<Company Name>

Breakpoint
Software Architecture Document

Version 1.0

Revision History

Date	Version	Description	Author
<01/12/24>	<1.0>		

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

Das Dokument beschreibt die Architektur der Wiki-Plattform, die es Benutzern ermöglicht, Inhalte zu erstellen, zu verwalten und gemeinsam zu bearbeiten. Die Plattform unterstützt mehrere Geräte (Desktop, Mobilgeräte) und gewährleistet Sicherheit, Zuverlässigkeit und hohe Performance.

1.3 Definitions, Acronyms, and Abbreviations

ASR: Architecture Significant Requirements
REST: Representational State Transfer
MVC: Model-View-Controller
JWT: JSON Web Token
RBAC: Role-Based Access Control

1.4 Overview

Das Dokument enthält:

- Ziele und Einschränkungen (Kapitel 3)
- Use-Case Realisierungen (Kapitel 4)
- Prozess-, Deployment- und Datenansichten (Kapitel 6, 7, 9)
- Qualitätsmerkmale (Kapitel 11)

2. Architectural Representation

Die Architektur basiert auf dem 4+1-Sichtenmodell von Kruchten. Folgende Sichten sind enthalten:

- Use-Case View: Beschreibung der zentralen Anwendungsfälle.
- Logical View: Decomposition der Software in Pakete und Klassen.
- Process View: Darstellung der Thread- und Prozess-Architektur.
- Deployment View: Zuordnung von Prozessen auf physische Nodes.
- Implementation View: Darstellung von Layern und Komponenten.

3. Architectural Goals and Constraints

Die wichtigsten Ziele und Einschränkungen:

- Performance: Ladezeiten unter 2 Sekunden für Seitenaufrufe und Suchergebnisse.
- Zuverlässigkeit: 99.9% Systemverfügbarkeit durch Redundanzmechanismen.
- Sicherheit: Schutz sensibler Daten durch rollenbasierte Zugriffssteuerung und Verschlüsselung.
- Modifizierbarkeit: Modularer Aufbau für schnelle Erweiterungen.
- Plattformunabhängigkeit: Responsives Design für Desktop und Mobilgeräte.

4. Use-Case View

4.1 Use-Case Realizations

Use-Case Realization: User Authentication

Zentraler Anwendungsfall: Benutzer-Authentifizierung
Akteure: Benutzer und System
Hauptablauf:

1. Benutzer navigiert zur Login-Seite.
2. Benutzer gibt Benutzername und Passwort ein.
3. System überprüft die Eingaben mit den gespeicherten Benutzerdaten.
4. Bei Übereinstimmung wird der Benutzer authentifiziert und auf das Dashboard weitergeleitet.

Alternative Flows:

- Passwort vergessen:

    1. Benutzer klickt auf "Passwort vergessen".
    2. System fordert die Eingabe der registrierten E-Mail-Adresse an.
    3. System sendet eine E-Mail mit einem Passwort-Zurücksetzungslink.
    4. Benutzer setzt das Passwort zurück und kehrt zur Login-Seite zurück.

- Fehlerhafte Anmeldedaten:
    1. Benutzer gibt falsche Anmeldedaten ein.
    2. System zeigt eine Fehlermeldung an.
    3. Nach mehreren Fehlversuchen wird der Benutzer aufgefordert, die Passwort-Zurücksetzung zu verwenden.

Zusätzliche Anforderungen:

- Sicherheit:
    - Passwörter müssen sicher gehasht gespeichert werden.
    - Nach 5 fehlgeschlagenen Login-Versuchen wird der Benutzerzugang vorübergehend gesperrt.
- Usability:
    - Benutzer werden durch Hinweise und Fehlernachrichten bei Problemen unterstützt.

Interaktionen:

- Benutzer interagiert über das Frontend mit den Eingabefeldern.
- System kommuniziert mit der Datenbank, um Anmeldedaten zu überprüfen.
- Nach erfolgreicher Anmeldung wird eine Session erstellt.

Diagramm: Ein Sequenzdiagramm illustriert den Ablauf zwischen Benutzer, Frontend, Backend und Datenbank.

