Pflichtenheft – Company & Location Service

Service-Name: company-service
Version: 1.0
Stand: 12.02.2026
Geltungsbereich: Modul/Service zur Verwaltung von Companies (Mandanten) und deren Locations (Standorte) inkl. Hauptstandort-Regel, Standortstatus (OPEN/CLOSED), Soft-Delete (Papierkorb), Audit und Eventing (Outbox – empfohlen)

1. Einleitung

1.1 Ziel

Ziel ist die zentrale Verwaltung von **Company**-Stammdaten (Mandant) und **Locations** (Standorte) innerhalb eines Multi-Mandanten-Systems. Jede Company besitzt mindestens eine Location und genau eine **Hauptlocation**.

Der Service stellt insbesondere sicher:

- beliebig viele Companies (Mandanten) im System
- pro Company mindestens 1 Location
- pro Company genau 1 Hauptlocation
- Hauptlocation kann jederzeit gewechselt werden
- Locations können geschlossen werden (CLOSED), aber es muss mindestens eine **OPEN** Location verbleiben und die Hauptlocation muss immer OPEN sein
- Soft-Delete (Papierkorb) + Restore für Company/Location inkl. Audit (wer/wann)
- **Cascade Trash Locations**: Company trash → alle Locations trash (Soft-Delete)
- Integration via Events (Outbox empfohlen) für lose Kopplung mit anderen Services

1.2 Abgrenzung / Nicht-Ziele (MVP)

Nicht Bestandteil des MVP:

- Authentifizierung (Login, Token-Ausstellung, MFA, Refresh Tokens) → kommt später als eigener Auth-Service
- Benutzerverwaltung/Registrierung → User-Service / Auth-Service
- Verwaltung von Kommunikationsdaten (E-Mail, Telefon, Adresse) → Communication-Service
- Vollständiges revisionssicheres Audit-Log (append-only) → optional später (Events liefern Basis)
- Volltext-/Geo-Suche über externe Suchsysteme (Elastic/OpenSearch) → als Erweiterung vorgesehen

1.3 Begriffe

- **Company:** Mandant (Tenant). Fachliches Objekt für Unternehmensdaten.
- **Location:** Standort, der zu genau einer Company gehört.
- **Hauptlocation:** genau ein Standort pro Company (die „primäre“ Location).
- **OPEN/CLOSED:** fachlicher Status eines Standorts (offen/geschlossen).
- **Soft-Delete / Papierkorb (TRASHED):** Datensatz bleibt in DB, ist aber logisch gelöscht (`trashedAt/trashedBy`). Restore möglich.
- **Audit:** Felder, die speichern „wer“ und „wann“ erstellt/geändert/gelöscht hat.
- **UUID/ULID:** eindeutige IDs als String. ULID ist sortierbar nach Zeit, UUID ist weit verbreitet.

2. Rahmenbedingungen / Technologie

2.1 Technologie-Stack (MUSS)

- Backend: Java + Spring (Spring Boot)
- Datenbank: MariaDB (MySQL kompatibel)
- API: REST/JSON
- IDs: UUID/ULID (String)

2.2 Datenbank-Migrationen (SOLL)

- Schemaänderungen werden versioniert über ein Migrationstool (z. B. Flyway oder Liquibase).

2.3 OpenAPI/Swagger Dokumentation (MUSS)

Alle Controller und DTOs werden ausführlich mit OpenAPI/Swagger annotiert:

- Endpunktbeschreibung (Zweck/Use Case)
- DTO-Schemas (Request/Response)
- Beispiele (Examples)
- Fehlerfälle (Statuscodes + Error DTO)
- Parameter (Path/Query/Body) inkl. Pflicht/Optional/Default
- Paging/Sort-Parameter (falls Liste)

Definition of Done (MUSS): Kein Endpoint gilt als fertig ohne vollständige Swagger-Doku (Beschreibung, Parameter, DTO-Schema, Examples, Error Responses).

2.4 Zeit- und Datumsstandard (MUSS)

- Alle Zeitpunkte in Responses/Requests als UTC (ISO-8601), z. B. `2026-02-12T12:00:00Z`.
- Serverseitige Audit-Zeitpunkte werden in UTC gesetzt.

