blocpress
document automation

Element Design Concept

Goals

G-1: Automatisierte Template-basierte Dokumentengenerierung

Description: blocpress-render soll LibreOffice-Templates (ODT) mit strukturierten JSON-Daten automatisiert zusammenführen und Dokumente in den Formaten ODT, PDF und RTF generieren. Die Generierung muss deterministisch, performant und über die öffentliche REST-API zugänglich sein.
Systemziel: Automatisierte Dokumentengenerierung | Geschäftsprozess: Dokument automatisiert generieren

G-2: Zentrale Verwaltung von Templates und Bausteinen

Description: blocpress-workbench soll alle Templates, Bausteine und deren Versionen im Schema `workbench` speichern und verwalten. Änderungen müssen nachvollziehbar sein und der Freigabestatus jederzeit erkennbar.
Systemziel: Zentrale Template-Verwaltung | Entities: Information Architecture

G-3: Workflow-gesteuerte Qualitätssicherung

Description: blocpress-proof soll einen mehrstufigen Freigabeprozess mit definiertem Vier-Augen-Prinzip (Template-Designer ≠ Prüfer) implementieren. Templates dürfen nur nach erfolgreicher Prüfung in das production-Schema übernommen werden.
Systemziel: Workflow-gesteuerte Qualitätssicherung | Geschäftsprozess: Template erstellen und freigeben

G-4: Turnusmäßige Compliance-Reviews

Description: blocpress-proof soll automatisch Review-Zyklen (1, 3 oder 5 Jahre) überwachen, fällige Reviews identifizieren und für Prüfer sichtbar machen.

Systemziel: Compliance-Unterstützung | Geschäftsprozess: Turnusmäßiges Compliance Review

G-5: Automatisierte Regressionstests

Description: blocpress-proof soll Testdaten verwalten, Test-PDFs generieren (via blocpress-render), mit Baselines vergleichen und Abweichungen erkennen.
Systemziel: Umfassende Testunterstützung | Geschäftsprozess: Template testen und Regressionstests

G-6: Performante Inhaltssuche

Description: blocpress-workbench soll eine Integration mit Elasticsearch bereitstellen, um fachliche Konstrukte über alle Templates und Bausteine hinweg innerhalb von 2 Sekunden zu finden.
Systemziel: Intelligente Inhaltssuche | Geschäftsprozess: Fachliche Konstrukte recherchieren

User Interfaces

UI-1: Template-Verwaltungsoberfläche (blocpress-workbench)

Description: Web Component `<bp-workbench>` zum Hochladen, Bearbeiten, Anzeigen und Verwalten von Templates. Wird von Template-Designern genutzt. Geladen via Import Map in blocpress-studio.

Actions:

- Template hochladen (UC-1)
- Template zur Freigabe einreichen (UC-2)
- Template-Details anzeigen (UC-3)
- Template-Versionen vergleichen (UC-4)
- Baustein erstellen/bearbeiten (UC-5)

Visible Data:

- Template-Liste: Name, Version, Status, Ersteller, Erstellungsdatum (aus E-1: Template)
- Template-Details: ID, Name, Beschreibung, Version, Status, User-Fields, Wiederholungsgruppen, Bedingungen (aus E-1: Template)
- Validierungsergebnisse: Fehler, Warnungen (aus TF-1: Template validieren)
- Baustein-Liste: Name, Typ, verwendende Templates (aus E-2: Baustein)

UI-2: Freigabe-Oberfläche (blocpress-proof)

Description: Bestandteil des Web Components `<bp-proof>` zur Prüfung und Freigabe von Templates. Wird von Prüfern genutzt. Geladen via Import Map in blocpress-studio.

Actions:

- Template zur Prüfung öffnen (UC-6)
- Kommentar hinzufügen (UC-7)
- Template freigeben (UC-8)
- Template ablehnen (UC-9)
- Testdokument mit Template generieren (UC-10)

Visible Data:

- Eingereichte Templates: Name, Version, Einreicher, Einreichdatum (aus E-1: Template, E-4: Freigabeprozess)
- Template-Details zur Prüfung: Komplette Metadaten, Validierungsergebnisse (aus E-1: Template)
- Freigabe-Historie: Prüfer, Kommentare, Datum, Entscheidung (aus E-4: Freigabeprozess)

UI-3: Compliance-Review-Oberfläche (blocpress-proof)

Description: Bestandteil des Web Components `<bp-proof>` zur Durchführung turnusmäßiger Reviews. Wird von Prüfern genutzt.

Actions:

- Fällige Reviews anzeigen (UC-11)
- Review durchführen (UC-12)
- Review-Ergebnis speichern (UC-13)

Visible Data:

- Review-Liste: Template-Name, Review-Typ, Fälligkeitsdatum, Status (aus E-5: Compliance Review)
- Template-Inhalte zur Review (aus E-1: Template)
- Review-Historie: Vergangene Reviews, Kommentare (aus E-5: Compliance Review)

UI-4: Test-Management-Oberfläche (blocpress-proof)

Description: Bestandteil des Web Components `<bp-proof>` zur Verwaltung von Testdaten und Regressionstests. Wird von Prüfern genutzt.

Actions:

- Test Case erstellen (UC-14)
- Testdaten bearbeiten (UC-15)
- Test ausführen (UC-16)
- Baseline aktualisieren (UC-17)

- Testpool verwalten (UC-18)

Visible Data:

- Test Cases: Name, Template, Status, letzte Ausführung (aus E-6: Test Case)
- Testdaten: JSON-Struktur (aus E-6: Test Case)
- Test-Ergebnisse: Bestanden/Fehlgeschlagen, Abweichungen (aus E-6: Test Case)
- Testpools: Name, enthaltene Tests, Erfolgsrate (aus E-7: Testpool)

UI-5: Such-Oberfläche (blocpress-workbench)

Description: Bestandteil des Web Components `<bp-workbench>` zur Recherche fachlicher Konstrukte. Wird von Template-Designern genutzt.

Actions:

- Nach Begriff suchen (UC-19)
- Suchergebnisse filtern (UC-20)
- Template an Fundstelle öffnen (UC-21)

Visible Data:

- Suchergebnisse: Template-Name, Baustein-Name, Kontext, Position (aus E-8: Content Index via TI-7)
- Suchfilter: Template-Typ, Baustein-Kategorie, Status (aus E-1, E-2)

UI-6: Administrator-Oberfläche (blocpress-admin)

Description: Web Component `<bp-admin>` zur Verwaltung von Benutzern, Rollen und System-Konfiguration. Wird von Administratoren genutzt. Geladen via Import Map in blocpress-studio.

