HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Collaborative Decision Management and Architectural Refactoring (CDAR) Tool

Bachelorarbeit

Technischer Bericht

Abteilung Informatik

Hochschule für Technik Rapperswil

Frühjahressemester 2014

Autoren:	Marcel Tinner, Daniel Zigerlig
Betreuer:	Prof. Dr. Olaf Zimmermann
Experte:	Dr. Gerald Reif, [ipt]
Gegenleser:	Prof. Oliver Augenstein

12. Juni 2014

Abstract

Architekturentscheidungen und ihre Begründungen werden im Rahmen von Softwareprojekten meist als strukturierte Texte erfasst; in der Literatur werden zahlreiche Templates für diese Dokumentationsaufgabe vorgeschlagen. Ein rein textbasierter Ansatz hat aber zahlreiche Nachteile, z.B. eingeschränkte Teamfähigkeit und fehlende Skalierbarkeit. Die Bachelorarbeit untersucht, ob diese Nachteile überwunden werden können, indem eine existierende Wiki-Engine mit einem neu zu erstellenden entscheidungsbaumorientierten Rich Client im Browser kombiniert wird.

Zu den Herausforderungen dieser Bachelorarbeit gehört die Erstellung eines Gesamtkonzeptes für User Interface und Software-Architektur, die Implementierung eines Prototypen, die Integration eines nativen Webclients und der Wiki-Engine MediaWiki sowie ein modularer und konfigurierbarer Lösungsansatz als Grundlage für zukünftige Erweiterungen. Neben einer benutzerfreundlichen Prototypen-Implementierung ist auch eine anforderungsgerechte, erweiterbare Tool-Architektur gefordert.

Das Ergebnis dieser Bachelorarbeit ist ein funktionsfähiger Prototyp in Form einer Java- und JavaScript-basierten Webapplikation mit einer HTTP- Schnittstelle, die sich an den REST-Designprinzipien orientiert. Die Software- Architektur des Prototypen ermöglicht die einfache Anbindung weiterer fachlicher Services an die Server-Komponente. Diese leitet die textuellen Entscheidungsbeschreibungen (Dokumentation) über einen API-Aufruf an die Wiki-Engine MediaWiki weiter. Dies führt zu zwei wesentlichen Vorteilen. Erstens müssen die Hauptfunktionen von Wikis, z.B. die Pflege einer Änderungshistorie, nicht nachimplementiert werden. Zweitens lässt sich in Zukunft die Wiki-Engine einfach austauschen oder auch erweitern, um zum Beispiel mit bereits vorhandenen Wikis weitere Informationen in die Applikation zu integrieren.

Eigenständigkeitserklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selbst und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Regeln zitiert haben,
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Ort, Datum:

Rapperswil, 12.6.2014

Marcel Tinner

Daniel Zigerlig

Danksagungen

Wir danken folgenden Personen für Ihre Unterstützung während der Bachelorarbeit:

- Prof. Dr. Olaf Zimmermann für die Betreuung unserer Bachelorarbeit
- Lukas Wegmann für hilfreiche Vorschläge nach durchgeführtem Code Review
- Jonas Furrer für interessante Inputs bezogen auf die HTTP API
- Raphael Faes für das Korrekturlesen unseres Berichts
- Unseren Partnerinnen für Geduld, Rat und motivierende Worte

Inhaltsverzeichnis

1 Management Summary 11
1.1 Ausgangslage 11
1.2 Zielsetzung 11
1.3 Vorgehen 12
1.4 Resultat 13
1.5 Ausblick 14

2 Einführung 15
2.1 Einleitung 15
2.2 Beispielszenario 15
2.2.1 Ist-Zustand Bau Architekt 15
2.2.2 Soll-Zustand Architekt 16
2.3 Beispiel eines Wissenseintrages 18
2.3.1 AD about Integration Style (IBM UMF Template for Decision Log) 18
2.3.2 Y-Statements in Telco Case Study (EIP Pattern Selection) 18

3 Analyse der Anforderungen 19
3.1 Übersicht über die zentralen Konzepte der Domäne 19
3.2 Allgemeine Beschreibung des Software-Werkzeuges 20
3.3 Produkt Funktion 20
3.3.1 Funktionen des Wissensproduzentes 21
3.3.2 Funktionen des Wissenskonsumenten 22
3.4 Benutzer Charakteristik 22
3.5 Annahmen und Abhängigkeiten 22
3.6 Funktionale Anforderungen 23
3.6.1 Aktoren und Stakeholder 23
3.6.2 Use Case Diagramm 24
3.6.3 Use Case Beschreibungen 25
3.7 Weitere Anforderungen 28
3.8 Anforderungen an die Testumgebung 29

4 Domainmodell der CDAR Applikation 30
4.1 Übersicht des Domainmodells 30
4.2 Detailbeschreibung des Domainmodells 31
4.2.1 Wissensbaum und Entscheidungsprojekt 31
4.2.2 Verzeichnis 31
4.2.3 Wissensnode 31
4.2.4 Wissenssubnode 32
4.2.5 Wissenslink 32
4.2.6 Beziehungen der Klassen 33
4.2.7 Beispieldaten 33

5 Architekturdesign 34
5.1 Übersicht des Systems 34
5.1.1 CDAR Applikation 34
5.1.2 Wiki-Software 34
5.1.3 Datenbank 34
5.2 Externes Design der Applikation 35
5.2.1 Vorgehen 35
5.2.2 Mockups 35

6 Evaluation der Technologien zur Umsetzung der Architektur 40
6.1 Servertechnologien 40
6.1.1 Tomcat Anwendungsserver 40

6.1.2 Jersey ... 40
6.1.3 Datenbanktechnologie ... 41
6.2 Clienttechnologien ... 41
6.2.1 JavaScript-Framework ... 41
6.2.2 JavaScript-Plugin Entscheidungsverwaltung ... 42
6.2.3 JavaScript-Plugin Entscheidungsführungsgraphen ... 42
6.3 Wiki-Software ... 43
6.4 Übersicht der eingesetzten Libraries und Plugins ... 44
6.4.1 CDAR-Browseranwendung (Client) ... 44
6.4.2 CDAR-Serveranwendung ... 44
7 Implementierung der Architektur ... 45
7.1 Datenhaltung ... 45
7.1.1 Client ... 45
7.1.2 Server ... 45
7.1.3 MediaWiki ... 45
7.2 Parallele Bearbeitung ... 46
7.3 Kommunikation ... 47
7.3.1 Beispielkommunikation Node erstellen ... 48
7.3.2 Beispielkommunikation Node kopieren ... 49
7.3.3 Beispielkommunkation Node löschen ... 50
7.3.4 Beispielkommunkation Node Drag & Drop ... 51
7.3.5 Beispielkommunikation Abfrage Wiki-Seite ... 52
7.4 Sicherheit und Session Handling ... 53
7.4.1 MediaWiki Single-Sign-on ... 54
7.5 Baum- und Graphenstrukturen ... 56
7.5.1 Entscheidungsverwaltung ... 56
7.5.2 Entscheidungsführungsgraphen ... 57
7.6 Aufbau der Serverapplikation ... 58
7.7 Aufbau der Clientapplikation ... 59
7.7.1 Routen in Zusammenhang mit Views und Controllers ... 60
7.7.2 Beschreibungen der Services ... 62
7.8 Entity Relationship Model der Datenbank ... 64
7.8.1 Beispielinhalt der Datenbank ... 65
8 Beschreibung der RESTful Server WebAPI ... 73
8.1 Übersicht der Schnittstelle ... 74
8.1.1 User Ressourcen ... 74
8.1.2 Property File Ressourcen ... 74
8.1.3 Allgemeine Ressourcen ... 75
8.1.4 Wissensproduzent Ressourcen ... 77
8.1.5 Wissenskonsument Ressourcen ... 78
8.2 HTTP Statuscodes ... 78
8.3 Detailbeschreibung der Ressourcen ... 79
8.3.1 Property File Konstanten ... 79
8.3.2 Benutzer ... 79
8.3.3 Baum ... 82
8.3.4 Export/Import ... 83
8.3.5 Verzeichnisse ... 85
8.3.6 Nodes ... 86
8.3.7 Subnodes ... 90
8.3.8 Verbindungen ... 94
8.3.9 Templates ... 96
8.3.10 Kommentare ... 97
9 Schlussfolgerung ... 99

9.1 Zusammenfassung 99
9.2 Ausblick 100

A Qualitätssicherungsmassnahmen I
A.1 Kennzahlen I
A.2 Usability Tests III

B Verzeichnis der Abbildungen und Tabellen VI
B.1 Abbildungsverzeichnis VI
B.2 Tabellenverzeichnis VII

C Glossar IX

D Literatur XI

E Projektplanung XIII
E.1 Iterationsplanung XIII
E.2 Arbeitsplanung XIV
E.3 Meilensteine XVII
E.4 Meetings XVIII
E.5 Tools und Unterstützungssoftware XVIII

F Auswertung Zeitrapportierung XIX
F.1 Projektaufwand XIX
F.2 Zeitaufwand pro Iteration XX

G Aufgabenstellung XXI
G.1 Scan Aufgabenstellung XXI

H Benutzerhandbuch XXVI
H.1 Installationsanleitung XXVI
H.2 Betriebsanleitung XXVIII

1 Management Summary

1.1 Ausgangslage

Das Dokumentieren von Architektur-Entscheidungen im Rahmen eines Softwareprojektes wird meist in Form von Textdokumenten realisiert. Dies führt zu zahlreichen Nachteilen wie beispielsweise die fehlende Skalierung als auch eine eingeschränkte Teamfähigkeit.
Seit 2004 ist das Dokumentieren von Architektur-Entscheidungen ein aktives Forschungsthema. Neben dem Nutzen von Wiki-Engines existieren verschiedene Forschungsprototypen, welche die Wiederverwendbarkeit von Dokumentationen unterstützen und zusätzlich eine Benutzerführung ermöglichen sollen. Zum Zeitpunkt dieser Bachelorarbeit hat sich keines dieser Tools auf dem Markt etabliert [Nel].

1.2 Zielsetzung

Die vorliegende Bachelorarbeit hat das Ziel, eine neue technische Richtung des in der Ausgangslage beschriebenen Forschungstrends zu untersuchen. Durch die Nutzung einer Wiki-Engine in Kombination mit einem neu zu konzipierendem Rich-Client im Browser werden vorhandene Werkzeuge eingesetzt und kombiniert. Neben einer Prototypen-Implementation ist auch eine anforderungsgerechte, erweiterbare Software-Architektur gefordert. Nachfolgende Herausforderungen beschreiben die Hauptziele dieser Bachelorarbeit:

- Gesamtkonzept und Implementierung für User-Interface und Software-Architektur
- Kombination eigener Webclient und MediaWiki
- Grundstein schaffen für weitere Softwarekomponenten (Erweiterbarkeit)

1.3 Vorgehen

Um die Problembeschreibung zu lösen, haben wir ein iteratives und agiles Vorgehensmodell ähnlich zu „Rational Unified Process (RUP)“ gewählt.
Anfangs haben wir die Problemstellung analysiert. Dazu erstellten wir ein Domänenmodell basierend auf Literatur sowie Vorarbeiten des Institutsprojekts[1]. Mögliche Risiken wurden zu Beginn der Bachelorarbeit mittels Mini-Prototypen und Mockups gemildert. Anhand der nicht-funktionalen Anforderungen erstellten wir eine Framework-Auswahl, welche nach einer vollständigen Domainanalyse schliesslich zu einer Software-Architektur und Implementierung führte.
Die Anforderungen an die Software entwickelten sich während der Arbeit iterativ und inkrementell aus Vorgaben des Betreuers und Erfahrungen, welche wir während der Arbeit mit unserer Software selbst sammeln konnten.

Zur Förderung der Wiederverwendbarkeit und Benutzerführung entschied sich das Projektteam frühzeitig, zwei Benutzerrollen einzuführen. Den Wissensproduzenten und -konsumenten. Ersterer trägt seine Entscheidungen und deren Folgerungen in die Applikation ein. Nachfolgend arbeitet ein Wissenskonsument mit den bereitgestellten Daten und kann sich potentielle Architekturentscheidungen anzeigen lassen [Ifs] [Ola].

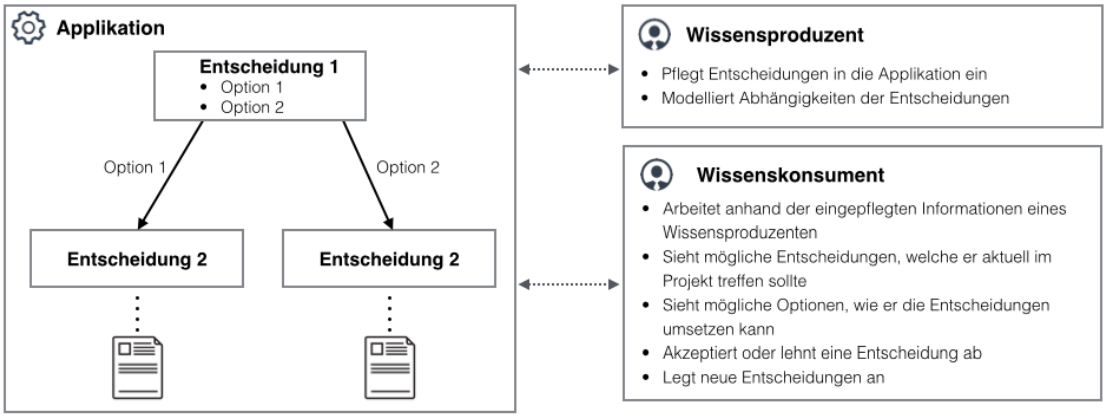


Abbildung 1: Rollenaufteilung in der Applikation

[1] http://www.ifs.hsr.ch/Architectural-Refactoring-for.12044.0.html?&L=4

1.4 Resultat

Das Ergebnis dieser Bachelorarbeit ist ein funktionsfähiger Prototyp in Form einer Webapplikation sowie einer Server-Komponente mit einer RESTful WebAPI. Die etablierte Software-Architektur ermöglicht das einfache Anbinden weiterer Services an den erstellten Server.
In der Webanwendung kann der Benutzer als Wissensproduzent oder –konsument arbeiten. Als Wissensproduzent kann er mögliche Entscheide in die Applikation aufnehmen und dokumentieren. Der Server leitet alle Entscheidungsbeschreibungen beziehungsweise Dokumentationen an MediaWiki weiter. Diese kann sich als eigenständige Instanz auch ausserhalb des Applikationsservers befinden und lässt sich konfigurieren. Somit ist eine Betrachtung und Bearbeitung der erstellten Wissenseinträge auch ausserhalb der Applikation möglich. Dies führt zu zwei wesentlichen Vorteilen. Die Hauptfunktionen von Wikis, beispielsweise die Änderungshistorie, stellt die Software zur Verfügung und muss somit nicht nachimplementiert werden. Zweitens lässt sich in Zukunft die Wiki-Software einfach austauschen oder auch erweitern, um zum Beispiel mit bereits vorhandenen Wikis weitere Informationen in die Applikation zu integrieren
Die eingetragenen Informationen lassen sich über einen Entscheidungsführungsgraphen miteinander verknüpfen, um kausale und oder temporäre Abhängigkeiten zwischen Entscheidungen zu spezifizieren.

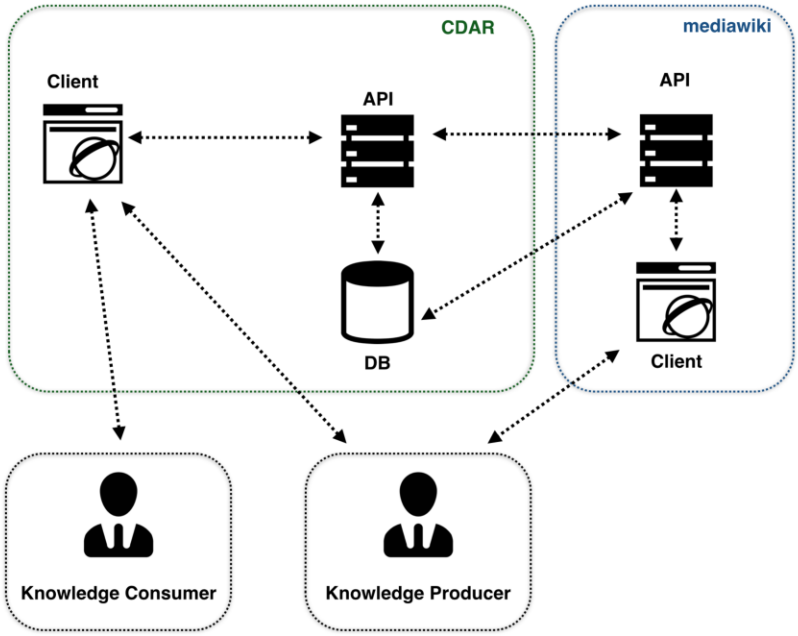


Abbildung 2: Architekturübersicht

Nach dem erfolgreichen Einpflegen von Daten kann der Benutzer in der Rolle als Wissenskonsumenten neue Projekte aufgrund bereits vorhandener Informationen erstellen. In dieser Ansicht ermöglicht das Tool, die Entscheidungen im jeweiligen Projekt zu werten und zu entscheiden. Somit kann ein Benutzer mit Hilfe der vorgetragenen Daten durch sein jeweiliges Projekt geführt werden. Er sieht seine Auswahlmöglichkeiten und kann abwägen, was für seine momentane Situation am sinnvollsten ist.
Die eingepflegten Daten können mittels einer Benutzerverwaltung anderen Benutzern der Software freigeschaltet werden. Die Applikation bietet darüber hinaus eine Export-, Import- sowie Reporting-Funktionalität an, um das Teilen der Daten über verschiedene Server und das Ausdrucken der eingepflegten Informationen zu ermöglichen.

