blocpress

document automation

System Design Concept

Goals

Das technische System blocpress soll folgende Ziele im Rahmen der digitalen Lösung erreichen:

Automatisierte Dokumentengenerierung: Das System soll LibreOffice-Templates mit JSON-Daten zusammenführen und automatisiert professionelle Dokumente (ODT, PDF, RTF) generieren. Die Generierung erfolgt deterministisch und performant über eine REST-API.
Fachliche Grundlage: Vision im Solution Design Concept

Zentrale Template-Verwaltung: Das System soll Templates und Bausteine zentral speichern, versionieren und für autorisierte Nutzer bereitstellen. Änderungen an Templates müssen nachvollziehbar sein und der aktuelle Freigabestatus jederzeit erkennbar.
Fachliche Grundlage: Information Architecture im Solution Design Concept

Workflow-gesteuerte Qualitätssicherung: Das System soll einen mehrstufigen Freigabeprozess mit Rollen und Berechtigungen implementieren, sodass Templates nur nach erfolgreicher Prüfung in der Produktion verwendet werden können.
Geschäftsprozess: Template erstellen und freigeben

Compliance-Unterstützung: Das System soll turnusmäßige Reviews automatisch planen, überwachen und durchsetzen. Templates mit abgelaufenen Review-Fristen müssen gekennzeichnet und zur Prüfung vorgelegt werden.
Geschäftsprozess: Turnusmäßiges Compliance Review

Umfassende Testunterstützung: Das System soll Testdaten verwalten, Abnahme-PDFs generieren und Regressionstests automatisiert ausführen. Abweichungen von erwarteten Ergebnissen müssen erkannt und gemeldet werden.
Geschäftsprozess: Template testen und Regressionstests

Intelligente Inhaltssuche: Das System soll alle Templates und Bausteine indexieren und eine performante Volltextsuche nach fachlichen Konstrukten ermöglichen, um Konsistenz über alle Dokumente hinweg zu gewährleisten.
Geschäftsprozess: Fachliche Konstrukte recherchieren

Skalierbare Container-Architektur: Das System soll als Docker-Container betrieben werden können und gleichzeitige Anfragen mehrerer Anwendungen effizient verarbeiten.

System Architecture

Die Systemarchitektur von blocpress besteht aus folgenden Elementen:

User Types

Template-Designer: Nutzt blocpress-workbench (via blocpress-studio) zum Hochladen, Bearbeiten und Verwalten von Templates und Bausteinen. Arbeitet primär in LibreOffice und lädt fertige Templates hoch.

Prüfer (QM / Freigeber / Compliance): Konsolidierte Rolle, die blocpress-proof (via blocpress-studio) nutzt. Prüft Templates, gibt frei oder lehnt ab, führt Compliance-Reviews durch, verwaltet Testdaten und führt Regressionstests aus.

Administrator: Nutzt blocpress-admin (via blocpress-studio) zur Verwaltung von Benutzerkonten, Rollen, Berechtigungen und System-Konfiguration.

API-Konsument (Entwickler): Nutzt die öffentliche REST-API von blocpress-render zur programmatischen Dokumentengenerierung aus Anwendungssystemen heraus.

Fachliche Kundensegmente: Value Proposition im Solution Design Concept

Software Elements

Die Anwendung ist als Self-Contained Systems (SCS) strukturiert — fünf unabhängig deploybare Module mit eigener UI, eigenem Backend und eigenem Datenbank-Schema:

blocpress-studio (Portal-Shell): Statische HTML/JS-Anwendung (nginx). Verantwortlich für Navigation, JWT-Authentifizierung und Integration der Micro-Frontends via Import Maps. Kein eigenes Backend. Lädt die Web Components der fachlichen Module.

