft mit Entity X" wird `FileEntityLink` als Join genutzt, nicht Volltext-Suche

---

Block D — Detail-Views

File-Detail (`/{tenant}/files/:id`)

Zu zeigende Daten:

- Stammdaten (Display-Name, Dateityp, Größe, Checksum, Erstellt-Datum + Ersteller, Verantwortlicher, Quelle)
- Aktuelle Version (Nummer + Datum + Ersteller dieser Version)
- Tags

- Notizen
- Vorschau der aktuellen Version (PDF/Bild/Office; Fallback bei `not_supported`: Klartext-Hinweis + Download-Funktion)
- Verknüpfte Entitäten (gruppiert nach Typ, mit Link-Role)
- Versionsverlauf (Anzahl + Link auf Sub-View)
- Activity-Stream zu dieser File (Upload, Versionen, Verknüpfungs-Änderungen, Downloads — siehe Block I)

Benötigte UI-Elemente:

- Vorschau-Komponente (PDF-Viewer, Bild-Viewer mit Zoom, Office-Viewer, Video/Audio-Player)
- Inline-Edit für Stammdaten (Display-Name, Notes, Tags, Verantwortlicher)
- Verknüpfungs-Verwaltung mit Auto-Complete pro Entitäts-Typ und Rollen-Eingabefeld
- Action-Funktionen für Operationen (siehe unten)
- Versions-Sektion mit Anzeige aller Versionen (Nummer, Datum, Ersteller, Change-Note, Größe, Download-Funktion pro Version)

Operationen:

- Download (aktuelle Version)
- Download spezifische Version
- Neue Version hochladen (siehe Block F)
- Display-Name umbenennen (Original-Datei in Storage bleibt unverändert; nur `display_name` ändert sich)
- Notes editieren
- Tags hinzufügen / entfernen
- Verantwortlichen ändern
- Entitäts-Verknüpfung hinzufügen / entfernen / Rolle ändern
- Vorschau erneut generieren (manueller Trigger bei `failed`)
- Archivieren / Reaktivieren
- Hard-Delete (Multi-Step-Confirm: Klartext-Warnung über alle Versionen + Verknüpfungen + Activity-Einträge die mit gelöscht werden)
- DSGVO-Export (Datei + Metadaten als ZIP — über Modul 35)
- Permalink kopieren (auth-pflichtige URL — kein Public-Link in Phase 1)

Versionsverlauf-Sub-View (`/{tenant}/files/:id/versions`)

Zu zeigende Daten:

- Liste aller Versionen chronologisch absteigend
- Pro Version: Nummer, Datum, Ersteller, Größe, Change-Note, Aktiv-Markierung (für `current_version`)

Benötigte UI-Elemente:

- Versionsliste
- Pro Version: Download-Funktion, Vorschau-Funktion, „Als aktuelle Version setzen" (Rollback)

Operationen:

- Vorschau älterer Version
- Download älterer Version
- Rollback zu älterer Version (legt **neue** Version mit Inhalt der gewählten alten Version an — kein Löschen, Audit-Trail bleibt)
- Change-Note nachträglich editieren (eigene oder mit `files.admin`)

Block E — Upload-Workflow (Chunked Upload mit Progress)

Anforderungen

- **Chunk-Größe:** 5 MB (mobil-optimiert; Trade-off zwischen Overhead und Resume-Granularität)
- **Parallel-Chunks:** max. 3 parallel pro Upload (Bandbreiten-schonend für mobile/baustelle)
- **Resume:** bei Verbindungsabbruch wird `FileUploadProgress` persistiert; Client kann nach Reconnect die fehlenden Chunks nachsenden
- **Maximum-Größe pro File:** 500 MB Default (Tenant-Admin-konfigurierbar bis 2 GB; Phase 2: Storage-Limits pro Subscription-Plan)
- **Mehrfach-Upload:** mehrere Dateien gleichzeitig (Queue)
- **Drag-and-Drop:** Browser + native Capacitor-File-Picker auf Mobile
- **Foto-Direkt-Upload:** Kamera-Integration über Capacitor (Modul 36); Foto wird sofort hochgeladen
- **Offline-Modus:** Foto/Datei wird in IndexedDB-Queue gehalten, Upload startet bei Reconnect (LP-8)

