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

11. November 2015, Essen

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

API	Application Programming Interface
CSS	Cascading Style Sheets
CVS	Concurrent Version System
GNS	Gesellschaft für Nuklear-Service mbH
GUI	Graphical User Interface
Hosting	Bereitstellen einer Web-Anwendung auf einem Web-Server
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JEE	Java Platform, Enterprise Edition
JSON	JavaScript Object Notation
KIS	Abteilung der GNS für Softwareentwicklung
REST	Representational State Transfer
URL	Uniform Resource Locator

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 deren 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.1 auf Seite I

2.2. Abweichung vom Projektantrag

Abweichend vom Projektantrag werden das Pflichten- sowie das Lastenheft nur in Auszügen im Projektbericht aufgeführt. Dies ist dem Umfang des vorliegenden Projektberichts geschuldet.

2.3. Resourcenplanung

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

Art der Resource	Beschreibung
Sachliche Ressourcen	
Projektbüro	
Computer	
IDE	Eclipse for JEE Developers[1]
Demilitarisierte Zone	zu Testzwecken
Personelle Ressourcen	
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 voraussetzen.

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 Quellcode Angaben 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" definiert den überarbeiteten Workflow wie folgt:

- 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 Stundensätze[3]:

[2] `http://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
[3] Beruhend 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 schätzungsweise **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 Minuten**.

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

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

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

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

$$T_{\Delta rel} = \frac{T_{\Delta abs}}{15m} = 71\% \tag{3}$$

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.

$$\begin{aligned} T_{\Delta jahr} &= T_{\Delta abs} * N_{Workflow/Monat} * N_{Monat/Jahr} \\ &= T_{\Delta abs} * 10 * 12 = 1272m \\ &= 21,2h \end{aligned} \tag{4}$$

$$\begin{aligned} E_{jahr} &= T_{\Delta jahr} * Stundensatz_{Mitarbeiter} \\ &= T_{\Delta jahr} * \frac{30€}{h} \\ &= 636€ \end{aligned} \tag{5}$$

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.

$$\begin{aligned} K_{ges} &= T_{Projekt} * Stundensatz_{Projekt-Mitarbeiter} \\ &= 70h * \frac{16€}{h} \\ &= 1120€ \end{aligned} \tag{6}$$

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

Eine detaillierte, nach Phasen unterteilte Übersicht der Kosten findet sich im Anhang A.4 auf Seite V

3.2.3. Amortisationsdauer

$$T_{amortisation} = \frac{K_{ges}}{E_{jahr}/a} = \frac{1120€}{636€/a} = 1,76a \tag{7}$$

Zieht man die Ergebnisse aus Kapitel 3.2.1 und 3.2.2 heran und betrachtet diese relativ zueinander, wie in Formel 7 geschehen, lässt sich feststellen, dass sich das Projekt nach etwa 1,8 Jahren amortisiert hat.

3.3. Risikoanalyse

Im Folgenden wird eine Risikoanalyse und -bewertung durchgeführt. Eine detaillierte Übersicht der Risikobewertung findet sich im Anhang A.2 auf Seite II.
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 primär durch die Erfahrung und intensive fachliche Vorbereitung des Projektmitarbeiters 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 Auswirken 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 gemäßigtem 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 A.3 auf Seite IV.
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. Nutzwertanalyse

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

Architekturbereich	beste Alternative
Backend	JEE-Stack
Frontend	React

Tabelle 5: Ergebnis der Nutzwertanalyse

Die zugrunde liegenden Nutzwertanalysen befinden sich im Anhang A.5 auf Seite VI.

3.6. 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 neues 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 A.6 auf Seite VII Use-Case-Diagramme, welche die Anwendungsfälle detailliert darstellen.

3.7. Qualitätsanforderungen

Die Qualitätsanforderungen der Anwendung sind hinlänglich im 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

Eine detaillierte Übersicht der Qualitätsanforderungen findet sich im Anhang A.7 auf Seite IX.

3.8. 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 herunterzuladen. 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 A.8 auf Seite X.

3.9. 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 des Projektmitarbeiters 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 3.5 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 Applicationservers.

Datenbank MySQL
Als Datenbanksystem wird MySQL verwendet. Dies ist in erster Linie mit der bereits vorhandenen Erfahrung des Projektmitarbeiters 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 ausreichend leistungsfähigem Intel / AMD Prozessor
- ausreichend Arbeitsspeicher
- Debian Linux Betriebssystem
- MySQL Datenbankserver
- Tomcat Applicationserver
- 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 sich dagegen in Kapitel 3.5.

4.2.1. Serverarchitektur

Als Grundlage der Serverarchitektur wird das REST-Prinzip[4] 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 (CRUD[5]) von persistenten Daten. Sowohl von solchen, die den Application-Lifecycle überleben (Datenbankeinträge), als auch von flüchtigen, wie etwa Sessiondaten. 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 Jackson[6], 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-RS[7] bereitstellt.