Actions:

- Benutzer verwalten (UC-22)
- Rollen zuweisen (UC-23)
- System-Konfiguration ändern (UC-24)

Visible Data:

- Benutzerliste: Name, E-Mail, Rollen, Status (aus E-9: Benutzer)
- Rollendefinitionen: Name, Berechtigungen (aus E-9: Benutzer)
- System-Parameter: Review-Zyklen, Limits, API-Keys (aus E-10: Konfiguration)

Use Cases

UC-1: Template hochladen

Goal: G-2: Zentrale Verwaltung von Templates und Bausteinen

Prerequisites:

- Benutzer ist authentifiziert
- Benutzer hat Rolle "Template-Designer"

Actors:

- Template-Designer (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)

Main Scenario:

1. Template-Designer wählt über UI-1 die Aktion "Template hochladen"
2. Template-Designer wählt eine ODT-Datei vom lokalen Dateisystem
3. Web-Oberfläche sendet die Datei über TI-1 (REST-API: POST /api/templates) an blocpress-server
4. blocpress-server ruft TF-1 (Template validieren) auf
5. TF-1 analysiert User-Fields, Wiederholungsgruppen und IF-Bedingungen
6. blocpress-server speichert Template-Binärdaten und Metadaten in E-1 (Template) mit Status "Entwurf"
7. blocpress-server sendet Indexierungs-Anfrage über TI-7 (Elasticsearch API) an Content Search Engine
8. blocpress-server gibt Template-ID und Validierungsergebnisse an Web-Oberfläche zurück
9. Web-Oberfläche zeigt Template-Details und Validierungsergebnisse auf UI-1 an

Alternative Scenarios:

4a. Validierung schlägt fehl (ungültige ODT-Struktur) 5a. blocpress-server gibt Fehlermeldung zurück 6a. Web-Oberfläche zeigt Fehlermeldung auf UI-1 an 7a. Use Case endet ohne Template-Speicherung

UC-2: Template zur Freigabe einreichen

Goal: G-3: Workflow-gesteuerte Qualitätssicherung

Prerequisites:

- UC-1 (Template hochladen) erfolgreich abgeschlossen

- Template hat Status "Entwurf"

Actors:

- Template-Designer (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)

Main Scenario:

1. Template-Designer wählt Template auf UI-1 aus
2. Template-Designer wählt Aktion "Zur Freigabe einreichen"
3. Web-Oberfläche sendet Anfrage über TI-1 (REST-API: POST /api/templates/{id}/submit) an blocpress-server
4. blocpress-server ruft TF-2 (Workflow-Status ändern) auf
5. TF-2 prüft, ob Benutzer berechtigt ist (Designer ≠ Freigeber)
6. TF-2 ändert Status in E-1 (Template) von "Entwurf" zu "Eingereicht"
7. TF-2 erstellt Eintrag in E-4 (Freigabeprozess) mit Status "Eingereicht"
8. blocpress-server bestätigt erfolgreiche Einreichung an Web-Oberfläche
9. Web-Oberfläche zeigt Bestätigung auf UI-1 an
10. Eingereichte Templates sind für Qualitätsmanager auf UI-2 sichtbar

Alternative Scenarios:

5a. Benutzer ist auch Freigeber (Vier-Augen-Prinzip verletzt) 6a. TF-2 gibt Fehlermeldung zurück 7a. Web-Oberfläche zeigt Fehlermeldung auf UI-1 an 8a. Use Case endet ohne Statusänderung

UC-3: Template-Details anzeigen

Goal: G-2: Zentrale Verwaltung von Templates und Bausteinen

Prerequisites:

- Benutzer ist authentifiziert
- Mindestens ein Template existiert

Actors:

- Template-Designer (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)

Main Scenario:

1. Template-Designer wählt Template aus Liste auf UI-1 aus

2. Web-Oberfläche sendet Anfrage über TI-1 (REST-API: GET /api/templates/{id}) an blocpress-server
3. blocpress-server lädt Template-Metadaten aus E-1 (Template)
4. blocpress-server lädt zugehörige Bausteine aus E-2 (Baustein)
5. blocpress-server gibt vollständige Template-Informationen an Web-Oberfläche zurück
6. Web-Oberfläche zeigt Template-Details auf UI-1 an (Name, Version, Status, User-Fields, Wiederholungsgruppen, Bausteine)

UC-8: Template freigeben

Goal: G-3: Workflow-gesteuerte Qualitätssicherung

Prerequisites:

- Template hat Status "Eingereicht" oder "In Prüfung"
- Benutzer hat Rolle "Freigeber"
- UC-2 (Template zur Freigabe einreichen) erfolgreich abgeschlossen

Actors:

- Qualitätsmanager/Freigeber (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)

Main Scenario:

1. Freigeber öffnet eingreichtes Template auf UI-2
2. Freigeber prüft Template-Inhalt und Validierungsergebnisse
3. Freigeber wählt Aktion "Freigeben"
4. Web-Oberfläche sendet Anfrage über TI-1 (REST-API: POST /api/templates/{id}/approve) an blocpress-server
5. blocpress-server ruft TF-2 (Workflow-Status ändern) auf
6. TF-2 prüft, ob Benutzer berechtigt ist und nicht der ursprüngliche Designer ist
7. TF-2 ändert Status in E-1 (Template) zu "Freigegeben"
8. TF-2 aktualisiert E-4 (Freigabeprozess) mit Freigeber, Freigabedatum und Status "Freigegeben"
9. blocpress-server ruft TF-4 (Review-Datum berechnen) auf
10. TF-4 erstellt Eintrag in E-5 (Compliance Review) mit berechnetem nächsten Review-Datum
11. blocpress-server bestätigt erfolgreiche Freigabe an Web-Oberfläche
12. Web-Oberfläche zeigt Bestätigung auf UI-2 an

Alternative Scenarios:

6a. Freigeber ist identisch mit Designer (Vier-Augen-Prinzip verletzt) 7a. TF-2 gibt Fehlermeldung zurück 8a. Web-Oberfläche zeigt Fehlermeldung auf UI-2 an 9a. Use Case endet ohne Freigabe

UC-10: Testdokument mit Template generieren

Goal: G-3: Workflow-gesteuerte Qualitätssicherung

Prerequisites:

- Template existiert
- Benutzer hat Rolle "Qualitätsmanager" oder "Test-Manager"

Actors:

