

Projektbericht: Erstellung einer Webanwendung zur Verbesserung des Workflows des Erstellens von Benutzerhandbüchern

Auszubildender

Oliver Herrmann

Geb.: 23.09.1992

GNS mbH

Tel.: +49 201 109 - 1523

Email: oliver.herrmann@gns.de

Ausbildungsberuf: Fachinformatiker Anwendungsentwicklung

Ausbildungsbetrieb

GNS mbH

Frohnhauser Str. 67

45127 Essen

8. November 2015, Essen

Inhaltsverzeichnis

| | |
|---|-----------|
| Abbildungsverzeichnis | 5 |
| Tabellenverzeichnis | 6 |
| Abkürzungsverzeichnis | 8 |
| 1. Einleitung | 9 |
| 1.1. Projektumfeld | 9 |
| 1.2. Projektziel | 9 |
| 1.3. Projektbegründung | 9 |
| 1.4. Projektschnittstellen | 10 |
| 1.5. Projektabgrenzung | 10 |
| 2. Projektplanung | 11 |
| 2.1. Projektphasen | 11 |
| 2.2. Abweichung vom Projektantrag | 11 |
| 2.3. Ressourcenplanung | 11 |
| 2.4. Entwicklungsprozess | 12 |
| 3. Analysephase | 12 |
| 3.1. Ist-Analyse | 12 |
| 3.2. Wirtschaftlichkeitsanalyse | 12 |
| 3.2.1. Nutzen des Projekts | 13 |
| 3.2.2. Kosten des Projekts | 14 |
| 3.3. Risikoanalyse | 15 |
| 3.4. "Make or Buy"-Bewertung | 16 |
| 3.5. Projektkosten | 16 |
| 3.6. Amortisationsdauer | 16 |
| 3.7. Nutzwertanalyse | 16 |
| 3.8. Anwendungsfälle | 17 |
| 3.9. Qualitätsanforderungen | 17 |
| 3.10. Lastenheft / Fachkonzept | 18 |
| 3.11. Zwischenstand | 18 |
| 4. Entwurfsphase | 18 |
| 4.1. Zielplattform | 18 |
| 4.2. Architekturdesign | 19 |
| 4.2.1. Serverarchitektur | 20 |
| 4.2.2. Clientarchitektur | 20 |
| 4.3. Entwurf der Benutzeroberfläche | 21 |
| 4.3.1. Visuelles Konzept | 21 |
| 4.3.2. Prototyp | 22 |

| | |
|---|-----------|
| 4.4. Datenmodell | 22 |
| 4.5. Geschäftslogik | 22 |
| 4.6. Maßnahmen zur Qualitätssicherung / Testkonzept | 22 |
| 4.7. Pflichtenheft | 23 |
| 4.8. Zwischenstand | 23 |
| 5. Implementierungsphase | 23 |
| 5.1. Iterationsplanung | 23 |
| 5.2. Implementierung der Datenstrukturen | 23 |
| 5.3. Implementierung der Geschäftslogik | 24 |
| 5.4. Implementierung der Benutzeroberfläche | 25 |
| 5.5. Implementierung der Tests | 25 |
| 5.6. Zwischenstand | 26 |
| 6. Abnahmephase | 26 |
| 6.1. Zwischenstand | 26 |
| 7. Einführungsphase | 26 |
| 7.1. Zwischenstand | 26 |
| 8. Dokumentation | 26 |
| 8.1. Zwischenstand | 26 |
| 9. Fazit | 26 |
| 9.1. Soll- / Ist-Vergleich | 26 |
| 9.2. Lesson-Learned | 26 |
| 9.3. Ausblick | 26 |
| Anhang | 27 |
| A. Projektphasen | 27 |
| B. Risikoanalyse | 28 |
| C. "Make or Buy"-Bewertung | 30 |
| D. Kostenübersicht nach Projektphasen | 31 |
| E. Nutzwertanalyse | 32 |
| F. Use-Case Diagramme | 33 |
| G. Qualitätsanforderungen (Auszug aus Pflichtenheft) | 35 |
| H. Lastenheft (Auszug) | 36 |

| | |
|-------------------------------|----|
| I. Oberflächenprototyp | 38 |
| J. Entity-Relationship-Modell | 40 |
| K. Komponentendiagramm | 41 |
| L. Pflichtenheft (Auszug) | 42 |
| M. Iterationsplan | 43 |
| N. Listings | 44 |

Abbildungsverzeichnis

| | | |
|----|---|----|
| 1. | Use-Case "Wiki konvertieren" | 33 |
| 2. | Use-Case "Wiki bearbeiten" | 34 |
| 3. | Prototyp - Layout für Desktop-Browser | 38 |
| 4. | Prototyp - Layout für Mobile-Browser | 39 |
| 5. | ER-Modell der Anwendung | 40 |
| 6. | Komponentendiagramm | 41 |

Tabellenverzeichnis

| | | |
|-----|--|----|
| 1. | Projektphasen | 11 |
| 2. | Ressourcen | 11 |
| 3. | Stundensätze | 13 |
| 4. | ”Make or Buy”-Bewertung | 16 |
| 5. | Ergebnis der Nutzwertanalyse | 17 |
| 6. | Qualitätsanforderungen an die Anwendung - Auszug | 18 |
| 7. | Zwischenstand nach der Analysephase | 18 |
| 8. | Zwischenstand nach der Entwurfsphase | 23 |
| 9. | Zwischenstand nach der Implementierungsphase | 26 |
| 10. | Projektphasen detailliert | 27 |
| 11. | Risikoanalyse detailliert | 29 |
| 12. | Detaillierte ”Make or Buy”-Bewertung | 30 |
| 13. | Kosten unterteilt nach Projektphasen | 31 |
| 14. | Detaillierte Nutzwertanalyse bezüglich der Backend-Architektur | 32 |
| 15. | Detaillierte Nutzwertanalyse bezüglich der Front-Architektur | 32 |
| 16. | Qualitätsanforderungen an die Anwendung (Pflichtenheft) | 35 |

Listings

| | | |
|----|--------------------------------|----|
| 1. | DataBaseUpdater.java | 44 |
| 2. | BaseApi.java | 45 |

Abkürzungsverzeichnis

| | |
|---------------|--|
| CVS | Concurrent Version System |
| GNS | Gesellschaft für Nuklear-Service mbH |
| Hosting | Bereitstellen einer Web-Anwendung auf einem Web-Server |
| JSON | JavaScript Object Notation |
| KIS | Abteilung der GNS für Softwareentwicklung |
| REST | Representational State Transfer |

1. Einleitung

1.1. Projektumfeld

Die GNS (Gesellschaft für Nuklear-Service) mbH ist ein mittelständisches Unternehmen, das 1977 gegründet wurde und aktuell etwa 550 Mitarbeiter beschäftigt. Die Firmenzentrale befindet sich in Essen. Die GNS betreibt des weiteren Standorte in Duisburg, Mülheim an der Ruhr und Karlsruhe. Das Hauptgeschäftsfeld der GNS ist die Entwicklung sowie der Vertrieb von Behältern zur Lagerung und zum Transport von radioaktivem Material, als auch die Beladung solcher Behältnisse.

Ebenfalls zum Geschäftsbereich der GNS zugehörig ist der Ver- und Betrieb von Softwarelösungen. Hier ist beispielhaft das AVK (Abfallfluss- Verfolgungs- und Produktkontroll-System) zu nennen, welches 1991 entwickelt wurde und heute in allen deutschen Kernkraftwerken verwendet wird. Daneben werden von der GNS eine Vielzahl von stark spezialisierten Softwarelösungen betrieben, wovon viele als digitale Dienstleistung für Kunden (z.B. Kernkraftwerke und Zwischenlager) bereitgestellt werden.

Auftraggeber des Projekts "Erstellung einer Webanwendung zur Verbesserung des Workflows des Erstellens von Benutzerhandbüchern" ist die GNS, in Person Friedrich Bauriedel, Leiter der Abteilung KIS.

1.2. Projektziel

Ziel des Projekts ist, den regelmäßig anfallenden Arbeitsschritt "Erstellen eines Benutzerhandbuchs aus einem Wiki" für die Mitarbeiter der GNS zu vereinfachen und zu beschleunigen. Ebenfalls soll der Workflow normiert und zentralisiert werden, sodass der Aufwand für vorlaufende Arbeiten minimiert wird.

1.3. Projektbegründung

Wie in Abschnitt 1.1 erwähnt betreibt die GNS eine Vielzahl an hoch spezialisierten Softwarelösungen für diverse Kunden. Aufgrund der Komplexität vieler dieser Anwendungen ist es nötig den Benutzern ein Handbuch zur Verfügung zu stellen. Zu diesem Zweck betreibt die GNS betriebsinterne Wikis, in denen Inhalte bezüglich der korrekten Bedienung einer Anwendung von Mitarbeitern der GNS gepflegt werden.

2012 wurde von der GNS der sogenannte "WikiConverter" entwickelt. Dies ist ein Tool, welches die Inhalte eines MediaWikis extrahiert und diese in ein PDF- bzw. HTML-Dokument konvertiert. Ziel der Entwicklung war es, die mithilfe des WikiConverters erzeugten Dateien als interaktive Hilfe in die jeweiligen Anwendungen einzubetten bzw. als Benutzerhandbuch den Kunden bereitstellen zu können.

Allerdings wurde bei der Entwicklung des WikiConverters wenig Wert auf eine intuitive Benutzerführung gelegt, was dazu führte, dass die Bedienung von diesem äußerst komplex ist.

Durch die in den letzten Jahren gesteigerte Anzahl der von der GNS betriebenen Softwarelösungen, sowie die stetige Verkürzung der Release-Intervalle dieser, ist der Arbeitsschritt "Erstellen von Benutzerhandbüchern" immer mehr in den Vordergrund gerückt, was schließlich dazu führte, dass eine Überarbeitung von diesem unumgänglich wurde.

Zu diesem Zweck wurde von der Abteilung KIS Mitte 2015 das Projekt "Erstellung einer Webanwendung zur Verbesserung des Workflows des Erstellens von Benutzerhandbüchern" initiiert, welches zum Ziel hat, die Erstellung von Benutzerhandbüchern zu vereinfachen und zu zentralisieren.