Use-Case Realization: Wiki Page Creation

Zentraler Anwendungsfall: Erstellung einer Wikiseite
Akteure: Redakteure (mit Bearbeitungsrechten) und das System

Hauptablauf:

1. Redakteur navigiert zur Option "Neue Seite erstellen".
2. Redakteur gibt Titel und Inhalte in die entsprechenden Felder ein.
3. System prüft die Eingaben auf Vollständigkeit und Validität.
4. Redakteur speichert die Seite.
5. System speichert die Seite in der Datenbank und macht sie für andere Benutzer verfügbar.

Alternative Flows:

- Ungültige Eingaben:
  1. System zeigt Fehlerhinweise an (z. B. "Titel fehlt" oder "Inhalt zu kurz").
  2. Redakteur korrigiert die Eingaben und versucht erneut, die Seite zu speichern.
- Speicherfehler:
  1. System erkennt Probleme bei der Datenbankverbindung.
  2. System zeigt eine Fehlermeldung an und protokolliert den Vorfall.
  3. Redakteur wird informiert, dass die Seite nicht gespeichert werden konnte.

Zusätzliche Anforderungen:

- Versionierung:
  - Jede gespeicherte Seite erhält eine neue Versionsnummer, um Änderungen nachverfolgen zu können.
- Usability:
  - Redakteure werden mit Tooltips und Vorschlägen beim Ausfüllen der Felder unterstützt.

Interaktionen:

- Redakteur interagiert über das Frontend mit den Eingabefeldern.
- System übernimmt die Validierung und Speicherung der Inhalte in der Datenbank.
- Andere Benutzer können die erstellte Seite nach erfolgreicher Speicherung aufrufen.

Diagramm:
Ein Sequenzdiagramm zeigt die Interaktion zwischen Redakteur, System und Datenbank während der Seitenerstellung.

Use-Case Realization: User Authentication

Zentraler Anwendungsfall: Benutzer-Authentifizierung
Akteure: Benutzer und System
Hauptablauf:

5. Benutzer navigiert zur Login-Seite.
6. Benutzer gibt Benutzername und Passwort ein.
7. System überprüft die Eingaben mit den gespeicherten Benutzerdaten.
8. Bei Übereinstimmung wird der Benutzer authentifiziert und auf das Dashboard weitergeleitet.

Alternative Flows:

- Passwort vergessen:

    5. Benutzer klickt auf "Passwort vergessen".
    6. System fordert die Eingabe der registrierten E-Mail-Adresse an.
    7. System sendet eine E-Mail mit einem Passwort-Zurücksetzungslink.
    8. Benutzer setzt das Passwort zurück und kehrt zur Login-Seite zurück.

- Fehlerhafte Anmeldedaten:
    4. Benutzer gibt falsche Anmeldedaten ein.
    5. System zeigt eine Fehlermeldung an.
    6. Nach mehreren Fehlversuchen wird der Benutzer aufgefordert, die Passwort-Zurücksetzung zu verwenden.

Zusätzliche Anforderungen:

- Sicherheit:
    - Passwörter müssen sicher gehasht gespeichert werden.
    - Nach 5 fehlgeschlagenen Login-Versuchen wird der Benutzerzugang vorübergehend gesperrt.
- Usability:
    - Benutzer werden durch Hinweise und Fehlernachrichten bei Problemen unterstützt.

Interaktionen:

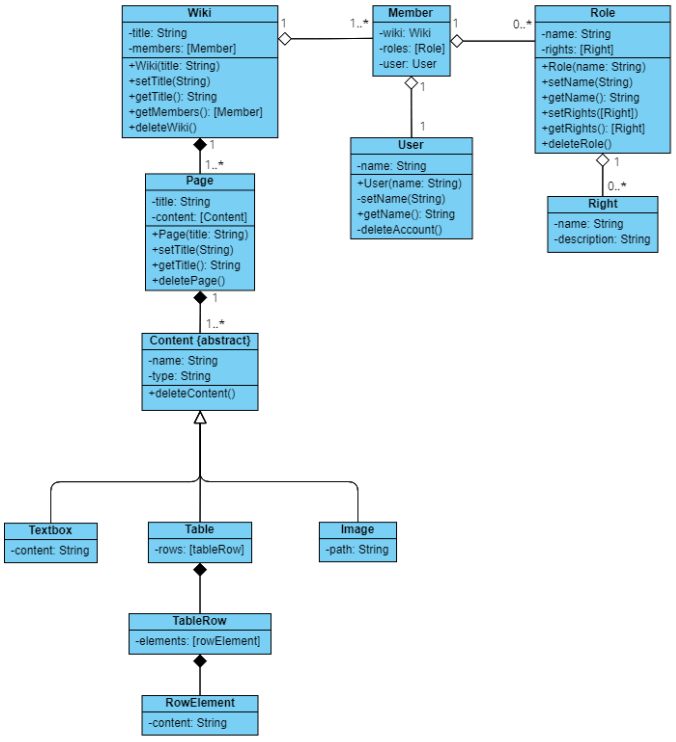
- Benutzer interagiert über das Frontend mit den Eingabefeldern.
- System kommuniziert mit der Datenbank, um Anmeldedaten zu überprüfen.
- Nach erfolgreicher Anmeldung wird eine Session erstellt.