- Qualitätsmanager (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)
- Message Queue (Software Element aus System Design)

Main Scenario:

1. Qualitätsmanager öffnet Template auf UI-2
2. Qualitätsmanager wählt Aktion "Testdokument generieren"
3. Qualitätsmanager gibt JSON-Testdaten ein oder wählt vorhandene Testdaten
4. Web-Oberfläche sendet Anfrage über TI-1 (REST-API: POST /api/documents/generate) mit Template-ID, JSON-Daten und gewünschtem Format (ODT/PDF/RTF)
5. blocpress-server validiert JSON-Daten gegen Template-Schema
6. blocpress-server erstellt Eintrag in E-3 (Dokumentengenerierung) mit Status "In Bearbeitung"
7. blocpress-server sendet Generierungsanfrage über TI-5 (Message Queue) asynchron
8. blocpress-server gibt Generierungs-ID an Web-Oberfläche zurück
9. Web-Oberfläche zeigt "Generierung läuft" auf UI-2 an
10. blocpress-server ruft TF-5 (Dokument generieren) auf (asynchron via Message Queue)
11. TF-5 lädt Template aus E-1 (Template)
12. TF-5 befüllt Template mit JSON-Daten (User-Fields, Wiederholungsgruppen, IF-Bedingungen)
13. TF-5 exportiert Dokument im gewünschten Format (ODT/PDF/RTF) via LibreOffice headless
14. blocpress-server speichert generiertes Dokument in E-3 (Dokumentengenerierung)
15. blocpress-server aktualisiert Status in E-3 auf "Erfolgreich"
16. Web-Oberfläche pollt Status über TI-1 (REST-API: GET /api/documents/{id}/status)
17. blocpress-server gibt Download-Link zurück
18. Web-Oberfläche zeigt Download-Link auf UI-2 an

Alternative Scenarios:

5a. JSON-Daten entsprechen nicht dem Template-Schema 6a. blocpress-server gibt Validierungsfehler zurück 7a. Web-Oberfläche zeigt Fehlermeldung auf UI-2 an 8a. Use Case endet ohne Dokumentengenerierung

13a. LibreOffice-Export schlägt fehl 14a. blocpress-server aktualisiert Status in E-3 auf "Fehlgeschlagen" mit Fehlerprotokoll 15a. Web-Oberfläche zeigt Fehlermeldung auf UI-2 an 16a. Use Case endet mit Fehler

UC-12: Review durchführen

Goal: G-4: Turnusmäßige Compliance-Reviews

Prerequisites:

- Compliance Review ist fällig (Status "Anstehend")
- Benutzer hat Rolle "Compliance-Reviewer"

Actors:

- Compliance-Reviewer (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)

Main Scenario:

1. Compliance-Reviewer öffnet fälliges Review auf UI-3
2. Web-Oberfläche sendet Anfrage über TI-1 (REST-API: GET /api/reviews/{id}) an blocpress-server
3. blocpress-server lädt Review-Details aus E-5 (Compliance Review)
4. blocpress-server lädt zugehöriges Template aus E-1 (Template)
5. Web-Oberfläche zeigt Template-Inhalte und Review-Informationen auf UI-3 an
6. Compliance-Reviewer prüft inhaltliche Aktualität gegen rechtliche/fachliche Anforderungen
7. Compliance-Reviewer gibt Review-Ergebnis ein ("Freigegeben" oder "Überarbeitung erforderlich")
8. Compliance-Reviewer fügt Kommentare hinzu
9. Web-Oberfläche sendet Review-Ergebnis über TI-1 (REST-API: POST /api/reviews/{id}/complete) an blocpress-server
10. blocpress-server aktualisiert E-5 (Compliance Review) mit Reviewer, Review-Datum, Ergebnis und Kommentaren
11. blocpress-server ruft TF-4 (Review-Datum berechnen) auf
12. TF-4 berechnet nächstes Review-Datum und speichert es in E-5
13. blocpress-server bestätigt erfolgreiche Review-Speicherung an Web-Oberfläche

14. Web-Oberfläche zeigt Bestätigung auf UI-3 an

Alternative Scenarios:

7a. Review-Ergebnis ist "Archivieren" 8a. blocpress-server ändert Template-Status in E-1 zu "Archiviert" 9a. Template wird nicht mehr für Produktiv-Generierung verwendet 10a. Weiter mit Schritt 13

UC-14: Test Case erstellen

Goal: G-5: Automatisierte Regressionstests

Prerequisites:

- Template existiert und ist freigegeben
- Benutzer hat Rolle "Test-Manager"

Actors:

- Test-Manager (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)

Main Scenario:

1. Test-Manager wählt Template auf UI-4 aus
2. Test-Manager wählt Aktion "Test Case erstellen"
3. Test-Manager gibt Test-Name und Beschreibung ein
4. Test-Manager erstellt oder lädt JSON-Testdaten
5. Test-Manager wählt "Test ausführen und als Baseline speichern"
6. Web-Oberfläche sendet Anfrage über TI-1 (REST-API: POST /api/testcases) mit Template-ID, Name, Beschreibung und JSON-Testdaten
7. blocpress-server erstellt Eintrag in E-6 (Test Case) mit Status "Baseline fehlt"
8. blocpress-server ruft TF-5 (Dokument generieren) auf mit JSON-Testdaten
9. TF-5 generiert PDF-Dokument
10. blocpress-server speichert generiertes PDF als Baseline in E-6 (Test Case)
11. blocpress-server aktualisiert Status in E-6 auf "Bestanden"
12. blocpress-server gibt Test-Case-ID an Web-Oberfläche zurück
13. Web-Oberfläche zeigt Test Case Details und Baseline-PDF auf UI-4 an

UC-16: Test ausführen

Goal: G-5: Automatisierte Regressionstests

Prerequisites:

- Test Case existiert mit Baseline
- Benutzer hat Rolle "Test-Manager" oder Test wird automatisch ausgeführt

Actors:

- Test-Manager (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)
- Message Queue (Software Element aus System Design)

Main Scenario:

1. Test-Manager wählt Test Case auf UI-4 aus
2. Test-Manager wählt Aktion "Test ausführen"
3. Web-Oberfläche sendet Anfrage über TI-1 (REST-API: POST /api/testcases/{id}/execute) an blocpress-server
4. blocpress-server lädt Test Case aus E-6 (Test Case)
5. blocpress-server ruft TF-5 (Dokument generieren) auf mit JSON-Testdaten aus E-6
6. TF-5 generiert aktuelles PDF-Dokument
7. blocpress-server ruft TF-6 (PDF vergleichen) auf mit Baseline-PDF und aktuellem PDF
8. TF-6 vergleicht PDFs pixel- oder textbasiert
9. TF-6 gibt Vergleichsergebnis zurück (Bestanden/Fehlgeschlagen, Abweichungen)
10. blocpress-server speichert aktuelles PDF und Vergleichsergebnis in E-6 (Test Case)
11. blocpress-server aktualisiert Status und letzte Ausführung in E-6
12. blocpress-server gibt Test-Ergebnis an Web-Oberfläche zurück
13. Web-Oberfläche zeigt Test-Ergebnis auf UI-4 an (bei Abweichungen: visueller Vergleich)

Alternative Scenarios:

8a. PDFs sind identisch 9a. TF-6 gibt "Bestanden" zurück 10a. Weiter mit Schritt 10

8b. PDFs unterscheiden sich 9b. TF-6 gibt "Fehlgeschlagen" mit Abweichungsdetails zurück 10b. Weiter mit Schritt 10

UC-19: Nach Begriff suchen

Goal: G-6: Performante Inhaltssuche

Prerequisites:

- Benutzer ist authentifiziert

- Content Search Engine ist aktiv und indexiert

Actors:

- Template-Designer (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)
- Content Search Engine (Software Element aus System Design)

Main Scenario:

1. Template-Designer gibt Suchbegriff (z.B. "Beitragsanpassung") in Suchfeld auf UI-5 ein
2. Web-Oberfläche sendet Such-Anfrage über TI-1 (REST-API: GET /api/search?query=...) an blocpress-server
3. blocpress-server sendet Such-Anfrage über TI-7 (Elasticsearch API) an Content Search Engine
4. Content Search Engine durchsucht Index und gibt Treffer zurück (Template-ID, Baustein-ID, Kontext, Position)
5. blocpress-server reichert Treffer mit Metadaten aus E-1 (Template) und E-2 (Baustein) an
6. blocpress-server gibt Suchergebnisse an Web-Oberfläche zurück
7. Web-Oberfläche zeigt Suchergebnisse auf UI-5 an (gruppiert nach Templates und Bausteinen)

Alternative Scenarios:

4a. Keine Treffer gefunden 5a. Content Search Engine gibt leere Liste zurück 6a. blocpress-server gibt "Keine Ergebnisse" an Web-Oberfläche zurück 7a. Web-Oberfläche zeigt "Keine Ergebnisse gefunden" auf UI-5 an

UC-22: Benutzer verwalten

Goal: G-3: Workflow-gesteuerte Qualitätssicherung (unterstützend)

Prerequisites:

- Benutzer hat Rolle "Administrator"

Actors:

- Administrator (User Type aus System Design)
- Web-Oberfläche (Software Element aus System Design)
- blocpress-server (dieses Element)

Main Scenario:

1. Administrator öffnet Benutzerverwaltung auf UI-6
2. Web-Oberfläche sendet Anfrage über TI-1 (REST-API: GET /api/users) an blocpress-server

3. blocpress-server lädt Benutzerliste aus E-9 (Benutzer)
4. Web-Oberfläche zeigt Benutzerliste auf UI-6 an
5. Administrator wählt Benutzer aus und ändert Rollen
6. Web-Oberfläche sendet Update über TI-1 (REST-API: PUT /api/users/{id}/roles) an blocpress-server
7. blocpress-server aktualisiert Rollen in E-9 (Benutzer) und speichert das JWT Subject (sub-Claim) als Referenz
8. blocpress-server bestätigt erfolgreiche Aktualisierung an Web-Oberfläche
9. Web-Oberfläche zeigt Bestätigung auf UI-6 an

Technical Functions

TF-1: Template validieren

Input:

- templateBinary: byte[] (ODT-Datei)

Output:

- validationResult: ValidationResult
- isValid: boolean
- errors: List<ValidationError>
- warnings: List<ValidationWarning>
- userFields: List<UserField>
- repetitionGroups: List<RepetitionGroup>
- conditions: List<Condition>

Main Functional Flow:

1. Lade templateBinary als ODT-Dokument über LibreOffice API
2. Prüfe, ob Dokument gültige ODT-Struktur hat
3. Extrahiere alle User-Fields aus dem Dokument
4. Für jedes User-Field: Prüfe, ob Name in gültiger Punkt-Notation ist (z.B. "kunde.name")
5. Identifiziere alle Sections und Tables im Dokument
6. Für jede Section/Table: Prüfe, ob sie als Wiederholungsgruppe markiert ist (enthält Array-Referenz)
7. Extrahiere alle IF-Bedingungen (OpenDocument-Felder mit Condition)
8. Validiere, dass alle referenzierten Felder in Bedingungen existieren
9. Erstelle ValidationResult mit isValid=true/false, Liste von Fehlern, Warnungen und extrahierten

Elementen

10. Gebe ValidationResult zurück

Alternative Functional Flows:

2a. ODT-Struktur ist ungültig (keine valide ZIP-Datei oder fehlendes content.xml) 3a. Setze isValid=false 4a. Füge Fehler "Ungültige ODT-Datei" zu errors hinzu 5a. Gebe ValidationResult zurück

4a. User-Field hat ungültige Punkt-Notation (z.B. enthält Leerzeichen) 5a. Füge Warnung zu warnings hinzu 6a. Fahre mit nächstem Field fort

8a. IF-Bedingung referenziert nicht existierendes Field 9a. Füge Fehler zu errors hinzu 10a. Setze isValid=false

TF-2: Workflow-Status ändern

Input:

- templateId: Long
- newStatus: String ("Entwurf", "Eingereicht", "In Prüfung", "Freigegeben", "Abgelehnt", "Archiviert")
- userId: Long
- comment: String (optional)

Output:

- success: boolean
- errorMessage: String (optional)

Main Functional Flow:

1. Lade Template mit templateId aus E-1 (Template)
2. Lade aktuellen Status des Templates
3. Lade Benutzer mit userId aus E-9 (Benutzer)
4. Prüfe, ob Statusübergang erlaubt ist (z.B. "Entwurf" → "Eingereicht" erlaubt)
5. Prüfe Vier-Augen-Prinzip: Wenn newStatus="Freigegeben", prüfe dass userId ≠ Template.erstellerId
6. Aktualisiere Status in E-1 (Template) auf newStatus
7. Wenn Freigabeprozess in E-4 (Freigabeprozess) für Template existiert, aktualisiere ihn
8. Sonst erstelle neuen Eintrag in E-4 mit templateId, Status, userId, Datum
9. Wenn comment vorhanden, speichere in E-4
10. Setze success=true
11. Gebe success zurück