1.5 Ausblick

Eine mögliche zukünftige Erweiterung dieser Applikation wäre beispielsweise das Anbinden einer Arbeitsplanungssoftware. In diese könnten die jeweiligen Entscheidungen nach dem Entscheiden direkt als Arbeitspakete exportiert werden. Diese Aufgabenstellung ist bereits definiert und wird im Rahmen einer weiteren Bachelorarbeit im Herbstsemester 2014 ebenfalls an der HSR realisiert.

2 Einführung

2.1 Einleitung

Diese Arbeit hat gemäss der Aufgabenstellung (siehe G Aufgabenstellung) zum Ziel, eine webbasierte Wissensmanagement-Entscheidungsführungsapplikation zu entwickeln. Aktuelle Technologien im Webbereich sollen als Grundlage für die Applikation dienen.

2.2 Beispielszenario

Grundsätzlich lässt sich die Problembeschreibung, welche diese Bachelorarbeit behandelt, auf sämtliche Themengebiete anwenden. Nachfolgendes Beispiel zeigt zwecks Verständlichkeit anhand eines fiktiven Bau Architekten, wie die zu erstellende Applikation die Arbeit von wiederkehrenden Entscheidungen im Laufe eines Arbeitsprozesses verbessern soll.

2.2.1 Ist-Zustand Bau Architekt

Ein Architekturbüro hat je nach Kundenwunsch verschiedene Möglichkeiten, ein Projekt umzusetzen. Sämtliche vergangenen Projekte wurden sauber dokumentierten und in Form einer gedruckten Dokumentation im Archiv hinterlegt.
Sobald das Bau Architekturbüro einen Kundenauftrag erhält, welcher sich in den Anforderungen und Gegebenheiten sehr einem in früheren Jahren realisierten Projekt ähnelt, bietet sich den Mitarbeitern der folgende Ablauf.

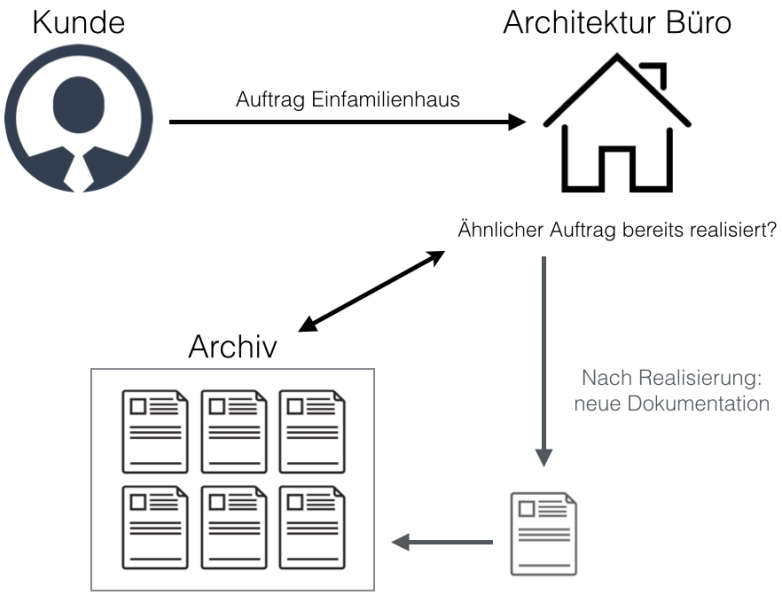


Abbildung 3: Ist-Zustand Bau Architekt

Die Mitarbeiter durchsuchen das Archiv nach ähnlichen Projekten. Sind diese gefunden und durchgelesen, erfolgt die Realisierung des aktuellen Kundenauftrages. Entscheidungen aus fürheren Projekten werden erneut gefällt und dokumentiert. Hier kann es aufgrund von potentiellen Neuentwicklungen zu anderen Entscheidungen als in früheren Jahren kommen. Diese werden in der neuen Dokumentation niedergeschrieben und begründet.
Dies führt zu sehr viel redundanten Informationen die an mehreren Stellen, mit anderen Begründungen und Entscheidungen, geführt werden
Falls nun einige Zeit später ein weiteres Einfamilienhaus angefordert wird, hat ein Architekt die Auswahl von mehreren Dokumentationen. Er muss beide durchlesen, um die richtigen Entscheidungen beziehungsweise sämtliche Möglichkeiten zu finden.

2.2.2 Soll-Zustand Architekt

Mit Hilfe der CDAR Applikation soll der Ablauf eines Projektes beim fiktiven Bau Architekten in Zukunft wie nachfolgend beschrieben ablaufen.

2.2.2.1 Einpflegen der Informationen bei neuem Projekt

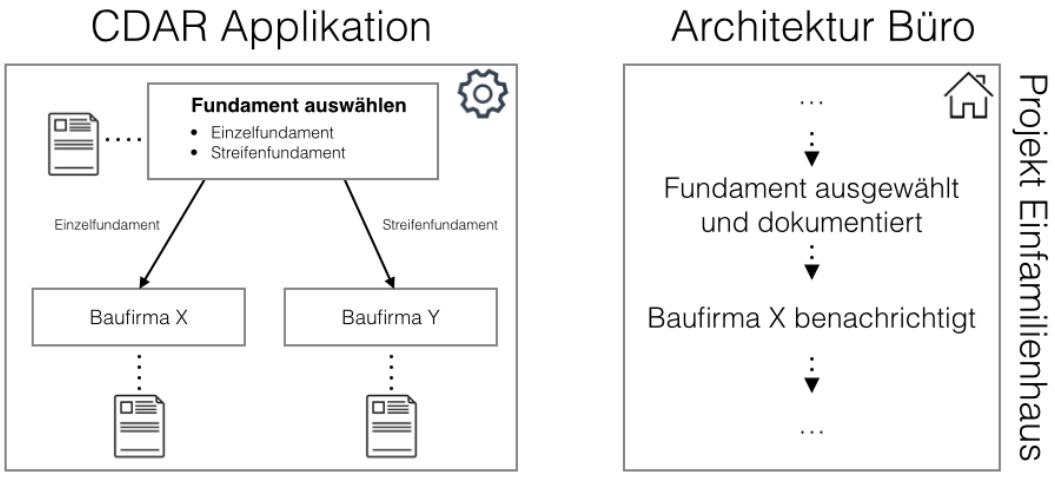


Abbildung 4: Projektablauf Einfamilienhaus

Sämtliche wichtige Entscheide im Laufe des Projektes werden durch den Wissens-produzenten in die CDAR Applikation eingepflegt. Dabei zeigen die Mitarbeiter auf, welche Möglichkeiten sie bei der Entscheidung hatten und was für Folgeentscheide diese mit sich gezogen haben. Als Beispiel dient die Auswahl eines Fundamentes. Entscheidet sich der Bau Architekt für ein Einzelfundament, kommt lediglich die Baufirma X zum Einsatz. Bei der Entscheidung für ein Streifenfundament kommt jedoch nur Baufirma Y in Frage. Sämtliche Entscheide sind schriftlich dokumentiert und hinterlegt.

2.2.2.2 Auf vorhandene Informationen zugreifen bei wiederkehrendem Projekt

Stösst das Architekturbüro auf einen weiteren Auftrag wie beispielsweise den des Einfamilienhauses, bietet die CDAR-Applikation die Möglichkeit, auf das bereits realisierte Projekt zuzugreifen. Die Mitarbeiter können das neue Projekt in der Rolle des Wissenskonsumenten in Angriff nehmen und sehen die früher erstellten Entscheidungen und Möglichkeiten anhand des eingetragenen Wissensmodelles. Bei neuen Änderungen wie zum Beispiel einer weiteren Baufirma kann diese einfach als Option hinzugefügt, ausgewählt und dokumentiert werden. Dies fördert die Nachvollziehbarkeit von anders gefällten Entscheidungen.

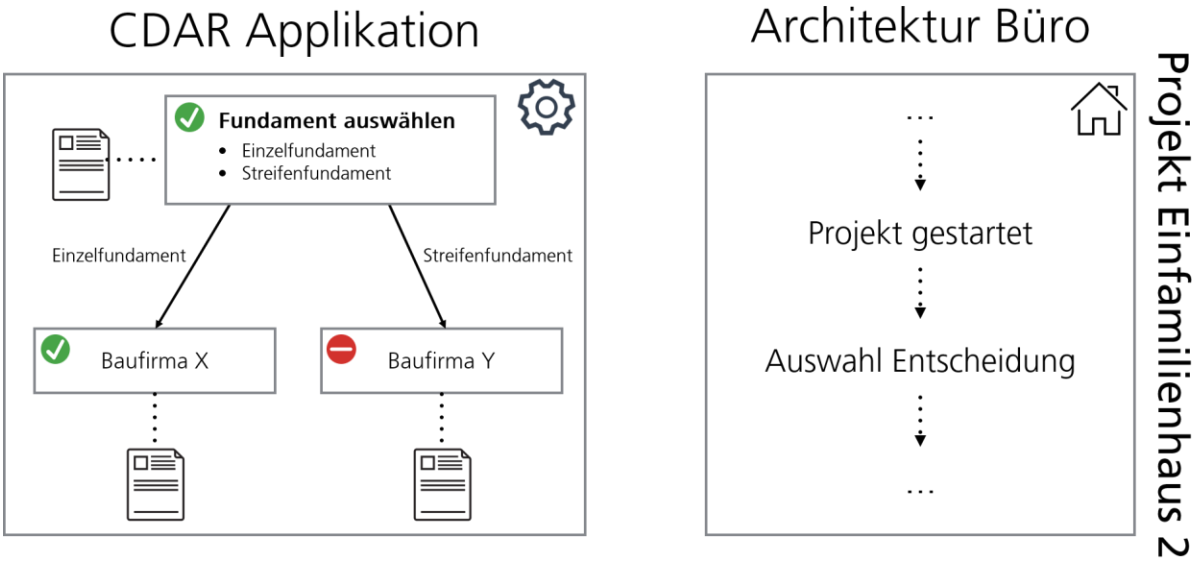


Abbildung 5: Einfamilienhaus mit CDAR-Applikation

Dieses Beispiel soll die grundlegende Funktionalität der Applikation erläutern und zeigt die generische Nutzung dieser Applikation. Das Werkzeug könnte in verschiedenen beruflichen Märkten seine Anwendung finden, im Rahmen dieser Bachelorarbeit gilt jedoch der Kontext von Cloud Computing.

2.3 Beispiel eines Wissenseintrages

Nachfolgende Beispiele zeigen den Stil potentieller Wissenseinträge, welche die Software beim Nutzen durch einen Software Architekten beinhalten wird [Ola14].

2.3.1 AD about Integration Style (IBM UMF Template for Decision Log)

Subject Area	Process and service layer design	**Topic**	Integration
Name	Integration Style	**AD ID**	2
Decision Made	We decided for RPC and the Messaging pattern (Enterprise Integration Patterns)		
Issue or Problem	How should process activities and underlying services communicate?		
Assumptions	Process model and NFRs/QA requirements are valid and stable.		
Motivation	If logical layers are physically distributed, they must be integrated.		
Alternatives	File transfer, shared database, no physical distribution (local calls)		
Justification	This is an inherently synchronous scenario: VSP users as well as internal Telco staff expect immediate responses to their requests. Messaging system will ensure guaranteed delivery and buffer requests to unreliable data sources.		
Implications	Need to select, install, and configure a message-oriented middleware provider.		
Derived Requirements	Many finer grained patterns are now eligible and have to be decidd upon: message construction, channel design, message routing, message transformation, system management (see Enterprise Integration Patterns book).		
Related Decisions	Next, we have to decide on one or more integration technologies implementing the selected two integration styles. Many alternatives exist, e.g. Java Message Service (JMS) providers.		

Tabelle 1: Beispiel eines Wissenseintrages

2.3.2 Y-Statements in Telco Case Study (EIP Pattern Selection)

- AD 1: Apply Messaging pattern and RPC pattern
  - In the context of the order management scenario at T,
  - Facing the need to process customer orders synchronously, without loosing any messages,
  - We decided to apply the Messaging pattern and the RPC pattern
    - And neglected File Transfer, Shared Database, no physical distribution (local calls)
  - to achieve guaranteed delivery and request buffering when dealing with unreliable data sources
  - accepting that follow-on detailed design work has be performed and that we need to select, install, and configure a message-oriented middleware provider.

3 Analyse der Anforderungen

3.1 Übersicht über die zentralen Konzepte der Domäne

Die nachfolgenden Begriffe etablierten sich im Laufe dieser Bachelorarbeit.

Begriff	Erklärung
Wissensproduzent (engl. Knowledge Producer)	Beschreibt eine Benutzerrolle. Ein Wissensproduzent pflegt Informationen in die Applikation ein.
Wissenskonsument (engl. Knowledge Consumer)	Beschreibt eine Benutzerrolle. Ein Wissenskonsument arbeitet mit Hilfe von Wissenseinträge, welche durch einen Wissensproduzenten vorgängig in die Software eingepflegt wurden.
Wissensbaum (auch Wissensmodell, engl. Knowledge Tree)	Allgemeine Beschreibung für ein Projekt eines Wissensproduzenten
Projektbaum (auch Anwendungsprojekt, engl. Project Tree)	Allgemeine Beschreibung für ein Projekt eines Wissenskonsumenten
Wissensnode (auch Entscheidung, Knoten, engl. Decision / Node)	Eintrag im System in Form einer Architekturentscheidung oder auch einem Architectural Refactoring. In jedem Fall sichtbar in der Entscheidungsverwaltung und optional im Entscheidungsführungsgraphen.
Wissenssubnode (auch Option, im Code: Subnode)	Option eines Wissensnode. Ein Wissensnode kann verschiedene Möglichkeiten bieten. Wissenssubnode bzw. Options beschreiben diese Möglichkeiten.
Entscheidungsverwaltung	Komplettansicht in Form eines Baumes aller Einträge.
Entscheidungsführungsgraph (auch Graph)	Graphansicht welche die Abhängigkeiten zwischen den Wissensnodes bzw. Wissenssubnodes visualisiert.
Wissenslink (auch Verbindung)	Verbindung im Entscheidungsführungsgraphen. Beschreibt von welchem Wissensnode man anhand welcher Option zum nachfolgenden Wissensnode gelangt.
Verzeichnis (engl. directory)	Verzeichnis beziehungsweise Ordner in der Entscheidungs-verwaltung

Tabelle 2: CDAR-Terminologie

3.2 Allgemeine Beschreibung des Software-Werkzeuges

Die Applikation ermöglicht einzelne Wissensprojekte anzulegen und vorhandene Informationen in das System einzupflegen. Die eingetragenen Informationen sind unter Umständen voneinander abhängig. Deshalb besteht eine Verknüpfung zwischen den eingetragenen Informationen. Dies soll mit Hilfe eines Entscheidungsführungsgraphen dargestellt werden. Ausserdem soll dem Wissensproduzenten ermöglicht werden, durch sämtlichen eingepflegten Informationen zu navigieren und diese mit in den erarbeiteten Baum einzupflegen.
Ein Anwender des Systems kann auch als Wissenskonsument mit der Applikation arbeiten. Hierbei kann er neue Projekte auf Grundlage von Wissensbäumen erstellen. Der Wissenskonsument soll sich in einer weiteren Graphansicht für eingepflegte Designentscheidungen entscheiden können. Dabei kann er nachlesen, welche Informationen diesbezüglich eingepflegt wurden und was für Folgen sich für das Projekt ergeben.

3.3 Produkt Funktion

Dieses Kapitel beschreibt die Funktionen und Anforderungen an das Produkt.
Grundsätzlich ist die Applikation in zwei Bereiche beziehungsweise Ansichten unterteilt:

- Ansicht des Wissensproduzenten und seinen jeweiligen Wissensprojekten
- Ansicht des Wissenskonsumenten und seinen jeweiligen Anwendungsprojekten

Auf die einzelnen Bereiche und deren spezifische Anforderungen wird in den nachfolgenden Kapiteln eingegangen. Allgemeine Anforderungen an die Applikation sind in der folgenden Tabelle beschrieben.

#	Funktion / Anforderung / Einschränkungen
1	Jeder Benutzer hat einen Account (inkl. Benutzername und Passwort)
2	Ein Benutzer kann seine Rolle wechseln (Wissensproduzent/-konsument)
3	Es können neue Benutzer erstellt werden
4	Ein Benutzer kann sich nur einmal gleichzeitig am System anmelden.

Tabelle 3: Allgemeine Produkt Funktionen

3.3.1 Funktionen des Wissensproduzentes

Die möglichen Funktionen der Applikation aus Sicht eines Wissensproduzenten beinhalteten in erster Linie das Einpflegen von Wissenselementen in Form von Architekturentscheidungen bzw. Architectural Refactorings. Hierbei besteht die Möglichkeit, neue Wissensprojekte zu erstellen oder in Vorhandene weiteres Wissen einzupflegen. Beim jeweiligen Projekt gibt es zwei Ansichten auf bereits eingetragene Daten. Diese bestehen aus einer Entscheidungsverwaltung, in welcher sämtliche Einträge ersichtlich sind und zudem aus einem Entscheidungsführungsgraphen, in welchem die Entscheidungen miteinander verknüpft werden können. Ein Wissensproduzent erhält zudem die Möglichkeit, seine eingepflegten Informationen zu exportieren bzw. wieder zu importieren.
In der nachfolgenden Tabelle sind die Anforderungen an den Bereich des Wissensproduzenten beschrieben.

#	Funktion / Anforderung / Einschränkungen
1	Ein Wissensproduzent kann Wissensbäume verwalten
2	Einem vorhandenen Wissensbaum können wiederkehrende Architekturentscheide hinzugefügt werden.
3	Die Applikation ermöglicht das Hinzufügen von Unterpunkten (Options) an einem erstellten Architekturentscheid.
4	Die Applikation bietet eine Export- sowie Import-Funktion zum Zwecke von Datenaustausch zwischen Wissensproduzent.
5	Die Applikation ermöglicht es dem Wissensproduzent, sämtliche erfasste Daten durch die Entscheidungsverwaltung anzuzeigen. Es existieren Möglichkeiten zur Ordnung, Gruppierung sowie Filterung der Daten.
6	Eine baumartige Ansicht ermöglicht es dem Wissensproduzent, einen Ablauf in einem Projekt anhand der eingepflegten Daten zu modellieren sowie umzubauen.
7	Die Applikation ermöglicht es dem Benutzer, den kompletten Inhalt der Applikation in strukturierten Daten zu exportieren.