3. Fachliche Anforderungen

3.1 Mandantenmodell / Zuordnung (MUSS)

Jeder Company- und Location-Datensatz besitzt:

- `companyId` (die ID der Company als Mandanten-ID; UUID/ULID)

Jede Location gehört genau zu einer Company:

- `location.companyId` verweist auf die zugehörige Company.

Wichtige Konsequenz: Alle DB-Zugriffe sind **company-scharf**.

3.2 IDs (UUID/ULID) & globale Eindeutigkeit (MUSS)

- `companyId` und `locationId` sind UUID/ULID Strings.
- **locationId ist global eindeutig**, damit die Route `/api/v1/location/{locationId}` ohne companyId möglich ist.
- Der Service MUSS trotzdem prüfen, dass die Location zur Company im Auth-/Tenant-Kontext gehört (siehe 4.1).

3.3 Company-Felder (MUSS/SOLL)

MUSS (MVP):

- `companyId`
- `name`
- `mainLocationId`
- Audit: `createdAt/By`, `modifiedAt/By`, `trashedAt/By`
- `version` (Optimistic Locking) (MUSS, siehe 3.12)

SOLL (typisch):

- `displayName`
- `timezone` (Default, z. B. `Europe/Berlin`)
- `locale` (z. B. `de-DE`)
- `logoFileRef` (Referenz auf File-Service/Storage)

3.4 Location-Felder (MUSS/SOLL)

MUSS (MVP):

- `locationId`
- `companyId`
- `name`
- `status` (OPEN/CLOSED)
- Close-Meta: `closedAt/By`, optional `closedReason`
- Audit: `createdAt/By`, `modifiedAt/By`, `trashedAt/By`
- `version`

SOLL (typisch):

- `locationCode` (pro Company optional eindeutig)
- `timezone` (fallback auf Company timezone)
- `trashedCause` (`MANUAL` | `CASCADE`) oder `trashedByCascade` (boolean) zur sauberen Restore-Logik

3.5 Hauptlocation-Regel (MUSS)

- Pro Company existiert **genau eine** Hauptlocation.
- Hauptlocation wird über `company.mainLocationId` definiert.
- Hauptlocation MUSS existieren, OPEN sein und darf nicht trashed sein.

3.6 Mindestanzahl aktiver Locations (MUSS)

- Pro Company muss jederzeit mindestens **eine OPEN Location** existieren (nicht trashed).
- Das Schließen oder Trashen der letzten OPEN Location ist nicht erlaubt.

3.7 Standort schließen / wieder öffnen (MUSS)

- Locations können geschlossen werden (`status=CLOSED`).
- Eine Location darf **nicht** geschlossen werden, wenn sie aktuelle Hauptlocation ist.
- Ein geschlossenes Objekt kann wieder geöffnet werden (`status=OPEN`).

3.8 Soft-Delete, Restore, Retention (MUSS/SOLL)

MUSS:

- Soft-Delete über `trashedAt/trashedBy`
- Standard-GET/LIST liefern nur `trashedAt IS NULL` (außer `includeTrashed=true`)

MUSS (Entscheidung): Cascade Trash Locations

- Wenn eine Company getrashed wird, werden alle ihre Locations ebenfalls getrashed.

SOLL:

- Restore-Endpunkte für Company und Location
- Retention/Purge: endgültiges Löschen nach konfigurierbarer Frist (z. B. 90 Tage) per täglichem Job

3.9 Firmenlogo (SOLL)

- Company kann ein Logo haben, gespeichert als **Referenz** `logoFileRef` (kein Base64 in DB).
- Upload/Storage erfolgt außerhalb dieses Services (z. B. File-Service/Object Storage).

3.10 Kommunikation (Abgrenzung, MUSS)

- Kommunikationsdaten werden **nicht** im company-service gespeichert.
- Integration erfolgt über Communication-Service via `ownerType` + `ownerId`.

Empfehlung (festgelegt):

- pro Owner mehrere Adressen erlaubt, aber **genau eine primäre** (`primary=true`).

3.11 Suche (SOLL)

- Endpunkte zur Liste/Suche von Locations pro Company mit Paging/Sort.
- Optional Filter: status, nameContains, includeTrashed.

3.12 Audit & Versionierung (MUSS)

MUSS:

- `createdAt/By`, `modifiedAt/By`, `trashedAt/By`
- `closedAt/By` (Location Close)

MUSS:

- `version` (Optimistic Locking) zur Vermeidung von Überschreibkonflikten, insbesondere bei `setMainLocation`.