Klartext-Progress-Anzeige

- Pro Datei: Name, Fortschritt in Prozent + Datentransfer (z.B. „12 MB / 47 MB"), geschätzte Restzeit
- Bei Fehler: Klartext-Reason (siehe Klartext-Hinweise unten) + Retry-Funktion
- Bei Erfolg: Übergang in Files-Browser oder direkt in Detail-View (User-Wahl)

Upload-API-Verlauf (logisch)

1. Client ruft Init-Endpoint auf mit `original_filename`, `total_size_bytes`, `mime_type`, optional `target_file_id` für Versions-Upload, optional Verknüpfungs-Vorgaben
2. Server legt `FileUploadProgress` an, gibt `upload_id` und Chunk-Größe zurück
3. Client streamt Chunks mit `(upload_id, chunk_index, chunk_data)`
4. Server validiert Chunk-Checksum, persistiert in temporären Storage-Bereich, inkrementiert `received_chunks`
5. Bei Letztem Chunk: Server baut die Datei zusammen (Server-side-Concat oder Multi-Part-Upload bei Scaleway), validiert Gesamt-Checksum, legt `File` (oder neue `FileVersion`) an, löscht temporäre Chunks, setzt `FileUploadProgress.status = completed`
6. Server triggert Preview-Generation-Job (siehe Block G)
7. Client erhält finale `file_id` und ggf. URL

Edge-Cases

- Upload-Abbruch durch User: `FileUploadProgress.status = cancelled`, temporäre Chunks werden vom Cleanup-Job entfernt
- Upload-Expiry (24h ohne Aktivität): `expired`, Chunks gelöscht
- Server-Fehler bei Concat: `failed` mit `failure_reason`, User kann Re-Upload starten (komplett neu, nicht resume)
- Duplikat-Check (per Checksum): bei exaktem Match einer existierenden, nicht-archivierten File des gleichen Tenants Klartext-Hinweis: *„Diese Datei existiert bereits als ‚{display_name}'. Möchtest du eine neue Version anlegen, eine Verknüpfung hinzufügen oder trotzdem als neue Datei speichern?*"

Block F — Versionierung

Trigger für neue Version

- **Automatisch beim Upload mit gleichem** `display_name` an dieselbe File-ID (z.B. über „Neue Version hochladen"-Funktion in der Detail-View)

- **Vorschlag bei Upload mit gleichem Display-Name** in einem Entity-Kontext: *„Es gibt bereits eine Datei ‚angebot.pdf' in diesem Projekt. Möchtest du eine neue Version anlegen oder eine separate Datei?"*
- **Manuell** über „Neue Version hochladen" in File-Detail-View (Display-Name kann abweichen, Versionierung wird trotzdem fortgeführt)

Anforderungen

- Versionsnummern sind fortlaufend (1, 2, 3, ...) und nicht editierbar
- Beim Anlegen einer neuen Version: optional Change-Note, sonst Default-Klartext (z.B. „Version hochgeladen am {Datum}")
- Vorherige Versionen bleiben in Storage und Datenbank
- Hard-Delete der File entfernt **alle** Versionen
- Storage-Pfad-Konvention: `tenants/{tenantId}/files/{fileId}/v{versionNumber}/original`
- Preview wird pro Version separat generiert

Rollback

- „Als aktuelle Version setzen" auf einer alten Version legt **neue** Version mit Inhalt der gewählten alten Version an
- Kein In-Place-Rollback (Audit-Trail-Integrität)

Klartext-Hinweise

- Beim Versions-Upload: *„Datei wird als Version {n} gespeichert. Vorherige Versionen bleiben verfügbar."*
- Beim Rollback: *„Inhalt von Version {n} wird als neue Version {m} gespeichert. Vorherige Versionen bleiben unverändert."*

Block G — Preview-Generation

Unterstützte Typen (Phase 1)

- **PDF:** direkt
- **Bilder:** JPEG, PNG, GIF, WebP, HEIC (HEIC-Konvertierung nach JPEG für Browser-Kompatibilität)
- **Office:** DOCX, XLSX, PPTX, ODT, ODS, ODP — Konvertierung nach PDF (LibreOffice-Headless oder vergleichbar; Server-side Worker)
- **Video:** Thumbnail (erster Frame oder konfigurierbar) + Browser-natives Playback im Detail-View
- **Audio:** keine Vorschau, nur Player
- **Plain-Text / Markdown:** direkte Anzeige
- **Sonstige:** `not_supported` → nur Download-Funktion