Im Rahmen des Projekts wird eine Web-Anwendung entwickelt, welche die Kernfunktionalität des WikiConverters ummantelt und eine einheitliche, graphische Benutzerschnittstelle zu dieser anbietet. So wird die Komplexität des WikiConverters vor den Benutzern verborgen, um diesen eine einfachere und intuitivere Benutzung zu ermöglichen.

Als messbares Ziel des Projekts ist eine Reduzierung der durchschnittlichen Arbeitszeit für den Workflow "Erstellen eines Benutzerhandbuchs" um mindestens 25% veranschlagt.

1.4. Projektschnittstellen

Die technischen und sozialen Schnittstellen des Projekts sind ausschließlich betriebsinterner Natur.

Technisch betrachtet greift das Projekt die bestehende Anwendung "WikiConverter" auf und adaptiert dessen Kernfunktionalität.

Aus projektspezifischer Sicht sind die Schnittstellen rein intern, da sowohl Auftraggeber als auch Endnutzer der Anwendung Mitarbeiter bzw. Abteilungen der GNS sind.

Die Projektabschließende Abnahme erfolgt gegenüber dem Auftraggeber, hier die Abteilung KIS der GNS.

1.5. Projektabgrenzung

Ausdrücklich nicht Teil des Projekts ist das Hosting der im Rahmen des Projekts entwickelten Anwendung sowie alle in Verbindung mit dem Hosting stehenden administrativen Maßnahmen. Diese werden von nicht direkt am Projekt beteiligten Mitarbeitern der GNS durchgeführt.

2. Projektplanung

2.1. Projektphasen

| Projekphase | Geplante Zeit |
|----------------------|---------------|
| Anforderungsanalyse | 7h |
| Projektplanung | 4h |
| Software-Architektur | 14h |
| Implementierung | 26h |
| Projektabschluss | 3h |
| Sonstiges | 16h |
| Gesamt | 70h |

Tabelle 1: Projektphasen

Eine detaillierte Übersicht der einzelnen Phasen findet sich im Anhang A auf Seite 27

2.2. Abweichung vom Projektantrag

Abweichend vom Projektantrag werden das Pflichten- sowie das Lastenheft nur in Auszügen im Projektbericht aufgeführt. Dies geschieht aus dem Grund den Umfang des Projektbericht nicht übermäßig zu gestalten.

2.3. Ressourcenplanung

Im Folgenden werden alle für die Durchführung des Projekts bezogenen Ressourcen aufgelistet. Diese werden durch den Auftraggeber, hier die GNS, bereitgestellt.

| Art der Resource | Beschreibung |
|----------------------------|---|
| Sachliche Resource | |
| Projektbüro | |
| Computer | |
| IDE | Eclipse for JEE Developers ¹ |
| Demilitarisierte Zone | zu Testzwecken |
| Personelle Resource | |
| Projektleiter | Oliver Herrmann |
| Projektmitarbeiter | Oliver Herrmann |

Tabelle 2: Ressourcen

¹<http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/mars1>, abgerufen am 27.10.2015

2.4. Entwicklungsprozess

Grundlage für den Entwicklungsprozess bildet das iterative Wasserfallmodell ². Dies wurde gewählt, da es gut mit Projekten geringen bis mittleren Umfangs harmoniert. Andere Modelle wie z.B. Scrum oder Extreme-Programming wurden nicht in Erwägung gezogen, da sie die Projektarbeit im Team erfordern.

3. Analysephase

3.1. Ist-Analyse

Der jetzige Stand bezüglich des Workflows "Erstellen eines Benutzerhandbuchs" sieht vor, dass der ausführende Mitarbeiter den WikiConverter aus dem internen CVS auscheckt, im Quellcodeangaben zu dem zu konvertierenden Wiki (URL, Hauptseite) ergänzt und den Quellcode anschließend kompiliert und ausführt. Das Ergebnis der Ausführung sind die generierten PDF- und HTML-Dateien.

Das Projekt "Erstellung einer Webanwendung zur Verbesserung des Workflows des Erstellens von Benutzerhandbüchern" sieht vor, den Workflow wie folgt zu definieren:

- Der Benutzer loggt sich online in der Anwendung ein.
- Der Benutzer wählt ein existierendes Wiki aus oder pflegt ein neues ein. (Nötige Angaben sind URL und Hauptseite des Wikis)
- Der Benutzer kann über die graphische Oberfläche das Konvertieren starten.
- Nach erfolgreicher Konvertierung kann der Benutzer die generierten Dateien herunterladen.

Außerdem werden aus Konvertierungsvorgängen erstellte Dateien dauerhaft gespeichert. Diese können ebenfalls vom Benutzer heruntergeladen werden, falls eine neue Konvertierung nicht benötigt wird.

3.2. Wirtschaftlichkeitsanalyse

Im folgenden wird die Wirtschaftlichkeit des Projekts betrachtet. Da das Projekt keinen direkten Gewinn erwirtschaftet sondern der monetäre Vorteil durch die Erleichterung der Arbeit der betroffenen Mitarbeiter erzielt wird, wird zuerst der Nutzen des Projekts messbar festgehalten. Dieser wird dann den Kosten des Projekts gegenübergestellt um eine qualifizierte Aussage über die Wirtschaftlichkeit des Projekts zu machen.

Grundlage der Analyse sind folgende Stundensätze³:

²http://www.pi.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_PI/LV__SE_RE/R_01_Wasserfallmodell__Ausarbeitung__Schwaiger.pdf abgerufen am 05.10.2015

³Beruhend auf Tarifverträgen bzw. betriebsinternen Angaben

| Art der Beschäftigung | Stundensatz |
|-----------------------|-------------|
| Fachinformatiker | 20 € |
| Auszubildender | 6 € |
| Gemeinkosten | 10 € |

Tabelle 3: Stundensätze

Hieraus ergibt sich ein *effektiver* Stundensatz von 30 € für die Arbeit eines Fachinformatikers, bzw. 16 € für die Arbeit eines Auszubildenden (hier Projektmitarbeiter).

3.2.1. Nutzen des Projekts

Der Nutzen des Projekts manifestiert sich primär darin, dass der regelmäßig auftretende Workflow "Erstellen eines Benutzerhandbuchs" beschleunigt wird. Da die exakte Bestimmung der durchschnittlich für diesen Workflow anfallenden Zeit von vielen, auch personellen, Faktoren abhängt, wird er für die Zwecke der Wirtschaftlichkeitsanalyse konservativ abgeschätzt. Diese Schätzung wird fortan als Grundlage für weitere Berechnungen herangezogen, um so ein Worst-Case Ergebnis ausdrücken zu können. Der aktuelle durchschnittliche zeitliche Aufwand für den Workflow ergibt sich aus

- Einarbeitung in den Workflow und den WikiConverter im Allgemeinen
- Auschecken der Quellen
- Anpassungen am Quellcode
- Kompilieren der Quellen
- Ausführen der Anwendung
- Sichern der Ergebnisse

und beträgt nach Schätzungen **15 Minuten**.

Der neue, vom Projekt angestrebte Workflow setzt sich dagegen wie folgt zusammen

Alternative 1

- Login
- Konvertierung anstoßen
- herunterladen der Ergebnisse

Geschätzte Dauer: 5 Minute.

Alternative 2

- Login
- Auswählen eines bereits existierenden Ergebnisses
- herunterladen der Ergebnisse

Geschätzte Dauer: 2 Minuten.

Hier ist anzumerken, dass durch die zentrale Speicherung der konvertierten Ergebnisse dem Nutzer ermöglicht wird, den zeitintensiven Schritt der Konvertierung auszulassen, wenn ein adäquates Ergebnis bereits existiert. Dies ist z.B. dann der Fall, wenn die Konvertierung kürzlich durch einen dritten Mitarbeiter vorgenommen wurde und seitdem keine relevanten Änderungen am zu Grunde liegenden Wiki auftraten.

Konservativ abgeschätzt wird davon ausgegangen, dass in 80% der Fälle Alternative 1 und in den übrigen 20% Alternative 2 eintreten wird.

$$T_{\emptyset} = 0.8 * 5m + 0.2 * 2m = 4.4m \quad (1)$$

Aus Formel 1 ergibt sich eine durchschnittliche Dauer des Workflows von **4,4 Minuten**.

$$T_{\Delta abs} = 15m - 4,4m = 10,6m \quad (2)$$

$$T_{\Delta rel} = \frac{T_{\Delta abs}}{15m} = 71\% \quad (3)$$

Dies entspricht nach Formel 3 einer Einsparung von **71 %** und verspricht ein Einhalten des in Kapitel 1.3 ausgesprochenen Ziels, die durchschnittliche Arbeitszeit des Workflows um mindestens mindestens 25% zu reduzieren.

$$T_{\Delta jahr} = T_{\Delta abs} * 10 * 12 = 1272m = 21,2h \quad (4)$$

$$K_{jahr} = T_{\Delta jahr} * \frac{30\text{€}}{h} = 636\text{€} \quad (5)$$

Unter der Annahme, dass der Workflow durchschnittlich zehnmal pro Monat von einem Fachangestellten durchgeführt wird ergibt sich daraus eine Zeitersparnis von 21,2 Stunden pro Jahr, was nach Formel 5 einer jährlichen monetären Ersparnis von **636 €** entspricht.

3.2.2. Kosten des Projekts

Die Kosten des Projekts werden allein durch die Kosten der Entwicklung bestimmt. Es werden keine nicht bereits vorhandenen Ressourcen für das Projekt verwendet.

$$K_{ges} = 70h * \frac{16\text{€}}{h} = 1120\text{€} \quad (6)$$

Bei einem kalkulierten Projektumfang von 70 Stunden ergibt sich aus Formel 6 ein Kostenumfang von **1120 €**, unter der Voraussetzung, dass als Projektmitarbeiter ausschließlich Auszubildende eingesetzt werden.

3.3. Risikoanalyse

Im folgenden wird eine Risikoanalyse und -bewertung durchgeführt. Eine detaillierte Übersicht der Risikobewertung findet sich im Anhang B auf Seite 28.