Tabelle 4: Produkt Funktionen des Wissensproduzenten

3.3.2 Funktionen des Wissenskonsumenten

In der Applikation können Wissenskonsumenten ihre Anwendungsprojekte verwalten und diesen Wissensprojekte zuweisen. Anschliessend bietet die Applikation eine Ansicht auf die eingepflegten Wissenseinträgen des jeweiligen Wissensprojektes. Der Benutzer sieht jeweils, für welche Refactorings bzw. Änderungen er sich bereits entschieden hat. Dabei können die einzelnen Punkte in einer Detailansicht genauer betrachtet und zudem kommentiert werden. In der nachfolgenden Tabelle sind die Anforderungen an den Bereich des Wissenskonsumenten beschrieben.

#	Funktion / Anforderung / Einschränkungen
1	Ein Wissenskonsument kann Anwendungsprojekte verwalten
2	Einem Anwendungsprojekt können ein oder mehrere Wissensprojekte zugewiesen werden
3	Den einzelnen Entscheidungen können verschiedene Zustände zugewiesen werden.

Tabelle 5: Produkt Funktionen des Wissenskonsumenten

3.4 Benutzer Charakteristik

Die zu erstellende Applikation ist für Unternehmen verschiedenster Art und Grösse geplant. Im Rahmen dieser Bachelorarbeit erfolgt die Nutzung des Tools mit Fokus auf das Thema Cloud Computing. Grundsätzlich soll die Applikation jedoch generisch und somit auf verschiedene Themenbereiche anwendbar sein.

3.5 Annahmen und Abhängigkeiten

Die Ausführung des Clients benötigt einen aktuellen Internetbrowser mit aktivierter JavaScript Unterstützung. Serverseitig braucht die Applikation neben einer funktionierenden MediaWiki-Umgebung zudem einen Tomcat-Server (Version 7) sowie eine MySQL Server Instanz (Version 5.6).
Um eine problemlose Darstellung sicherzustellen, wird von einer aktuellen Version eines verbreiteten Browsers ausgegangen (siehe auch H.2.9 Kompatibilität und Einschränkungen). Ausserdem benötigt die Applikation eine ständige Netzwerkverbindung.

3.6 Funktionale Anforderungen

3.6.1 Aktoren und Stakeholder

Nachfolgend sind die Aktoren und Stakeholder des Systems aufgeführt.

Unangemeldeter Benutzer
Ein unangemeldeter Benutzer der Applikation hat die Möglichkeit, sich einen neuen Account einzurichten oder sich einzuloggen. Sämtliche weitere Funktionalitäten der Applikation sollen dem unangemeldeten Benutzer nicht zugänglich sein.

Wissensproduzent
Ein Wissensproduzent ist zuständig für das Einpflegen von Wissenseinträgen ins System. Er kann neue Wissensbäume erstellen, Daten eintragen und daraus einen Entscheidungsbaum modellieren.

Wissenskonsument
Ein Wissenskonsument kann neue Anwendungsprojekte anlegen und bearbeiten. Dabei kann er die Wissenseinträge sowie den Entscheidungsführungsgraphen von bereits eingepflegten Daten in sein Projekt laden. Nun ermöglicht die Applikation dem Benutzer, sein Projekt anhand dieser Informationen zu führen und der Wissenskonsument sieht seine Entscheidungsmöglichkeiten. Er kann sich bei einer anstehenden Entscheidung in Form von verschiedenen Status entscheiden.

3.6.2 Use Case Diagramm

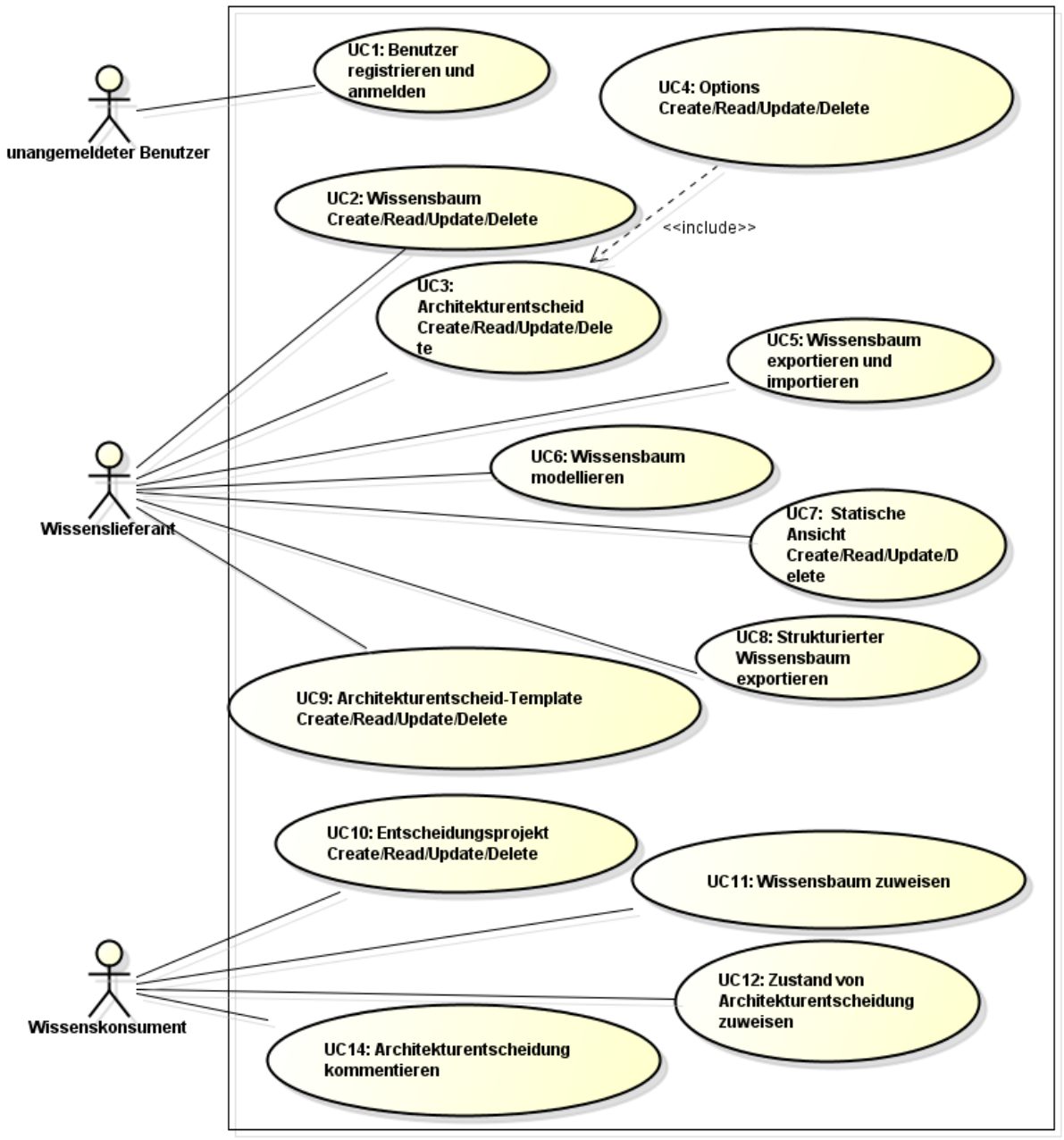


Abbildung 6: Use Case Diagramm

3.6.3 Use Case Beschreibungen

Nachfolgend die Beschreibungen sämtlicher Use Cases im Brief-Format.

3.6.3.1 UC1: Benutzer erstellen

Use Case Name	UC01: Benutzer erstellen
Aktoren	Unangemeldeter Benutzer
Beschreibung	Sämtliche Funktionen der Applikation, ausgenommen Registrierung, sind erst nach einem erfolgreichen Einloggen verfügbar. Im Rahmen dieser Bachelorarbeit können sämtliche unangemeldete Benutzer ein neues Konto erstellen.

3.6.3.2 UC2: Wissensbaum Create/Read/Update/Delete

Use Case Name	UC2: Wissensbaum Create/Read/Update/Delete
Aktoren	Wissensproduzent
Beschreibung	Bevor der Wissensproduzent mit dem Einpflegen von Wissenseinträgen beginnen kann, muss er in der Applikation einen Wissensbaum erzeugen. Nachträgliches Ändern des Titels sowie das Löschen von kompletten Wissensbäumen wird durch die Applikation unterstützt.

3.6.3.3 UC3: Architekturentscheid Create/Read/Update/Delete

Use Case Name	UC3: Architekturentscheid Create/Read/Update/Delete
Aktoren	Wissensproduzent
Beschreibung	Beinhaltet das Erstellen, Lesen, Bearbeiten und Löschen von Architekturentscheidungen in einen vorhandenen Wissensbaum.

3.6.3.4 UC4: Options Create/Read/Update/Delete

Use Case Name	UC4: Options Create/Read/Update/Delete
Aktoren	Wissensproduzent
Beschreibung	Eine eingetragene Architekturentscheidung beinhaltet unter Umständen verschiedene Möglichkeiten zur Realisation. Die Applikation erlaubt das Hinzufügen von mehreren Optionen an eine Entscheidung.

3.6.3.5 UC5: Export/Import Wissensbaum

Use Case Name	UC5: Wissensbaum exportieren und importieren
Aktoren	Wissensproduzent
Beschreibung	Um andere Wissensproduzenten, welche nicht über Zugriff auf die angelegten Projekte verfügen, mit bereits eingetragenen Daten zu unterstützen, soll die Applikation eine Export- sowie Import-Funktion von Wissenseinträgen ermöglichen.

3.6.3.6 UC6: Wissensbaum modellieren

Use Case Name	UC6: Wissensbaum modellieren
Aktoren	Wissensproduzent
Beschreibung	Bereits erfasste Architekturentscheidungen lassen sich in einer Graphstruktur miteinander verbinden.

3.6.3.7 UC7: Entscheidungsverwaltung Create/Read/Update/Delete

Use Case Name	UC7: Entscheidungsverwaltung Create/Read/Update/Delete
Aktoren	Wissensproduzent
Beschreibung	Angesichts einer übersichtlichen Darstellung sämtlicher Einträge, soll die Applikation die Möglichkeit bieten, diese zu gruppieren. Darüber hinaus soll die Darstellung eine Filterung aller Einträge ermöglichen.

3.6.3.8 UC8: Strukturierter Wissensbaum exportieren

Use Case Name	UC8: Strukturierter Wissensbaum exportieren
Aktoren	Wissensproduzent
Beschreibung	Zwecks der Ansicht des eingetragenen Wissens, soll die Applikation den Export der Inhalte in einem leserlichen Format ermöglichen.

3.6.3.9 UC9: Architekturentscheid-Template Create/Read/Update/Delete

Use Case Name	UC9: Architekturentscheid-Template Create/Read/Update/Delete
Aktoren	Wissensproduzent
Beschreibung	Die Applikation soll das Erstellen von Architekturentscheidungs-templates unterstützen. Dies vereinfacht das Erstellen von einheitlichen Einträgen.

3.6.3.10 UC10: Entscheidungsprojekt Create/Read/Update/Delete

Use Case Name	UC10: Entscheidungsprojekt Create/Read/Update/Delete
Aktoren	Wissenskonsument
Beschreibung	Beinhaltet das Erstellen, Lesen, Bearbeiten und Löschen von Entscheidungsprojekten.

3.6.3.11 UC11: Wissensbaum zuweisen

Use Case Name	UC11: Wissensbaum zuweisen
Aktoren	Wissenskonsument
Beschreibung	Zuweisung von Wissensbäumen an einem erstellten Entscheidungs-projekt.

3.6.3.12 UC12: Zustand von Architekturentscheidung zuweisen

Use Case Name	UC12: Zustand von Architekturentscheidung zuweisen
Aktoren	Wissenskonsument
Beschreibung	Um in der Applikation zwischen angenommenen, abgelehnten, noch nicht entschiedenen und zu überdenkenden Architektur-entscheidungen zu unterscheiden, soll diese das Setzen von Zuständen unterstützen.

3.6.3.13 UC13: Architekturentscheidung kommentieren

Use Case Name	UC13: Architekturentscheidung kommentieren
Aktoren	Wissenskonsument
Beschreibung	Zwecks der Begründung von Entscheidungen oder das Anbringen von Korrekturvorschlägen sollen Wissenskonsumenten eine Architektur-entscheidungen kommentieren können.

3.7 Weitere Anforderungen

#	Name	Beschreibung
1	Antwortzeit	Der Client reagiert auf eine Anfrage innerhalb von max. vier Sekunden, andernfalls wird der Benutzer über eine Ladeaktivität informiert.
2	Browserkompatibilität	Die Applikation unterstützt die Internetbrowser Internet Explorer 11, Firefox 29 und Google Chrome 35.
3	Effizienz	Der Client ermöglicht das Hinzufügen von Wissenssubnodes mit weniger als vier Klicks.
4	Sicherheit	Die Applikation kontrolliert den Zugriff auf geschützte Ressourcen.
5	Austauschbarkeit	Die Software-Layers sind so konzipiert, dass ein Layer nur wenig Abhängigkeiten aufweist. Damit in Zukunft beispielsweise auch mit einer nicht relationaler Datenbank gearbeitet werden kann.
6	Installierbarkeit	Das Installieren der Server-Applikation in einer frisch aufgesetzten Umgebung soll in weniger als 3 Stunden möglich sein.
7	Lizenzierung	Die Lizenzen von Software-Produkten und Technologien von Drittanbietern sollen möglichst populär und wenig restriktiv sein. Sie dürfen durch ihre Lizenzierung die Hoheit über den eigenen Code nicht einschränken[1].

Tabelle 6: Weitere Anforderungen

[1] http://opensource.org/licenses

3.8 Anforderungen an die Testumgebung

Im Rahmen dieser Bachelorarbeit wird die Software auf mehreren verschiedenen Testumgebungen geprüft. Die nachfolgende Hardwarespezifikation beschreibt die Leistungsschwächste Umgebung, auf welcher die Applikation ohne merkbare Einschränkungen benutzt werden soll.

	Client	Virtueller Server
RAM	2.5 GB	2 GB
CPU	Core Duo T2300 1.66 GHz	Xenon E5-2660 v2 2.2 GHz
System Typ	32 Bit	64 Bit
OS	Windows 8.1	Windows Server 2012

Tabelle 7: Testumgebungssetup

4 Domainmodell der CDAR Applikation

Das folgende Domainmodell bietet einen vereinfachten Überblick über die vorhandenen Objekte in der CDAR Applikation.

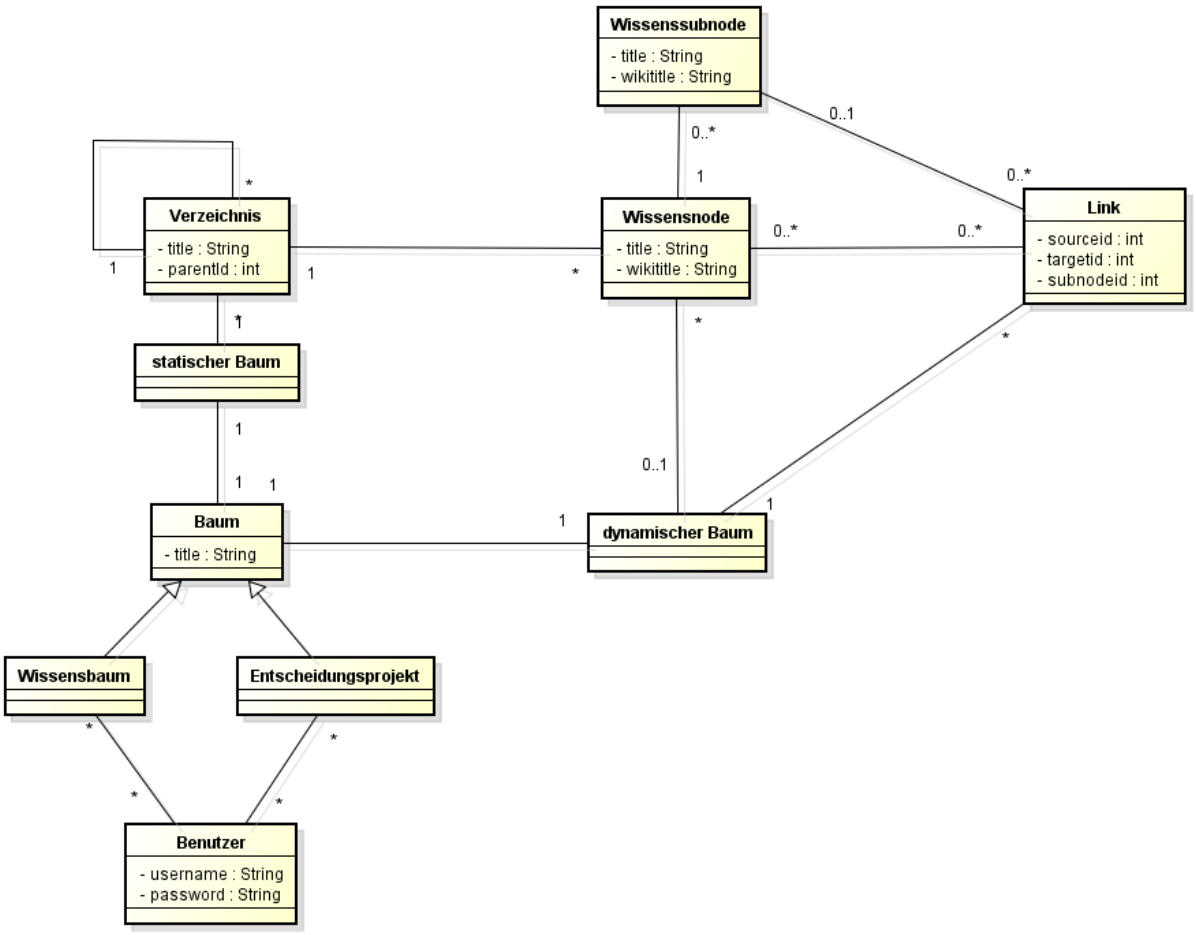


Abbildung 7: Domainmodel

4.1 Übersicht des Domainmodells

Damit sämtliche Änderungen nachvollziehbar sind, wird jeder Entität ein Benutzer zugeordnet. Dieser kann in der Rolle als Wissensproduzent Wissensbäume anlegen. Befindet er sich in der Rolle des Wissenskonsumentes, besteht die Möglichkeit, Entscheidungsprojekte anzulegen. Wissensbäume wie auch Entscheidungsprojekte haben jeweils eine Entscheidungsverwaltung, als auch einen Entscheidungsführungsgraphen. Die eingepflegten Wissenseinträge (Wissensnodes) befinden sich immer in einem Verzeichnis in der Entscheidungsverwaltung. Zudem besteht die Möglichkeit, diese dem Entscheidungsführungsgraph hinzuzufügen und anzuzeigen. Ein Link stellt eine Verbindung zwischen zwei Wissensnodes beziehungsweise zwischen einem Wisssensnode und einem Wissenssubnode dar.