Alternative Functional Flows:

1a. Template mit templateId existiert nicht 2a. Setze success=false, errorMessage="Template nicht gefunden" 3a. Gebe success und errorMessage zurück

4a. Statusübergang nicht erlaubt (z.B. "Freigegeben" → "Entwurf") 5a. Setze success=false, errorMessage="Statusübergang nicht erlaubt" 6a. Gebe success und errorMessage zurück

5a. Vier-Augen-Prinzip verletzt (userId = Template.erstellerId) 6a. Setze success=false, errorMessage="Designer darf nicht freigeben (Vier-Augen-Prinzip)" 7a. Gebe success und errorMessage zurück

TF-4: Review-Datum berechnen

Input:

- templateId: Long
- reviewCycle: Integer (1, 3 oder 5 Jahre)

Output:

- nextReviewDate: LocalDate

Main Functional Flow:

1. Lade Template mit templateId aus E-1 (Template)
2. Lade aktuelles Datum (heute)
3. Berechne nextReviewDate = heute + reviewCycle Jahre
4. Gebe nextReviewDate zurück

TF-5: Dokument generieren

Input:

- templateId: Long
- templateVersion: Integer (optional, default: neueste freigegebene Version)
- jsonData: String (JSON-Struktur)
- outputFormat: String ("ODT", "PDF", "RTF")

Output:

- documentBinary: byte[]
- success: boolean
- errorMessage: String (optional)

Main Functional Flow:

1. Lade Template mit templateId und templateVersion aus E-1 (Template)
2. Prüfe, ob Template Status "Freigegeben" hat
3. Lade Template-Binärdaten aus E-1
4. Parse jsonData zu JSON-Objekt
5. Öffne Template mit LibreOffice headless API
6. Für jedes User-Field im Template: Extrahiere Wert aus JSON via Punkt-Notation und setze Field-Wert
7. Für jede Wiederholungsgruppe (Section/Table): Iteriere über Array aus JSON und dupliziere Gruppe
8. Für jede IF-Bedingung: Evaluiere Bedingung basierend auf JSON-Daten und zeige/verstecke Inhalt
9. Exportiere Dokument im gewünschten outputFormat (ODT, PDF oder RTF) über LibreOffice API
10. Konvertiere exportiertes Dokument zu byte[]
11. Schließe LibreOffice-Dokument
12. Setze success=true
13. Gebe documentBinary und success zurück

Alternative Functional Flows:

2a. Template hat nicht Status "Freigegeben" 3a. Setze success=false, errorMessage="Template nicht freigegeben" 4a. Gebe success und errorMessage zurück

4a. jsonData ist kein valides JSON 5a. Setze success=false, errorMessage="Ungültige JSON-Daten" 6a. Gebe success und errorMessage zurück

6a. Referenziertes Feld in Punkt-Notation existiert nicht in JSON 7a. Setze Field-Wert auf "" (leerer String) oder Default-Wert 8a. Fahre mit nächstem Field fort

9a. LibreOffice-Export schlägt fehl 10a. Setze success=false, errorMessage="Export fehlgeschlagen: " + Fehlerdetails 11a. Schließe LibreOffice-Dokument 12a. Gebe success und errorMessage zurück

TF-6: PDF vergleichen

Input:

- baselinePdf: byte[]
- currentPdf: byte[]

Output:

- comparisonResult: ComparisonResult
- isIdentical: boolean
- differences: List<Difference> (Seite, Position, Beschreibung)

Main Functional Flow:

1. Lade baselinePdf und currentPdf als PDF-Dokumente
2. Prüfe, ob beide PDFs gleiche Seitenanzahl haben
3. Für jede Seite: Extrahiere Text aus baselinePdf und currentPdf
4. Vergleiche Text Zeile für Zeile
5. Bei Unterschieden: Erstelle Difference-Objekt mit Seitennummer, Position und Beschreibung
6. Füge Difference zu differences-Liste hinzu
7. Nach Vergleich aller Seiten: Wenn differences leer, setze isIdentical=true, sonst false
8. Erstelle ComparisonResult mit isIdentical und differences
9. Gebe ComparisonResult zurück

Alternative Functional Flows:

2a. Seitenanzahl unterschiedlich 3a. Setze isIdentical=false 4a. Erstelle Difference "Seitenanzahl unterschiedlich: Baseline X Seiten, Current Y Seiten" 5a. Gebe ComparisonResult zurück

1a. baselinePdf oder currentPdf ist ungültiges PDF 2a. Setze isIdentical=false 3a. Erstelle Difference "PDF-Vergleich fehlgeschlagen: Ungültiges PDF" 4a. Gebe ComparisonResult zurück

TF-7: Compliance-Reviews überprüfen

Input:

- keines (läuft als geplanter Job)

Output:

- fälligeReviews: List<ComplianceReview>

Main Functional Flow:

1. Lade aktuelles Datum (heute)
2. Lade alle Templates mit Status "Freigegeben" aus E-1 (Template)
3. Für jedes Template: Lade zugehörige Compliance Reviews aus E-5 (Compliance Review)
4. Wenn kein Review existiert oder letztes Review abgeschlossen: Berechne nächstes Review-Datum via TF-4
5. Wenn nächstes Review-Datum ⇐ heute + 30 Tage oder bereits überschritten:
6. Erstelle neuen Review-Eintrag in E-5 mit Status "Anstehend"
7. Füge Review zu fälligeReviews-Liste hinzu
8. Fällige Reviews sind für Compliance-Reviewer auf UI-3 sichtbar
9. Gebe fälligeReviews zurück

Technical Interfaces

TI-1: REST-API

Description: Haupt-REST-API für alle Client-Anfragen (Web-Oberfläche, externe Anwendungen). Wird von Web-Oberfläche (Software Element) und externen Anwendungssystemen genutzt.