blocpress-workbench (Quarkus): Template-Entwicklung — Upload, Validierung (TF-1 via blocpress-core), Bearbeitung, Bausteinverwaltung, Content Search (Elasticsearch), Vorschau-Generierung. Arbeitsumgebung für Template-Designer. Nutzt Schema `workbench`.
Use Cases: UC-1 bis UC-5, UC-19 bis UC-21
Detaillierte Beschreibung: Goals im Element Design Concept

blocpress-proof (Quarkus): Prüfung und Freigabe — Workflow (TF-2, Vier-Augen-Prinzip), Testdaten, Regressionstests (TF-6), Compliance-Reviews (TF-7), Stufenübergabe nach production. Arbeitsumgebung für Prüfer. Nutzt Schema `proof`.
Use Cases: UC-6 bis UC-9, UC-11 bis UC-18
Technical Functions: TF-2, TF-6, TF-7

blocpress-render (Quarkus): Dokumentengenerierung — einzige öffentliche REST-API. Liest freigegebene Templates aus Schema `production`. Nutzt blocpress-core (Merge-Pipeline) und LibreOffice headless (Format-Export, Version ≥ 24). Stateless, horizontal skalierbar.
Technical Function: TF-5: Dokument generieren

blocpress-admin (Quarkus): Administration — Benutzerverwaltung, Rollenzuweisung,

Systemkonfiguration, Audit-Logs. Nutzt Schema `admin`.
Use Cases: UC-22 bis UC-24

Shared Library:

blocpress-core: Gemeinsame Java-Library (Maven-Abhängigkeit, kein eigenständiger Container). Enthält ODT-Parsing (odfdom), Template-Validierung und Merge-Pipeline (Text-Block-Expansion, Bedingungsauswertung, Schleifenbehandlung, Feldersetzung). Keine LibreOffice-Abhängigkeit. Wird von blocpress-workbench (TF-1: Validierung) und blocpress-render (TF-5: Merge-Schritt) genutzt.

Gemeinsame Infrastruktur:

PostgreSQL-Datenbank (Version ≥ 18): Eine Datenbankinstanz mit vier getrennten Schemata (`workbench`, `proof`, `production`, `admin`). Template-Tabellen haben in workbench/proof/production identische Struktur für kopie-basierte Stufenübergabe. Binärdaten als bytea (< 5.000 Dokumente). Storage Abstraction Layer für zukünftigen S3-Wechsel.
Technical Interface: TI-2: Datenbank-Interface
Entities: Entities im Element Design Concept

Content Search Engine (Elasticsearch ≥ 7.x): Suchindex für Templates und Bausteine, genutzt von blocpress-workbench.
Technical Interface: TI-7: Elasticsearch API

Message Queue (RabbitMQ/Kafka): Asynchrone Dokumentengenerierung und Regressionstests.
Technical Interface: TI-5: Message Queue Interface

Hardware Elements

Container Platform: Docker-Host oder Kubernetes-Cluster zum Betrieb aller SCS-Container (studio, workbench, proof, render, admin). Mindestens 16 GB RAM und 4 CPU-Cores für produktive Umgebung.

Datenbank-Server: Dedizierter Server oder Container für PostgreSQL (Version ≥ 18) mit persistentem Storage. Eine Instanz mit vier Schemata (workbench, proof, production, admin). Mindestens 8 GB RAM, 100 GB SSD-Storage.

Load Balancer: Hardware- oder Software-Load-Balancer zur Verteilung von Last auf mehrere blocpress-render-Instanzen (horizontal skalierbar für Dokumentengenerierung).

Client-Geräte: Desktop-Computer oder Laptops für Template-Designer, Prüfer und Administratoren mit modernem Webbrowser (Chrome, Firefox, Edge) und LibreOffice-Installation.

Partner Elements

JWT-basierte Authentifizierung: blocpress-studio übernimmt die Authentifizierung und reicht das JWT als Property an die Web Components weiter. Jedes SCS-Modul validiert JWTs eigenständig via Quarkus SmallRye JWT. Es besteht keine Laufzeitabhängigkeit zu einem externen Identity Provider.