4.2 Detailbeschreibung des Domainmodells

4.2.1 Wissensbaum und Entscheidungsprojekt

Die Klassen Wissensbaum und Entscheidungsprojekt stehen jeweils für ein komplettes Projekt, welches ein Benutzer entweder in der Rolle des Wissensproduzenten (Wissensbaum) oder des Wissenskonsumenten (Entscheidungsprojekt) anlegt. Da sich die Funktionalität der Applikation je nach Projektart unterscheidet, wird dies auch im Domainmodell differenziert. Grundsätzlich weisen die Klassen Wissensbaum und Entscheidungsprojekt jedoch keine unterschiedliche Attribute auf, es wird lediglich zwecks Übersichtlichkeit unterschieden.

4.2.1.1 Attribute

Attribut	Typ	Beschreibung
Title	String	Titel des Wissensbaumes beziehungsweise des Entscheidungsprojektes.

Tabelle 8: Attribute Entscheidungsprojekt

4.2.2 Verzeichnis

Die Klasse Verzeichnis repräsentiert im der Entscheidungsverwaltung ein Verzeichnis. Jeder Wissensnode befindet sich in einem Verzeichnis. Diese ermöglichen eine Verschachtelung, um mehrere Verzeichnishierarchien zu erlauben.

4.2.2.1 Attribute

Attribut	Typ	Beschreibung
Title	String	Titel des Verzeichnisses
ParentId	Int	Id des übergeordneten Verzeichnisses.

Tabelle 9: Attribute Verzeichnis

4.2.3 Wissensnode

Bei der Klasse Wissensnode handelt es sich um die Wissenseinträge eines Benutzers. Diese enthält neben einem Titel auch den Wiki-Titel der referenzierten Wiki-Seite.

4.2.3.1 Attribute

Attribut	Typ	Beschreibung
Title	String	Titel des Verzeichnisses
Wikititle	String	Titel der Wiki-Seite

Tabelle 10: Attribute Wissensnode

4.2.4 Wissenssubnode

Ein Wissenseintrag weist den Benutzer der Applikation auf eine mögliche Architektur-Entscheidung hin. Die Umsetzung einer solchen Entscheidung kann womöglich in unterschiedlichen Formen erfolgen. Diese können mittels der Klasse Wissenssubnode, als Option einem Wissensnode hinzugefügt werden.

4.2.4.1 Attribute

Attribut	Typ	Beschreibung
Title	String	Titel des Verzeichnisses
Wikititle	String	Titel der Wiki-Seite

Tabelle 11: Attribute Wissenssubnode

4.2.5 Wissenslink

Um eine Modellierung eines Entscheidungsführungsgraphen zu ermöglichen, müssen die einzelnen Wissenseinträge untereinander verknüpft werden. Die Klasse Wissenslink verbindet zwei Wissenseinträge miteinander. Optional gilt eine Verbindung lediglich in Bezug auf eine spezifische Option, hierfür ist das optionale Feld „SubnodeId“ vorhanden.

4.2.5.1 Attribute

Attribut	Typ	Beschreibung
SourceId	String	Id des Wissensnodes, von welchem die Verbindung abgeht
TargetId	String	Id des Wissensnodes, zu welchem die Verbindung führt
SubnodeId	Int	Optionales Feld, falls Verbindung einen Bezug auf einen Wissenssubnode hat

Tabelle 12: Attribute Wissenslink

4.2.6 Beziehungen der Klassen

Multiplizität	Source Klasse	Destination Klasse	Beschreibung
:	Benutzer	Wissensbaum	Benutzer kann mehrere Wissensbäume anlegen. Die Nutzung der Bäume kann durch mehrere Benutzern erfolgen, falls der Ersteller dies beabsichtigt.
:	Benutzer	Entscheidungs-projekt	Benutzer kann mehrere Entscheidungsprojekte anlegen. Die Nutzung der Projekte kann durch mehrere Benutzer erfolgen, falls der Ersteller dies beabsichtigt.
1:1	Baum	Entscheidungs-führungsgraphen	Jeder Baum beinhaltet einen Entscheidungs-führungsgraphen.
1:1	Baum	Entscheidungsverwaltung	Jeder Baum beinhaltet eine Entscheidungsverwaltung.
1:*	Entscheidung-sverwaltung	Verzeichnis	Die Entscheidungsverwaltung kann eine beliebige Anzahl an Verzeichnissen beinhalten.
1:*	Verzeichnis	Verzeichnis	Ein Verzeichnis kann selbst wieder mehrere Verzeichnisse beinhalten.
1:*	Verzeichnis	Wissensnode	Ein Wissensnode existiert in genau einem Verzeichnis. Ein Verzeichnis kann mehrere Wissensnode beinhalten.
0..1:*	Entscheidungs-führungsgraphen	Wissensnode	Boolsche Verbindung ob ein Node im Entscheidungs-führungsgraph sichtbar ist.
1:0..*	Wissensnode	Wissenssubnode	Ein Wissensnode kann mehrere Optionen (Wissenssubnodes) beinhalten.
0..*:0..*	Wissensnode	Link	Ein Wissensnode kann in mehreren Verbindungen auftreten.
0..1:0..*	Wissenssubnode	Link	Ein Wissenssubnode kann in mehreren Links verwendet werden.

Tabelle 13: Beziehung der Klassen

4.2.7 Beispieldaten

Eine Übersicht mit Beispieldaten findet sich in Kapitel „7.8.1 Beispielinhalt der Datenbank“.

5 Architekturdesign

5.1 Übersicht des Systems

Die nachfolgende Grafik zeigt die Software Architektur und das Zusammenspiel der implementierten Komponenten mit MediaWiki.

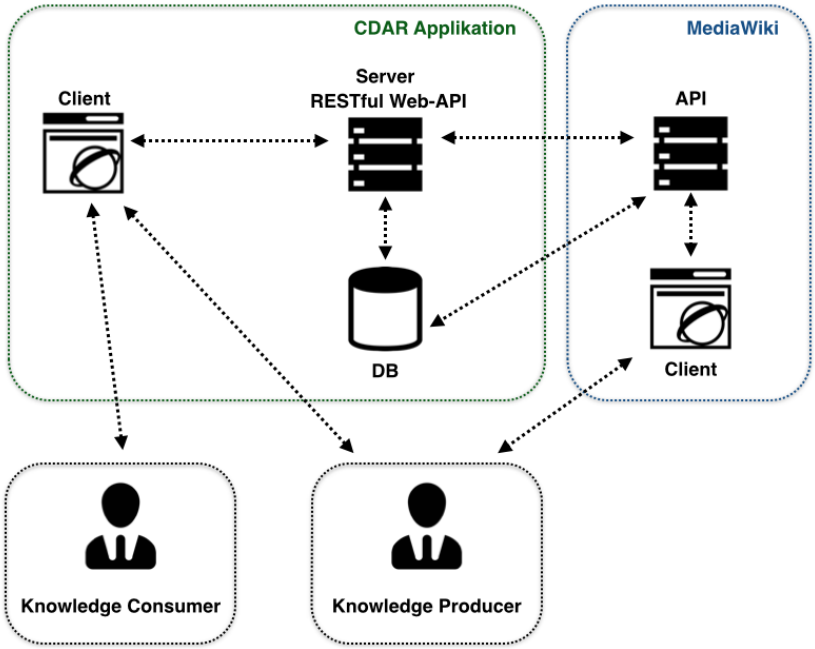


Abbildung 8: Softwarearchitektur

5.1.1 CDAR Applikation

Die implementierte Software Architektur besteht aus dem Webclient, einer RESTful WebAPI sowie einer Datenbank. Der Benutzer arbeitet mit der Applikation über das erstellte Webinterface. Sämtliche Aufrufe gehen via HTTP an den Server und werden persistiert. Handelt es sich um einen Aufruf eines Wissenseintrages, beispielsweise das Lesen oder Schreiben von Wissenseinträgen, leitet die WebAPI den Aufruf an eine konfigurierte Wiki-Software weiter.

5.1.2 Wiki-Software

MediaWiki besitzt neben einer API auch eine Weboberfläche. Über diese erfolgt ein Zugriff auf die MediaWiki-Engine direkt aus dem Browser, ohne Zutun des CDAR-Tools. Im Rahmen dieser Bachelorarbeit haben beide Benutzerrollen ebenfalls Zugriff auf diese Schnittstelle. Erstellt der Nutzer der Applikation einen Wissenseintrag, leitet die CDAR Applikation diese Anfrage mittels der API-Schnittstelle von MediaWiki weiter. Somit sind sämtliche Wissenseinträge auch innerhalb als auch ausserhalb der Applikation verfügbar.

5.1.3 Datenbank

Die darunterliegende Datenbank kann entweder gemeinsam für die Wiki-Software sowie die CDAR-Applikation genutzt werden. Optional können selbstverständlich auch einzelne Instanzen genutzt werden.

5.2 Externes Design der Applikation

5.2.1 Vorgehen

Früh erstellte Mockups erlaubten es, Unklarheiten zu erkennen und darauf zu reagieren. Die Erstellung erfolgte mit dem Tool Balsamiq Mockups [Bal].

5.2.2 Mockups

5.2.2.1 Login und Registrierung

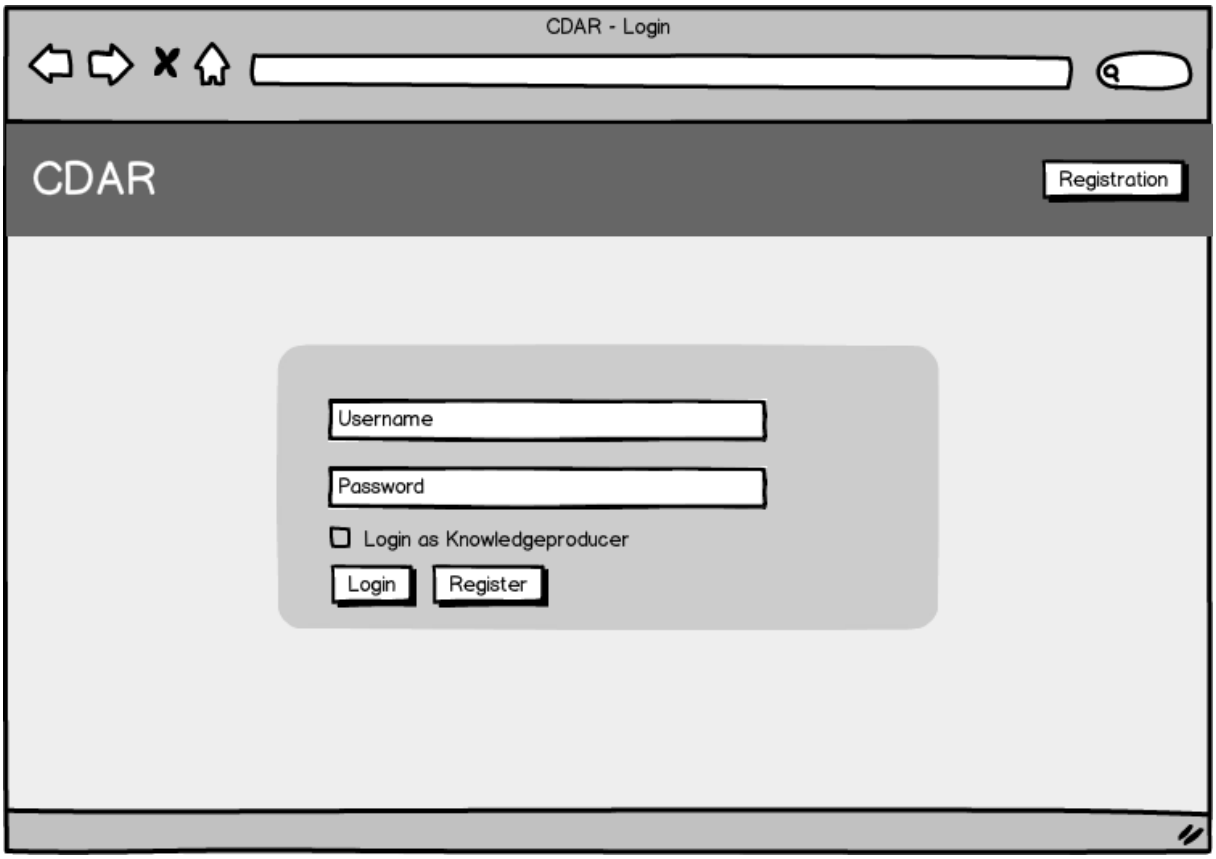


Abbildung 9: Mockup Login und Registrierung

Sämtliche Funktionen der Applikation sind nur in einem angemeldeten Zustand benutzbar. Die Software ermöglicht jedem Benutzer, einen neuen Benutzeraccount anzulegen. Diese Ansicht zeigt das Login, welches mittels Username sowie Passwort erfolgt. Gleich zu Beginn steht es dem Benutzer frei, sich als Wissensproduzent oder Wissenskonsument anzumelden.

5.2.2.2 Wissensproduzent

Die nachfolgenden Ansichten betreffen die User Rolle Wissensproduzent.

5.2.2.3 Hauptseite

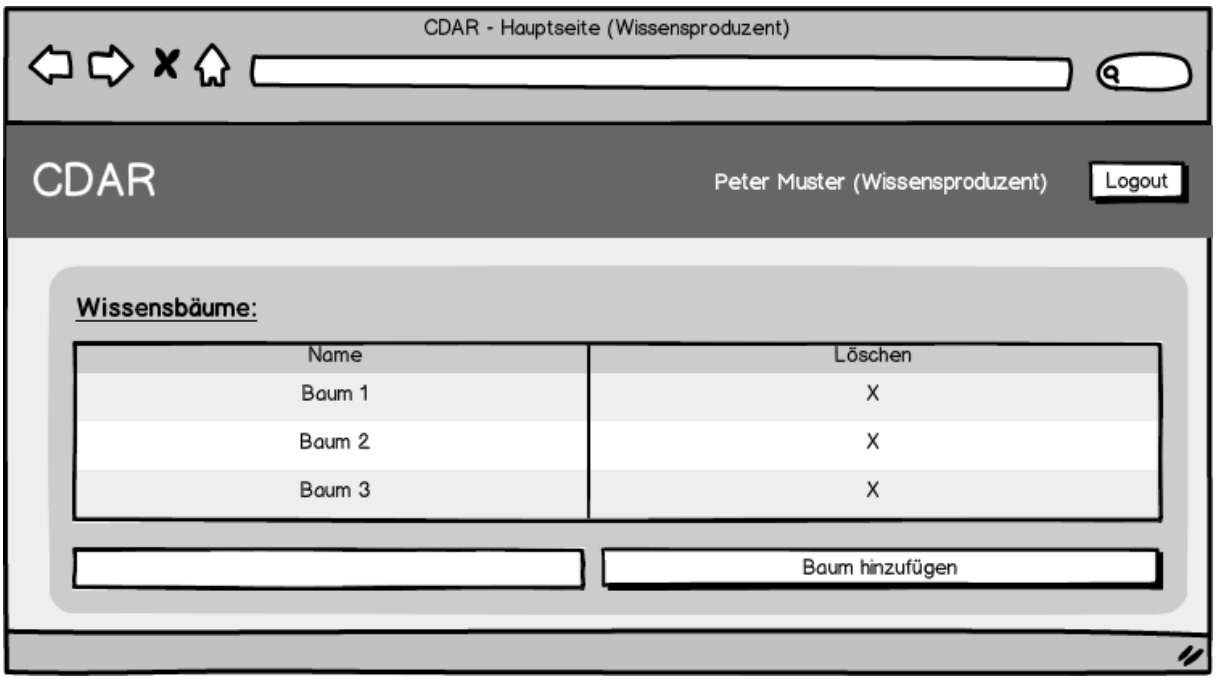


Abbildung 10: Mockup Hauptseite Wissensproduzent

Die Startseite des Wissensproduzenten bietet dem Benutzer eine Übersicht über vorhandene Wissensbäume. Die Wissensbäume können an dieser Stelle aufgerufen oder gelöscht werden. Zusätzlich besteht die Möglichkeit, weitere Wissensbäume hinzuzufügen.