Input:

Verschiedene Endpoints:

- `POST /api/templates` - Template hochladen
- Parameter: file (multipart/form-data), name (String), description (String)
- `POST /api/templates/{id}/submit` - Template zur Freigabe einreichen
- Parameter: id (Long)
- `GET /api/templates/{id}` - Template-Details abrufen
- Parameter: id (Long)
- `POST /api/templates/{id}/approve` - Template freigeben
- Parameter: id (Long), comment (String, optional)
- `POST /api/documents/generate` - Dokument generieren
- Parameter: templateId (Long), jsonData (String), outputFormat (String)
- `GET /api/documents/{id}/status` - Generierungsstatus abrufen
- Parameter: id (Long)
- `POST /api/testcases` - Test Case erstellen
- Parameter: templateId (Long), name (String), jsonData (String)
- `POST /api/testcases/{id}/execute` - Test ausführen
- Parameter: id (Long)
- `GET /api/search` - Nach Begriff suchen
- Parameter: query (String), filter (String, optional)

Output:

JSON-Response je nach Endpoint:

- Template-Objekt mit Metadaten
- Generierungs-ID
- Test-Ergebnis
- Suchergebnisse

Action:

Ruft entsprechende Technical Functions auf:

- Template hochladen: TF-1 (Template validieren), Schreibt E-1 (Template)
- Template freigeben: TF-2 (Workflow-Status ändern), Aktualisiert E-1, E-4
- Dokument generieren: TF-5 (Dokument generieren), Schreibt E-3
- Test ausführen: TF-5 (Dokument generieren), TF-6 (PDF vergleichen), Aktualisiert E-6

TI-2: Datenbank-Interface (PostgreSQL)

Description: Zugriff auf PostgreSQL-Datenbank zum Lesen und Schreiben aller Entities. Wird intern von blocpress-server genutzt.

Input:

SQL-Queries über JPA/Hibernate

Output:

Entity-Objekte oder Anzahl betroffener Zeilen

Action:

CRUD-Operationen auf alle Entities (E-1 bis E-10)

TI-3: LibreOffice API

Description: Zugriff auf LibreOffice headless (Version ≥ 24) zur Verarbeitung von ODT-Templates. Wird intern von TF-1, TF-5 genutzt.

Input:

- ODT-Dokument (byte[])
- Export-Format (ODT/PDF/RTF)
- Field-Werte (Map<String, String>)

Output:

- Exportiertes Dokument (byte[])
- Validierungsergebnisse

Action:

Ruft LibreOffice UNO-API auf:

- Dokument öffnen: `XComponentLoader.loadComponentFromURL()`
- Fields setzen: `XTextFieldsSupplier.getTextFields()`
- Dokument exportieren: `XStorable.storeToURL()` mit Filter für PDF/RTF

TI-4: JWT-Authentifizierung

Description: blocpress-server validiert eingehende JWTs direkt, ohne einen externen Identity Provider zu benötigen. Der öffentliche Schlüssel (Public Key) und der erwartete Issuer werden in `application.properties` konfiguriert. Die Umsetzung erfolgt über die Quarkus SmallRye JWT Extension (`smallrye-jwt`). Für Testzwecke kann ein langlebiger Demo-JWT mit dem konfigurierten Schlüssel erzeugt werden.

Input:

- JWT Bearer Token (im `Authorization`-Header)

Output:

- Authentifizierungsergebnis (success/failure)
- Benutzerinformationen aus JWT-Claims (sub, name, email, groups)

Action:

Validiert JWT-Signatur gegen den konfigurierten Public Key, prüft Issuer und Expiration. Extrahiert das Subject (`sub`-Claim) und gleicht es mit E-9 (Benutzer) ab.

TI-5: Message Queue Interface

Description: Asynchrone Kommunikation für zeitaufwändige Operationen (Dokumentengenerierung, Regressionstests). Wird intern von blocpress-server genutzt.

Input:

- Queue-Name (String)
- Message-Payload (JSON)

Output:

- Message-ID

Action:

Sendet Message an RabbitMQ/Kafka Queue, Consumer in blocpress-server verarbeitet Message asynchron

TI-7: Elasticsearch API

Description: Zugriff auf Content Search Engine zur Indexierung und Suche. Wird von blocpress-server für UC-19 (Nach Begriff suchen) genutzt.

Input:

- Indexierungs-Request: templateId (Long), content (String)

- Such-Request: query (String)

Output:

- Indexierungs-Bestätigung
- Such-Ergebnisse: List<SearchHit> mit templateId, baustein_id, kontext, position

Action:

Kommuniziert mit Elasticsearch über REST-API:

- Indexierung: `PUT /templates/_doc/{id}`
- Suche: `GET /templates/_search?q={query}`

Entities

E-1: Template

Description: Speichert Metadaten und Binärdaten von LibreOffice-Templates sowie deren Versionen und Status. Existiert mit identischer Struktur in drei Schemata: `workbench` (Entwurf), `proof` (Prüfung), `production` (Freigegeben).
Geschäftsentität: Template in der Information Architecture

Attributes:

ID	Name	Data Type	Description
1	id	Long	Eindeutige Template-ID (Primary Key)
2	name	String(255)	Name des Templates
3	description	String(2000)	Beschreibung des Templates
4	version	Integer	Versionsnummer (1, 2, 3, ...)
5	status	String(50)	Status: "Entwurf", "Eingereicht", "In Prüfung", "Freigegeben", "Abgelehnt", "Archiviert"
6	ersteller_id	Long	Foreign Key zu E-9 (Benutzer)
7	erstellungsdatum	Timestamp	Erstellungszeitpunkt
8	freigeber_id	Long	Foreign Key zu E-9 (Benutzer), nullable

ID	Name	Data Type	Description
9	freigabedatum	Timestamp	Freigabezeitpunkt, nullable
10	template_binary	bytea	ODT-Datei als Binärdaten (PostgreSQL BLOB)
11	user_fields	JSONB	Extrahierte User-Fields aus Validierung
12	wiederholungsgruppen	JSONB	Extrahierte Wiederholungsgruppen
13	bedingungen	JSONB	Extrahierte IF-Bedingungen
14	review_zyklus	Integer	Review-Zyklus in Jahren (1, 3 oder 5)

E-2: Baustein

Description: Speichert wiederverwendbare Dokumentfragmente, die in mehreren Templates verwendet werden können. Schema: `workbench`.
Geschäftsentität: Baustein in der Information Architecture

Attributes:

ID	Name	Data Type	Description
1	id	Long	Eindeutige Baustein-ID (Primary Key)
2	name	String(255)	Name des Bausteins
3	description	String(2000)	Beschreibung
4	typ	String(50)	Typ: "Kopfzeile", "Fußzeile", "Klausel", "Abschnitt"
5	inhalt	bytea	ODT-Fragment als Binärdaten
6	version	Integer	Versionsnummer
7	ersteller_id	Long	Foreign Key zu E-9 (Benutzer)
8	erstellungsdatum	Timestamp	Erstellungszeitpunkt