Technical Interface: TI-4: JWT-Validierung

Observability (optional): Alle SCS-Module exportieren Traces, Metriken und Logs über OpenTelemetry (OTLP-Protokoll). Der Betreiber kann einen beliebigen OTLP-kompatiblen Collector einsetzen (z.B. Grafana Alloy, Jaeger, Datadog Agent). Ohne konfiguriertem Endpoint laufen die Module ohne externe Monitoring-Abhängigkeit.

Backup-System: Bestehendes Backup-System der Organisation zur regelmäßigen Sicherung der PostgreSQL-Datenbank (inkl. Binärdaten).

System Scenarios

Szenario: Template hochladen und zur Freigabe einreichen

Ziel: Ein Template-Designer erstellt ein neues Template und reicht es zur Qualitätsprüfung ein.

1. Der Template-Designer meldet sich über blocpress-studio an (Authentifizierung erfolgt über JWT)
2. blocpress-studio lädt das Web Component `<bp-workbench>` und reicht das JWT als Property weiter
3. Der Template-Designer lädt eine LibreOffice-Datei (.odt) über die Workbench-Oberfläche hoch
4. Das Web Component sendet die Datei an blocpress-workbench über die interne REST-API
5. blocpress-workbench speichert die Datei im Schema `workbench` (PostgreSQL) und erstellt einen Metadaten-Eintrag
6. blocpress-workbench validiert die Template-Datei über blocpress-core (ODT-Parsing ohne LibreOffice)
7. blocpress-core analysiert User-Fields, Wiederholungsgruppen und IF-Bedingungen und meldet Ergebnisse zurück
8. blocpress-workbench speichert die Validierungsergebnisse und aktualisiert den Template-Status auf "Entwurf"
9. Der Template-Designer sieht die Validierungsergebnisse und reicht das Template zur Prüfung ein
10. blocpress-workbench kopiert das Template in das Schema `proof` (Stufenübergabe workbench → proof)
11. Das Template ist nun in blocpress-proof für den Prüfer sichtbar

Use Cases: UC-1: Template hochladen, UC-2: Zur Freigabe einreichen
Geschäftsprozess: Template erstellen und freigeben

Szenario: Dokument über API generieren

Ziel: Eine externe Anwendung generiert ein Dokument basierend auf einem freigegebenen Template.

1. Die externe Anwendung sendet einen POST-Request mit Template-ID und JSON-Daten an blocpress-render (einzige öffentliche API)
2. blocpress-render authentifiziert die Anfrage (JWT-Validierung)
3. blocpress-render validiert die JSON-Daten gegen das Template-Schema
4. blocpress-render lädt das freigegebene Template aus dem Schema `production` (PostgreSQL)
5. blocpress-render legt die Generierungsanfrage in die Message Queue für asynchrone Verarbeitung
6. Der LibreOffice Document Processor nimmt die Anfrage aus der Queue entgegen
7. Der Document Processor befüllt das Template mit den JSON-Daten (User-Fields, Wiederholungsgruppen, IF-Bedingungen)
8. Der Document Processor exportiert das befüllte Dokument im gewünschten Format (ODT, PDF oder RTF)
9. Das generierte Dokument wird im Schema `production` gespeichert
10. blocpress-render aktualisiert den Generierungs-Status auf "Erfolgreich"
11. Die externe Anwendung erhält die Generierungs-ID als Response und kann den Status abfragen
12. Beim Status-Abruf liefert blocpress-render einen Download-Link für das generierte Dokument zurück

Technical Function: TF-5: Dokument generieren
Use Case: UC-10: Testdokument mit Template generieren
Geschäftsprozess: Dokument automatisiert generieren

Szenario: Automatischer Regressionstest nach Template-Änderung

Ziel: Nach einer Template-Änderung werden automatisch alle zugehörigen Regressionstests ausgeführt.