4.2.2. Clientarchitektur

Clientseitig wird eine auf dem Flux[8] Konzept basierende Model-View-Architektur verwendet.

[4] `http://www.infoq.com/articles/designing-restful-http-apps-roth`, abgerufen am 07.10.2015
[5] Create, 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

Diese weist folgende Vorteile auf:

Trennung von Daten und Darstellung Durch die strikte Trennung von Daten (repräsentiert durch zustandslose Stores) und deren Darstellung (zustandsbehaftete Komponenten) werden Fehler in der Geschäfts- 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 (je 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 Design[9] entwickelt, um die Attraktivität und den Nutzwert der Anwendung für verschiedene Endgeräte wie z.B. Smartphones oder Tablets zu gewährleisten.

[9] `http://alistapart.com/article/responsive-web-design`, abgerufen am 07.10.2015

Visuelle Richtlinien Die Darstellung lehnt sich dabei zum einen an die Richtlinien von Material Design[10] an, um einen modernen Gesamteindruck zu erzielen. Zum anderen orientiert sich das Farbschema der Oberfläche am Corporate Design des Auftraggebers.

4.3.2. Prototyp

Eine bildhafte Darstellung eines Prototypen, welcher die hier aufgeführten visuellen Konzepte verdeutlichen soll, findet sich in Anhang A.9 auf Seite XII.

4.4. Datenmodell

Das der Anwendung zugrunde liegende Datenmodell wird in Form eines Entity-Relationship-Modells in Anhang A.10 auf Seite XIV dargestellt.

4.5. Geschäftslogik

Den Hauptprozess der Anwendung stellt das Erstellen von Handbüchern dar. Eine schematische Beschreibung von diesem findet sich in Anhang A.6 auf Seite VII in Form eines Use-Case-Diagramms.
Ein umfassenderer Überblick über den logischen Aufbau der Anwendung findet sich in Anhang A.11 auf Seite XV.

4.6. Maßnahmen zur Qualitätssicherung / Testkonzept

Im Folgenden sind die qualitätssichernden Maßnahmen des Projekts beschrieben.

Dokumentation

Um die Wartbarkeit der Anwendung zu erleichterten und langfristig zu gewährleisten, werden die Quelltexte dokumentiert. Dies geschieht nach vorherrschenden Standards, speziell JavaDoc[11] bzw. JsDoc[12].
Des Weiteren werden Informationen bezüglich der Entwicklungsumgebung und -toolchain in Hilfedateien 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 entstandene Fehler frühzeitig zu erkennen.

[10] `https://www.google.com/design/spec/material-design`, abgerufen am 07.10.2015
[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

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.

4.7. Pflichtenheft

Ein Auszug des im Rahmen der Entwurfsphase erstellten Pflichtenhefts findet sich in Anhang A.12 auf Seite XVI.

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	4h	+ 2h
Gesamt	**18h**	**17h**	**- 1h**

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 Forstschritt der Entwicklung besser verifizieren zu können und die logischen Abhängigkeiten der Teilmodule zu verdeutlichen.
Der Iterationsplan findet sich in Anhang A.13 auf Seite XVII.

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 Workbench[13] 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.

[13] `https://www.mysql.de/products/workbench/`, abgerufen am 27.10.2015

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.

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 Maven[14] 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 Dantebankschemas 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 in Anhang A.14 auf Seite XVIII.

Testdatengenerierung Zur programmatischen Erstellung von Testdatensätzen wird eine Klasse `DataBaseSeeder` implementiert. Diese stellt die Funktionalität neue Testdatensätze zu erstellen bereit, 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 jede 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`

[14] `https://maven.apache.org/`, abgerufen am 27.10.2015

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 in Anhang A.14 auf Seite XX.

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. Dies wird mit Hilfe der CSS3 Spezifikation FlexBox[15] 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 Queries[16] erreicht.

Design

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

Wie in Kapitel 4.6 beschrieben, werden zur Sicherung der Softwarequalität diverse automatisierte Tests entwickelt. Im Folgenden wird näher auf die Implementierung dieser eingegangen.

[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

Clientseitige Tests

Die Funktionalität der clientseitigen Geschäftslogik wird mit Hilfe einer umfassenden Testsuite sichergestellt. Als Testframework kommt Jest[17] zum Einsatz.
Die Tests sind so konzipiert, dass sie die natürliche Interaktion des Benutzers mit der graphischen Schnittstelle möglichst präzise nachbilden, um Fehler in der Oberfächenimplementierung frühzeitig aufzudecken.
Da die Ausführung der Tests und die Ausgabe deren Ergebnisse über einen Browser erfolgt, wird von der expliziten Führung eines Testprotokolls abgesehen.