Anforderungen

- Preview-Generation als asynchroner Worker-Job (Modul 17 / interne Worker-Queue)
- Status-Spalte `preview_status` mit Werten `pending | generating | ready | failed | not_supported`
- Bei `failed`: manueller Re-Trigger über Detail-View (Permission `files.write`)
- Preview-Output landet in `tenants/{tenantId}/files/{fileId}/v{versionNumber}/preview.{ext}` (PDF für Office, JPEG für Bilder/Video-Thumbnail)
- Thumbnail-Generation für Listen-Grid (kleinere Auflösung, separater Storage-Pfad)
- Maximaler Worker-Timeout pro Job (z.B. 5 Min); bei Überschreitung `failed`

Klartext-Hinweise

- `pending` / `generating`: *„Vorschau wird erstellt — kann einen Moment dauern."*

- `failed`: *„Vorschau konnte nicht erstellt werden. Datei kann heruntergeladen und lokal geöffnet werden. [Erneut versuchen]"*
- `not_supported`: *„Für diesen Dateityp ist keine Vorschau verfügbar. [Herunterladen]"*

Block H — Verknüpfungen an Entitäten

Unterstützte Entitäts-Typen (Phase 1)

- `contact` (Modul 1)
- `company` (Modul 1)
- `lead` (Modul 1, 2)
- `deal` (Modul 3)
- `project` (Modul 4)
- `order` (Modul 4 / KWP-Sync)
- `user` (Modul 27 — z.B. HR-Dokumente am Mitarbeiter; Permissions besonders, siehe HR-Hub Modul 14)
- `form_submission` (Modul 5 — bei File-Upload-Feldern)
- `task` (Modul 26)
- `appointment` (Modul 6 / 25)

Anforderungen

- Verknüpfungen sind M:N (siehe Datenmodell `FileEntityLink`)
- Pro Verknüpfung optional `link_role` (Freitext)
- Verknüpfungen werden im Activity-Stream der Ziel-Entity gespiegelt (Activity-Typ `file_uploaded` mit `metadata.entity_type` und `link_role`)
- Hinzufügen einer Verknüpfung erfordert `files.write` **und** Lese-Permission auf die Ziel-Entity
- Entfernen einer Verknüpfung erfordert `files.write`

Verknüpfung über Customer-Portal-Upload

- Endkunden-Uploads (Modul 15) landen automatisch in Files
- `source = customer_portal`
- Auto-Verknüpfung mit dem `Project` / `Deal` / `Order`, in dessen Kontext der Upload erfolgt
- `created_by_user_id` ist NULL (Upload kommt von `PortalUser`, separate User-Klasse); zusätzlich `created_by_portal_user_id` (Pflicht-Feld in diesem Fall — für Audit-Trail; nicht in Datenmodell oben aufgeführt um Spec klein zu halten, ist aber bei Implementierung zu ergänzen)
- Klartext-Hinweis im Files-Browser: Quelle als „Customer-Portal" + Name des hochladenden Endkunden

Verknüpfung über Form-Submission

- Form-Builder (Modul 5) hat File-Upload-Felder
- Bei Submission landen Files in dieser Modul-Struktur
- Auto-Verknüpfung mit der `FormSubmission`
- `source = form_submission`

Block I — Activity-Stream-Integration

Files erzeugen Activities im zentralen Activity-Stream (Modul 1, Block E):

- `file_uploaded` (subject = die Ziel-Entity; metadata enthält `file_id`, `display_name`, `link_role`)

- `file_version_added` (subject = die File; ggf. zusätzlich gespiegelt an verknüpfte Entities)
- `file_link_added` / `file_link_removed`
- `file_archived` / `file_restored`
- `file_renamed` (Display-Name)
- `file_deleted` (Hard-Delete; subject = NULL nach Delete, aber Activity bleibt mit `metadata.deleted_file_name`)
- `file_downloaded` (Phase 2 — optional Tracking; nicht in Phase 1 wegen Audit-Last)

`summary_text`**-Beispiele (Klartext-DE):**
- „Max Mustermann hat ‚angebot.pdf' an Deal ‚Heizungstausch Schmidt' angehängt."
- „Sandra Müller hat Version 3 von ‚wartungsvertrag.pdf' hochgeladen."
- „Anna Becker (Customer-Portal) hat ‚foto-zaehler.jpg' an Projekt ‚Heizung Schmidt' hochgeladen."