3.13 Normalisierung / Datenqualität (SOLL)

- `nameNormalized` (trim + lowercase) für Company/Location zur Suche.
- `locationCodeNormalized` (falls locationCode verwendet wird).

3.14 Domain-Events / Outbox (SOLL, empfohlen)

- Outbox Pattern: Domain-Event wird in derselben DB-Transaktion geschrieben.
- Events (minimal):
    - `CompanyCreated`, `CompanyUpdated`, `CompanyTrashed`, `CompanyRestored`, `CompanyMainLocationChanged`
    - `LocationCreated`, `LocationUpdated`, `LocationClosed`, `LocationReopened`, `LocationTrashed`, `LocationRestored`

4. Nicht-funktionale Anforderungen

4.1 Sicherheit / Mandantentrennung (MUSS)

- `companyId` wird serverseitig gegen Auth-/Tenant-Kontext geprüft (kein Cross-Company Zugriff).
- Audit-UserIDs werden aus dem Auth-Kontext befüllt.
- Zugriff über Mandantengrenzen ist ausgeschlossen (403).

Spezialfall Location-Routen ohne companyId:
Bei `/api/v1/location/{locationId}` MUSS serverseitig geprüft werden:

- `location.companyId == companyId aus Auth-/Tenant-Kontext`
- plus IAM-Permission im Company-Kontext.

4.2 Logging/Tracing (SOLL)

- Jede Anfrage loggt correlationId, companyId, Endpoint, Dauer, Statuscode.
- Keine sensiblen Daten im Klartext loggen (z. B. Tokens).

4.3 Caching (SOLL)

Siehe Abschnitt 8 (Caching).

5. Datenmodell (logisch)

5.1 Entity: Company

Kernfelder:

- `companyId` (UUID/ULID, String)
- `name`, optional `displayName`
- `mainLocationId` (UUID/ULID, String)
- optional `timezone`, `locale`, `logoFileRef`
- Audit: `createdAt/By`, `modifiedAt/By`, `trashedAt/By`
- `version`

Indizes (MUSS/SOLL):

- MUSS: PK(companyId)
- SOLL: `(nameNormalized)` für Suche

5.2 Entity: Location

Kernfelder:

- `locationId` (UUID/ULID, String)
- `companyId` (FK logisch)
- `name`, optional `locationCode`, optional `timezone`
- `status` (OPEN/CLOSED)
- Close: `closedAt/By`, optional `closedReason`
- Soft-Delete: `trashedAt/By`, optional `trashedCause`
- Audit: `createdAt/By`, `modifiedAt/By`
- `version`

Indizes (MUSS/SOLL):

- MUSS: PK(locationId)
- MUSS: `(companyId, status, trashedAt)` (z. B. für „mindestens eine OPEN“)
- SOLL: `(companyId, nameNormalized)`
- SOLL: Unique `(companyId, locationCode)` falls locationCode genutzt

Hauptlocation-Regel: technisch (Constraint) oder fachlich (Service-Logik + Transaktion) erzwingen. Empfehlung: fachlich + Optimistic Locking.

5.3 Optional: OutboxEvent (SOLL)

- `eventId`, `eventType`, `occurredAtUtc`, `companyId`, optional `locationId`, `actorSubjectId`, `payloadJson`, `status`, `retryCount`

6. REST-API Spezifikation (inkl. Zweck)

6.1 Company

POST `/api/v1/companies`
Zweck: Company anlegen (inkl. initialer Location als Hauptlocation).
Hinweis: notwendig, weil `companyId` erst bei Anlage entsteht.

GET `/api/v1/companies/{companyId}`
Zweck: Company laden (ohne trashed standardmäßig).

PUT `/api/v1/companies/{companyId}`
Zweck: Company ändern (Name, DisplayName, timezone, locale, logoFileRef …).

PUT `/api/v1/companies/{companyId}/main-location`
Zweck: Hauptlocation wechseln.
Regeln: Ziel-Location muss zur Company gehören, OPEN und nicht trashed sein.

PUT `/api/v1/companies/{companyId}/logo`
Zweck: Logo-Referenz setzen.