5.2.2.4 Baumansicht

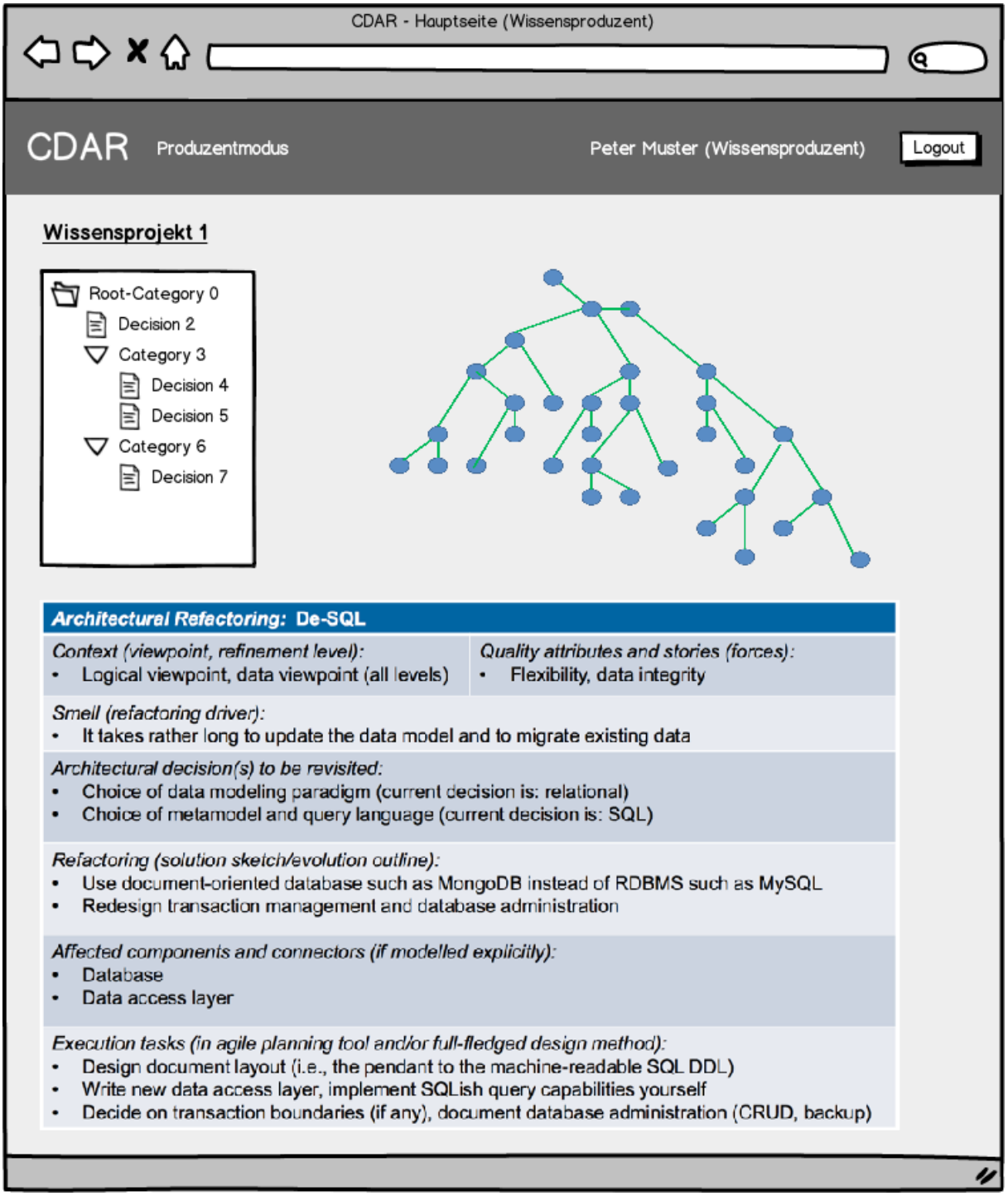


Abbildung 11: Mockup Baumansicht

In der oberen Hälfte befinden sich die Baumstruktur und der Entscheidungsführungsgraph. Die untere Hälfte zeigt die Wiki-Seite des selektierten Wissenseintrages.

5.2.2.4.1 Entscheidungsverwaltung

Die Entscheidungsverwaltung beinhaltet sämtliche eingepflegte Wissenseinträge. Sie ermöglicht dem Benutzer das Ordnen der Einträge in verschiedene Verzeichnisse, sodass er eine Übersicht über seine eingepflegten Daten erhält und durch sie navigieren kann.

5.2.2.4.2 Entscheidungsführungsgraphen

Sobald der Anwender einige Wissenseinträge angelegt hat, lassen sich diese in den Entscheidungsführungsgraphen ziehen. Durch das Verbinden dieser Einträge besteht die Möglichkeit, einen Ablauf zu modellieren. Dies entscheidet schlussendlich, wie sich später ein Wissenskonsument durch ein Projekt führen lässt.

5.2.2.5 Entscheidungsansicht

Nach Auswahl eines Wissenseintrages in der Entscheidungsverwaltung beziehungsweise im Graphen, erscheint am unteren Rand eine Ansicht, welche die Architekturentscheidung respektive das Architectural Refactoring anzeigt. Darüber hinaus kann der Anwender den Eintrag verändern, ergänzen oder auch löschen. Diese Funktionalität wird an dieser Stelle nur erwähnt und ist auf den Wireframes nicht ersichtlich.

5.2.2.6 Wissenskonsument

Nachfolgende Mockups beziehen sich auf Ansichten, auf welche der User in der Rolle als Wissenskonsument trifft.

5.2.2.6.1 Hauptseite

Abbildung 12: Mockup Hauptseite Wissenskonsument

Nach einem erfolgreichen Login des Benutzers in der Rolle als Wissenskonsument, zeigt die Applikation eine Übersicht der angelegten Projekte. Über eine Eingabe kann auch ein neues Projekt hinzugefügt werden.

5.2.2.6.2 Projektansicht

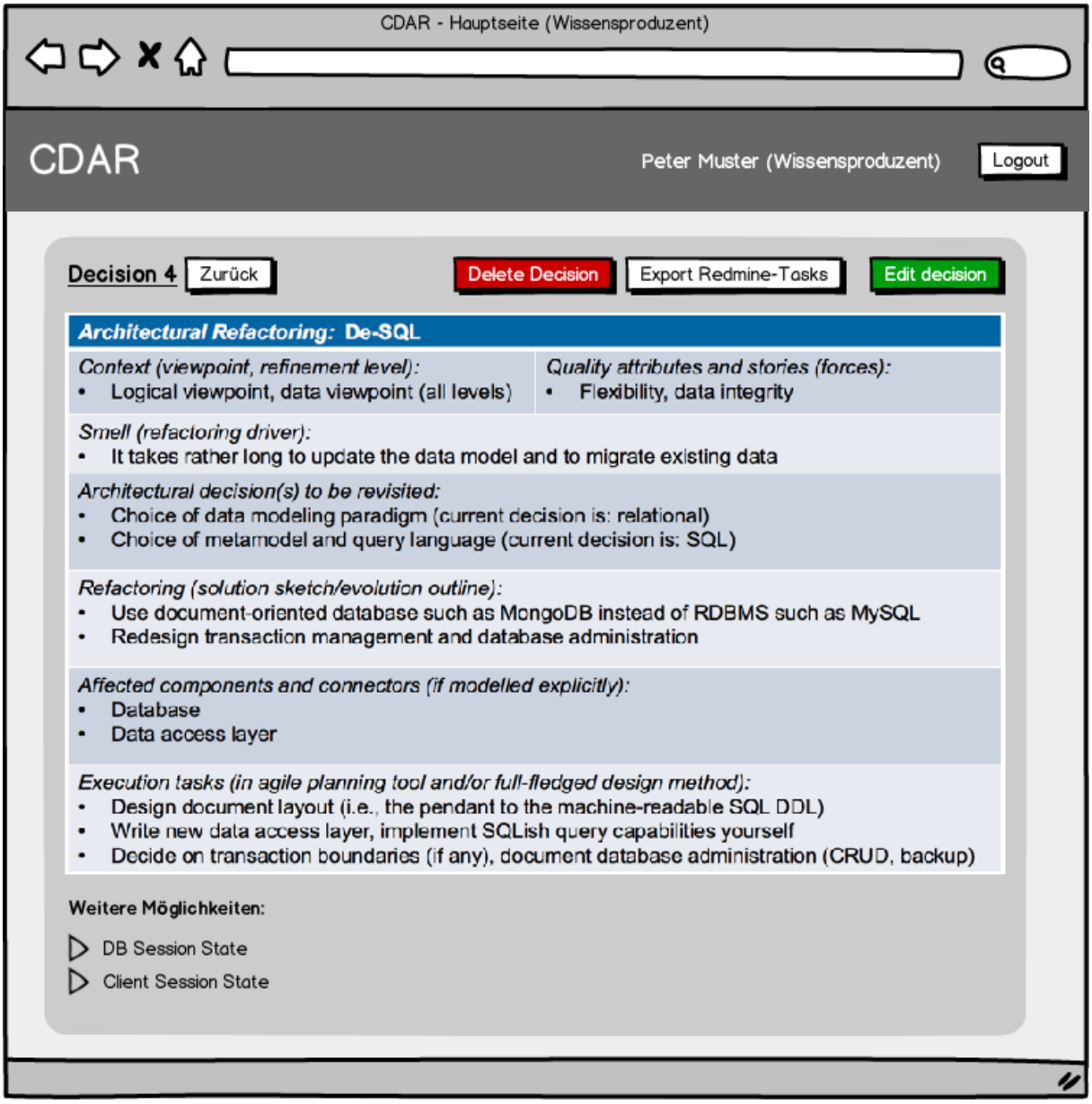


Abbildung 13: Mockup Projektansicht

Grundsätzlich ähnelt sich diese Ansicht stark der unter Punkt 5.2.2.4 gezeigten Ansicht des Wissensproduzenten. Ein Wissenskonsument hat jedoch zusätzlich die Möglichkeit, sich für einen Wissenseintrag zu entscheiden. Die einzelnen Einträge sind im Entscheidungsführungsgraphen farblich markiert um den Ablauf des Projektes nachvollziehen zu können.
Darüber hinaus können die einzelnen Anwendungsprojektmitglieder ihre Entscheidungen anhand einer Kommentarfunktion kommentieren. So lassen sich auch zu einem späteren Zeitpunkt die gewählten Entscheidungen nachvollziehen.

6 Evaluation der Technologien zur Umsetzung der Architektur

6.1 Servertechnologien

6.1.1 Tomcat Anwendungsserver

Beschreibung
Apache Tomcat ist eine OpenSource Implementierung der Apache Software Foundation. Die Software ermöglicht das Hosting von Webservern und Webcontainern und erlaubt in Java geschriebene Web-Anwendungen auszuführen. Lizenziert ist die Technologie unter der Apache Lizenz [Apa].

Vorteile

- Open-Source Implementierung
- Im Projektteam vorwissen vorhanden durch Arbeit mit der Technologie im Rahmen der Studienarbeit

Alternativen

- Glassfish
- JBoss

6.1.2 Jersey

Beschreibung
Die Implementierung einer erweiterbaren und wartbaren RESTful Schnittstelle ist keine einfache Aufgabe. Um den Implementierungsaufwand möglichst gering zu halten, haben wir uns für das Java Jersey Framework entschieden.
Es gibt viele Implementierungen des REST Architekturstils in Java, bei Jersey handelt es sich um die Open Source Referenz Implementierung. Sowohl in der Entwicklung als auch der Produktion ist diese Implementierung sehr weit verbreitet [Jer].

Vorteile

- Weit verbreitet und somit grosse Community
- Einfach anzuwenden durch die Nutzung von Annotations

6.1.3 Datenbanktechnologie

Für den Datenbankzugriff wurde MySQL mit einfachen JDBC Abfragen gewählt. In Zukunft wäre auch die Nutzung von einer NoSQL-Datenbank möglich. Die Kapselung der verschiedenen Layer erlaubt einen Austausch der Data-Access-Schicht ohne Refactorings durch alle Schichten.

6.2 Clienttechnologien

6.2.1 JavaScript-Framework

Beschreibung
AngularJS ist ein OpenSource-JavaScript-Framework von Google. Mit Angular lassen sich Single Page Applications für den Browser entwickeln, welche clientseitig lediglich HTML, CSS sowie JavaScript benötigen [Ang].

Vorteile

- Vorwissen vorhanden durch Miniprojekt im Modul „Internettechnologien“
- Grosse Community
- Setzt auf REST-Designprinzipien und transferiert Daten als JSON-Objekte

Nachteile

- Sehr mächtig, bietet viele Möglichkeiten

Alternativen

- Ember.js
- Backbone.js

6.2.2 JavaScript-Plugin Entscheidungsverwaltung

Beschreibung

JsTree ist ein JQuery-Plugin zur Visualisierung von interaktiven Baumdarstellungen. Dabei sind Elemente wie in einem Betribesystemverzeichnis verschachtelt. Die Library ist OpenSource und steht unter der MIT-Lizenz [JsT].

Vorteile

- Aktive Entwicklung
- Kein grosses Vorwissen notwendig

Nachteile

- Wenig Beispielcode durch den Hersteller vorhanden
- Produkt ist abhängig von einer einzelnen Privatperson

6.2.3 JavaScript-Plugin Entscheidungsführungsgraphen

Beschreibung

JsPlumb ist eine Java-Script-Library zur Visualisierung von Graphen. Die Knoten des Graphen können verschoben und miteinander verknüpft werden. Die Library steht unter einer Duallizenzierung von MIT und GPLv2 [JsP].

Vorteile

- Aktive Weiterentwicklung
- Informative Webseite mit Codebeispielen

Nachteile

- Bietet kein automatisches Anordnen des Graphen
- Produkt ist abhängig von einer einzelnen Privatperson

Alternativen

- D3.js

6.3 Wiki-Software

MediaWiki wurde ursprünglich für die Wikipedia programmiert und ist als freies Softwarepaket erhältlich. Viele im Internet zugängliche Wikisysteme werden durch MediaWiki betrieben. Die Wiki-Engine steht unter der GPL-Lizenz [Med].

Vorteile:

- Grosse Verbreitung und somit grosse Community
- Aktive Weiterentwicklung
- Bekannte Syntax

Nachteile:

- Kein brauchbarer WYSIWYG-Editor

Alternativen:

Es gibt unzählige verschiedene Wiki-Software auf dem Markt. Eine nach Programmiersprache sortierte Liste ist unter folgendem Link zu finden.
Link: http://de.wikipedia.org/wiki/Liste_von_Wiki-Software

6.4 Übersicht der eingesetzten Libraries und Plugins

Nachfolgend gelistete Software und deren Lizenzen beinhaltet die in der Bachelorarbeit erstellte Software.

6.4.1 CDAR-Browseranwendung (Client)

Name	Lizenz
AngularJS	MIT
Bootstrap	MIT
JsPlumb	MIT + GPL v2
JQuery	MIT
JQuery UI	MIT
JQuery Noty	MIT
JsPlumb Liviz	Keine spezifische Lizenzierung[1]
JsTree	MIT
XEditable	MIT

Tabelle 14: Lizenzen Clientteil

6.4.2 CDAR-Serveranwendung

Name	Lizenz
Apache Maven	Apache v2
Bliki	EPL
Genson	Apache v2
Java Jersey	CDDL + GPL
Java MySQL Connector	GPL
Joda-Time	Apache v2
JSTL	CDDL + GPL
JUnit	EPL
MediaWiki	GNU
Wiki-Java	GNU GPL v3

Tabelle 15: Lizenzen Serverteil

[1] https://github.com/Indb/jsPlumb_Liviz.js

7 Implementierung der Architektur

7.1 Datenhaltung

Die Speicherung der Daten erfolgt in einer eigenen MySQL-Datenbank. Sämtliche Wissenseinträge, beispielsweise Architekturentscheidungen persistiert die Datenbank von MediaWiki.

Abbildung 14: Datenhaltung

7.1.1 Client

Der Browser persistiert in Form von Cookies die User-Id sowie den Accesstoken des eingeloggten Benutzers. Diese Daten werden clientseitig benötigt, um bei einer Abfrage dem Server mitzuteilen, dass der Benutzer eingeloggt ist.

7.1.2 Server

Der Server speichert sämtliche Daten in einer eigenen MySQL-Datenbank. Hierzu gehören beispielsweise die Userdaten, angelegte Bäume und Projekte sowie auch die Repräsentation der Baum- und Graphenstruktur. Die zugehörigen Namen der Wiki-Seiten sind pro Entscheidung als Referenz in der Datenbank gespeichert.

7.1.3 MediaWiki

Um die diversen Wiki-Vorteile, beispielsweise die sichtbare Änderungshistorie zu nutzen, speichert der Server sämtliche Einträge und deren Änderungen ins MediaWiki. Ausgenommen hiervon sind lediglich Templates, welche sich ein Benutzer anlegt. Diese sind weniger sensibel und nicht unter vielen, zeitgleichen Änderungen betroffen. Aus diesem Grunde und zudem um die Performance zu steigern, persistiert der Server die Templates.

7.2 Parallele Bearbeitung

Während dieser Bachelorarbeit wurde entschieden, dass keine parallele Bearbeitung am selben Projekt unterstützt werden soll, um potentielle Überschreibungen zu vermeiden. Versucht ein Benutzer ein Projekt zu bearbeiten, welches sich bereits in Bearbeitung befindet, so erscheint im Client folgende Meldung.

Abbildung 15: Lockingmeldung

Der Benutzer wird informiert, bevor Veränderungen sichtbar sind. Das heisst, wenn der Benutzer eine neue Entscheidung erstellen will, wird dieser darüber informiert und kann die Aktion nicht vollenden. Die Entscheidung welche er erstellen wollte, erscheint nicht und wird auch nicht gespeichert.

Diese Reihenfolge trifft auf folgende zwei Szenarien nicht zu.

- Verbindung erstellen
- Ordner / Entscheidungen kopieren

In diesen Fällen sind Änderungen am Projekt sichtbar bevor der Benutzer darüber informiert wird, dass eine Sperrung auf das Projekt besteht. Die vorgenommenen Änderungen werden aber nicht persistiert. Ein Beispiel für den genauen Ablauf der Kommunikation ist im Kapitel „7.3.2 Beispielkommunikation Node kopieren“ festgehalten.