Zusammenfassend lässt sich das Ergebnis der Risikoanalyse in drei Teilbereiche gruppieren.

Gruppe 1 Unvermeidbare Risiken

Hierzu zählen z.B. krankheitsbedingte personelle Ausfälle. Diese stellen ein unvermeidbares Risiko dar. Eine Strategie zur Vermeidung ist hier nicht vorgesehen.

Gruppe 2 Bedingt vermeidbare Risiken

Hierzu zählen vor allem Risiken aus Fehlern der Planung. Diese Risiken sind bedingt vermeidbar. Dem Auftreten solcher Risiken wird insbesondere durch die im Qualitätssicherungskonzept vorgesehenen Zwischenstandsanalysen entgegengewirkt.

Gruppe 3 Leicht vermeidbare Risiken

Dies sind in erster Linie Risiken die in der Entwicklung auftreten können. Das Eintreten dieser Risiken wird in erster Linie durch die Erfahrung und intensive fachliche Vorbereitung der Projektmitarbeiter vermieden. Des weiteren hilft das im Qualitätssicherungskonzept beinhaltete Testkonzept hier für eine frühzeitige Erkennung von Konsequenzen dieser Risikogruppe.

Die möglichen Konsequenzen lassen sich wiederum in zwei Bereiche gruppieren.

Gruppe A Zeitliche Konsequenzen

Dies sind Risikofolgen, welche in erster Linie eine Ausdehnung des Projektzeitraums bzw. eine Verzögerung des Projektabschlusses mit sich ziehen. Da das Projekt allerdings für rein betriebsinterne Zwecke konzipiert ist und keinen direkten Gewinn erwirtschaftet beschränken sich die negativen Auswirkungen dieser Gruppe auf ein Minimum. Dies hat zur Folge, dass Zeitliche Konsequenzen, sofern sie nicht in sehr hoher Quantität auftreten, nicht zum Abbruch des Projekts führen.

Gruppe B Finanzielle Konsequenzen

Dies sind Risikofolgen, welche primär eine Erhöhung des monetären Aufwands des Projekts mit sich ziehen. Erzeugt werden diese Konsequenzen vor allem durch Verzögerungen im Projektablauf.

Da das Projektbudget ausschließlich für die Beschäftigung der Projektmitarbeiter aufgewendet wird, ergibt sich dessen Betrag aus der geplanten Projektdauer. Dies hat zur Folge, dass finanzielle Konsequenzen direkt mit zeitlichen Konsequenzen gleichgesetzt werden können und daher, bei gemäßigttem Auftreten, nicht zum Projektabbruch führen.

3.4. "Make or Buy"-Bewertung

Im Folgenden wird abgewägt, ob sich das Projekt bezüglich Nutzen und Wirtschaftlichkeit gegenüber bereits existierenden Lösungen behauptet.

| Alternative | Nutzen | Wirtschaftlichkeit |
|--|--------|--------------------|
| MediaWiki Erweiterung | ⊖ ⊖ ⊖ | ⊖ ⊖ ⊖ ⊖ |
| Erstellung der Anwendung durch Dienstleister | ⊕ ⊕ | ⊖ ⊖ ⊖ |
| Eigenständige Erstellung der Anwendung | ⊕ ⊕ ⊕ | ⊖ ⊖ |

Tabelle 4: "Make or Buy"-Bewertung

Eine detaillierte Übersicht der "Make or Buy"-Bewertung findet sich im Anhang C auf Seite 30.

Als Ergebnis der "Make or Buy"-Analyse lässt sich festlegen, dass eine eigenständige Entwicklung der Anwendung aus wirtschaftlicher und nutzenorientierter Sicht die beste Alternative darstellt.

3.5. Projektkosten

Wie in Kapitel 3.2.2 bereits erläutert, setzen sich die Kosten ausschließlich aus den Mitarbeiterkosten der Projektmitarbeiter zusammen. Da für das Projekt nur ein Mitarbeiter veranschlagt ist, sind die Kosten äquivalent zu der kalkulierten Dauer des Projekts.

$$K_{ges} = T_{ges} * K_{proj_mitarbeiter} = 70h * \frac{16€}{1h} = 1120€ \quad (7)$$

Aus Formel 7 wird ersichtlich, dass sich die Gesamtkosten des Projekts auf **1120 €** belaufen.

Eine detaillierte, nach Phasen unterteilte Übersicht der Kosten findet sich im Anhang D auf Seite 31

3.6. Amortisationsdauer

$$T_{amortisation} = \frac{1120€}{636€/1a} = 1,76a \quad (8)$$

Zieht man die Ergebnisse aus Kapitel 3.2.1 und 3.2.2 heran und betrachtet diese relativ zueinander, wie in Formel 8 geschehen, lässt sich feststellen, dass sich das Projekt nach etwa 1,8 Jahren amortisiert hat.

3.7. Nutzwertanalyse

Für die entwicklungsrelevanten Bereiche "Backend" und "Frontend" werden jeweils Nutzwertanalysen durchgeführt, um im Vorhinein die optimale Architektur zu bestimmen.

| Architekturbereich | beste Alternative |
|--------------------|-------------------|
| Backend | JEE-Stack |
| Frontend | React |

Tabelle 5: Ergebnis der Nutzwertanalyse

Die zugrunde liegenden Nutzwertanalysen befinden sich im Anhang E auf Seite 32.

3.8. Anwendungsfälle

Die Anwendung deckt alle für den Workflow "Erstellen eines Handbuchs" relevanten Anwendungsfälle ab. Auch vor- und nachbereitende Anwendungsfälle werden von der Anwendung abgedeckt.

Es folgt eine Auflistung aller relevanten Anwendungsfälle.

1. vorgelagerte Anwendungsfälle
 - a) Anlegen eines Benutzers
 - b) Login durch einen Benutzer
 - c) Anlegen eines neuen Wikis
2. Hauptanwendungsfälle
 - a) Konvertieren eines Wikis
 - b) Herunterladen eines Konvertierungsergebnisses
 - c) Bearbeiten eines Wikis
3. nachgelagerte Anwendungsfälle
 - a) Löschen eines Benutzers
 - b) Löschen eines Wikis
 - c) Löschen eines Konvertierungsergebnisses

Zu den Anwendungsfällen 2a und 2c finden sich im Anhang F auf Seite 33 Use-Case-Diagramme, welche die Anwendungsfälle detailliert darstellen.

3.9. Qualitätsanforderungen

Die Qualitätsanforderungen der Anwendung sind hinlänglich im dem dem Projekt zugehörigen Pflichtenheft beschrieben.

Hier wird ein gekürzter Auszug aus diesem dargestellt.

| Qualitätsmerkmal | sehr gut | gut | normal | nicht relevant |
|------------------|----------|-----|--------|----------------|
| Funktionalität | | X | | |
| Zuverlässigkeit | | | X | |
| Benutzbarkeit | | X | | |
| Effizienz | | | X | |
| Wartbarkeit | | | X | |
| Übertragbarkeit | | X | | |

Tabelle 6: Qualitätsanforderungen an die Anwendung - Auszug

Eine detaillierte Übersicht der Qualitätsanforderungen findet sich im Anhang G auf Seite 35.

3.10. Lastenheft / Fachkonzept

Das Grobkonzept der Anwendung ist wie folgt beschrieben:

”Die Anwendung ist eine Webapplikation, welche es dem Benutzer erlaubt den Inhalt eines Wikis in eine PDF- bzw. HTML-Datei zu konvertieren und diese anschließend herunterladen. Dazu kann der Benutzer selbst neue Wikis einpflegen. Gespeicherte Wikis sowie konvertierte Inhalte können mit anderen Nutzern geteilt werden.”

Eine detaillierte Übersicht über die Zielbestimmungen, den Produkteinsatz und die Produktübersicht findet sich in Anhang H auf Seite 36.

3.11. Zwischenstand

| Teilphase | Geplant | Tatsächlich | Differenz |
|------------------------------|-----------|-------------|-------------|
| Sichtung des Lastenhefts | 1h | 1h | |
| Erstellen des Pflichtenhefts | 2h | 3h | + 1h |
| Wirtschaftlichkeitsanalyse | 1h | 2h | + 1h |
| Risikoanalyse | 1h | 1h | |
| ”Make or Buy” Analyse | 1h | 1h | |
| Anwendungsfälle definieren | 1h | 1h | |
| Gesamt | 7h | 9h | + 2h |

Tabelle 7: Zwischenstand nach der Analysephase

4. Entwurfsphase

4.1. Zielplattform

Im folgenden wird dargelegt, aus welchen Gründen die gewählte Zielplattform anvisiert wird.

Programmiersprache JAVA, TypeScript

Das Backend der Anwendung wird mit der Programmiersprache Java in Version 7 umgesetzt. Dies ist primär mit der bereits vorhandenen Erfahrung der Projektmitarbeiter begründet. Außerdem ist die bereits existierende Ziellaufzeitumgebung (s. 4.1) für Java-Anwendungen ausgelegt. Für die Frontend-Programmierung wird TypeScript in Version 1.6 verwendet. Dies ist damit begründet, dass wie in Kapitel 3.7 beschrieben der Einsatz eines JavaScript-Frameworks anvisiert wird. Des weiteren bietet TypeScript im Gegensatz zu JavaScript umfassendere statische Typprüfungen. Dies erleichtert die Programmierung und verhindert spätere Laufzeitfehler.

Application-Server Tomcat

Als Application-Server wird Tomcat in Version 7 eingesetzt. Dies liegt vor allem an dem leichtgewichtigen Umfang und den damit verbundenen geringen Installations- und Administrationsaufwand des Application-Servers.

Datenbank MySQL

Als Datenbanksystem wird MySQL verwendet. Dies ist in erster Linie mit der bereits vorhandenen Erfahrung der Projektmitarbeiter begründet.

Server

Die spätere Laufzeitumgebung inklusive der Installation der Anwendung in dieser ist, wie in Kapitel 1.5 beschrieben, zwar nicht Teil des Projekts, der Vollständigkeit halber ist hier allerdings in Kürze die vom Kunden bereitgestellt Server-Umgebung beschrieben.