5.6. Zwischenstand

Teilphase	**Geplant**	**Tatsächlich**	**Differenz**
Erstellen der Datenbanktabellen	2h	1h	- 1h
Entwicklung des Backends	7h	10h	+ 3h
Entwicklung des Frontends	9h	15h	+ 6h
Dokumentation des Quellcodes	2h	2h	
Testing	2h	2h	
Soll-Ist-Vergleich	2h	1h	- 1h
Puffer für Korrekturen	2h	-	- 2h
Gesamt	**26h**	**31h**	**+ 5h**

Tabelle 9: Zwischenstand nach der Implementierungsphase

6. Abnahmephase

Nach erfolgreichem Abschluss der Implementierungsphase wurde die Anwendung durch den Auftraggeber, hier die Abteilung KIS der GNS mbH, abschließend abgenommen.
Zu diesem Zwecke wurden alle im Lastenheft aufgeführten Anforderungen an die Anwendung vorgeführt, um die Korrektheit dieser zu bestätigen.

6.1. Zwischenstand

Teilphase	**Geplant**	**Tatsächlich**	**Differenz**
Abnahme der Anwendung	1h	1h	
Gesamt	**1h**	**1h**	

Tabelle 10: Zwischenstand nach der Abnahmephase

[17] `https://facebook.github.io/jest/`, abgerufen am 28.10.2015

7. Einführungsphase

Im Rahmen der Einführung wurde vom Projektleiter eine Anwenderschulung abgehalten, um die Vorgehensweise des überarbeiteten Workflows "Erstellen eines Handbuchs aus einem MediaWiki" anhand der Anwendung zu visualisieren.

7.1. Zwischenstand

Teilphase	Geplant	Tatsächlich	Differenz
Anwenderschulung	2h	1h	- 1h
Gesamt	**2h**	**1h**	**- 1h**

Tabelle 11: Zwischenstand nach der Einführungsphase

8. Dokumentation

8.1. Projektdokumentation

Die Projektdokumentation beschreibt alle Phasen des Projekts, die im Rahmen von diesem durchlaufen wurden.

8.2. Benutzerdokumentation

Eine dedizierte Benutzerdokumentation wurde im Rahmen des Projekts nicht erstellt. Dies ist zum einen der geringen Komplexität der Anwendung bzw. der intuitiven Gestaltung dieser geschuldet. Zum anderen wurde von der Erstellung eines Benutzerhandbuchs abgesehen, da alle Punkte, die dieses beinhalten sollte, bereits in der Anwenderschulung (siehe Kapitel 7) abgehandelt wurden.

8.3. Entwicklerdokumentation

Ein dediziertes Entwicklerhandbuch wurde im Rahmen des Projekts nicht erstellt. Allerdings wurden alle Quelltexte ausreichend dokumentiert (siehe Kapitel 4.6) und einzelne Hilfedateien erstellt, in denen Details zur Entwicklung und verwendeten Tools festgehalten wurden.
Die Anwendungsdokumentation entspricht damit in Form und Umfang dem Standard des Auftraggebers.

8.4. Zwischenstand

Teilphase	Geplant	Tatsächlich	Differenz
Projektdokumentation	12h	11h	- 1h
Gesamt	**12h**	**11h**	**- 1h**

Tabelle 12: Zwischenstand nach der Dokumentationsphase

9. Fazit

9.1. Soll- / Ist-Vergleich

Rückblickend betrachtet lässt sich feststellen, dass die im Rahmen des Projekts entwickelte Anwendung alle Anforderungen des Lasten- / bzw. Pflichtenhefts vollständig erfüllt.
Weiter lässt sich festhalten, dass der in Kapitel 2.1 erläuterte Projektplan eingehalten werden konnte. Die geplanten und tatsächlichen Zeitaufwände der einzelnen Projektphasen sind in Tabelle 13 dargestellt. Dieser lässt sich entnehmen, dass einzelne Abweichungen der geplanten Zeiten sich gegenseitig kompensierten und so eine Einhaltung des gesetzten Projektzeitraums erzielt wurde.

Phase	Geplant	Tatsächlich	Differenz
Anforderungsanalyse	7h	9h	+ 2h
Entwurfsphase	18h	17h	- 1h
Implementierungsphase	26h	31h	+ 5h
Abnahmephase	1h	1h	
Einführungsphase	2h	1h	- 1h
Dokumentationsphase	12h	11h	- 1h
Puffer	4h	-	- 4h
Gesamt	**70h**	**70h**	**0h**

Tabelle 13: Soll-Ist Endstand

9.2. Lessons-Learned

Im Zuge des Projekts konnte der Projektleiter umfassende Erfahrung bezüglich der Planung und Durchführung eines Softwareprojekts sammeln. Dabei wurde erkennbar, wie wichtig eine detaillierte Planung aller Phasen eines Projekts ist, um dieses erfolgreich zum Abschluss zu bringen.
Vor allem Entscheidungen bezüglich Implementierung- und Architekturdetails wollen wohl durchdacht sein. Um Fehlentscheidungen dieser Bereiche aufzudecken erwies sich auch eine regelmäßige Kontrolle des Ist-Stands als äußerst hilfreich.