E-3: Dokumentengenerierung

Description: Speichert Metadaten und Ergebnisse von Dokumentengenerierungen. Schema:

`production`.
Geschäftsentität: Dokumentengenerierung in der Information Architecture

Attributes:

ID	Name	Data Type	Description
1	id	Long	Eindeutige Generierungs-ID (Primary Key)
2	template_id	Long	Foreign Key zu E-1 (Template)
3	template_version	Integer	Version des verwendeten Templates
4	json_daten	JSONB	Eingabe-JSON-Daten
5	output_format	String(10)	"ODT", "PDF" oder "RTF"
6	generiertes_dokument	bytea	Generiertes Dokument als Binärdaten
7	zeitstempel	Timestamp	Generierungszeitpunkt
8	status	String(50)	"In Bearbeitung", "Erfolgreich", "Fehlgeschlagen"
9	fehlerprotokoll	Text	Fehlerdetails bei Fehlschlag, nullable
10	requestor_id	Long	Foreign Key zu E-9 (Benutzer) oder API-Key

E-4: Freigabeprozess

Description: Speichert Workflow-Status und Historie von Template-Freigaben.
Geschäftsentität: Freigabeprozess in der Information Architecture

Attributes:

ID	Name	Data Type	Description
1	id	Long	Eindeutige Prozess-ID (Primary Key)
2	template_id	Long	Foreign Key zu E-1 (Template)

ID	Name	Data Type	Description
3	workflow_status	String(50)	"Eingereicht", "In Prüfung", "Freigegeben", "Abgelehnt"
4	pruefer_id	Long	Foreign Key zu E-9 (Benutzer), nullable
5	pruefkommentare	Text	Kommentare des Prüfers, nullable
6	pruefdatum	Timestamp	Prüfzeitpunkt, nullable
7	freigeber_id	Long	Foreign Key zu E-9 (Benutzer), nullable
8	freigabedatum	Timestamp	Freigabezeitpunkt, nullable

E-5: Compliance Review

Description: Speichert turnusmäßige Reviews von Templates.
Geschäftsentität: Compliance Review in der Information Architecture

Attributes:

ID	Name	Data Type	Description
1	id	Long	Eindeutige Review-ID (Primary Key)
2	template_id	Long	Foreign Key zu E-1 (Template)
3	review_typ	String(20)	"1-Jahres-Review", "3-Jahres-Review", "5-Jahres-Review"
4	faelligkeitsdatum	Date	Geplantes Review-Datum
5	review_status	String(50)	"Anstehend", "In Bearbeitung", "Abgeschlossen", "Überfällig"
6	reviewer_id	Long	Foreign Key zu E-9 (Benutzer), nullable
7	review_datum	Timestamp	Tatsächliches Review-Datum, nullable

ID	Name	Data Type	Description
8	review_ergebnis	String(50)	"Freigegeben", "Überarbeitung erforderlich", "Archivieren", nullable
9	kommentare	Text	Review-Kommentare, nullable
10	naechstes_review_datum	Date	Nächstes geplantes Review-Datum, nullable

E-6: Test Case

Description: Speichert Test Cases mit Testdaten und Baseline-PDFs für Regressionstests.
Geschäftsentität: Test Case in der Information Architecture

Attributes:

ID	Name	Data Type	Description
1	id	Long	Eindeutige Test-ID (Primary Key)
2	template_id	Long	Foreign Key zu E-1 (Template)
3	template_version	Integer	Getestete Template-Version
4	test_name	String(255)	Name des Tests
5	beschreibung	String(2000)	Beschreibung
6	json_testdaten	JSONB	Test-JSON-Daten
7	erwartetes_pdf	bytea	Baseline-PDF als Binärdaten, nullable
8	aktuelles_pdf	bytea	Zuletzt generiertes PDF, nullable
9	test_status	String(50)	"Bestanden", "Fehlgeschlagen", "Baseline fehlt"
10	ersteller_id	Long	Foreign Key zu E-9 (Benutzer)
11	erstellungsdatum	Timestamp	Erstellungszeitpunkt
12	letzte_ausfuehrung	Timestamp	Zeitpunkt der letzten Ausführung, nullable

ID	Name	Data Type	Description
13	test_typ	String(50)	"Abnahmetest", "Regressionstest"
14	abweichungen	JSONB	Details zu Abweichungen bei Fehlschlag, nullable

E-7: Testpool

Description: Gruppiert mehrere Test Cases für gemeinsame Ausführung.
Geschäftsentität: Testpool in der Information Architecture

Attributes:

ID	Name	Data Type	Description
1	id	Long	Eindeutige Pool-ID (Primary Key)
2	name	String(255)	Name des Testpools
3	beschreibung	String(2000)	Beschreibung
4	letzte_ausfuehrung	Timestamp	Zeitpunkt der letzten Ausführung, nullable
5	erfolgsrate	Decimal(5,2)	Erfolgsrate in Prozent (0.00 - 100.00), nullable

E-8: Content Index

Description: Referenz auf Elasticsearch-Index für fachliche Konstrukte. Wird nicht in PostgreSQL gespeichert, sondern nur als Metadaten-Verweis.
Geschäftsentität: Content Index in der Information Architecture

Attributes:

ID	Name	Data Type	Description
1	id	String	Elasticsearch Document ID
2	template_id	Long	Referenz zu E-1 (Template)
3	baustein_id	Long	Referenz zu E-2 (Baustein), nullable
4	inhalt	Text	Volltext-Inhalt (in Elasticsearch indexiert)

ID	Name	Data Type	Description
5	letzte_aktualisierung	Timestamp	Zeitpunkt der letzten Indexierung

Hinweis: E-8 wird physisch in Elasticsearch gespeichert, nicht in PostgreSQL.

E-9: Benutzer

Description: Speichert Benutzerkonten und deren Rollen.
Geschäftsentität: Benutzer & Rolle in der Information Architecture

Attributes:

ID	Name	Data Type	Description
1	id	Long	Eindeutige Benutzer-ID (Primary Key)
2	name	String(255)	Vollständiger Name
3	email	String(255)	E-Mail-Adresse
4	rollen	JSONB	Array von Rollen: ["Template-Designer", "Prüfer", "Administrator", "API-Konsument"]
5	berechtigungen	JSONB	Array von Berechtigungen: ["Template erstellen", "Template bearbeiten", "Template prüfen", "Template freigeben", "API nutzen", "Review durchführen", "Tests verwalten"]
6	external_id	String(255)	JWT Subject (sub-Claim)
7	aktiv	Boolean	Account aktiv/deaktiviert
8	erstellungsdatum	Timestamp	Account-Erstellungszeitpunkt

E-10: Konfiguration

Description: Speichert System-Konfiguration und API-Keys.