- Server mit ausreichen leistungsfähigem Intel / AMD Prozessor
- ausreichend Arbeitsspeicher
- Debian Linux Betriebssystem
- MySQL Datenbankserver
- Tomcat Application-Server
- Java Laufzeitumgebung in Version 7 oder höher

4.2. Architekturdesign

Im folgenden wird dargelegt, aus welchen Gründen die jeweilige Softwarearchitektur des Back- und Frontends gewählt wurde. Eine detaillierte Übersicht über die Wahl Architekturspezifischer Frameworks findet dagegen sich in Kapitel 3.7.

4.2.1. Serverarchitektur

Als Grundlage der Serverarchitektur wird das REST-Prinzip⁴ in Verbindung mit JSON als Datenformat gewählt.

Daraus ergeben sich folgende Vorteile:

Separation of Concerns Der Server hat klar definierte und abgegrenzte Aufgabengebiete. Dazu zählen das Verwalten (CRUD⁵) von persistenten Daten. Sowohl von solchen, die den Application-Lifecycle überleben (Datenbankeinträge), als auch von flüchtigen, wie etwa Session-Daten. Außerdem ist der Server für die Zugriffsbeschränkung auf die Daten zuständig.

Nicht zu den Aufgaben des Servers gehören dagegen z.B. Oberflächen-Logik oder das halten des generellen Status einer Session.

Applikationsübergreifende Schnittstelle Aufgrund des REST-Prinzips ist es leicht möglich, Anwendungsdaten gegenüber dritten, anwendungsfremden Diensten bereitzustellen, um so den Nutzen der Anwendung langfristig zu sichern.

Datenformat Durch die Wahl von JSON als Datenformat entfällt zum einen eine clientseitige Konvertierung, da JSON in der JavaScript-Umgebung eines Browsers ein natives Datenformat darstellt. Zum anderen kann bei der serverseitigen Konvertierung der Daten auf vorhandene Bibliotheken, beispielsweise Jackson⁶, zurückgegriffen werden.

Allgemein Die Wahl der Architektur weist in Verbindung mit der Programmiersprache Java den Vorteil auf, dass der JEE-Stack bereits Standardkomponenten für diese Anwendungsbereiche wie etwa JAX-RS⁷ aufweist.

4.2.2. Clientarchitektur

Clientseitig wird eine auf dem Flux⁸ Konzept basierende Model-View-Architektur verwendet.

Diese weist folgende Vorteile auf:

⁴<http://www.infoq.com/articles/designing-restful-http-apps-roth>, abgerufen am 07.10.2015

⁵Create, Update, Read, Delete

⁶<http://wiki.fasterxml.com/JacksonHome>, abgerufen am 07.10.2015

⁷<https://jcp.org/en/jsr/detail?id=339>, abgerufen am 07.10.2015

⁸<https://facebook.github.io/flux/docs/overview.html>, abgerufen am 07.10.2015

Trennung von Daten und Darstellung Durch die strikte Trennung von Daten (repräsentiert durch zustandslose Stores) und deren Darstellung (zustandsbehaftete Komponenten) werden Fehler in der Geschäft- und Oberflächenlogik vermieden. Der unidirektionale Datenfluß der Flux-Architektur verhindert außerdem konkurrierende Zugriffe auf Daten.

Übersichtlichkeit Die starke Modularisierung (Aufteilung in Stores, Actions und Komponenten) erhöht die Übersichtlichkeit und erleichtert langfristig die Analyse bzw. das Refactoring des Frontends.

Erweiterbarkeit Aufgrund der bereits erläuterten Modularisierung und der strikten Objektorientierung des Frontends ist eine leichte Erweiterbarkeit von diesem gewährleistet.

4.3. Entwurf der Benutzeroberfläche

Die Benutzeroberfläche wird mit den Techniken HTML in Version 5 und CSS in Version 3 realisiert. JavaScript in Version 5 wird zur Unterstützung und Verbesserung der User Experience Verwendung finden. Umgesetzt wird die Oberfläche mit Hilfe des Frameworks React.

4.3.1. Visuelles Konzept

Die Benutzeroberfläche wird in einzelne Bereiche unterteilt, um dem Benutzer eine durchgehend gleichbleibende Grundstruktur zu bieten und so die Übersichtlichkeit der Anwendung zu maximieren.

Layout Zu diesem Zweck wird die Oberfläche in einen Header, welcher die Navigationsleiste, sowie Angaben zum aktuellen Status beherbergt, sowie einen Hauptbereich, welcher die statusabhängigen Formulare und Daten anzeigt unterteilt. Der Hauptbereich ist wiederum in Spalten und Zeilen (ja nach aktuellem Status) unterteilt, wobei die maximale Anzahl dieser Unterbereiche auf üblicherweise zwei bis drei beschränkt ist, um die Übersichtlichkeit zu gewährleisten.

Design Grundlagen Die Benutzeroberfläche wird nach den Grundlagen von Responsive Design⁹ entwickelt, um die Attraktivität und den Nutzwert der Anwendung für verschiedene Endgeräte wie z.B. Smartphones oder Tablets zu gewährleisten.

Visuelle Richtlinien Die Darstellung lehnt sich dabei zum einen an die Richtlinien von Material Design¹⁰ an, um einen modernen Gesamteindruck zu erzielen. Zum anderen orientiert sich das Farbschema der Oberfläche am Corporate Design des Auftraggebers.

⁹<http://alistapart.com/article/responsive-web-design>, abgerufen am 07.10.2015

¹⁰<https://www.google.com/design/spec/material-design>, abgerufen am 07.10.2015

4.3.2. Prototyp

Eine bildhafte Darstellung eines funktionslosen Prototypen, welcher die hier aufgeführten Visuellen Konzepte verdeutlichen soll findet sich in Anhang I auf Seite 38.

4.4. Datenmodell

Das der Anwendung zugrunde liegenden Datenmodell wird in Form eines Entity-Relationship-Modells in Anhang J auf Seite 40 dargestellt.

4.5. Geschäftslogik

Den Hauptprozess der Anwendung stellt das Erstellen von Handbüchern dar. Eine schematische Beschreibung von diesem findet sich in Anhang F auf Seite 33 in Form eines Use-Case-Diagramms.

Ein umfassenderer Überblick über den logischen Aufbau der Anwendung findet sich in Anhang K auf Seite 41.

4.6. Maßnahmen zur Qualitätssicherung / Testkonzept

Im Folgenden sind die qualitätssichernden Maßnahmen des Projekts beschrieben.

Dokumentation

Um die Wartbarkeit der Anwendung zu erleichtern und langfristig zu gewährleisten werden die Quelltexte dokumentiert. Dies geschieht nach vorherrschenden Standards, speziell JavaDoc¹¹ bzw. JsDoc¹².

Des weiteren werden Informationen bezüglich der Entwicklungsumgebung und -toolchain in readme-Dateien schriftlich festgehalten, um die Wartung durch Dritte zu vereinfachen.

Tests

Automatisierte Tests Für einzelne kritische Komponenten werden Unit-Tests implementiert. Diese helfen beim späteren Refactoring der Anwendung frühzeitig Fehler durch dieses zu erkennen.

Manuelle Tests Zur Feststellung der Fehlerfreiheit der Anwendungsoberfläche wird jedes Feature der Anwendung manuell, sprich durch händische Durchführung, getestet. Als Testumgebung werden aktuelle Versionen der populärsten Browser eingesetzt, hier Microsoft Internet Explorer, Mozilla Firefox und Google Chrome.

¹¹<http://www.oracle.com/technetwork/articles/java/index-jsp-135444.html>, abgerufen am 27.10.2015

¹²<http://usejsdoc.org/>, abgerufen am 27.10.2015

4.7. Pflichtenheft

Ein Auszug des im Rahmen der Entwurfsphase erstellten Pflichtenhefts findet sich in Anhang L auf Seite 42.

4.8. Zwischenstand

| Teilphase | Geplant | Tatsächlich | Differenz |
|---|------------|-------------|-------------|
| Erstellen des Projektplans | 2h | 2h | |
| Erstellen des Qualitätssicherungskonzepts | 2h | 2h | |
| Spezifizieren der Architektur-Grundlagen | 8h | 6h | - 2h |
| Erstellen des Datenmodells | 4h | 3h | - 1h |
| Erstellen des Testkonzepts | 2h | 2h | |
| Gesamt | 18h | 15h | - 3h |

Tabelle 8: Zwischenstand nach der Entwurfsphase

5. Implementierungsphase

5.1. Iterationsplanung

Der Implementierungsphase läuft die Erstellung eines Iterationsplans voraus. In diesem werden die einzelnen Teilschritte der Phase benannt, um den aktuellen Fortschritt der Entwicklung besser verifizieren zu können und die logischen Abhängigkeiten der Teilmodule zu verdeutlichen.

Der Iterationsplan findet sich in Anhang M auf Seite 43.

5.2. Implementierung der Datenstrukturen

Auf Basis des in Kapitel 4.4 beschriebenen Datenmodells wird die eigentliche Datenbank implementiert. Hierzu wird auf einem bestehenden MySQL-Server eine neue Datenbank und ein neuer Benutzer angelegt. Diesem werden daraufhin alle nötigen Privilegien gewährt.

Anschließend wird mit Hilfe des Tools MySQL Workbench¹³ ein SQL-Skript erzeugt, welches die Erstellung des Datenmodells abbildet. Dieses Skript wird in einzelnen Abschnitte nach Tabellen unterteilt und innerhalb des Projekts in einzelnen Dateien abgelegt.

Die Anwendung führt beim Start eine Prüfung des aktuellen Schemas durch und führt darauf basierend einzelne SQL-Skripte aus. Dadurch ist ein stets konsistentes Schema, auch bei späteren Änderungen des Datenmodells, gewährleistet. Des weiteren wird auf diesem Wege keine manuelle Konfiguration der Datenbanktabellen benötigt, wenn die Anwendung auf einem neuen System deployed wird.