Diese Sperrung bezieht sich ausschliesslich auf projektverändernde Aktionen. Im Rahmen dieser Bachelorarbeit kann die Sperrung nicht explizit freigegeben werden. Die Sperrung wird nach einer einstellbaren Zeit vom Server Freigegeben. Mehr Informationen zur Konfiguration sind im Kapitel „H.2.8 Konfigurationen der Applikation“ zu finden.

7.3 Kommunikation

Die Kommunikation zwischen Client und Server erfolgt über HTTP. Der Client spricht jeweils die RESTful WebAPI des Servers an, dieser erwartet und sendet Objekte als JSON [The]. JSON-Objekte haben den Vorteil, dass sie im Gegensatz zu XML weniger Overhead enthalten und zudem vom Client zur Weiterverarbeitung im Browser nicht umgewandelt werden müssen. Der Server kümmert sich jeweils um die Serialisierung von und zu Java-Objekten.

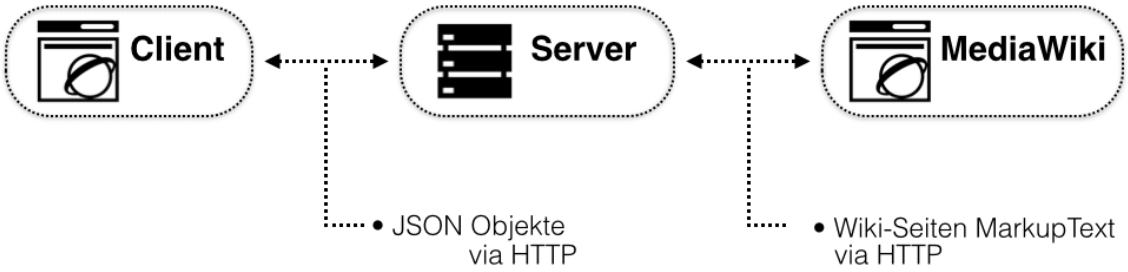


Abbildung 16: Kommunkation

Mit der Hilfe des wiki-java-Frameworks findet die Abfrage und Speicherung von Wiki-Seiten im MediaWiki statt [Wik]. Die Generierung von HTML-Text aus dem Wiki-Markup erfolgt mittels der Bliki-Bibliothek im Server [Jav].

7.3.1 Beispielkommunikation Node erstellen

Erstellt der Benutzer im Client einen Wissenseintrag, sendet dieser eine Anfrage an den Server, welcher die Tree-Identifikation und das Elternverzeichnis des neu erstellten Nodes beinhaltet. Der Server prüft, ob das Projekt bereits durch einen anderen Benutzer bearbeitet wird. Ist dies der Fall, wird der ganze Vorgang durch eine Lock-Antwort abgebrochen und es erscheint eine Meldung, welche dies dem Benutzer signalisiert. Ist jedoch das Projekt aktuell nicht in Bearbeitung durch einen anderen Benutzer, so erstellt der Server eine neue Wiki-Seite für den Node. Des Weiteren sendet der Server dem Client den erstellten Node als Antwort. Der Client zeichnet nun den neuen Node und setzt dessen Identität, welche durch den Server bzw. die Datenbank festgelegt wurde. Nachdem der Node gezeichnet wurde, kann ihm der Benutzer einen Titel geben. Nach dessen Bestätigung, sendet der Client eine Update-Anfrage. Der Server überprüft erneut auf eine Sperrung des Projektes. Falls keine vorhanden ist, aktualisiert er den Titel in der Datenbank und retourniert dem Client den aktualisierten Node.

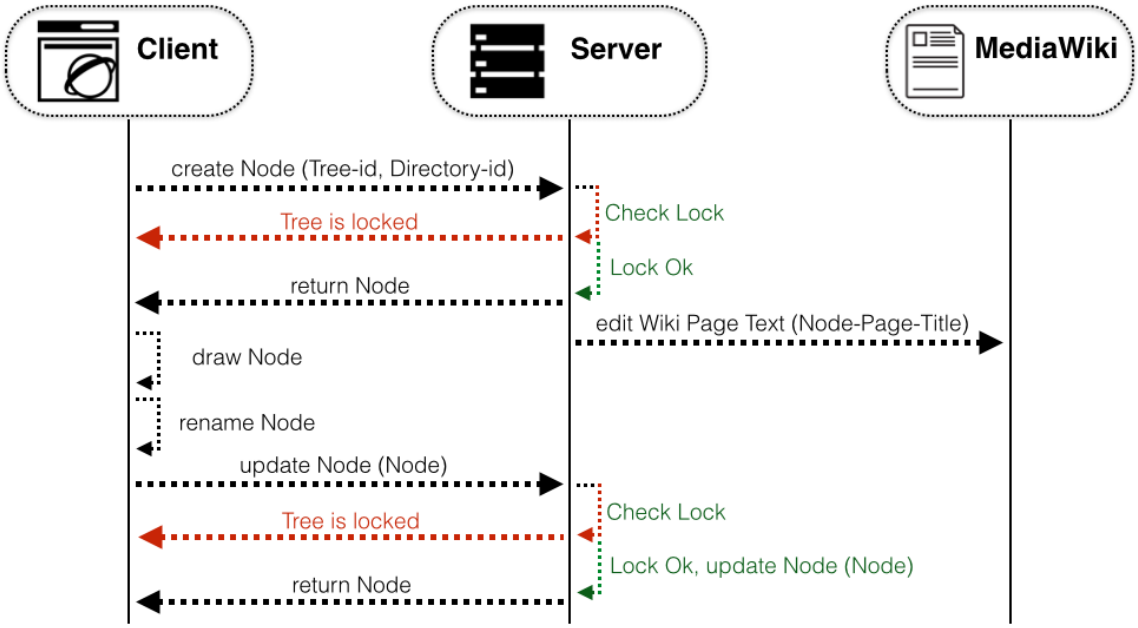


Abbildung 17: Kommunikation Node erstellen

7.3.2 Beispielkommunikation Node kopieren

Bei einem Kopiervorgang zeichnet jsTree die Entscheidung umgehend und es wird ein Event los gefeuert. Der Event liefert als Parameter den kopierten Node. Zum diesem Zeitpunkt ist jedoch das Elternverzeichnis noch nicht bekannt. Daher sendet der Client lediglich der Wert „0“ für das Elternverzeichnis, den Titel und die Tree-Identifikation. Nun überprüft der Server, ob das Projekt von einem anderen Benutzer momentan bearbeitet wird. Ist dies der Fall, wird der ganze Vorgang durch eine Lock-Antwort abgebrochen. Es erscheint eine entsprechende Meldung, welche dies dem Benutzer signalisiert.

Im Gegensatz zum Erstellen eines Nodes wird beim Kopieren der Node sofort gezeichnet, da der Event erst nach dem Kopiervorgang ausgelöst wird. **Die Entscheidung wird im Fall einer Sperrung jedoch nicht persistiert.**

Ist das Projekt aktuell nicht in Bearbeitung durch einen anderen Benutzer, so erstellt der Server eine neue Wiki-Seite mit demselben Inhalt wie der Originalnode. Ebenso werden dessen Optionen und deren Wiki-Einträge kopiert. Das Erstellen der Wiki-Einträge geschieht parallel. Währenddessen sendet der Server die kopierte Entscheidung an den Client zurück. Die kopierte Entscheidung besitzt den Standardstatus „Open“. Ebenso ist das Flag für die Visualisierung im Entscheidungsführungsgraphen nicht gesetzt.
Im retournierten Node ist unter anderem die Identifikation abgelegt, welche dem Client noch nicht bekannt ist. Diese wird nun im Clientobjekt gesetzt. Anschliessend gibt der Client dem Server den Befehl das Elternverzeichnis zu setzen. Dieser Vorgang wird ebenfalls auf eine Sperrung überprüft.

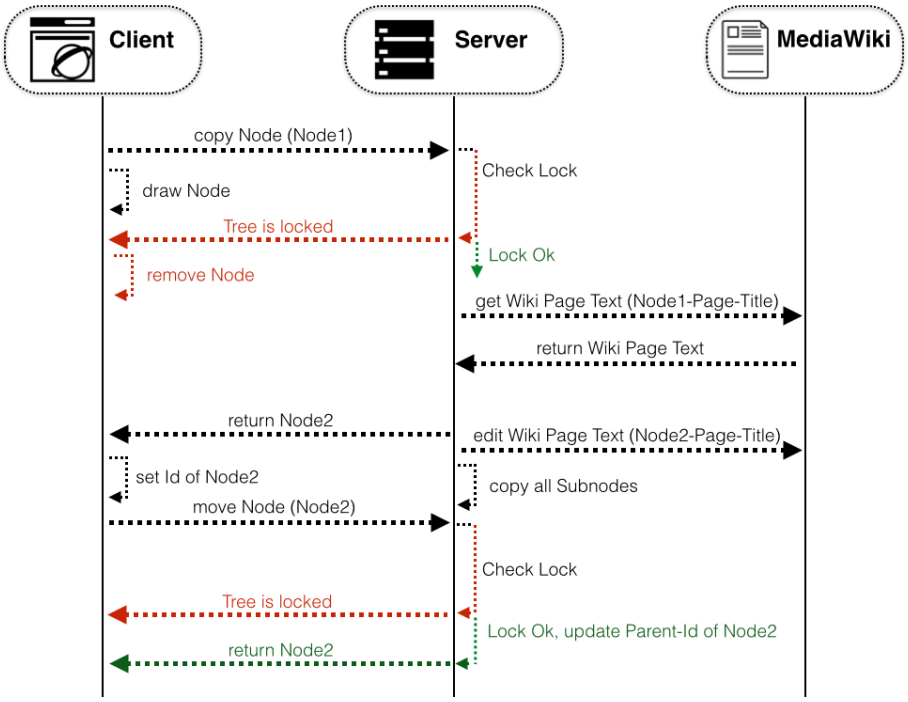


Abbildung 18: Kommunikation Entscheidung kopieren

7.3.3 Beispielkommunkation Node löschen

Löscht der Benutzer einen Wissenseintrag, sendet der Client eine „delete Node“ Anfrage mit der Identifikation des zu löschenden Nodes an den Server. Der Server prüft nun, ob das Projekt bereits durch einen anderen Benutzer bearbeitet wird. Ist dies der Fall, wird der ganze Vorgang durch eine Lock-Antwort abgebrochen. Es erscheint eine entsprechende Meldung, welche dies dem Benutzer signalisiert. Ist das Projekt aktuell nicht in Bearbeitung durch einen anderen Benutzer, so bestätigt der Server dies durch eine Bestätigung des Auftrages. Anschliessend löscht der Client die Entscheidung aus der Projektverwaltung. Ebenso veranlasst der Client das Löschen des Knotens sowie dessen Verbindungen im Entscheidungsführungsgraphen, falls der Node in diesem vorhanden ist.

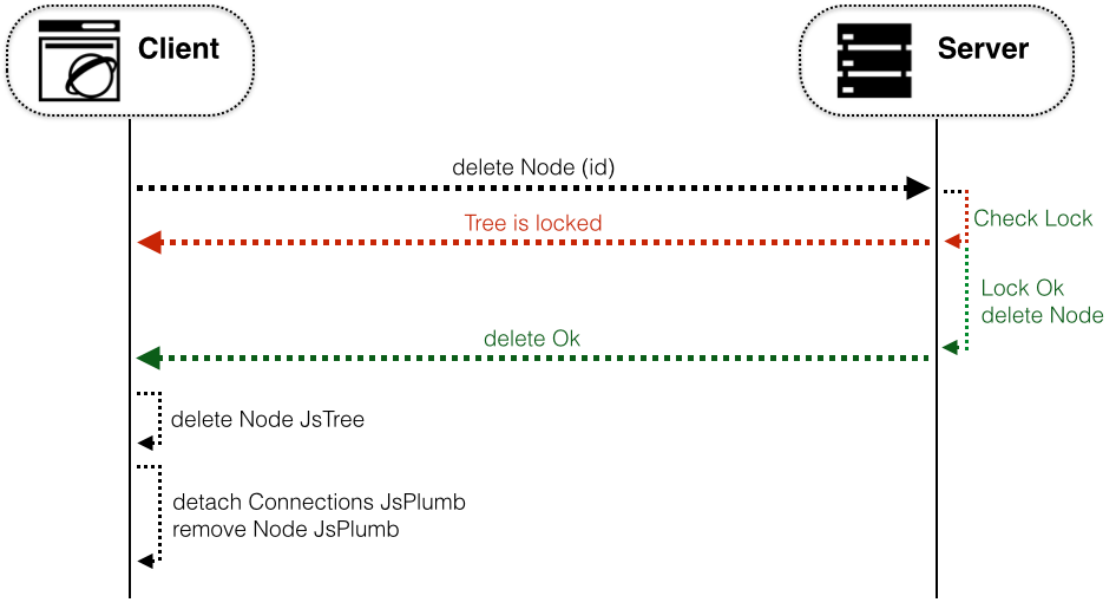


Abbildung 19: Beispielkommunikation Entscheidung löschen

7.3.4 Beispielkommunkation Node Drag & Drop

Der Verschiebevorgang einer Entscheidung von der Entscheidungsverwaltung in den Entscheidungsführungsgraphen löst zuerst den „Start Drag & Drop“ Event aus. Dabei überprüft der Client im nächsten Schritt, ob diese Entscheidung bereits im Graphen vorhanden ist. Dies wird durch den „Drag & Drop Move“ Event ausgelöst. Ist dies der Fall kann der Drop-Vorgang nicht vollendet werden. Diese Abfrage kann jedoch noch nicht überprüfen, ob ein Lock auf das Projekt besteht. Wenn der Benutzer die Entscheidung im Entscheidungsführungsgraphen loslässt (Drop) so löst dies ein Update-Befehl an den Server aus. Dieser Überprüft, ob eine Sperrung besteht. Ist dies der Fall, wird der ganze Vorgang durch eine Lock-Antwort abgebrochen. Es erscheint eine entsprechende Meldung, welche dies dem Benutzer signalisiert. Ist das Projekt aktuell nicht in Bearbeitung durch einen anderen Benutzer, so bestätigt der Server dies. Danach wird die Entscheidung im Entscheidungsführungsgraphen gezeichnet.

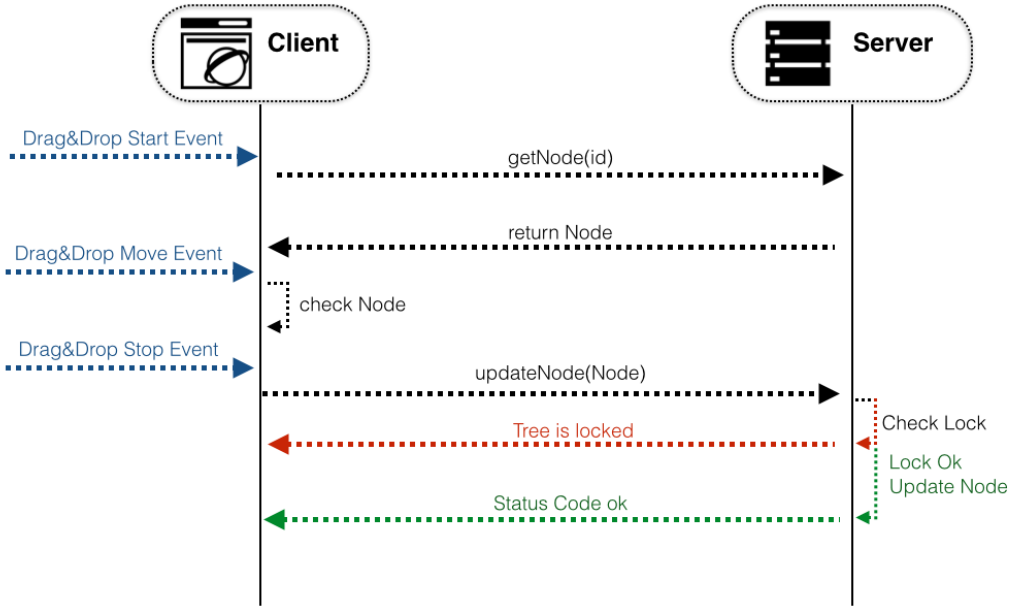


Abbildung 20: Beispielkommunikation Entscheidung Drag & Drop

7.3.5 Beispielkommunikation Abfrage Wiki-Seite

Für die Anzeige der einzelnen Wissenseinträge im Graphen oder der Baumansicht erfolgt noch keine Abfrage der Inhalte der Wiki-Seite. Erst nachdem der Benutzer einen bestimmten Wissenseintrag abfragen will, ruft der Client beim Server den Inhalt der Wiki-Seite auf.