DELETE `/api/v1/companies/{companyId}/logo`
Zweck: Logo entfernen.

DELETE `/api/v1/companies/{companyId}`
Zweck: Soft-Delete Company (setzt trashedAt/trashedBy) + **Cascade Trash Locations**.

POST `/api/v1/companies/{companyId}/restore`
Zweck: Company wiederherstellen (Restore) + (empfohlen) Cascade-Restore der cascade-trashed Locations.
MUSS: konsistenten Zustand herstellen (gültige Hauptlocation + mindestens eine OPEN Location).

GET `/api/v1/companies` (SOLL)
Zweck: Companies listen (z. B. alle Companies, in denen das Subject Mitglied ist).
Query (SOLL): `includeTrashed`, Paging/Sort.

6.2 Locations (company-bezogen, empfohlen für UI)

GET `/api/v1/companies/{companyId}/locations` (SOLL)
Zweck: Locations einer Company listen.
Query (SOLL): `status`, `nameContains`, `includeTrashed`, Paging/Sort.

POST `/api/v1/companies/{companyId}/locations` (SOLL)
Zweck: Location anlegen (default OPEN).

6.3 Location (ID-basiert)

GET `/api/v1/location/{locationId}`
Zweck: Location laden (inkl. „ist Hauptlocation?“).

PUT `/api/v1/location/{locationId}`
Zweck: Location ändern (Name, Code, timezone …).

POST `/api/v1/location/{locationId}/close`
Zweck: Location schließen (CLOSED) + Close-Audit.
Regeln: nicht Hauptlocation; nicht die letzte OPEN Location.

POST `/api/v1/location/{locationId}/reopen`
Zweck: Location wieder öffnen (OPEN).

DELETE `/api/v1/location/{locationId}`
Zweck: Soft-Delete Location.
Regeln: nicht Hauptlocation; nicht die letzte OPEN Location.

POST `/api/v1/location/{locationId}/restore`
Zweck: Location wiederherstellen.

7. DTOs (Beispiele)

7.1 Create Company (mit initialer Location)

```
{
  "name": "InnoLogic GmbH",
  "displayName": "InnoLogic",
  "timezone": "Europe/Berlin",
  "locale": "de-DE",
  "logoFileRef": "file_abc123",
  "initialLocation": {
    "name": "Bremen HQ",
    "locationCode": "HB-01",
    "timezone": "Europe/Berlin"
  }
}
```

7.2 Company Response (mit Audit + Soft-Delete)

```
{
  "companyId": "01J3Z4Z8Q9F1K2M3N4P5R6S7T8",
  "name": "InnoLogic GmbH",
  "displayName": "InnoLogic",
  "mainLocationId": "01J3Z51A7B2C3D4E5F6G7H8J9K",
  "timezone": "Europe/Berlin",
  "locale": "de-DE",
  "logoFileRef": "file_abc123",
  "createdAt": "2026-02-12T10:00:00Z",
  "createdBy": "user-77",
  "modifiedAt": "2026-02-12T10:05:00Z",
  "modifiedBy": "user-77",
  "trashedAt": null,
  "trashedBy": null,
  "version": 1
}
```

7.3 Set Main Location

```
{ "locationId": "01J3Z51A7B2C3D4E5F6G7H8J9K" }
```

7.4 Location Response (inkl. main-Flag)

```
{
  "locationId": "01J3Z51A7B2C3D4E5F6G7H8J9K",
  "companyId": "01J3Z4Z8Q9F1K2M3N4P5R6S7T8",
  "name": "Bremen HQ",
  "locationCode": "HB-01",
  "timezone": "Europe/Berlin",
  "status": "OPEN",
  "isMain": true,
  "closedAt": null,
  "closedBy": null,
  "closedReason": null,
  "createdAt": "2026-02-12T10:00:00Z",
  "createdBy": "user-77",
  "modifiedAt": "2026-02-12T10:05:00Z",
  "modifiedBy": "user-77",
  "trashedAt": null,
  "trashedBy": null,
  "version": 1
}
```

7.5 Close Location

```
{ "reason": "Zusammenlegung / Umzug" }
```

8. Caching (SOLL)

8.1 Ziele

Reduktion von DB-Last/Latenz bei häufigen Reads (Company, Locations-Listen).