9.3. Ausblick

Obwohl die im Rahmen des Projekts entwickelte Anwendung alle im Lastenheft beschriebenen Anforderungen erfüllt, können in Zukunft neue Anforderungen auftreten. Beispielhaft sei hier das sogenannte ”Single-Sign-On” genannt, welches zeitnah für alle Anwendungen der GNS mbH umgesetzt werden soll.
Aufgrund des strukturierten und modularen Aufbaus der Anwendung sollten solche zukünftigen Erweiterungen allerdings keinerlei Probleme darstellen.

A. Anhang

A.1. Projektphasen

Projekphase	**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 Architekturgrundlagen	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 14: Projektphasen detailliert

A.2. 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	Zusätzlicher 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 Projektabschlusses
Auslassung relevanter Phasen	mangelnde Erfahrung	nachträgliche Planung	gering	Verzögerung des Projektabschlusses
Risiken der Analyse und Konzeption				
Fehlerhafte Ist-Analyse	ungenaue 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	ungenaue Spezifikation	Neukonzeption	gering	iterativer Rücksprung im Entwicklungsprozess

Risiko	**Ursache**	**Maßnahme**	**Eintrittswahr. nach der Maßnahme**	**Auswirkung nach der Maßnahme**
Realisierungrisiken				
Unerwartete Entwicklungs-probleme	nicht bestimm-bar	teilweise Neu-Konzeption	mittel	verlängerte Entwicklungs-zeit
Betreuungs- und Wartungsrisiken				
Breaking-Changes	MediaWiki Update	Anpassung der Geschäftslogik	hoch	keine

Tabelle 15: Risikoanalyse detailliert

A.3. "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 händischer Anpassung	⊖⊖	⊖⊖
Kompatibilitätsprobleme mit MediaWiki		⊖
Erstellung der Anwendung durch Dienstleister		
spezialisierte Anwendung	⊕⊕	⊖ ⊖ ⊖
Wartung durch Dienstleister nötig		⊖⊖
Risikoauslagerung		⊕⊕
Eigenständige Erstellung der Anwendung		
spezialisierte Anwendung	⊕⊕	⊖⊖
Wartung durch eigene Mitarbeiter	⊕	⊕
Tragen von Risiken		⊖

Tabelle 16: Detaillierte "Make or Buy"-Bewertung

A.4. Kostenübersicht nach Projektphasen

Grundlage der folgenden Kostenübersicht sind die in Anhang A.1 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 17: Kosten unterteilt nach Projektphasen

A.5. Nutzwertanalyse

A.5.1. Backend-Architektur

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

Eigenschaft	Gewichtung	JEE-Stack[18]	GNS-Framework[19]	SrpingMVC[20]
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 18: Detaillierte Nutzwertanalyse bezüglich der Backend-Architektur

A.5.2. Frontend-Architektur

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

Eigenschaft	Gewichtung	JSF[21]	Angular[22]	kein Framework	React[23]
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**	**23**
Nutzwert		**3,32**	**3,79**	**2,11**	**4,26**

Tabelle 19: Detaillierte Nutzwertanalyse bezüglich der Frontend-Architektur

[18] `http://www.oracle.com/technetwork/java/javaee/overview/index.html`, abgerufen am 08.11.2015

[19] ein betriebsinternes Framework zur Entwicklung von Webanwendungen

[20] `https://spring.io`, abgerufen am 08.11.2015

[21] `http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html`, abgerufen am 08.11.2015