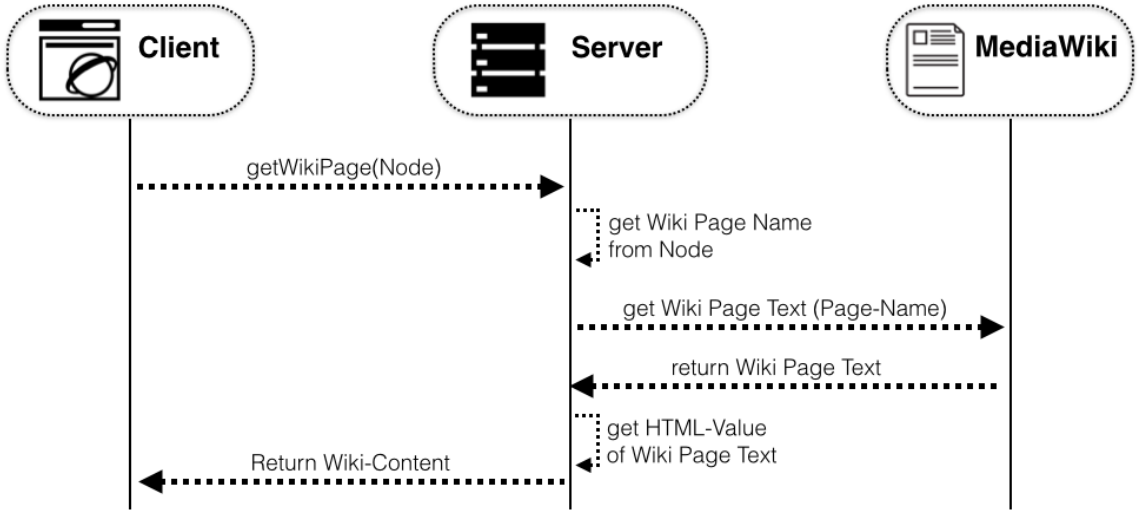


Abbildung 21: Beispielkommunikation Abfrage einer Wiki-Seite

Der Client beantragt beim Server die Wiki-Seite des jeweiligen Nodes. Nach Erhalt der Anfrage ermittelt der Server den Namen der dazugehörigen Wiki-Seite. Die API von MediaWiki liefert den Text mit Markup Language an den CDAR-Server. Dieser generiert mit der Antwort einen HTML formatierten Text, welcher im Client zur Anzeige kommt. Mit diesem überträgt der Server zudem auch den rohen Text mit Markup-Elementen, da dieser für die Bearbeitung des Eintrages im Client benötigt wird.
Die Generierung des HTML-Outputs im Server hat einen wesentlichen Vorteil. Würde MediaWiki den rohen Text sowie den HTML formatierten Text liefern, müsste der Wiki-Server zweimal angefragt werden.

7.4 Sicherheit und Session Handling

Sämtliche Funktionen der Applikation sind nur registrierten Benutzern zugänglich. Um zu erkennen, dass ein Benutzer angemeldet ist, sendet dieser bei jeder Abfrage an den Server seine User-Id sowie Accesstoken mit. Diese Daten überprüft der Server, falls die Id und der Accesstoken in der Datenbank gefunden wurden, ist die Abfrage gültig und wird bearbeitet. Andernfalls reagiert der Server mit einem entsprechenden HTTP-Statuscode.
Untenstehendes Diagramm visualisiert den Login-Prozess und die anschliessende Abfrage einer gesicherten Ressource:

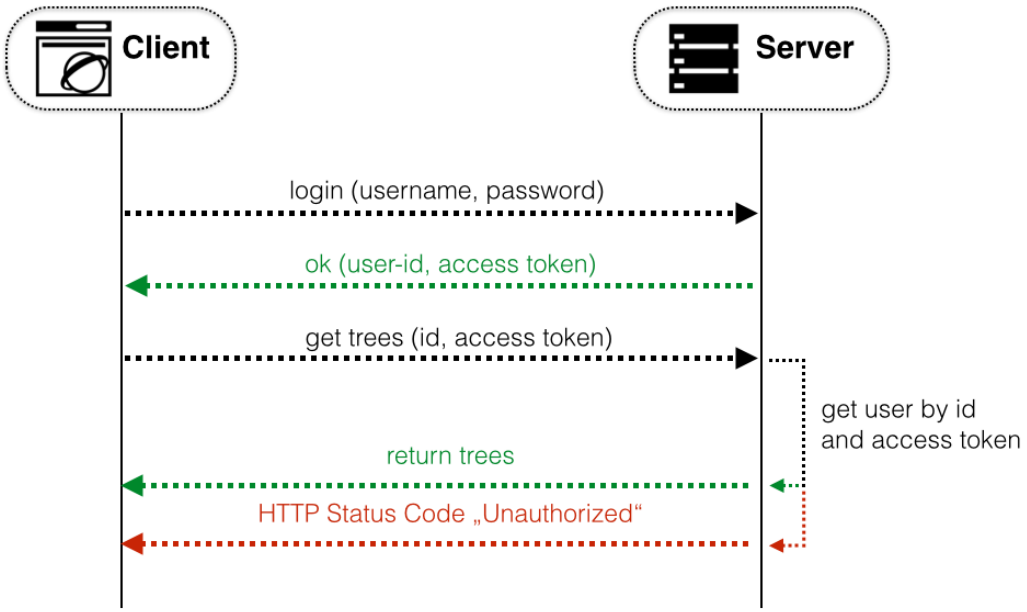


Abbildung 22: Kommunikation Sicherheit und Session Handling

7.4.1 MediaWiki Single-Sign-on

Möchte ein registrierter Benutzer eine Wiki-Seite im MediaWiki erstellen oder bearbeiten, muss er über einen Login im MediaWiki verfügen. Hierfür erstellten wir einen Single-Sign-On Mechanismus. Bei jeder Registrierung eines neuen Nutzers wird zeitgleich ein Benutzer im MediaWiki mit denselben Anmeldedaten erstellt.
Eine Registrierung in der Applikation ist auch möglich, wenn sich ein Benutzer schon vorgängig beim MediaWiki registriert hat. Hierfür muss er jedoch denselben Benutzernamen und Passwort bei der Registrierung angeben. Der Server versucht sich mit diesen Daten anzumelden, falls die Anmeldung gelingt, ist die Registrierung erfolgreich.

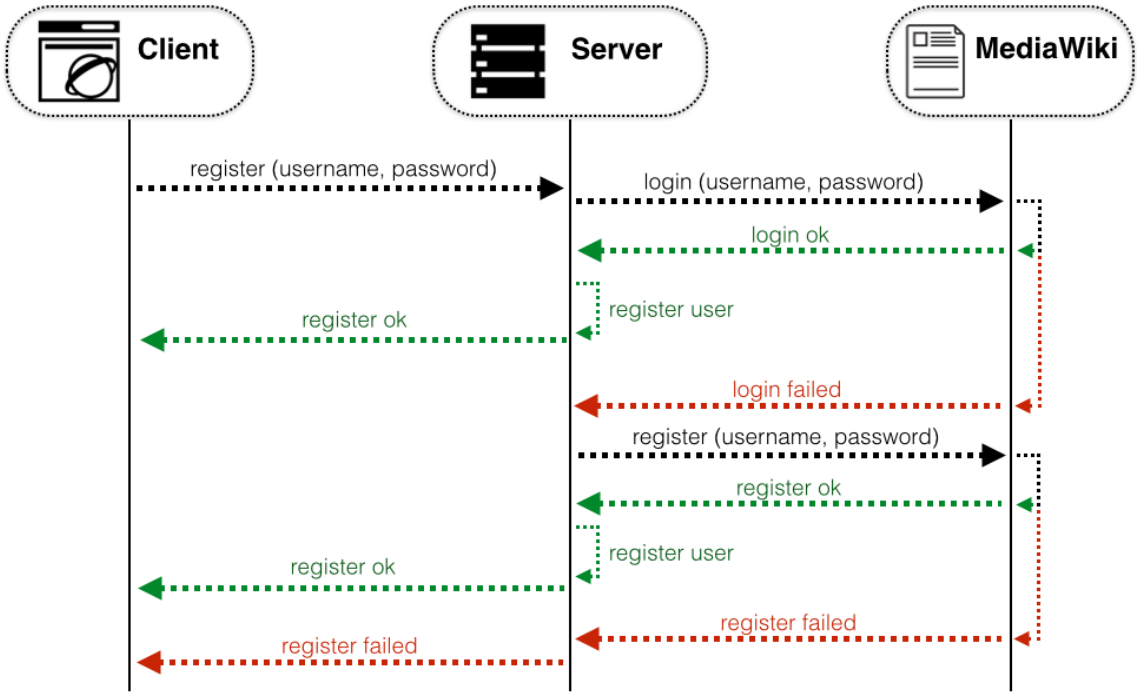


Abbildung 23: Kommunikation Single-Sing-On

Diese Implementierung hat zum Vorteil, dass sich der Benutzer nur bei der CDAR-Applikation registrieren muss und der ganze Anmeldeprozess von MediaWiki transparent erfolgt. Darüber hinaus ist eine Änderungskontrolle im MediaWiki sichtbar, da jeder Benutzer über einen eigenen Account verfügt. Leider bringt diese Implementierung auch einen wesentlichen Nachteil mit sich. Die Anmeldedaten müssen serverseitig im Klartext gespeichert werden. Dies aus dem Grund, dass sich der Benutzer auch unabhängig vom Entscheidungsführungswerkzeug in der MediaWiki-Umgebung anmelden können soll.

Lösungsvorschlag

Um das Problem der Übertragung und Speicherung des Passwortes im Klartext zu überwinden, könnten als erster Schritt die Anmelde- und Registrierungsdaten via TLS übertragen werden. Nach Erhalt der Daten im Server erfolgt eine Anmeldung beziehungsweise Registrierung mittels der Daten bei MediaWiki.

Das Kennwort kann anschliessend als Hashwert im Server gespeichert oder im Falle einer Anmeldung abgeglichen werden.

Damit diese Implementierung funktioniert, müsste gewährleistet werden, dass die Anmeldung an die CDAR-Applikation, welche die Anmeldung beim MediaWiki durchführt, während der ganzen Anmeldedauer des Benutzers erhalten bleibt. Sollte die Verbindung zu MediaWiki wegen Inaktivität des Benutzers abbrechen, müsste dieser auch in der CDAR-Applikation nach entsprechender Meldung abgemeldet werden.

Nachfolgendes Diagramm beschreibt einen erfolgreichen Versuch des Logins mit dem beschriebenen Lösungsvorschlag.

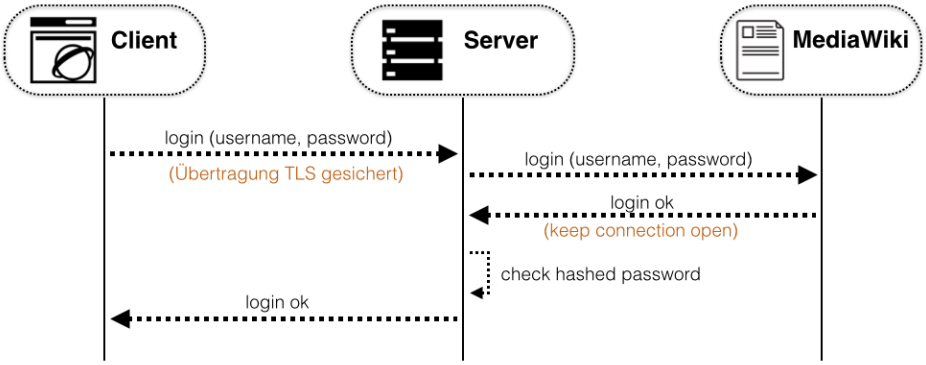


Abbildung 24: Lösungsvorschlag Single-Sign-On

7.5 Baum- und Graphenstrukturen

In der Applikation sind zwei Graphen unterschieden. Der eine ist mit einem Baum zu vergleichen, dies ist die Entscheidungsverwaltung, welche eine gruppierte Übersicht aller Wissenseinträge des Projektes darstellt. In der Entscheidungsverwaltung können keine Zyklen entstehen.
Der andere Graph hingegen stellt die Entscheidungsabhängigkeiten dar. In dieser Visualisierung können Zyklen entstehen. Diese sind zu vermeiden, werden jedoch durch die Applikation nicht verhindert.

7.5.1 Entscheidungsverwaltung

Die Entscheidungsverwaltung ist mit der JsTree Bibliothek umgesetzt. In der Verwaltung ist zwischen Ordnern und Entscheidungen zu unterscheiden. Diese sind auch in zwei verschiedenen Datenbanktabellen persistiert. Der jeweilige Ordner enthält die Identifikation des Elternordners. Entscheidungen hingegen haben eine zusätzliche Tabelle, in welcher dessen Identität und die Identität des Elternordners hinterlegt sind. Über diese Eigenschaften wird die Verzeichnisstruktur abgespeichert sowie auch aufgebaut.

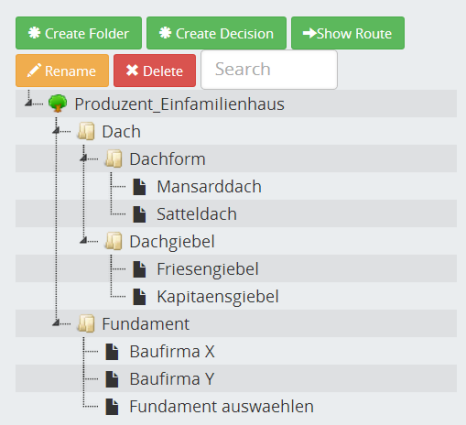


Abbildung 25: Entscheidungsverwaltung

7.5.2 Entscheidungsführungsgraphen

Der Entscheidungsführungsgraph enthält die Entscheidungen der Entscheidungsverwaltung. Die Knoten sind Verlinkungen der einzelnen Entscheidungen. Der Graph ist durch zwei Tabellen persistiert. Die eine wird für die Speicherung der Entscheidungen verwendet, die andere für die Verbindungen. Eine Verbindung besteht aus einem Start- und einem Zielknoten. Im Rahmen dieser Bachelorarbeit werden Zyklen nicht unterbunden, der Anwender ist verantwortlich, diese nicht zu erstellen.

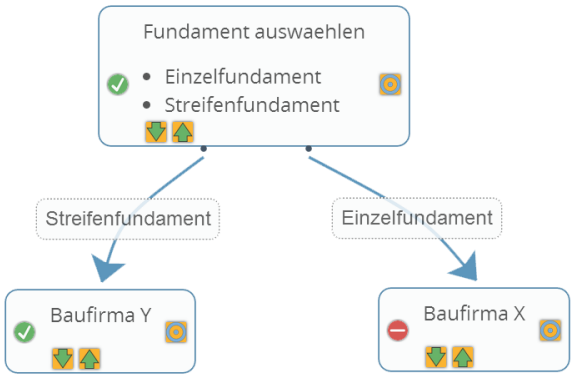


Abbildung 26: Entscheidungsführungsgraphen

Um die Startansicht des Entscheidungsführungsgraphen zu visualisieren, sucht die Applikation den Root-Knoten. Dazu sucht sie den Knoten, welcher keine ankommenden Verbindungen besitzt. Bestehen jedoch Zyklen, funktioniert diese Methode nicht. In diesem Fall wird jene Entscheidung als Root-Knoten bestimmt, welche zuerst erfasst und in den Entscheidungsführungsgraphen eingepflegt worden ist.

7.6 Aufbau der Serverapplikation

Layer	Funktion / Beschreibung
Presentation	Enthält die RESTful Schnittstelle, welche für die ganze Kommunikation zwischen Client und Server zuständig ist. Die Controller empfangen und versenden JSON-Objekte. Ein implementierter Security-Filter überprüft dabei die Authentifizierung aller Konsumenten bzw. Produzenten-Anfragen auf Id und den Accesstoken des Benutzers. Bei falschen Daten bricht der Filter die Abfrage ab.
Business Logic	Dieser Layer ist die Zwischenschicht vom Presentation Layer und dem Data Access Layer. Die Manager dieses Layers empfangen die Anweisungen und Objekte des oberen Layers, bearbeitet diese und sendet neue Befehle an den tiefergelegenen Data Access Layer. Dieser Layer besitzt ebenfalls die Entitäten, welche im Server verwendet werden. Ebenso werden die Entitäten in der REST-Kommunikation als JSON-Objekte genutzt. Komplexere Aufgaben, in welchen eine einzelne Klasse für die Logik nicht ausreicht, wie das Reporting und die Wiki-Verwaltung sind in eigenen Paketen abgelegt.
Data Access	Dieser Layer empfängt die Anweisung des oberen Layers und sendet die geforderten Objekte zurück. Der Data Access Layer kapselt sämtlichen Zugriff auf persistente Daten der CDAR-Applikation und Wiki-Seiten.

Tabelle 16: Layerbeschreibung

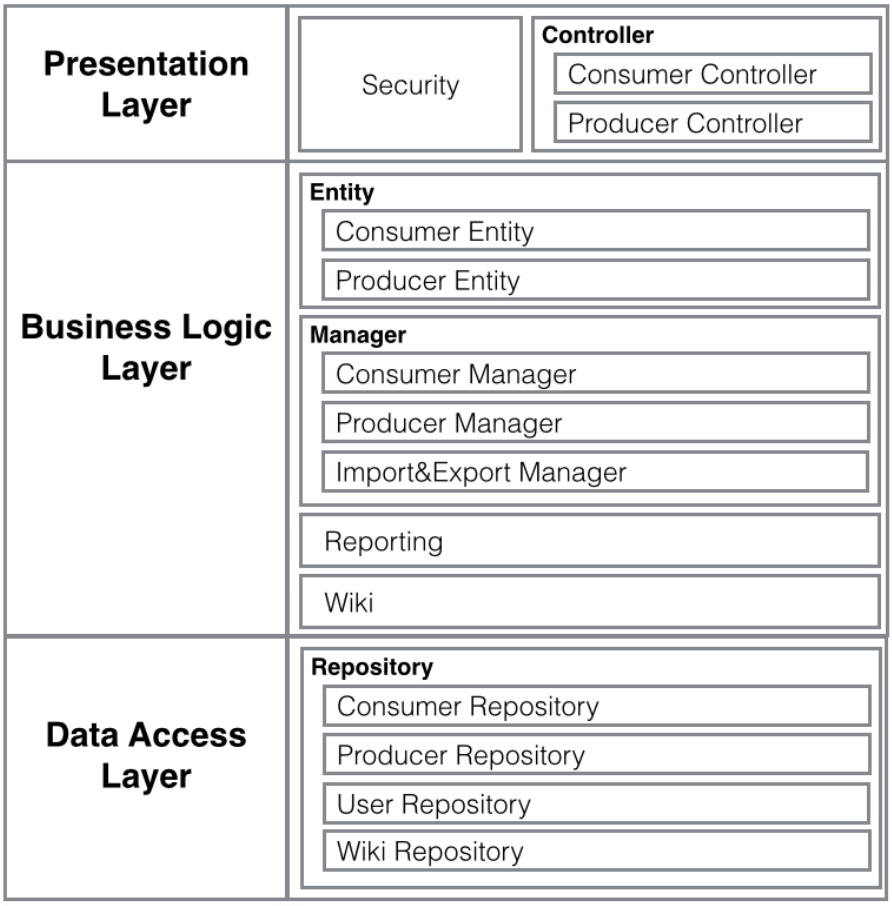


Abbildung 27: Layerübersicht Serverteil

7.7 Aufbau der Clientapplikation

Der CDAR-Webclient basiert auf dem AngularJS JavaScript-Framework und ist im Wesentlichen unterteilt in Views beziehungsweise HTML-Seiten, Services und Controllers. Ein Controller ist ein ViewModel kombiniert mit Logik. Er ermöglicht eine bidirektionale Datenverbindung zwischen einer View und dem Controller. Alle in der View genutzten Datenwerte werden durch den Controller angefordert, bearbeitet und gespeichert.
Durch Services können Logik und Bindungen an externe Ressourcen wie REST-Webservices implementiert werden. Diese sind als Singleton instanziiert und können den jeweiligen Controllern durch Dependency Injection hinzugefügt werden. Somit lässt sich eine Service-Implementierung in verschiedenen Controllern einbinden und nutzen.