8.2 Was wird gecacht?

SOLL:

- `GET /companies/{companyId}`
- `GET /companies/{companyId}/locations`
  KANN:
- `GET /location/{locationId}` (wenn sehr häufig)

8.3 TTL

- Company/Locations: 30–120 Sekunden

8.4 Invalidation (MUSS, falls Caching aktiv)

- Nach PUT/DELETE/RESTORE Company: Company-Cache + Locations-Cache invalidieren
- Nach Create/Update/Close/Reopen/Delete/Restore Location: Company-Cache (wegen main/Counts) + Locations-Cache + Location-Cache invalidieren

8.5 Mehrinstanzenbetrieb (SOLL)

- Bei ≥2 Instanzen: zentraler Cache (z. B. Redis) empfohlen.

9. Fehlerformat (SOLL)

9.1 Einheitliches Error-DTO

Mindestfelder:

- `timestamp`, `status`, `error`, `errorCode`, `message`, `path`, `correlationId`
- optional `details[]` (Feldfehler)

Abnahmekriterium: 400/401/403/404/409/500 liefern konsistent dieses Format.

Beispiel

```
{
  "timestamp": "2026-02-12T10:06:00Z",
  "status": 409,
  "error": "Conflict",
  "errorCode": "CANNOT_CLOSE_MAIN_LOCATION",
  "message": "Main location cannot be closed",
  "path": "/api/v1/location/01J...",
  "correlationId": "c-123",
  "details": []
}
```

Typische fachliche Fehlercodes (MUSS):

- `MAIN_LOCATION_REQUIRED`
- `MAIN_LOCATION_MUST_BE_OPEN`
- `CANNOT_CLOSE_MAIN_LOCATION`
- `CANNOT_TRASH_MAIN_LOCATION`
- `LAST_OPEN_LOCATION_REQUIRED`
- `COMPANY_TRASHED_OPERATION_NOT_ALLOWED`
- `OPTIMISTIC_LOCK_FAILED`

10. Pagination & Sort (SOLL)

10.1 Standard

- `page` (default 0), `size` (default 50, max z. B. 200)
- `sort` (z. B. `createdAt,DESC`)
- Response: `page/size/total/items`

11. Abnahmekriterien (Auszug)

- Company kann angelegt, geladen, geändert, soft-gelöscht und restored werden.
- Pro Company existiert genau eine Hauptlocation; Wechsel ist möglich und konsistent.
- Hauptlocation ist immer OPEN und nicht trashed.
- Eine Hauptlocation kann nicht geschlossen oder getrashed werden.
- Die letzte OPEN Location kann nicht geschlossen oder getrashed werden.
- Company Trash triggert Cascade Trash aller Locations.
- Mandantentrennung wird serverseitig erzwungen; Location-Endpunkte sind trotz fehlender companyId sicher.
- Swagger/OpenAPI ist vollständig (DoD: kein Endpoint ohne Swagger).
- Fehlerantworten sind konsistent im Error-DTO.

MUSS/SOLL/KANN Übersicht

Bereich	MUSS	SOLL	KANN
Technologie/Stack	Java+Spring Boot, REST/JSON, MariaDB, UUID/ULID	Flyway/Liquibase	—
OpenAPI/Swagger	Vollständige Doku je Endpoint (DoD)	globale Error Responses	—
Zeit/Datum	UTC ISO-8601	—	—
Mandantenmodell	Company/Location strikt company-scharf	—	—
Hauptlocation	genau eine, immer OPEN & nicht trashed	—	—
Location Status	OPEN/CLOSED, Close-Meta	reopen	—
Soft-Delete	trashedAt/By, includeTrashed	Restore, Retention/Purge	Restore-Auditfelder
Cascade Trash	Company trash → Locations trash	trashedCause zur Restore-Policy	—
Concurrency	version / Optimistic Locking	—	—
Suche	—	Locations-Liste mit Paging/Sort	erweiterte Suche
Events/Outbox	—	Outbox + Domain-Events	Broker/Advanced
Caching	—	TTL + Invalidation	Location-Cache, Search-Cache

Backlog (Epics → Stories → Tasks)

EPIC 0 – Projektgrundlagen

Story 0.1 – Repo & Grundsetup

- Spring Boot Projekt (Web, Validation, Security Placeholder, JPA/Jdbc, OpenAPI)
- MariaDB Connection + Profile (local/dev/prod)
- Standard Error DTO + Exception Mapping
- Logging + CorrelationId Filter
  Akzeptanzkriterien: App startet lokal, Swagger UI erreichbar, Healthcheck vorhanden.