Block J — Permissions-Anwendung

Permission-IDs

Permission	Erlaubt
`files.read_own`	Files sehen, bei denen User = `created_by_user_id` ODER `assigned_to_user_id` ODER über Entity-Verknüpfung indirekt sichtbar (z.B. eigenes Projekt)
`files.read_all`	Alle Files des Tenants sehen
`files.write`	Hochladen, neue Version, umbenennen, taggen, verknüpfen, archivieren
`files.delete`	Hard-Delete
`files.admin`	Storage-Limits ändern (Phase 2), alle Files (auch ohne Owner-Match) verwalten, DSGVO-Export beliebiger Files

Bundle-Mapping (Variante 2)

Bundle	Atomare Permissions
`files.none`	(keine)
`files.read_own`	`files.read_own`
`files.read_all`	`files.read_own` + `files.read_all`
`files.write`	`files.read_all` + `files.write`
`files.admin`	`files.read_all` + `files.write` + `files.delete` + `files.admin`

Anwendung in Routes

- `/{tenant}/files` → `files.read_own` oder `files.read_all`
- `/{tenant}/files/upload` → `files.write`
- `/{tenant}/files/:id` → `files.read_own` (mit Owner-/Entity-Sichtbarkeits-Check) oder `files.read_all`
- `/{tenant}/files/:id/versions` → analog
- `/{tenant}/admin/files/` → `files.admin` oder `tenant.admin`

Anwendung in Action-Funktionen

- Upload / neue Version / Umbenennen / Verknüpfen: `files.write`

- Hard-Delete: `files.delete`
- DSGVO-Export einzelner File: `files.admin` (außer eigene Files: `files.read_own` reicht)

Owner-Check-Logik bei `files.read_own`

User sieht eine File wenn **eine** der folgenden Bedingungen erfüllt ist:

1. `File.created_by_user_id = self`
2. `File.assigned_to_user_id = self`
3. Es existiert ein `FileEntityLink` zu einer Entity, auf die der User Lese-Zugriff hat (transitive Sichtbarkeit über Modul-Permissions: z.B. wenn User `projects.read_own` hat und das verlinkte Projekt ihm gehört → File sichtbar)

Implementierungs-Hinweis: Transitive Sichtbarkeit ist Performance-relevant. Index auf `FileEntityLink (tenant_id, entity_type, entity_id)` und Materialized View / Cache pro User erwägen.

HR-Sonderfall

Files an `user`-Entity (HR-Dokumente, z.B. Lohnabrechnungen) folgen HR-Hub-Permissions (Modul 14, separate Spec). Default: nur User selbst + `hr.admin` sehen die Datei, auch mit `files.read_all`.

Block K — Storage-Architektur

Anforderungen

- **Provider:** Scaleway Object Storage (EU, DSGVO-konform; LP-13)
- **Bucket-Strategie:** ein Bucket pro Region, Pfad-Präfix `tenants/{tenantId}/...` für Isolation
- **Verschlüsselung:** Server-side Encryption (Scaleway-managed Keys in Phase 1; KMS-Customer-Keys Phase 2)
- **Zugriff:** keine Public-URLs; signierte URLs mit kurzer Lebenszeit (z.B. 5 Min) für Browser-Download/Preview
- **Tenant-Isolation:** App-Code prüft, dass `storage_key` zu `tenant_id` aus dem aktuellen Request passt; zusätzliche Schutz-Schicht zur RLS in der DB
- **Backup:** Scaleway-Multi-Zone-Replikation; tägliche Snapshots
- **Lifecycle:** Hard-Delete der File löscht alle Versionen + Previews + Thumbnails aus Storage (Cleanup-Job, asynchron, mit Audit-Eintrag)
- **Storage-Verbrauch pro Tenant:** wird pro Upload aktualisiert, in Tenant-Admin-View sichtbar (Phase 2: Limits pro Subscription-Plan, siehe Modul 28)

Cleanup-Jobs

- Temporäre Chunks von expired/cancelled/failed `FileUploadProgress`: täglich
- Hard-gelöschte Files: sofortiger Trigger + Verifikation in nachgelagertem Job (24h später)
- Verwaiste Storage-Objekte (kein DB-Eintrag): wöchentlicher Reconciliation-Job mit Klartext-Report

---

Klartext-Hinweise / Fehler-Cases (LP-10)