Diagramm: Ein Sequenzdiagramm illustriert den Ablauf zwischen Benutzer, Frontend, Backend und Datenbank.

5. Logical View

5.1 Overview

Das Klassendiagramm repräsentiert die logische Struktur unseres Wiki-Systems und zeigt die wichtigsten Klassen, deren Attribute, Methoden und Beziehungen. Es bildet die Grundlage für die Implementierung und gewährleistet eine klare Trennung der Verantwortlichkeiten innerhalb der Anwendung.



5.2 Architecturally Significant Design Packages

Frontend-Komponenten: Darstellung und Benutzerinteraktion.
Backend-Logik: Datenmanagement und Sicherheitsprüfungen.
Datenbankmodelle: Tabellen für Benutzer, Artikel und Kategorien.

6. Process View

Die Prozesse sind nach folgenden Schichten gegliedert:

- Frontend-Thread: Darstellung und Benutzerinteraktion.
- API-Prozesse: Verarbeitung von Anfragen (CRUD-Operationen).
- Datenbankzugriff: MariaDB für persistente Speicherung und Abfragen.

6.1 Scenario Description: "Create New Page"

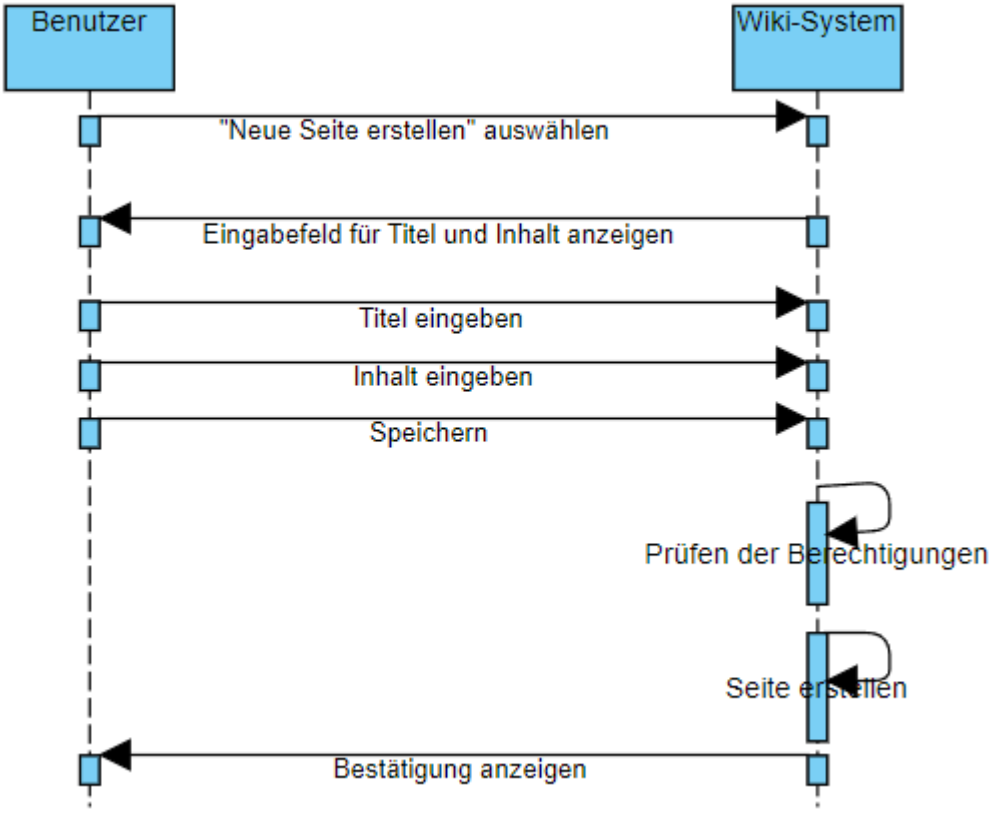
Dieses Sequenzdiagramm beschreibt den Prozess, wie ein Benutzer im Wiki-System eine neue Seite erstellt. Es illustriert die Interaktionen zwischen dem Benutzer und dem System, einschließlich der Überprüfung von Berechtigungen und der anschließenden Bestätigung der Aktion.