Story 0.2 – Migrationen

- Flyway/Liquibase integrieren
- Baseline Migration
  Akzeptanzkriterien: Schema wird automatisiert erstellt/aktualisiert.

EPIC 1 – Datenmodell & DB

Story 1.1 – DDL Company

- Tabelle company inkl. Audit/Soft-Delete/version
  Akzeptanzkriterien: CRUD möglich, Indizes vorhanden.

Story 1.2 – DDL Location

- Tabelle location inkl. Status/Close/Audit/Soft-Delete/version
- Indizes (companyId, status, trashedAt)
- (Optional) trashedCause
  Akzeptanzkriterien: Location kann company-scharf gespeichert werden.

Story 1.3 – Optional: Outbox

- Tabelle outbox_event inkl. Status/Retry
  Akzeptanzkriterien: Event kann in DB-Transaktion geschrieben werden.

EPIC 2 – Domain/DTO/Validation

Story 2.1 – DTOs + Entities

- Create/Update DTOs Company/Location, Responses, List Response
  Akzeptanzkriterien: DTOs decken Pflichtenheft-Felder ab.

Story 2.2 – Validierung & Fehlercodes

- Bean Validation, konsistente ErrorCodes, 400/403/404/409/500
  Akzeptanzkriterien: ungültige Requests liefern korrektes Error-DTO.

Story 2.3 – Normalisierung

- nameNormalized für Company/Location
  Akzeptanzkriterien: normalized Felder werden gesetzt und getestet.

EPIC 3 – Company API

Story 3.1 – CRUD Company

- POST /companies, GET/PUT/DELETE/RESTORE /companies/{companyId}
  Akzeptanzkriterien: Audit korrekt, Soft-Delete funktioniert.

Story 3.2 – Hauptlocation wechseln

- PUT /companies/{companyId}/main-location mit Optimistic Lock
  Akzeptanzkriterien: I-1/I-2 eingehalten, Konflikte → 409.

EPIC 4 – Location API

Story 4.1 – CRUD Location

- POST /companies/{companyId}/locations
- GET/PUT/DELETE/RESTORE /location/{locationId}
  Akzeptanzkriterien: company-scharfe Prüfungen bei /location/{id}.

Story 4.2 – Close/Reopen

- close/reopen Endpoints, Regeln (nicht main, nicht letzte OPEN)
  Akzeptanzkriterien: 409 Fehlercodes bei Regelverstoß.

EPIC 5 – Cascade Trash & Restore Konsistenz

Story 5.1 – Company Trash Cascade

- Company trash setzt alle Locations trashed (CASCADE)
  Akzeptanzkriterien: keine Location bleibt sichtbar nach Company Trash (Default).

Story 5.2 – Restore Policy

- Company restore stellt mainLocation + mind. eine OPEN Location her
  Akzeptanzkriterien: konsistenter Zustand nach restore.

EPIC 6 – Security & Audit Integration

Story 6.1 – Mandantentrennung

- companyId gegen Auth-Kontext prüfen, 403 bei mismatch
  Akzeptanzkriterien: Cross-Company nicht möglich.

Story 6.2 – Audit Actor

- createdBy/modifiedBy/trashedBy/closedBy aus Security Context
  Akzeptanzkriterien: Audit korrekt.

EPIC 7 – Events/Outbox (empfohlen)

Story 7.1 – Domain Events

- Events bei Änderungen erzeugen (Outbox)
  Akzeptanzkriterien: Outbox enthält Events nach CRUD/State Changes.

EPIC 8 – Retention/Purge (optional später)

Story 8.1 – Purge Job

- trashed > X Tage endgültig löschen
  Akzeptanzkriterien: Daten werden nach Retention entfernt.

EPIC 9 – QA & Release

Story 9.1 – Tests

- Unit Tests (Regeln, Normalisierung), Integrationstests (Endpoints)
  Akzeptanzkriterien: CI grün, Kernpfade getestet.

Story 9.2 – Swagger Review

- Swagger vollständig gemäß DoD
  Akzeptanzkriterien: Review-Checkliste erfüllt.