Attributes:

ID	Name	Data Type	Description
1	id	Long	Eindeutige Konfigurations-ID (Primary Key)
2	schluessel	String(255)	Konfigurations-Schlüssel (z.B. "default_review_cycle")
3	wert	String(2000)	Konfigurations-Wert
4	datentyp	String(50)	"String", "Integer", "Boolean", "JSON"
5	beschreibung	String(2000)	Beschreibung des Parameters
6	aenderbar	Boolean	Kann zur Laufzeit geändert werden

Quality Requirements

QR-1: Performance bei Dokumentengenerierung

Description: TF-5 (Dokument generieren) muss für Standard-Dokumente (bis 20 Seiten, bis 10 Wiederholungsgruppen) innerhalb von 5 Sekunden abgeschlossen sein. Dies ist kritisch für das Benutzererlebnis bei UC-10 (Testdokument generieren) und die API-Performance bei TI-1.

QR-2: Suchgeschwindigkeit

Description: UC-19 (Nach Begriff suchen) muss innerhalb von 2 Sekunden Ergebnisse liefern, auch bei 1.000+ Templates. TI-7 (Elasticsearch API) muss entsprechend optimiert sein mit geeigneten Indizes.

QR-3: Deterministische Dokumentengenerierung

Description: TF-5 (Dokument generieren) muss deterministisch sein: Identische Eingaben (Template-Version + JSON-Daten) müssen zu identischen Ausgabedokumenten führen. Dies ist essentiell für TF-6 (PDF vergleichen) und die Regressionstests in UC-16.

QR-4: Transaktionale Integrität bei Workflow-Änderungen

Description: TF-2 (Workflow-Status ändern) muss atomar sein: Entweder alle Änderungen (E-1, E-4) werden committed oder keine. Bei Fehler muss vollständiger Rollback erfolgen.

QR-5: Vier-Augen-Prinzip

Description: TF-2 (Workflow-Status ändern) muss sicherstellen, dass Designer ≠ Freigeber. Diese Prüfung ist zwingend erforderlich für UC-8 (Template freigeben) und darf nicht umgangen werden können.

QR-6: Skalierbarkeit der Message Queue

Description: TI-5 (Message Queue Interface) muss bis zu 100 gleichzeitige Dokumentengenerierungen in der Queue verwalten können ohne Performance-Degradation.

QR-7: Datensicherheit bei Binärdaten

Description: E-1.10 (template_binary), E-3.6 (generiertes_dokument), E-6.7 (erwartetes_pdf) und E-6.8 (aktuelles_pdf) müssen verschlüsselt in PostgreSQL gespeichert werden, wenn personenbezogene Daten enthalten sein können. Zugriff nur über authentifizierte TI-2 (Datenbank-Interface).

QR-8: API-Rate-Limiting

Description: TI-1 (REST-API) muss Rate Limiting implementieren: Maximal 1.000 Requests pro Stunde pro API-Key. Dies verhindert Missbrauch und gewährleistet faire Ressourcennutzung.

QR-9: Audit-Logging

Description: Alle Aufrufe von TF-2 (Workflow-Status ändern), UC-8 (Template freigeben), UC-12 (Review durchführen) müssen in einem Audit-Log protokolliert werden (separate Tabelle, nicht modifizierbar). Log muss User-ID, Zeitstempel, Aktion und alte/neue Werte enthalten.

QR-10: Backup & Recovery für Binärdaten

Description: PostgreSQL-Datenbank inklusive aller Binärdaten (E-1.10, E-3.6, E-6.7, E-6.8) muss täglich gesichert werden. Recovery Time Objective: 4h, Recovery Point Objective: 24h.

Constraints

C-1: Quarkus Framework

Description: blocpress-server muss mit Quarkus Framework implementiert werden. Dies ermöglicht schnelle Startup-Zeiten, geringen Memory-Footprint und native Kompilierung.

Referenz: Quarkus.io Dokumentation

C-2: LibreOffice Version

Description: TF-1 (Template validieren) und TF-5 (Dokument generieren) müssen LibreOffice Version ≥ 24 headless nutzen. Ältere Versionen werden nicht unterstützt.

Referenz: LibreOffice Release Notes

C-3: PostgreSQL Version

Description: TI-2 (Datenbank-Interface) muss PostgreSQL Version ≥ 18 nutzen. JSONB-Indizes und bytea-Speicherung sind erforderlich.

Referenz: PostgreSQL 18 Release Notes

C-4: Template-Format

Description: UC-1 (Template hochladen) akzeptiert nur ODT- und OTT-Formate. Andere Formate (DOCX, DOC) werden nicht unterstützt.

Referenz: OpenDocument Format Specification (OASIS)

C-5: Export-Formate

Description: TF-5 (Dokument generieren) unterstützt nur Export nach ODT, PDF und RTF. Andere Formate (DOCX, HTML) werden nicht unterstützt.

C-6: Datenschutz (DSGVO)

Description: Alle Entities mit personenbezogenen Daten (E-3.4 json_daten, E-6.6 json_testdaten) müssen DSGVO-konform verarbeitet werden. Löschfunktion muss implementiert sein.

Referenz: DSGVO Art. 17 (Recht auf Löschung)

C-7: Storage Abstraction Layer

Description: Zugriff auf Binärdaten (E-1.10, E-3.6, E-6.7, E-6.8) muss über eine Storage Abstraction Layer erfolgen, die zukünftigen Wechsel von PostgreSQL bytea zu S3-kompatiblem Object Storage ermöglicht. Interface: `StorageService.save(byte[])` und `StorageService.load(id)`.

C-8: Datenvolumen

Description: Das System ist dimensioniert für < 5.000 Dokumente + Templates als Binärdaten in PostgreSQL. Bei Überschreitung muss Migration zu S3 über C-7 (Storage Abstraction Layer) erfolgen.

C-9: Docker-Container

Description: blocpress-server muss als Docker-Container betrieben werden können. Dockerfile und docker-compose.yml müssen bereitgestellt werden.

Referenz: Docker Best Practices

C-10: Elasticsearch Version

Description: TI-7 (Elasticsearch API) benötigt Elasticsearch Version ≥ 7.x. Kompatibilität mit Version 8.x muss gewährleistet sein.

Referenz: Elasticsearch Compatibility Matrix