6.2 Step-by-Step Workflow

- "Select Create New Page": The user initiates the process by clicking the "Create New Page" button in the system.
- Display Input Fields: The system presents input fields for the page title and content to the user.
- Enter Title and Content: The user fills in the page title and the desired content.
- Save: The user confirms by clicking the save option to create the page.
- Verify Permissions: The system checks if the user has the required permissions to create a new page.
- Create Page: If the permission check succeeds, the system creates the new page in the database.

6.3 Graphical Representation

Das Sequenzdiagramm zeigt die oben beschriebenen Schritte:



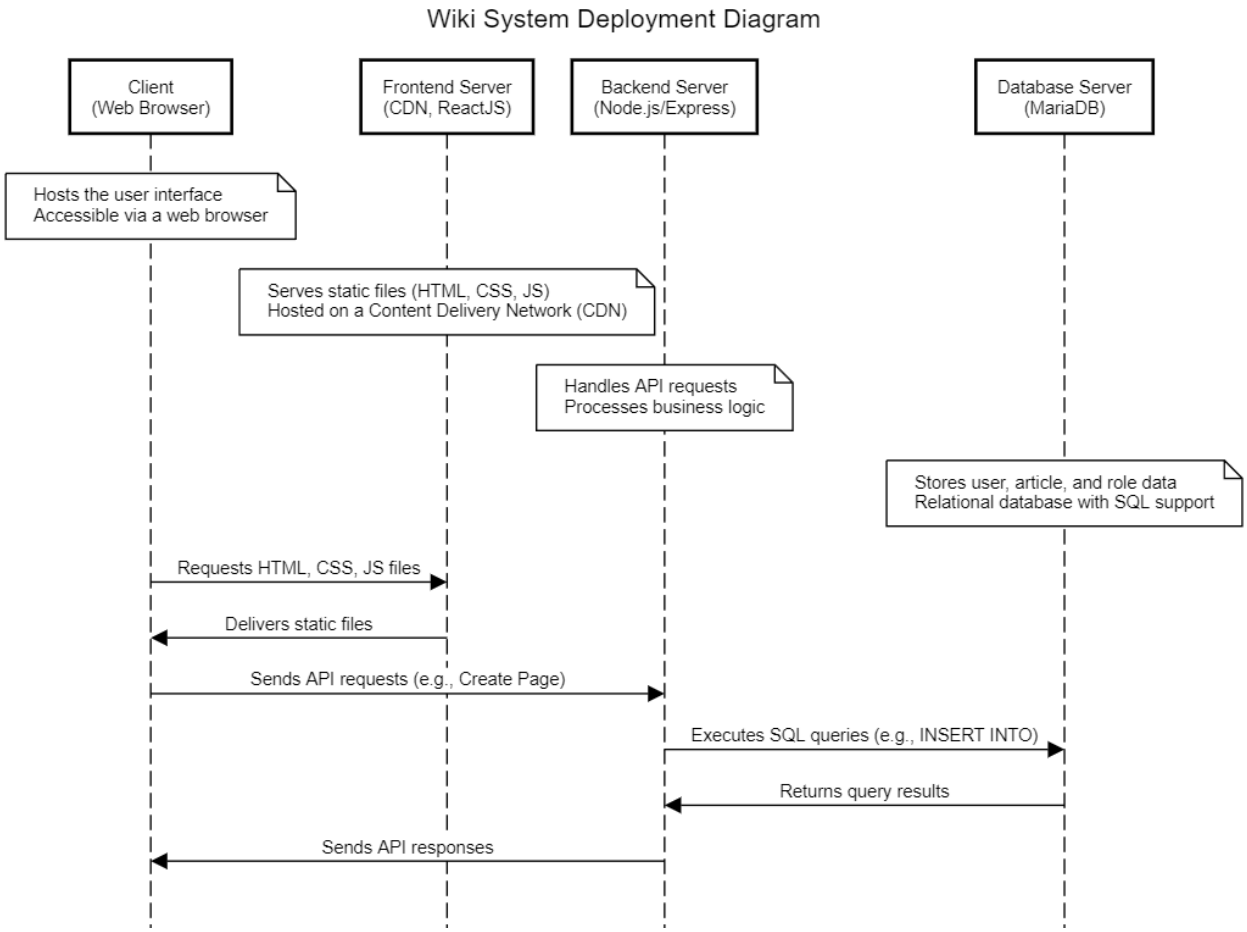
7. Deployment View

7.1 Overview

Die Deployment View beschreibt die physische Verteilung der Softwarekomponenten unserer Wiki-Plattform und zeigt, wie diese auf verschiedenen Hardware- und Netzwerkknoten bereitgestellt werden. Das System besteht aus vier Hauptknoten: dem Client, dem Frontend-Server, dem Backend-Server und dem Datenbank-Server.

7.2 Deployment Diagram Description

Das folgende Einsatzdiagramm zeigt die Interaktion zwischen den verschiedenen Knoten des Systems:



7.3 Description of Components

1. Client
   - Beschreibung: Der Client ist der Webbrowser, der die Benutzeroberfläche darstellt. Benutzer greifen über den Browser auf die Plattform zu.
   - Verantwortlichkeiten:
     - Darstellung der Benutzeroberfläche (HTML, CSS, JavaScript).
     - Senden von API-Anfragen an das Backend.

2. Frontend Server
   - Beschreibung: Der Frontend-Server ist ein Content Delivery Network (CDN), das statische Dateien wie HTML, CSS und JavaScript ausliefert.
   - Verantwortlichkeiten:
     - Bereitstellung von statischen Ressourcen.
     - Optimierte Auslieferung über ein globales CDN.

3. Backend Server