A.6. Use-Case Diagramme

A.6.1. Wiki konvertieren

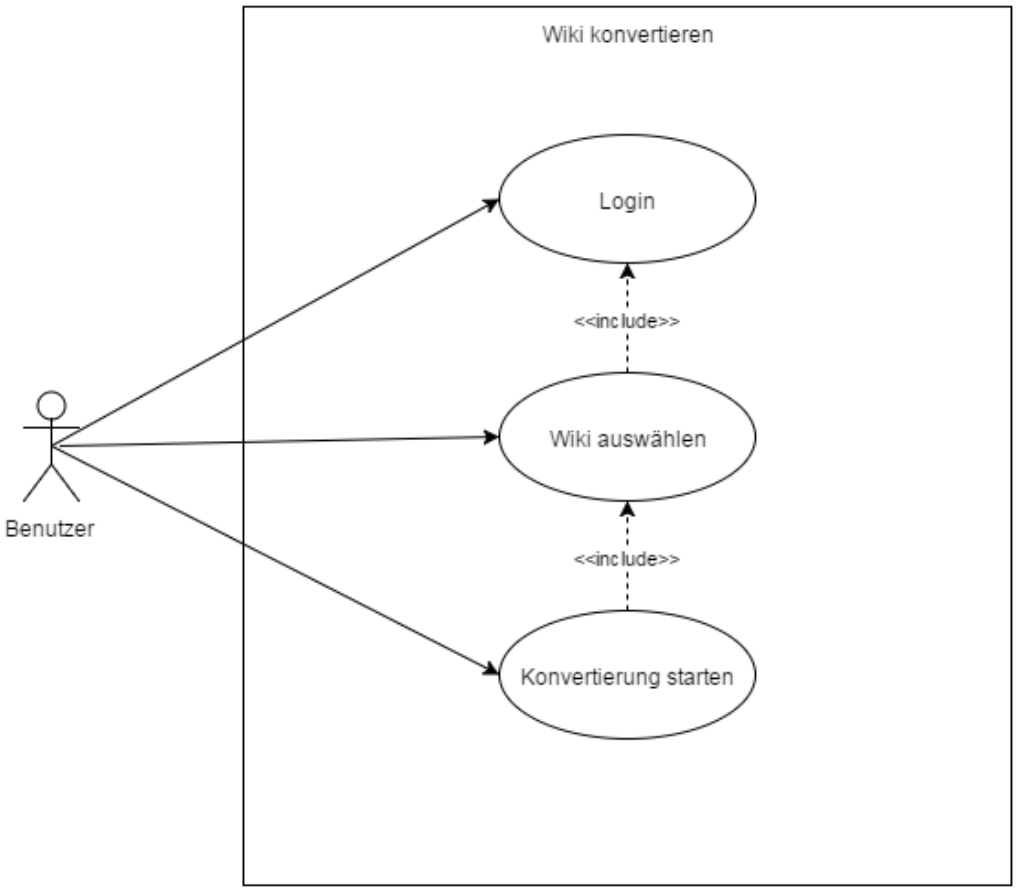

Abbildung 1: Use-Case "Wiki konvertieren"

[22] `https://angularjs.org`, abgerufen am 08.11.2015
[23] `https://facebook.github.io/react/`, abgerufen am 08.11.2015

A.6.2. Wiki bearbeiten

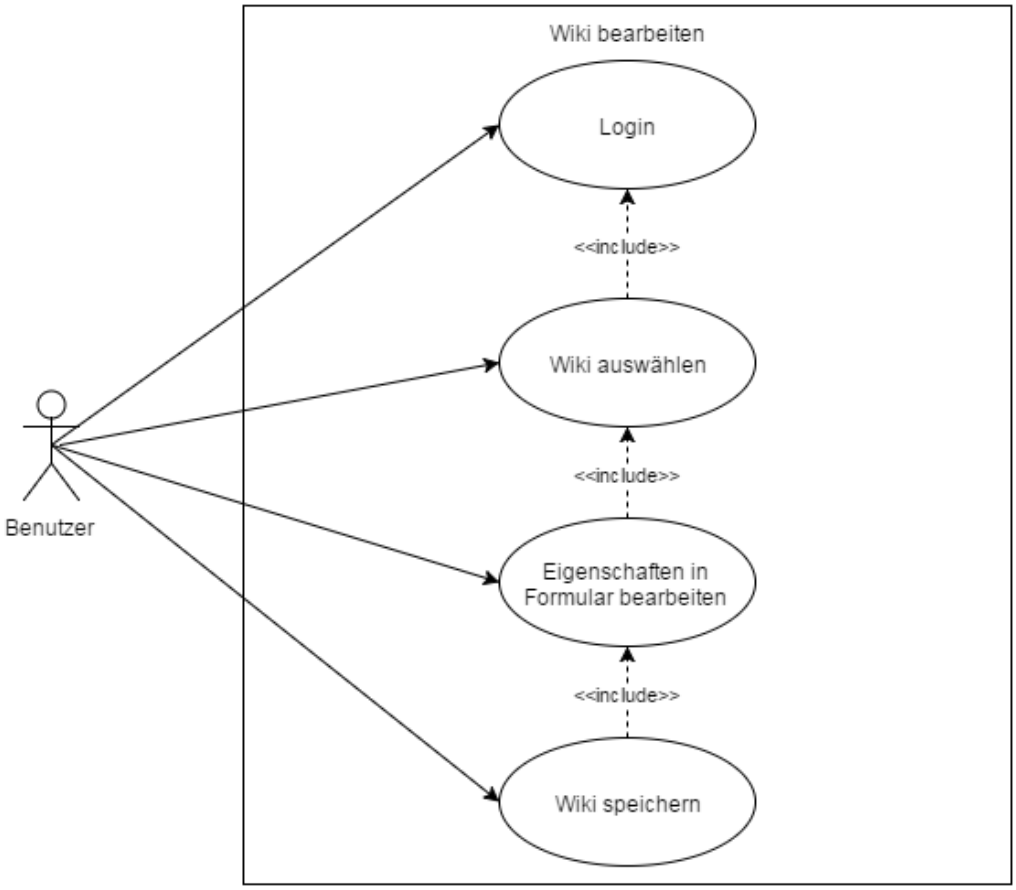

Abbildung 2: Use-Case "Wiki bearbeiten"