1. Ein Template wird von blocpress-workbench an blocpress-proof übergeben (Stufenübergabe workbench → proof)
2. blocpress-proof empfängt das Template im Schema `proof` und triggert das Test Framework
3. Das Test Framework (in blocpress-proof) ermittelt alle Test Cases, die dem Template zugeordnet sind
4. Für jeden Test Case lädt das Test Framework die gespeicherten JSON-Testdaten aus dem Schema `proof`
5. Das Test Framework sendet Generierungsanfragen an blocpress-render über die interne REST-

API

6. blocpress-render generiert PDFs für alle Test Cases
7. Das Test Framework lädt die generierten PDFs und die gespeicherten Baseline-PDFs
8. Das Test Framework vergleicht die PDFs pixel- oder textbasiert
9. Abweichungen werden dokumentiert und im Schema `proof` gespeichert
10. Der Prüfer öffnet blocpress-proof (via blocpress-studio) und sieht die Test-Ergebnisse
11. Der Prüfer entscheidet, ob Abweichungen akzeptabel sind oder das Template überarbeitet werden muss

Technical Functions: TF-5: Dokument generieren, TF-6: PDF vergleichen
Use Case: UC-16: Test ausführen
Geschäftsprozess: Template testen und Regressionstests

Szenario: Turnusmäßiges Compliance Review durchführen

Ziel: Der Compliance Manager erkennt ein fälliges Review und benachrichtigt den zuständigen Reviewer.

1. Der Compliance Manager (in blocpress-proof) läuft als geplanter Job und prüft täglich alle freigegebenen Templates im Schema `proof`
2. Für jedes Template berechnet der Compliance Manager das nächste Review-Datum basierend auf dem Review-Zyklus (1, 3 oder 5 Jahre)
3. Der Compliance Manager identifiziert Templates, deren Review-Datum in den nächsten 30 Tagen liegt oder bereits überschritten ist
4. Für jedes fällige Template erstellt der Compliance Manager einen Review-Eintrag im Schema `proof` mit Status "Anstehend"
5. Der Prüfer meldet sich über blocpress-studio an und sieht fällige Reviews in blocpress-proof
6. Der Prüfer öffnet ein Template zur Prüfung und vergleicht es mit aktuellen rechtlichen und fachlichen Anforderungen
7. Der Prüfer hinterlässt Kommentare und markiert das Template als "Freigegeben" oder "Überarbeitung erforderlich"
8. blocpress-proof aktualisiert den Review-Status und berechnet das nächste Review-Datum
9. Bei "Überarbeitung erforderlich" wird das Template in blocpress-workbench als "Überarbeitung erforderlich" markiert

Technical Function: TF-7: Compliance-Reviews überprüfen
Use Case: UC-12: Review durchführen
Geschäftsprozess: Turnusmäßiges Compliance Review

Szenario: Fachliche Konstrukte über alle Templates suchen

Ziel: Ein Template-Designer sucht nach einem fachlichen Konstrukt, um zu prüfen, in welchen Dokumenten es verwendet wird.

1. Der Template-Designer meldet sich über blocpress-studio an
2. Der Template-Designer gibt den Suchbegriff "Beitragsanpassung" in das Suchfeld der Workbench-Oberfläche ein
3. Das Web Component sendet eine Such-Anfrage an blocpress-workbench über die interne REST-API
4. blocpress-workbench leitet die Anfrage an die Content Search Engine (Elasticsearch) weiter
5. Die Content Search Engine durchsucht den Index nach dem Begriff in allen Templates und Bausteinen
6. Die Search Engine liefert eine Liste von Treffern mit Template-ID, Baustein-ID, Kontext und Position zurück
7. blocpress-workbench reichert die Suchergebnisse mit Metadaten (Template-Name, Status, Version) aus dem Schema `workbench` an
8. Die Workbench-Oberfläche zeigt die Suchergebnisse gruppiert nach Templates und Bausteinen an
9. Der Template-Designer wählt einen Treffer aus
10. Die Workbench-Oberfläche öffnet das entsprechende Template oder den Baustein und markiert die gefundene Stelle

Use Case: UC-19: Nach Begriff suchen
Geschäftsprozess: Fachliche Konstrukte recherchieren

Szenario: Index für Content Search aktualisieren

Ziel: Nach dem Upload oder der Änderung eines Templates wird der Suchindex automatisch aktualisiert.