¹³<https://www.mysql.de/products/workbench/>, abgerufen am 27.10.2015

5.3. Implementierung der Geschäftslogik

Die Entwicklung der Geschäftslogik erfolgt, wie bereits erwähnt, mit der Programmiersprache JAVA unter Zuhilfenahme der Entwicklungsumgebung Eclipse. Für die Verwaltung von Abhängigkeiten und Bibliotheken wird Apache Maven¹⁴ verwendet.

Die Anwendung wird dabei in drei Teilmodule, namentlich "Backend", "Konverter" und "Frontend" unterteilt, wobei für jedes dieser Module ein eigenes Eclipseprojekt angelegt wird, um die Übersichtlichkeit während der Entwicklung zu erhöhen.

Im Folgenden werden einzelnen Teile der Geschäftslogik und deren Implementierungsdetails erläutert.

Hilfsfunktionalitäten

Datenbankvalidierung Zur Validierung des Datenbankschemas wird eine Klasse `DatabaseUpdater` implementiert. Diese liest die Version des Datenbankschemas anhand einer Versionstabelle (Die beim Erststart auch durch die Klasse angelegt wird) aus und vergleicht diese mit den verfügbaren SQL-Skripten der Anwendung. Neue Skripte werden daraufhin auf der Datenbank ausgeführt, um das Datenbankschema beim Start der Anwendung stets auf den aktuellsten Stand zu bringen.

Ein Auszug der Klasse befindet sich Anhang N auf Seite 44.

Testdatengenerierung Zur programmatischen Erstellung von Testdatensätzen wird eine Klasse `DataBaseSeeder` implementiert. Diese bietet die Funktionalität neue Testdatensätze zu erstellen, welche beim Anwendungsstart ausgeführt wird.

Entitäten

Alle Entitäten des Datenmodells, die programmatisch editierbar sind, werden durch zugehörige Java-Beans abgebildet. Dieses Konzept folgt dem Standard von JPA. Um den Entwicklungsaufwand zu minimieren leiten alle Entitätsklassen von der Klasse `BaseModel` ab, welche Grundkonfiguration und Felder aller Tabellen beschreibt.

REST-API

Für jeden Entitätsklasse wird eine zugehörige API-Klasse erstellt. Diese erzeugen jeweils die über HTTP ansprechbaren REST-Endpunkte und sind für die Ausführung aller Datenbankzugriffe zuständig. Alle API-Klassen leiten von der generischen Klasse `BaseAPI` ab. In dieser sind bereits eine Reihe an ansprechbaren Pfaden / Methoden implementiert, sodass spezielle API-Klassen nur entitätsspezifische Methoden umsetzen müssen.

Ein Auszug der Klasse `BaseAPI` befindet sich Anhang N auf Seite 45.

¹⁴<https://maven.apache.org/>, abgerufen am 27.10.2015

Konverter

Grundlage des Konverter Moduls ist der Quelltext des bereits bestehenden WikiConverters. Dieser wird dahingehend angepasst, dass die Konvertierung in einem eigenen Thread ausgeführt und angestoßen werden kann.

5.4. Implementierung der Benutzeroberfläche

Die Benutzeroberfläche der Anwendung wird in HTML implementiert. Layout und Optik der Oberfläche werden mit Hilfe von CSS beschrieben. Unterstützend wird das Framework React zur Beschreibung komplexer GUI-Komponenten verwendet.

Layout

Das Layout der Oberfläche wird im Wesentlichen durch die Anordnung einzelner Komponenten in Spalten und Zeilen festgelegt. Die Umsetzung erfolgt mit der CSS3 Spezifikation FlexBox¹⁵ umgesetzt.

Des Weiteren passt sich das Layout dynamisch an das Ausgabeformat an, um so eine effiziente Anordnung der einzelnen Komponenten auf verschiedenen Endgeräten zu gewährleisten. Dies wird mit Hilfe der CSS Spezifikation Media Queries¹⁶ erreicht.

Design

Das Farbdesign orientiert sich an betriebsinternen Richtlinien, um die Corporate Identity der GNS zu adaptieren.

Das Design der einzelnen GUI-Komponenten orientiert sich in Zügen an Standards moderner (mobiler) Betriebssysteme, allen Voran Material Design, um einen modernen, vertrauten Gesamteindruck zu erzeugen.

5.5. Implementierung der Tests

Unit-TESTS!!! GUI-TESTS(Mocha!?)!!!

¹⁵<http://www.w3.org/TR/css-flexbox-1/>, abgerufen am 27.10.2015

¹⁶<http://www.w3.org/TR/mediaqueries-4/>, abgerufen am 27.10.2015

5.6. Zwischenstand

| Teilphase | Geplant | Tatsächlich | Differenz |
|---------------------------------|------------|-------------|-------------|
| Erstellen des Datenbanktabellen | 2h | 1h | - 1h |
| Entwicklung des Backends | 7h | 10h | + 3h |
| Entwicklung des Frontends | 9h | 20h | + 11h |
| Dokumentation des Quellcodes | 2h | 1h | - 1h |
| Testing | 2h | 2h | |
| Soll-Ist-Vergleich | 2h | 1h | - 1h |
| Puffer für Korrekturen | 2h | - | - 2h |
| Gesamt | 26h | 35h | + 9h |

Tabelle 9: Zwischenstand nach der Implementierungsphase

6. Abnahmephase

test

6.1. Zwischenstand

7. Einführungsphase

7.1. Zwischenstand

8. Dokumentation

8.1. Zwischenstand

9. Fazit

9.1. Soll- / Ist-Vergleich

9.2. Lesson-Learned

9.3. Ausblick

Anhang

A. Projektphasen

| Projektphase | Geplante Zeit |
|---|---------------|
| Anforderungsanalyse | Gesamt: 7h |
| Sichtung des Lastenhefts | 1h |
| Erstellen des Pflichtenhefts | 2h |
| Wirtschaftlichkeitsanalyse | 1h |
| Risikoanalyse | 1h |
| "Make or Buy" Analyse | 1h |
| Anwendungsfälle definieren | 1h |
| Projektplanung | Gesamt: 4h |
| Erstellen des Projektplans | 2h |
| Erstellen des Qualitätssicherungskonzepts | 2h |
| Software-Architektur | Gesamt: 14h |
| Spezifizieren der Architektur-Grundlagen | 8h |
| Erstellen des Datenmodells | 4h |
| Erstellen des Testkonzepts | 2h |
| Implementierung | Gesamt: 26h |
| Erstellen der Datenbanktabellen | 2h |
| Entwicklung des Backends | 7h |
| Entwicklung des Frontends | 9h |
| Dokumentation des Quellcodes | 2h |
| Testing | 2h |
| Soll-Ist-Vergleich | 2h |
| Puffer für Korrekturen | 2h |
| Projektabschluss | Gesamt: 3h |
| Anwenderschulung | 2h |
| Abnahme | 1h |
| Sonstiges | Gesamt: 16h |
| Projektdokumentation | 12h |
| Allgemeiner Puffer | 4h |
| Gesamt | 70h |

Tabelle 10: Projektphasen detailliert

B. Risikoanalyse

Die folgende Tabelle listet alle für das Projekt als relevant betrachteten Risiken auf. Ermittlungsgrundlage dieser Risiken ist neben der persönlichen Erfahrung der Projektbeteiligten eine interne Mitarbeiterbefragung

| Risiko | Ursache | Maßnahme | Eintrittswahr. nach der Maßnahme | Auswirkung nach der Maßnahme |
|---|-------------------------------------|--------------------------|--|--|
| Terminrisiken | | | | |
| Nicht einhalten des Abnahmetermins | uneingeplante Hindernisse | keine | mittel | Start der Anwendungsnutzung verzögert sich |
| Technische Risiken | | | | |
| Anwendung ist auf Zielplattform nicht lauffähig | Entwicklung gegen falsche Plattform | Anpassung der Anwendung | mittel | Zustätzlicher Zeitbedarf für Anpassungen |
| Personelle Risiken | | | | |
| Ausfall des Projektleiters | Krankheit | keine | gering | Projekt kommt zum erliegen |
| Ausfall eines Projektmitarbeiters | Krankheit | keine | gering | Projekt kommt zum erliegen |
| Planungsrisiken | | | | |
| Unterschätzung der Dauer einzelner Phasen | mangelnde Erfahrung | nachtrgäliche Planung | mittel | Verzögerung des Projektabschluss |
| Auslassung relevanter Phasen | mangelnde Erfahrung | nachträgliche Planung | gering | Verzögerung des Projektabschluss |
| Risiken der Analyse und Konzeption | | | | |
| Fehlerhafte Ist-Analyse | ungenauere Analyse | Nachbesserung d. Analyse | gering | iterativer Rücksprung im Entwicklungsprozess |
| Fehlerhafte Soll-Analyse | fehlerhafte Spezifikation | Nachbesserung d. Analyse | gering | iterativer Rücksprung im Entwicklungsprozess |
| Fehlerhafte Architektur / Konzept | ungenauere Spezifikation | Neukonzeption | gering | iterativer Rücksprung im Entwicklungsprozess |
| Realisierungsrisiken | | | | |
| Unerwartete Entwicklungsprobleme | nicht bestimmbar | teilweise Neukonzeption | mittel | verlängte Entwicklungszeit |

| Risiko | Ursache | Maßnahme | Eintrittswahr. nach der Maßnahme | Auswirkung nach der Maßnahme |
|---------------------------------|------------------|------------------------------|--|------------------------------------|
| Betreuungs- und Wartungsrisiken | | | | |
| Breaking-Changes | MediaWiki Update | Anpassung der Geschäftslogik | hoch | keine |

Tabelle 11: Risikoanalyse detailliert

C. "Make or Buy"-Bewertung

Grundlage der im Folgenden betrachteten Alternativen ist eine umfängliche Recherche bezüglich der Kernfrage "Welche Lösungen zur Erstellung von Handbüchern aus Wikis existieren bereits?". Betrachtet werden hier nur die stärksten Ergebnisse dieser Recherche, um die Übersichtlichkeit der Analyse zu bewahren.