A.7. Qualitätsanforderungen (Auszug 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 20: Qualitätsanforderungen an die Anwendung detailliert

A.8. Lastenheft (Auszug)

Zielbestimmung

Musskriterien

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

Wunschkriterien

- Handbuchversionen
  - Handbuchhistorie eines Wikis einsehen
  - Handbuch aus Historie löschen
  - Älteres Handbuch herunterladen
- 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.

A.9. Oberflächenprototyp

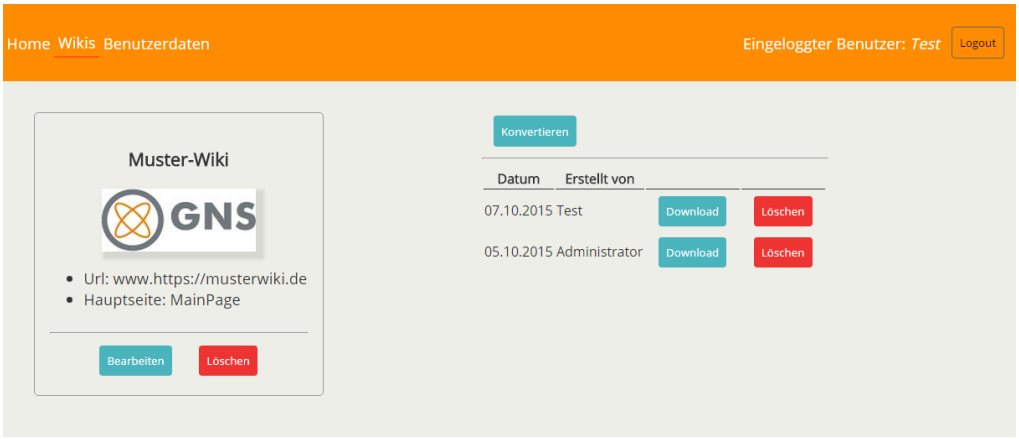

Abbildung 3: Prototyp - Layout für Desktop-Browser

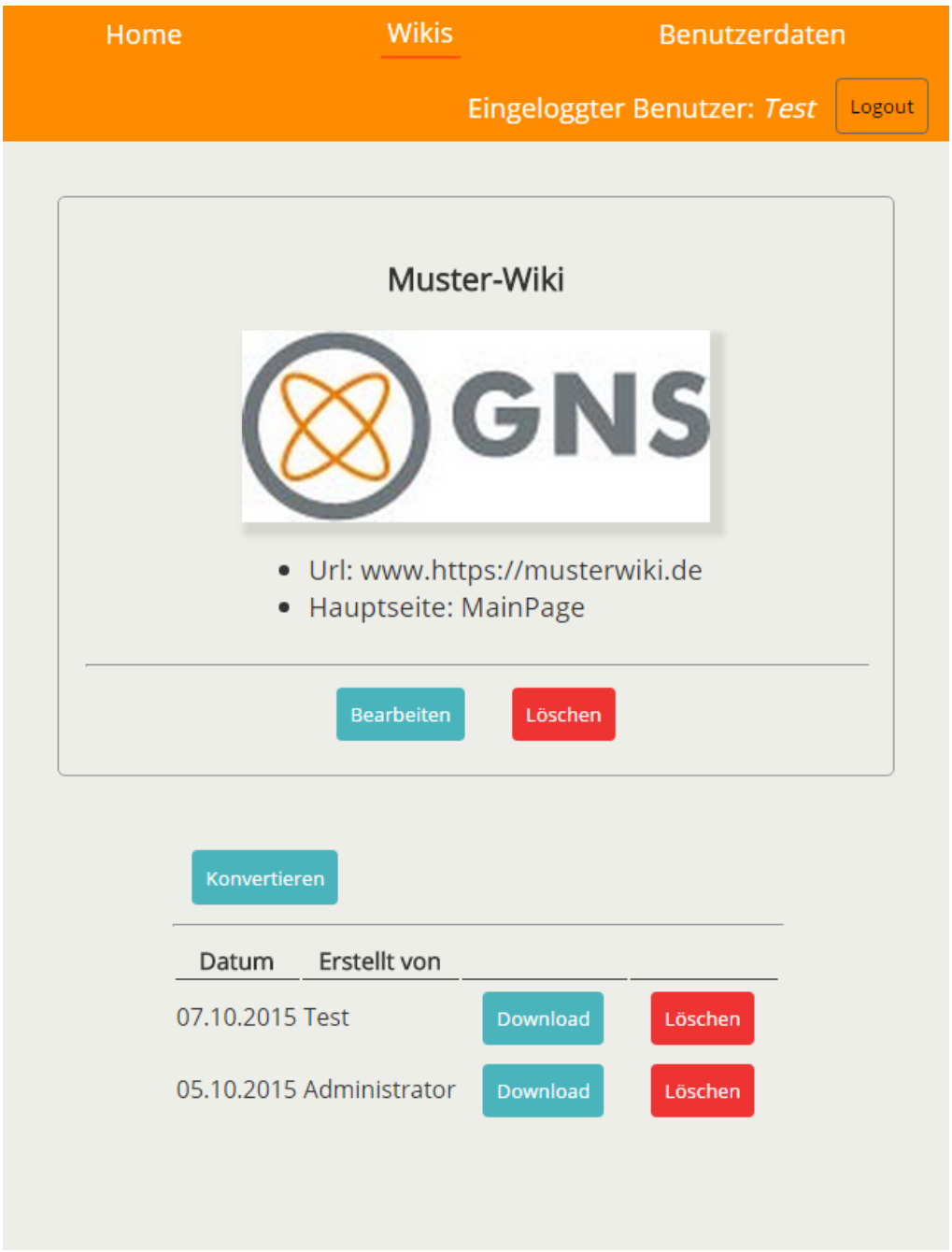

Abbildung 4: Prototyp - Layout für Mobile-Browser

A.10. Entity-Relationship-Modell

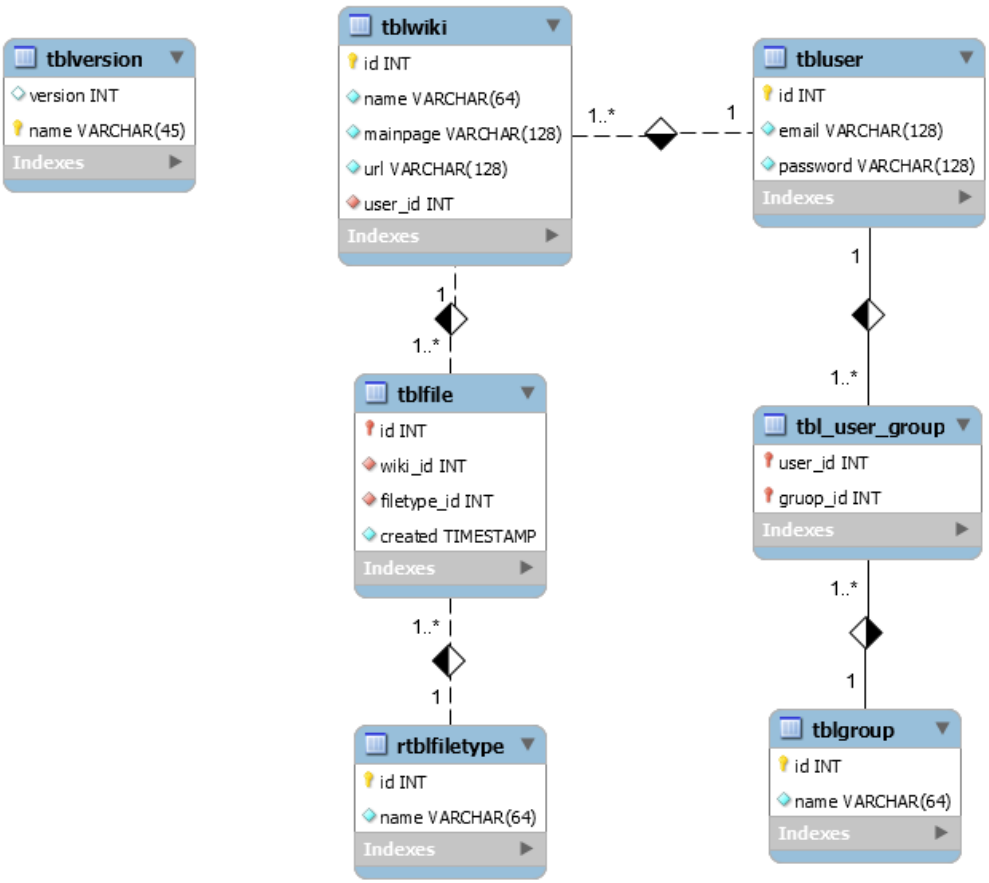

Abbildung 5: ER-Modell

A.11. Komponentendiagramm

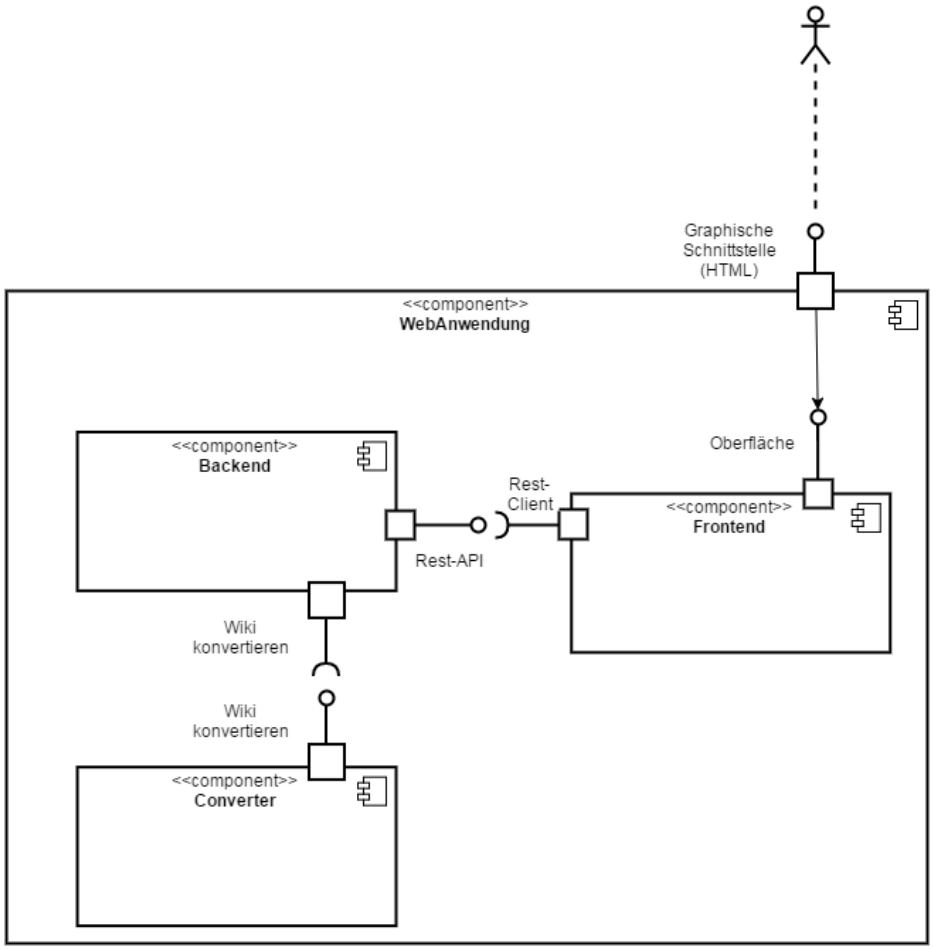

Abbildung 6: Komponentendiagramm

A.12. 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

A.13. 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