1. Ein Template-Designer lädt ein neues Template hoch oder aktualisiert ein bestehendes Template in blocpress-workbench
2. blocpress-workbench speichert das Template im Schema `workbench` (PostgreSQL) und aktualisiert die Metadaten
3. blocpress-workbench sendet eine Indexierungs-Anfrage an die Content Search Engine (Elasticsearch)
4. Die Content Search Engine extrahiert den Textinhalt des Templates (inkl. User-Fields, statischer Text, Tabellen)
5. Die Search Engine erstellt oder aktualisiert den Index-Eintrag mit Template-ID, extrahiertem Text und Positionsinformationen

6. Die Search Engine bestätigt die erfolgreiche Indexierung an blocpress-workbench
7. Bei Verwendung eines Bausteins in einem Template indexiert die Search Engine auch die Beziehung zwischen Baustein und Template

Quality Requirements

Verfügbarkeit: blocpress soll eine Verfügbarkeit von 99,5% während Geschäftszeiten (Mo-Fr, 08:00-18:00 Uhr) erreichen. Geplante Wartungsfenster außerhalb der Geschäftszeiten sind zulässig.

Performance - Dokumentengenerierung: Die Generierung eines Standard-Dokuments (bis 20 Seiten, bis 10 Wiederholungsgruppen) soll innerhalb von 5 Sekunden abgeschlossen sein. Bei größeren Dokumenten (20-100 Seiten) wird eine Generierungszeit von bis zu 30 Sekunden akzeptiert.

Performance - Suche: Die Suche nach fachlichen Konstrukten über alle Templates und Bausteine soll innerhalb von 2 Sekunden Ergebnisse liefern, auch bei einem Bestand von über 1.000 Templates.

Performance - Regressionstests: Ein einzelner Regressionstest soll innerhalb von 30 Sekunden abgeschlossen sein. Die parallele Ausführung von bis zu 10 Tests soll unterstützt werden.

Skalierbarkeit: Das System soll bis zu 100 gleichzeitige Dokumentengenerierungen verarbeiten können. blocpress-render ist horizontal skalierbar — bei Bedarf können weitere Instanzen hinzugefügt werden.

Zuverlässigkeit: Die Dokumentengenerierung soll deterministisch sein – identische Eingabedaten (Template-Version + JSON-Daten) müssen immer identische Ausgabedokumente erzeugen. Wiederholte Generierungen dürfen keine Abweichungen aufweisen.

Wartbarkeit: Alle Templates, Bausteine und generierten Dokumente müssen versioniert werden. Änderungen müssen nachvollziehbar sein und es soll möglich sein, ältere Versionen wiederherzustellen.

Nachvollziehbarkeit (Auditability): Alle Workflow-Aktionen (Freigaben, Ablehnungen), Compliance-Reviews, Testausführungen und API-Zugriffe müssen protokolliert werden. Logs müssen mindestens 2 Jahre aufbewahrt werden.

Sicherheit - Authentifizierung: blocpress-studio übernimmt die Authentifizierung und reicht das JWT als Property an Web Components weiter. Jedes SCS-Modul validiert JWTs eigenständig via Quarkus SmallRye JWT. Eine Laufzeitabhängigkeit zu einem externen Identity Provider besteht nicht.

Sicherheit - Autorisierung: Rollenbasierte Zugriffskontrolle (RBAC) muss implementiert sein. Template-Designer arbeiten in blocpress-workbench, Prüfer (konsolidierte Rolle) in blocpress-proof. Template-Designer dürfen nur Templates erstellen und bearbeiten, aber nicht freigeben.

Sicherheit - API: API-Zugriffe müssen über API-Keys oder OAuth 2.0 authentifiziert werden. Rate Limiting (max. 1000 Requests pro Stunde pro API-Key) muss implementiert sein.

Datenschutz: Personenbezogene Daten in Templates müssen DSGVO-konform verarbeitet werden. Das System muss die Möglichkeit bieten, personenbezogene Daten aus Test Cases und generierten Dokumenten zu löschen.