| Alternative | Nutzen | Wirtschaftlichkeit |
|--|--------|--------------------|
| MediaWiki Erweiterung | | |
| einfache Installation | | ⊕ |
| Installation für jedes Wiki nötig | | ⊖ |
| Workflow umständlich | ⊖ | ⊖ |
| Ergebnis bedarf händische Anpassung | ⊖⊖ | ⊖⊖ |
| Kompatibilitätsprobleme mit MediaWiki | | ⊖ |
| Erstellung der Anwendung durch Dienstleister | | |
| spezialisierte Anwendung | ⊕⊕ | ⊖ ⊖ ⊖ |
| Wartung durch Dienstleister nötig | | ⊖⊖ |
| Risikoausslagerung | | ⊕⊕ |
| Eigenständige Erstellung der Anwendung | | |
| spezialisierte Anwendung | ⊕⊕ | ⊖⊖ |
| Wartung durch eigene Mitarbeiter | ⊕ | ⊕ |
| Tragen von Risiken | | ⊖ |

Tabelle 12: Detaillierte "Make or Buy"-Bewertung

D. Kostenübersicht nach Projektphasen

Grundlage der Folgenden Kostenübersicht sind die in Anhang A aufgeführten Phasen des Projekts. Als Rechnungsgrundlage wird der in Kapitel 3.2 ermittelte Stundensatz von 16 € herangezogen.

| Phase | Kosten |
|----------------------|---------------|
| Anforderungsanalyse | 112 € |
| Projektplanung | 64 € |
| Software-Architektur | 224 € |
| Implementierung | 416 € |
| Projektabschluss | 48 € |
| Sonstiges | 256 € |
| Gesamt | 1120 € |

Tabelle 13: Kosten unterteilt nach Projektphasen

E. Nutzwertanalyse

E.1. Backend-Architektur

Im Folgenden wird über eine Nutzwertanalyse die beste Backend-Architektur ausgewählt. Die hier betrachteten Alternativen sind die stärksten Ergebnisse einer vorangegangenen Recherche.

| Eigenschaft | Gewichtung | JEE-Stack ¹⁷ | GNS-Framework ¹⁸ | SrpingMVC ¹⁹ |
|---------------------------------|------------|-------------------------|-----------------------------|-------------------------|
| REST-API integriert | 3 | 5 | 0 | 4 |
| Trennung von Back- und Frontend | 4 | 5 | 0 | 4 |
| Dokumentation | 2 | 3 | 1 | 5 |
| Testbarkeit | 2 | 2 | 1 | 3 |
| Refactoring (ggf. durch Dritte) | 3 | 3 | 4 | 2 |
| Gesamt | 14 | 18 | 6 | 18 |
| Nutzwert | | 3,86 | 1,14 | 3,57 |

Tabelle 14: Detaillierte Nutzwertanalyse bezüglich der Backend-Architektur

E.2. Frontend-Architektur

Im Folgenden wird über eine Nutzwertanalyse die beste Frontend-Architektur ausgewählt. Die hier betrachteten Alternativen sind die stärksten Ergebnisse einer vorangegangenen Recherche.

¹⁷<http://www.oracle.com/technetwork/java/javaee/overview/index.html>, abgerufen am 08.11.2015

¹⁸ein betriebsinternes Framework zur Entwicklung von Webanwendungen

¹⁹<https://spring.io>, abgerufen am 08.11.2015

| Eigenschaft | Gewichtung | JSF ²⁰ | Angular ²¹ | kein Framework | React ²² |
|-------------------|------------|-------------------|-----------------------|----------------|---------------------|
| Bootstrap-Aufwand | 2 | 3 | 3 | 5 | 2 |
| Modularität | 4 | 4 | 3 | 1 | 5 |
| Erweiterbarkeit | 5 | 3 | 4 | 0 | 5 |
| Dokumentation | 2 | 3 | 5 | 4 | 4 |
| User-Experience | 4 | 3 | 5 | 2 | 5 |
| Kompatibilität | 2 | 4 | 2 | 5 | 2 |
| Gesamt | 19 | 20 | 22 | 17 | 20 |
| Nutzwert | | 3,32 | 3,79 | 2,11 | 4,26 |

Tabelle 15: Detaillierte Nutzwertanalyse bezüglich der Frontend-Architektur

²⁰<http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html>, abgerufen am 08.11.2015

²¹<https://angularjs.org>, abgerufen am 08.11.2015

²²<https://facebook.github.io/react/>, abgerufen am 08.11.2015

F. Use-Case Diagramme

F.1. Wiki konvertieren

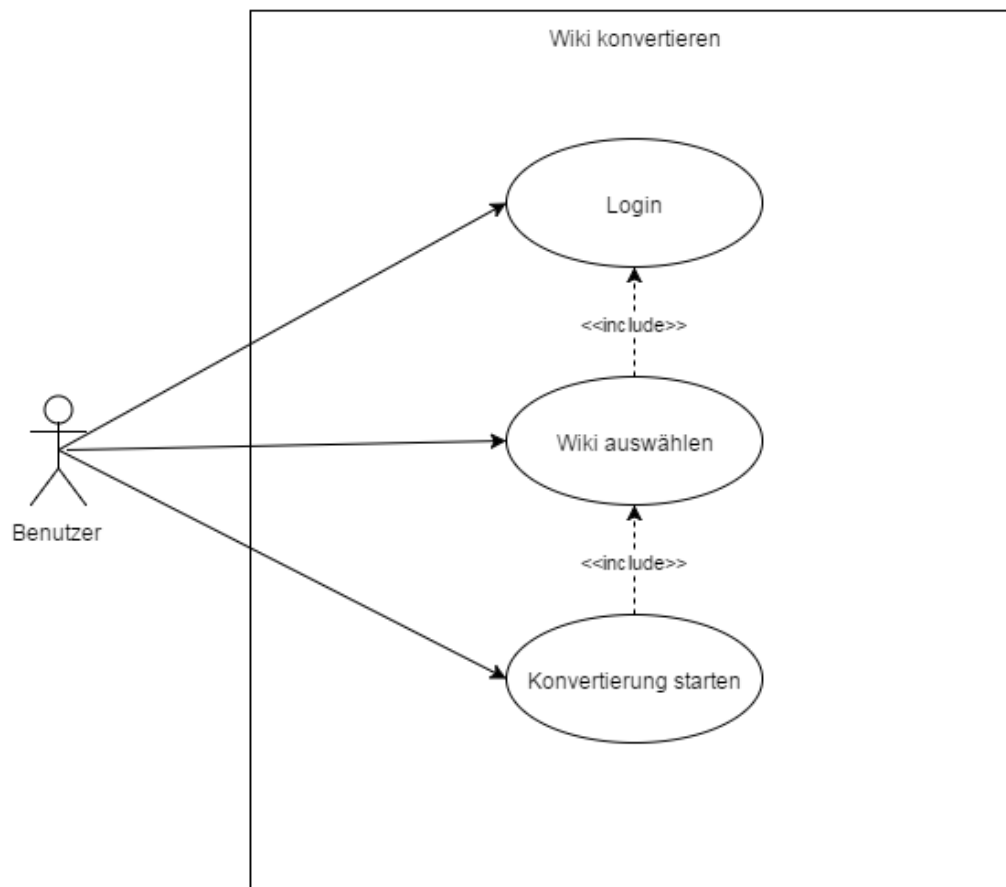


Abbildung 1: Use-Case "Wiki konvertieren"

F.2. Wiki bearbeiten

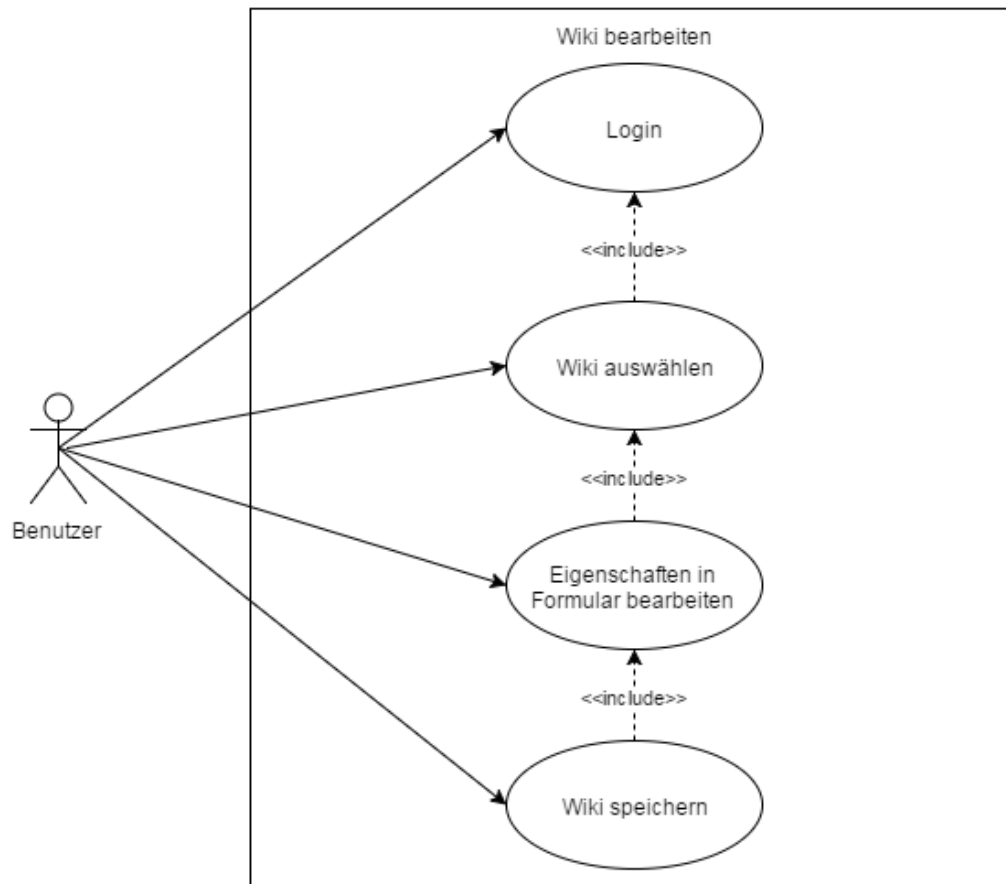


Abbildung 2: Use-Case "Wiki bearbeiten"