- **Upload-Abbruch (Netzwerk):** *„Verbindung unterbrochen. Bereits hochgeladene Daten bleiben — Upload wird fortgesetzt sobald du wieder online bist."*
- **Upload zu groß:** *„Datei ist {size} groß. Maximale Dateigröße in diesem Workspace: {max}. Frag deinen Workspace-Admin, wenn du größere Dateien hochladen musst."*
- **Mime-Typ blockiert (z.B. ausführbare Dateien):** *„Dateityp ‚{mime}' ist aus Sicherheitsgründen nicht erlaubt."*
- **Storage-Limit erreicht (Phase 2):** *„Speicher-Limit dieses Workspace ist erreicht. Bitte alte Dateien archivieren oder Plan upgraden."*

- **Preview fehlgeschlagen:** *„Vorschau konnte nicht erstellt werden. Datei kann heruntergeladen und lokal geöffnet werden. [Erneut versuchen]"*
- **Duplikat-Match:** *„Diese Datei existiert bereits als ‚{name}' in diesem Workspace. [Neue Version anlegen] [Verknüpfung hinzufügen] [Trotzdem neu speichern]"*
- **Hard-Delete-Warnung:** *„Diese Datei hat {n} Versionen und ist mit {m} Einträgen verknüpft. Beim Löschen gehen alle Versionen und Verknüpfungen verloren. Stattdessen archivieren? [Archivieren] [Endgültig löschen]"*
- **Permission-Block beim Anhängen an Entity:** *„Du hast keinen Zugriff auf {entity_name}. Bitte einen Kollegen mit Zugriff bitten, die Verknüpfung hinzuzufügen."*
- **Customer-Portal-Upload erkannt:** *„Diese Datei wurde von {endkunde_name} über das Kunden-Portal hochgeladen am {datum}."*

Phase 2 / spätere Features

- **OCR** für PDF/Bilder, Volltext-Index der OCR-Ergebnisse für globale Suche (Modul 24)
- **KI-Auto-Tagging** (Mistral/Scaleway) — schlägt Tags und Verknüpfungen basierend auf Inhalt vor
- **Storage-Limits** pro Subscription-Plan (Modul 28); UI-Banner bei Annäherung
- **Archivierungs-Workflow:** automatisches Archivieren von Files nach X Monaten ohne Zugriff (konfigurierbar)
- **Public-Share-Links** (z.B. Angebot per Link an Kunden) — separat von Customer-Portal
- **Watermarking** für PDFs (z.B. Empfänger-Name)
- **Versions-Vergleich** (Diff-View für PDFs/Office)
- **Folder-Strukturen** (in Phase 1 bewusst NICHT — Tags + Entity-Verknüpfungen + Filter ersetzen Folder)
- **Datei-Locking** beim parallelen Editieren (für Office-Online-Integration)
- **KMS-Customer-Keys** (Enterprise)

Architektur-Vorschriften (Querschnitt)

- **LP-10, AV #1:** Klartext-Diagnostik überall (Upload-Fehler, Preview-Fehler, Permission-Blocks)
- **LP-13, AV #12:** EU-Stack — Scaleway-Storage, EU-Verschlüsselung, kein US-CDN
- **AV #5:** Permissions-aware-Default in Listen und transitiver Sichtbarkeit
- **AV #6:** Audit-Trail-Default für alle schreibenden Aktionen (Upload, Verknüpfung, Delete)
- **LP-1:** Tenant-Isolation strikt (RLS + Storage-Pfad)
- **LP-8:** Offline-Modus für Foto-Upload auf Baustelle (IndexedDB-Queue, Sync-on-Reconnect)
- **LP-9:** Requirements-pur Spec ohne UI-Layout

Detail-Spec-Status: DRAFT

Erste vollständige Detail-Spec für Modul 23. **Vor Bau:** Verifizierung mit Noah (zusammen mit Pflichtenheft-Sweep) und Abgleich mit Modul 36 (Mobile-Offline-Queue) und Modul 15 (Customer-Portal-Upload-Flow).

Offene Detail-Punkte (nicht-blockierend):

- Konkrete Chunk-Größe und Parallel-Anzahl mit Mobile-Bandbreiten-Tests verifizieren
- Office-Preview-Worker: LibreOffice-Headless vs. SaaS-Konverter — Trade-off Self-Hosting vs. Skalierung
- HEIC-Konvertierungs-Library finalisieren (libheif vs. ImageMagick mit HEIC-Plugin)
- Storage-Limits pro Subscription-Plan in Phase 2 konkret definieren (zusammen mit Modul 28)
- HR-Hub-Permissions-Sonderfall in Modul 14 konsolidieren (Files an `user`-Entity)
- Transitive Sichtbarkeit: Performance-Tests mit großen Tenants (10.000+ Files)