A.14. Listings

```
1  public class DatabaseUpdater {
2
3    private static Logger logger = Logger.getLogger(DatabaseUpdater.class
       );
4
5    private String scriptPackage;
6    private String applicationName;
7
8    public DatabaseUpdater(String scriptPackage, String applicationName)
       {
9      this.scriptPackage = scriptPackage;
10     this.applicationName = applicationName;
11   }
12
13   public void update(EntityManager em) {
14     EntityTransaction transaction = em.getTransaction();
15     transaction.begin();
16
17     try {
18       this.checkIfVersionTableExists(em);
19       for(File script : this.getScripts()) {
20         this.executeSql(em, script);
21       }
22     } catch (Exception e) {
23       logger.error(e.getMessage(), e);
24       transaction.rollback();
25       return;
26     }
27
28     transaction.commit();
29     em.close();
30   }
31
32   protected void checkIfVersionTableExists(EntityManager em) throws
       SQLException {
33
34     java.sql.Connection conn = em.unwrap(java.sql.Connection.class);
35     conn.createStatement().executeUpdate(Version.getCreateScript());
36   }
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      if(resources.hasMoreElements()) {
66        URL p = resources.nextElement();
67        File f_package = new File(p.toURI());
68        File[] files = f_package.listFiles();
69
70        Arrays.sort(files, new Comparator<File>() {
71
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")
         int id) {
8      EntityManager em = createEntityManager();
9      try {
10       beforeRetrieve(request);
11       beforeGetOne(request, id);
12
13       T t = em.find(clazz, id);
14
15       afterGetOne(request, t);
16
17       return t;
18     } catch(WebApplicationException e) {
19       throw e;
20     } catch (Exception e) {
21       log(e);
22       throw new GeneralException(e);
23     } finally {
24       em.close();
25     }
26   }
27
28   (...)
```

Listing 2: BaseApi.java