G. Qualitätsanforderungen (Auszuf aus Pflichtenheft)

| Qualitätsmerkmal | sehr gut | gut | normal | nicht relevant |
|------------------------|----------|-----|--------|----------------|
| Funktionalität | | | | |
| Angemessenheit | | | X | |
| Richtigkeit | | X | | |
| Interoperabilität | X | | | |
| Sicherheit | | X | | |
| Ordnungsmäßigkeit | | | X | |
| Konformität | | | X | |
| Zuverlässigkeit | | | | |
| Reife | | | X | |
| Fehlertoleranz | | | X | |
| Wiederherstellbarkeit | | X | | |
| Konformität | | | X | |
| Benutzbarkeit | | | | |
| Verständlichkeit | X | | | |
| Erlernbarkeit | | X | | |
| Bedienbarkeit | | X | | |
| Attraktivität | | | X | |
| Konformität | | | | X |
| Effizienz | | | | |
| Zeitverhalten | | X | | |
| Verbrauchsverhalten | | | X | |
| Konformität | | | X | |
| Wartbarkeit | | | | |
| Analysierbarkeit | | | X | |
| Modifizierbarkeit | | X | | |
| Stabilität | | | X | |
| Testbarkeit | | | | X |
| Konformität | | | X | |
| Übertragbarkeit | | | | |
| Anpassbarkeit | | X | | |
| Installierbarkeit | | X | | |
| Koexistenz | X | | | |
| Austauschbarkeit | | X | | |
| Konformität | | | | X |

Tabelle 16: Qualitätsanforderungen an die Anwendung (Pflichtenheft)

H. Lastenheft (Auszug)

Zielbestimmung

Musskriterien

- Benutzerverwaltung
 - Benutzer anlegen
 - Benutzerdaten editieren
 - Benutzer löschen
- Wikiverwaltung
 - Wiki anlegen
 - Wikidaten editieren
 - Wiki löschen
 - Handbuch aus Wiki generieren
 - Handbuch runterladen

Wunschkriterien

- Handbuchversionen
 - Handbuchhistorie eines Wikis einsehen
 - Handbuch aus Historie löschen
 - Älteres Handbuch runterladen
- Handbuchgenerierung
 - Aktuellen Fortschritt während Generierung anzeigen
 - Erstellung für Wikis sperren, während aktuelle Generierung läuft
- Gruppenverwaltung
 - Gruppen anlegen
 - Gruppen löschen
 - Benutzer Gruppen zuteilen
 - Rechte Gruppen zuteilen

Abgrenzungskriterien

- Single-Sign-On über GNS internes ActiveDirectory
- Anlegen / Konfigurieren der Laufzeitumgebung innerhalb der GNS DMZ

Produkteinsatz

Anwendungsbereiche

Technischer/administrativer Anwendungsbereich.

Zielgruppen

Die Zielgruppe besteht ausschließlich aus Mitarbeitern der GNS mbH. Vornehmlich wird das Produkt durch Angestellte der Abteilung KIS benutzt werden.

Betriebsbedingungen

Das Produkt wird als Webanwendung bereitgestellt. Das Produkt ist über das Internet erreichbar. Es wird eine Up-Time von nahezu 100% angestrebt.

Produktübersicht

Das Produkt ist eine Webapplikation, welche es dem Benutzer erlaubt den Inhalt eines existierenden Mediawikis in eine PDF- respektive HTML-Datei zu konvertieren und diese anschließend herunterzuladen. Hierzu kann der Nutzer selbstständig neue Wikis indizieren um diese für das spätere Konvertieren bereitzustellen. Konvertierte Mediawikis können anderen Benutzern ebenfalls zum download bereitgestellt werden.