- Beschreibung:
  - Der Backend-Server ist ein Node.js/Express-Server, der die Geschäftslogik verarbeitet

und API-Endpunkte bereitstellt.

- Verantwortlichkeiten:
  - Verarbeitung von API-Anfragen.
  - Durchführung von Geschäftslogik (z. B. Berechtigungsprüfungen).
  - Kommunikation mit der Datenbank.

4. Database Server

- Beschreibung:
  - Der Datenbank-Server ist ein MariaDB-Server, der die Daten des Systems speichert.
- Verantwortlichkeiten:
  - Persistente Speicherung von Benutzern, Artikeln, Kategorien und Rollen.
  - Optimierung der Abfragen durch Indexierung.
  - Sicherstellung der Datenintegrität.

7.4 Data Flow

- Client → Frontend Server: Der Client fordert statische Dateien vom Frontend-Server an (z. B. HTML, CSS, JavaScript).
- Frontend Server → Client: Der Frontend-Server liefert die angeforderten Dateien an den Client aus.
- Client → Backend Server: Der Client sendet API-Anfragen (z. B. zum Erstellen oder Bearbeiten eines Artikels) an den Backend-Server.
- Backend Server → Database Server: Der Backend-Server führt SQL-Abfragen (z. B. INSERT, SELECT) an den Datenbank-Server aus.
- Database Server → Backend Server: Die Datenbank liefert die Abfrageergebnisse an den Backend-Server zurück.
- Backend Server → Client: Der Backend-Server sendet die Ergebnisse der Anfrage an den Client zurück.

8. Data View

ER-Modell (vereinfacht):

Tabellen:

- users (id, name, role, password_hash)
- articles (id, title, content, author_id)
- categories (id, name)

9. Size and Performance

Maximale Nutzeranzahl: 10.000 gleichzeitig aktive Benutzer.
Datenbankabfragen: <100ms für einfache Abfragen, 1.5s für komplexe.

10. Quality

Unsere Architektur zielt darauf ab, Zuverlässigkeit, Modifizierbarkeit, Performance und Sicherheit sicherzustellen. Mit den gewählten Taktiken (z. B. Caching, Redundanz) wird dies unterstützt.