Backup & Recovery: Tägliche Backups der PostgreSQL-Datenbank (inkl. aller Binärdaten) müssen erfolgen. Das System muss innerhalb von 4 Stunden aus einem Backup wiederhergestellt werden können (Recovery Time Objective: 4h). Datenverlust darf maximal 24 Stunden betragen (Recovery Point Objective: 24h).

Observability: Alle SCS-Module unterstützen OpenTelemetry (OTLP) für den Export von Traces, Metriken und Logs. Die Konfiguration eines OTLP-Endpoints ist optional — ohne Endpoint laufen die Module ohne externe Monitoring-Abhängigkeit. Quarkus Health-Checks (`/q/health/live`, `/q/health/ready`) stehen je Modul unabhängig zur Verfügung.

Fachliche Quality Requirements: Quality Requirements im Solution Design Concept
Element-spezifische Quality Requirements: Quality Requirements im Element Design Concept

Constraints

Technologische Constraints:

- blocpress ist als Self-Contained Systems (SCS) strukturiert: studio, workbench, proof, render, admin
- Alle SCS-Module (außer studio) basieren auf Quarkus Framework
- Jedes SCS-Modul wird als eigener Docker-Container betrieben
- LibreOffice Version ≥ 24 muss headless in blocpress-render integriert sein (Format-Export). blocpress-workbench nutzt blocpress-core für Validierung ohne LibreOffice
- Nur LibreOffice-kompatible Formate (ODT, OTT) werden als Template-Format unterstützt
- Export nach ODT, PDF und RTF
- PostgreSQL Version ≥ 18 als Datenbank mit vier getrennten Schemata (workbench, proof, production, admin)
- Identische Template-Tabellenstrukturen in workbench/proof/production für kopie-basierte Stufenübergabe
- Binärdaten werden in PostgreSQL gespeichert (< 5.000 Dokumente + Templates erwartet)
- Storage Abstraction Layer muss implementiert sein, um zukünftigen Wechsel zu S3-kompatiblem Object Storage zu ermöglichen
- Elasticsearch ab Version 7.x für Content Search

Datenschutz-Constraints:

- Verarbeitung personenbezogener Daten muss DSGVO-konform erfolgen
- Templates und generierte Dokumente dürfen nur in der EU bzw. in Regionen mit angemessenem Datenschutzniveau gespeichert werden
- Löschkonzept für personenbezogene Daten muss implementiert sein

Organisatorische Constraints:

- Der Freigabeprozess muss eine Vier-Augen-Prüfung gewährleisten (Designer ≠ Freigeber)
- Compliance-Reviews müssen nach 1, 3 oder 5 Jahren erfolgen (konfigurierbar pro Template)
- Templates ohne gültige Freigabe dürfen nicht für die Produktiv-Generierung verwendet werden

Lizenzierungs-Constraints:

- Verwendung von Open-Source-Komponenten (LibreOffice, PostgreSQL, Elasticsearch) unter Einhaltung der jeweiligen Lizenzbedingungen
- Kommerzielle Komponenten (falls verwendet) müssen lizenziert werden

Infrastruktur-Constraints:

- Das System muss in der bestehenden Container-Infrastruktur (Docker/Kubernetes) der Organisation betrieben werden können
- JWT-Validierung mittels konfiguriertem Issuer und Public Key ist erforderlich (kein externer Identity Provider zur Laufzeit)
- OpenTelemetry (OTLP) für Observability; Konfiguration eines Collectors ist optional
- Integration in bestehendes Backup-System muss möglich sein

Compliance-Constraints:

- Review-Zyklen und deren Dokumentation müssen branchenspezifischen Compliance-Anforderungen entsprechen (z.B. ISO 9001, branchenspezifische Vorgaben)
- Audit-Logs müssen manipulationssicher gespeichert werden
- Änderungen an Templates müssen lückenlos nachvollziehbar sein

Fachliche Constraints: Constraints im Solution Design Concept
Element-spezifische Constraints: Constraints im Element Design Concept