I. Oberflächenprototyp

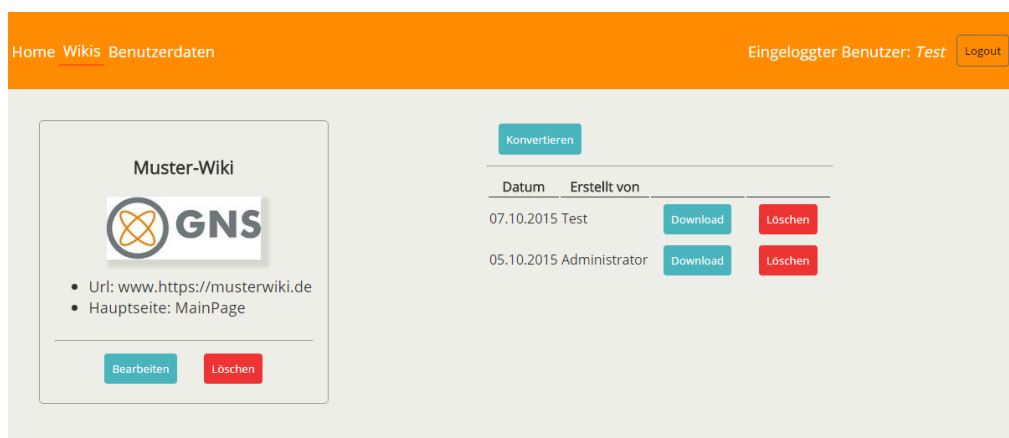


Abbildung 3: Prototyp - Layout für Desktop-Browser

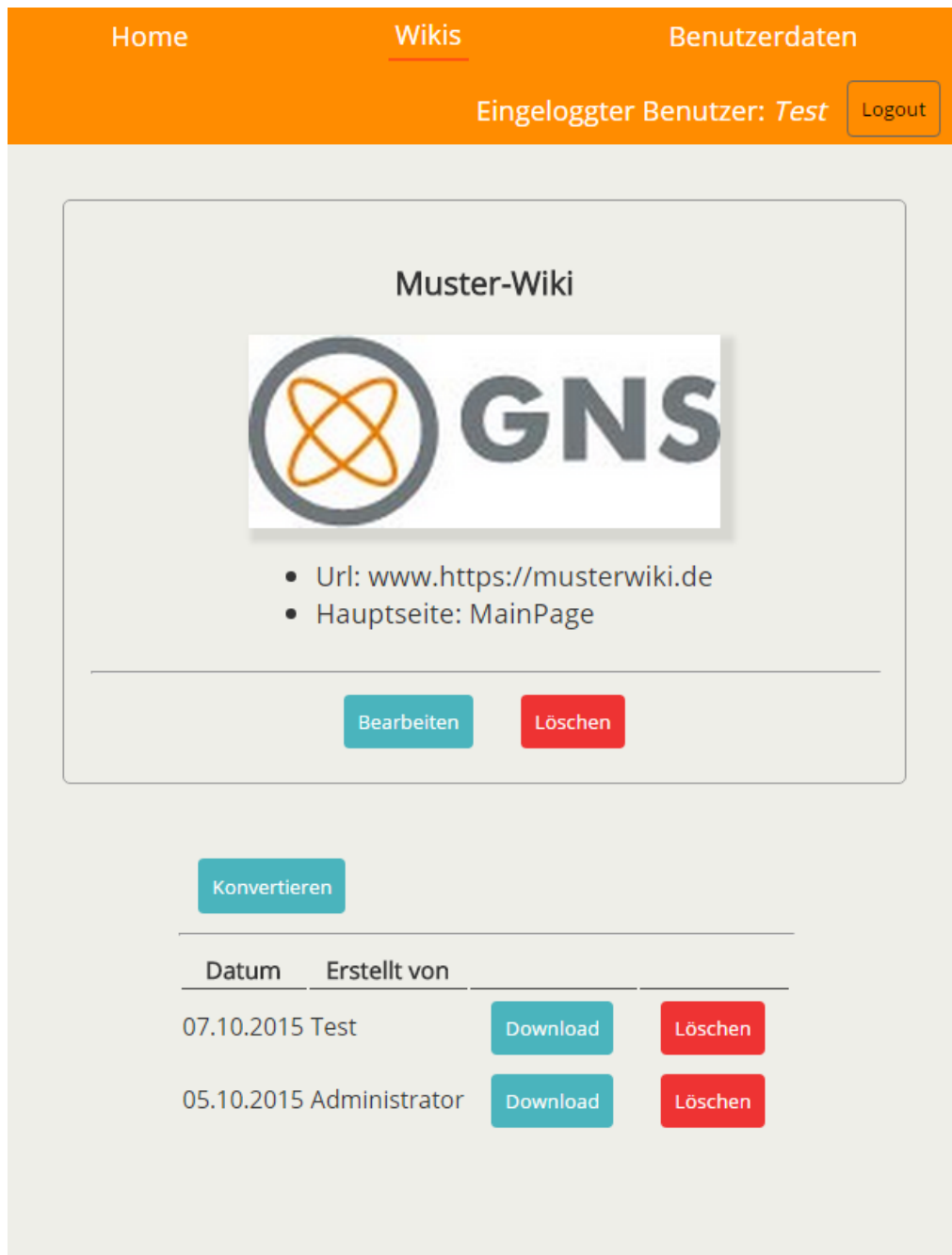


Abbildung 4: Prototyp - Layout für Mobile-Browser

J. Entity-Relationship-Modell

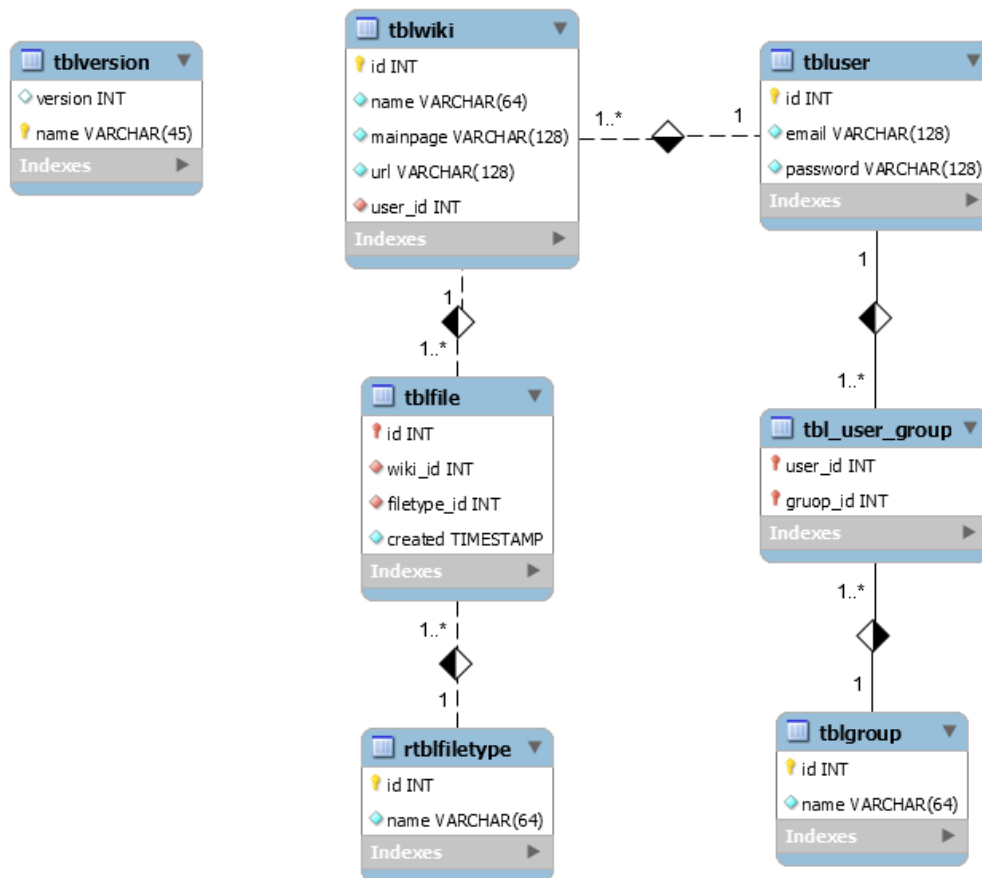


Abbildung 5: ER-Modell der Anwendung

K. Komponentendiagramm

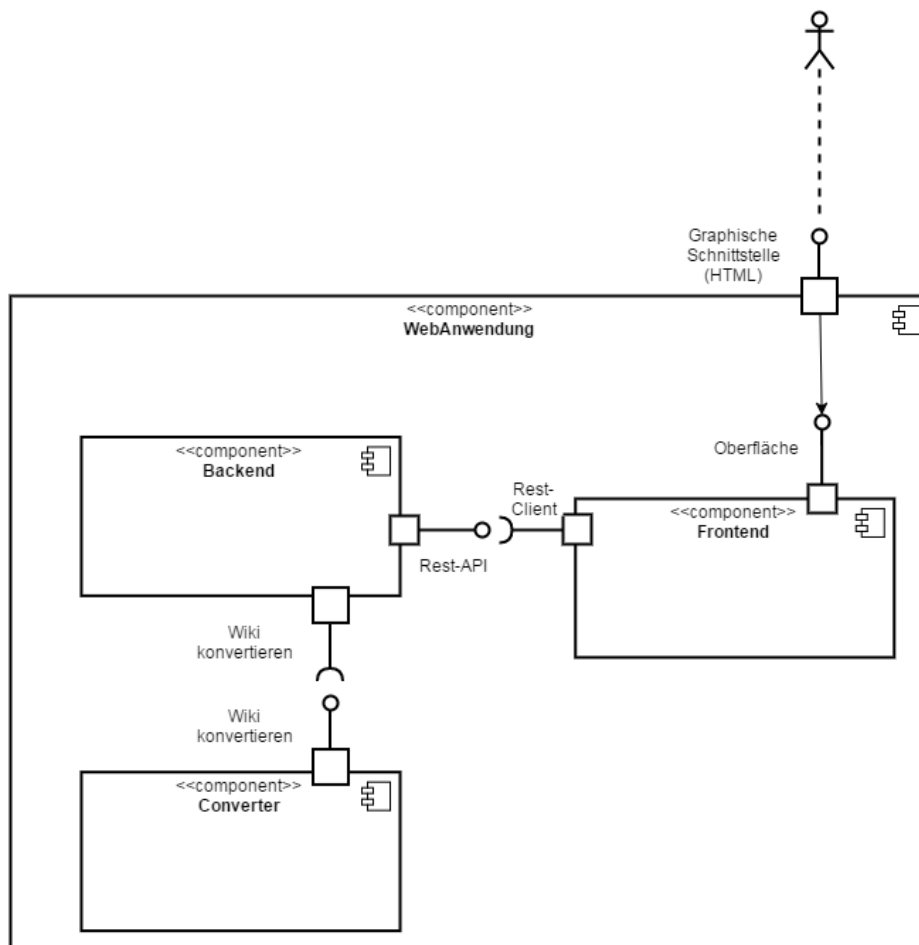


Abbildung 6: Komponentendiagramm

L. Pflichtenheft (Auszug)

Produktfunktionen

- /F10/
Geschäftsprozess: Login
Nachbedingung Erfolg: Benutzer ist angemeldet
Nachbedingung Fehlschlag: Fehlermeldung wird angezeigt
Beschreibung:
 1. Benutzer gibt Anmeldedaten ein

2. Benutzer klickt "Login"-Button

- /F20/

Geschäftsprozess: Wiki anlegen

Vorbedingung: Benutzer ist angemeldet

Nachbedingung Erfolg: Neues Wiki ist angelegt

Nachbedingung Fehlschlag: Fehlermeldung wird angezeigt

Beschreibung:

1. Benutzer navigiert zum Bereich "Wikis"
2. Benutzer klickt "Neu"-Button
3. Benutzer gibt Stammdaten ein
4. Benutzer klickt "Speichern"-Button

- /F30/

Geschäftsprozess: Handbuch erstellen

Vorbedingung:

- Benutzer ist angemeldet
- Wiki ist angelegt

Nachbedingung Erfolg: Handbuch steht zum Download bereit

Nachbedingung Fehlschlag: Fehlermeldung wird angezeigt

Beschreibung:

1. Benutzer navigiert zum Bereich "Wikis"
2. Benutzer wählt Wiki aus
3. Benutzer klickt "Konvertieren"-Button
4. Konvertierungsfortschritt wird graphisch dargestellt

M. Iterationsplan

- Anlegen der Projekte innerhalb der IDE
- Integration von Bibliotheken über Maven
- Erstellen der (Test-)Datenbank
- Erstellen des Datenmodells
- Abbildung des Datenmodells auf SQL-Skripte
- Implementierung von Utility-Funktionalitäten
 - Datenbankvalidierung
 - Testdatenerstellung

- Authorisierung und Authentifizierung
- Implementierung der Datenentitäten nach JPA
- Implementierung der REST-API nach JAX-RS
- Implementierung des Frontends
- Implementierung / Anpassung des Konverter-Moduls

N. Listings

```
1 public class DatabaseUpdater {
2
3     private static Logger logger = Logger.getLogger(DatabaseUpdater.class
4         );
5
6     private String scriptPackage;
7     private String applicationName;
8
9     public DatabaseUpdater(String scriptPackage, String applicationName)
10    {
11        this.scriptPackage = scriptPackage;
12        this.applicationName = applicationName;
13    }
14
15    public void update(EntityManager em) {
16        EntityTransaction transaction = em.getTransaction();
17        transaction.begin();
18
19        try {
20            this.checkIfVersionTableExists(em);
21            for(File script : this.getScripts()) {
22                this.executeSql(em, script);
23            }
24        } catch (Exception e) {
25            logger.error(e.getMessage(), e);
26            transaction.rollback();
27            return;
28        }
29
30        transaction.commit();
31        em.close();
32    }
33
34    protected void checkIfVersionTableExists(EntityManager em) throws
35        SQLException {
36
37        java.sql.Connection conn = em.unwrap(java.sql.Connection.class);
38        conn.createStatement().executeUpdate(Version.getCreateScript());
39    }
40}
```

```
37
38     protected void executeSql(EntityManager em, File file) throws
        SQLException {
39         Version version = em.find(Version.class, this.applicationName);
40         if(version == null) {
41             version = new Version();
42             version.name = this.applicationName;
43             version.version = 0;
44             em.persist(version);
45         }
46
47         Integer fileVersion = getFileVersion(file);
48
49         if(version.version >= fileVersion)
50             return;
51
52         String sql = this.readFile(file);
53         logger.debug("Executing Sql: " + sql);
54         java.sql.Connection conn = em.unwrap(java.sql.Connection.class);
55         conn.createStatement().executeUpdate(sql);
56
57         version.version = fileVersion;
58     }
59
60     protected File[] getScripts() throws Exception {
61         String pkg = this.scriptPackage;
62         ClassLoader cl = Thread.currentThread().getContextClassLoader();
63
64         Enumeration<URL> resources = cl.getResources(pkg.replace(".", "/"));
65         ;
66         if(resources.hasMoreElements()) {
67             URL p = resources.nextElement();
68             File f_package = new File(p.toURI());
69             File[] files = f_package.listFiles();
70
71             Arrays.sort(files, new Comparator<File>() {
72                 @Override
73                 public int compare(File f1, File f2) {
74                     return getFileVersion(f1).compareTo(getFileVersion(f2));
75                 }
76             });
77
78             return files;
79         }
80         else
81             return new File[]{};
82
83     }
84
85     (...)
```

Listing 1: DataBaseUpdater.java

```
1
2 public class BaseApi<T extends BaseModel> {
3
4     @GET
5     @Produces(MediaType.APPLICATION_JSON)
6     @Path("{id}")
7     public T getOne(@Context HttpServletRequest request, @PathParam("id")
8         int id) {
9         EntityManager em = createEntityManager();
10        try {
11            beforeRetrieve(request);
12            beforeGetOne(request, id);
13
14            T t = em.find(clazz, id);
15
16            afterGetOne(request, t);
17
18            return t;
19        } catch (WebApplicationException e) {
20            throw e;
21        } catch (Exception e) {
22            log(e);
23            throw new GeneralException(e);
24        } finally {
25            em.close();
26        }
27    }
28    (...)
```

Listing 2: BaseApi.java