Die nachfolgende Grafik zeigt die Struktur der Client-Applikation anhand der erstellten Views, Services und Controllers.

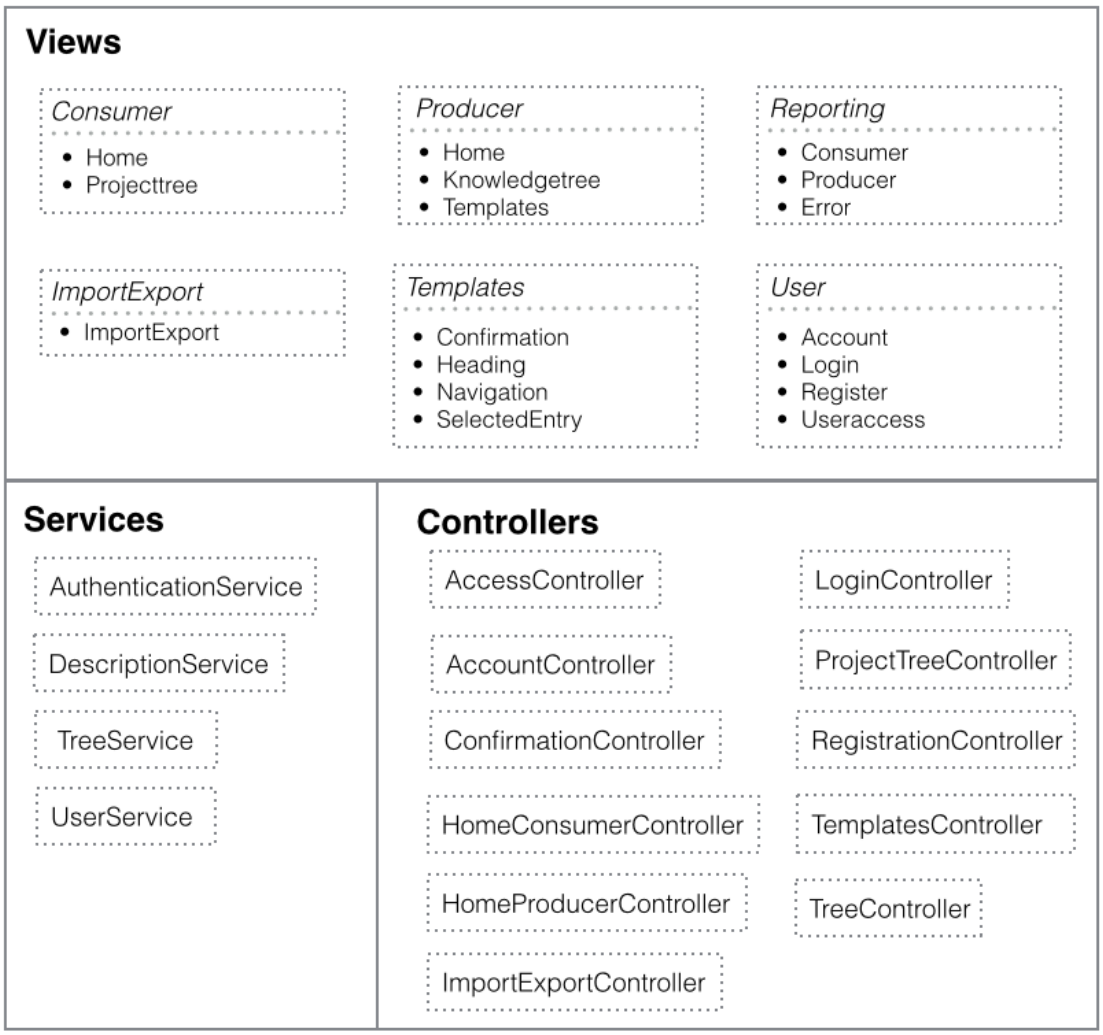


Abbildung 28: Aufbau des Clients

7.7.1 Routen in Zusammenhang mit Views und Controllers

Die Client Implementierung der CDAR-Applikation besteht aus einer Hauptseite, in welcher durch das ngRoute-Modul je nach URI-Aufruf die jeweiligen Views mit deren Controller nachgeladen werden. Dies hat den Vorteil, dass der Browser nur anfangs sämtliche JavaScript-Files laden muss, welche im Header dieser Hauptseite definiert wurden.

Nachfolgende Tabellen erläutern mögliche Routen mit deren Konfigurationen.

URI	Konfiguration und Beschreibung	
/login	View	Login
	Controller	LoginController
	Beschreibung	Benutzerlogin, sämtliche nicht validen Routen führen automatisch zum Login.
/registration	View	Register
	Controller	RegistrationController
	Beschreibung	Registrationsseite
/homeproducer	View	Producer/Home
	Controller	HomeProducer-Controller
	Beschreibung	Startseite mit einer Übersicht aller zugänglichen Projekten in der Rolle als Wissensproduzent
/homeconsumer	View	Consumer/Home
	Controller	HomeConsumerController
	Beschreibung	Startseite mit einer Übersicht aller zugänglichen Projekten als Wissens-konsument
/projecttree/ :treeId	View	Projecttree
	Controller	ProjectTreeController
	Beschreibung	Spezifische Ansicht eines Baumes in der Rolle als Wissenskonsument. Der Parameter :treeId steht als Platzhalter für eine jeweilige Baum-Id.

/projecttree/ :treeId/importexport	View	ImportExport
	Controller	ImportExport-Controller
	Beschreibung	Ansicht für Import und Export Funktionalitäten des Wissenskonsumenten. Der Parameter :treeId steht als Platzhalter für eine jeweilige Baum-Id.
/knowledgetree/ :treeId/importexport	View	ImportExport
	Controller	ImportExport-Controller
	Beschreibung	Ansicht für Import und Export Funktionalitäten des Wissensproduzenten. Der Parameter :treeId steht als Platzhalter für eine jeweilige Baum-Id.
/knowledgetree/ :treeId	View	Knowledgetree
	Controller	TreeController
		Spezifische Ansicht eines Baumes in der Rolle als Wissensproduzent. Der Parameter :treeId steht als Platzhalter für eine jeweilige Baum-Id.
/knowledgetree/ :treeId/templates	View	Templates
	Controller	Templates-Controller
	Beschreibung	Ansicht für die Bearbeitung und Erstellung von Wiki-Seiten Templates. Der Parameter :treeId steht als Platzhalter für eine jeweilige Baum-Id.
/knowledgetree/ :treeId/users	View	Useraccess
	Controller	AccessController
	Beschreibung	Ansicht um Benutzern den durch den Parameter :treeId definierten Baum freizugeben.
/projecttree/:treeId/users	View	Useraccess
	Controller	AccessController
	Beschreibung	Ansicht um Benutzern den durch den Parameter :treeId definierten Baum freizugeben.
/account	View	Account
	Controller	AccountController
	Beschreibung	Ansicht zur Bearbeitung des Benutzer-Passwortes sowie der Drill Hierarchie.

Tabelle 17: Zusammenhang Views und Controllers

7.7.2 Beschreibungen der Services

7.7.2.1 AuthenticationService

Der AuthenticationService behandelt sämtliche Kommunikation zwischen Client und Server betreffend der User-Ressource. Nachfolgende Tabelle zeigt sämtliche möglichen Funktionen des Services.

Methode	Beschreibung
addUser	Sendet mittels einer POST-Methode eine Anfrage an den Server. Beinhaltet ein Objekt mit Benutzername und Kennwort des zu registrierenden Benutzers.
updateUser	Methode um den Benutzer zu verändern. Im Rahmen dieser Bachelorarbeit lediglich zur Passwortänderung genutzt.
loginUser	Versucht den Benutzer beim Server anzumelden. Antwort enthält aktuellen Accesstoken bei erfolgreicher Anmeldung.
logoutUser	Löscht im Client den Accesstoken und die User-Identität im Cookie und leitet den Benutzer auf die Login-View weiter.

Tabelle 18: Authentification Service

7.7.2.2 DescriptionService

Beim Aufruf lädt der DescriptionService alle für den Client relevanten und konfigurierbaren Property-Values des Servers in den Cache. Der Service wird bei jedem Komplett-Refresh der Seite aufgerufen. Bei normalem Benutzerverhalten geschieht dies nur beim ersten Laden der Seite.

Value	Beschreibung und Beispiel
Directory	Name eines Verzeichnisses in der Entscheidungsverwaltung
Node	Name eines Wissenseintrages bzw. Nodes
Subnode	Name einer Option bzw. Subnode
Wikiurl	Wikiurl der MediaWiki-Installation
ExpandedLevel	Tiefe der Hierarchien, welche standardmässig in der Entscheidungsverwaltung offen sind
ConsumerDescription	Alternativer Name für den „Wissenskonsumenten“
ProducerDescription	Alternativer Name für den „Wissensproduzenten“

Tabelle 19: Description Service

7.7.2.3 TreeService

Der TreeService beinhaltet alle Abfragen, welche die RESTful-WebAPI zur Verfügung stellt, ausgenommen der Ressource User. Eine genaue Übersicht der Schnittstelle ist im Kapitel „8.1 Übersicht der Schnittstelle“ zu finden.

7.7.2.4 UserService

Der UserService bietet Methoden für den aktuell eingeloggten Benutzer an. Der Service dient als zentrale Schnittstelle der Browser-Cookies und der CDAR Client-Applikation. Er ermöglicht das Speichern und Auslesen des Cookies mit Benutzer-Identität und Accesstoken, welcher für alle gesicherten Abfragen an den Server benötigt werden. Darüber hinaus bietet der Service Methoden, um die aktuelle Benutzerrolle und Benutzerstatus abzufragen.

7.8 Entity Relationship Model der Datenbank

Der Abschnitt „4 Domainmodell der CDAR Applikation“ zeigt eine vereinfachte Version der Domäne und deren Entitäten. Aus der darauffolgenden Modellierung der Tabellen mit den zugehörigen Entitäten, Beziehungen und Zwischentabellen resultierte ein aufwendigeres Modell.

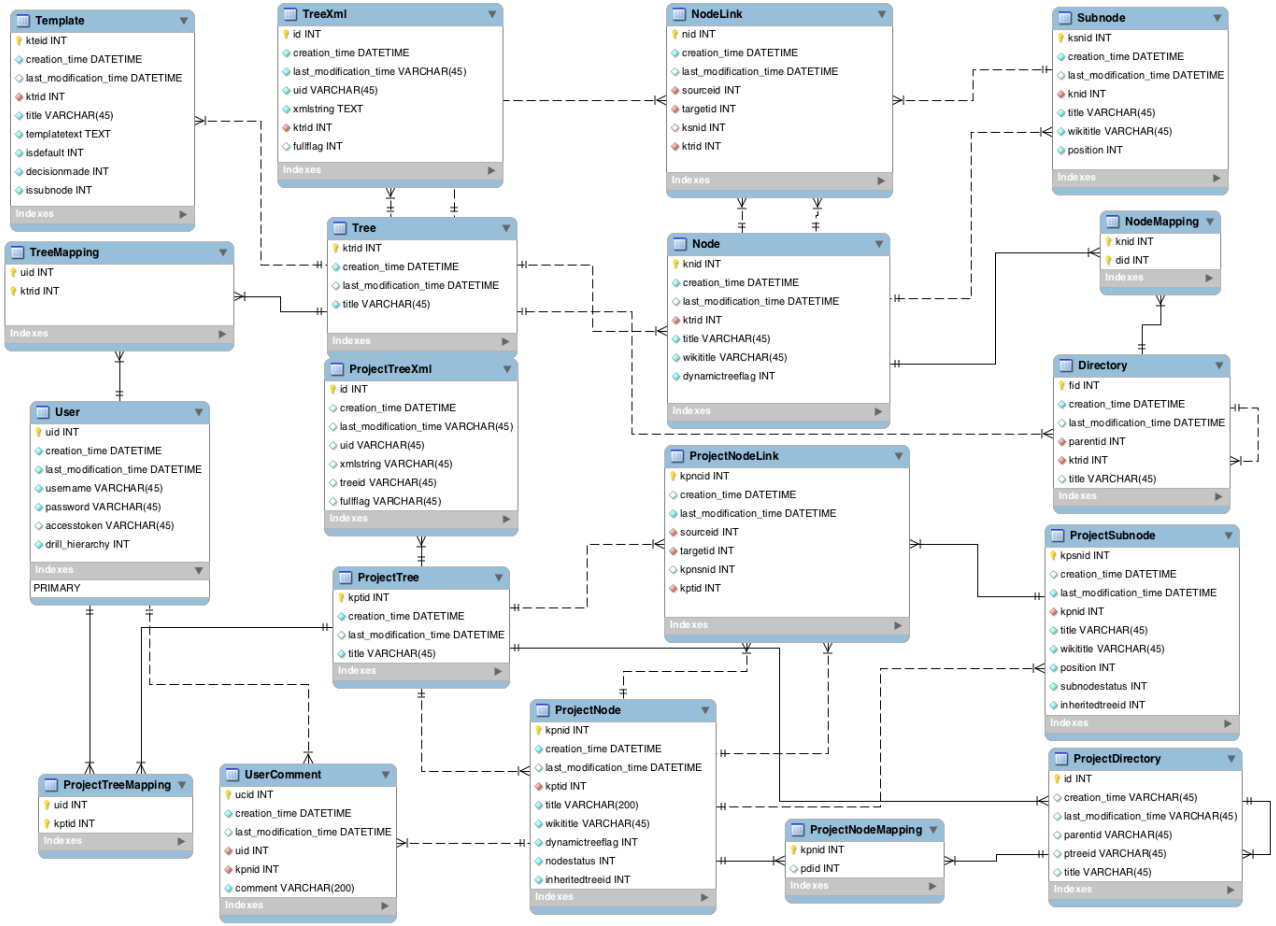


Tabelle 20: Entity Relationshop Model

7.8.1 Beispielinhalt der Datenbank

Nachfolgendes Beispiel zeigt die in der Datenbank persistierten Daten anhand des einfachen, aus der Einleitung bekannten Bau Architekten Musters.
Ein Mitarbeiter des Bau Architektur Büros hat das Beispiel, in der Rolle als Wissensproduzent, in die Software eingetragen und die Zusammenhänge modelliert. Das Projekt „Einfamilienhaus“ beinhaltet die drei Wissenseinträge „Fundament auswählen“, „Baufirma X“ und „Baufirma Y“. Diese sind in den Ordnern „Anfangsphase“ und „Baufirmen“ abgelegt.

Inhalt Entscheidungsverwaltung

Abbildung 29: Beispielinhalt Entscheidungsverwaltung

Inhalt Entscheidungsführungsgraphen

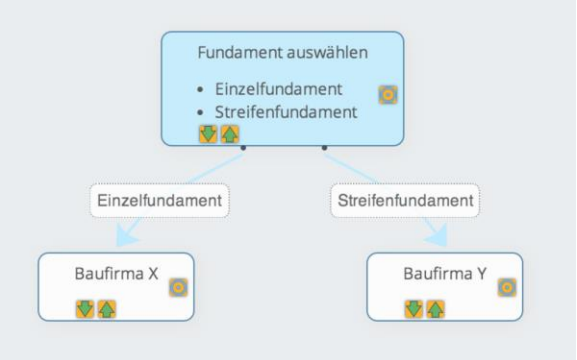


Abbildung 30: Beispielinhalt Entscheidungsführungs-graphen

Der Wissenseintrag „Fundament auswählen“ hat die zwei Optionen „Einzelfundament“ und „Streifenfundament“, welche je nach Auswahl zu einem anderen Wissenseintrag führen.

7.8.1.1 Tabelle User

Die in der CDAR-Applikation angelegten Benutzer werden in der Tabelle User abgespeichert.

Feld	Typ	Beispielinhalt und Beschreibung
Id	INT	1
		Primärschlüssel der Entität
creation_time	DATETIME	2014-05-28T11:52:19Z
		Datum und Zeit der Erstellung
last_modification_time	DATETIME	2014-06-05T14:59:14Z
		Datum und Zeit des letzten Aktualisierungszeitpunktes
username	VARCHAR(45)	BauArchitekt
		Name des Benutzers, selbiger Username kann auch für die Anmeldung des MediaWiki's benutzt werden
password	VARCHAR(45)	password
		Wert des Benutzerspasswortes im Klartext
accesstoken	VARCHAR(45)	f824b7166062f22e0ee791ad2ceb62a2fc174e45
		SHA1-Wert welcher zur Session-Überprüfung jeweils an den Server gesendet wird. Verändert sich bei jedem erfolgreichen Login.
drill_hierarchy	INT	5
		Integer-Wert der konfigurierbaren Hierarchie-Stufe des Entscheidungsführungsgraphen (Punkt H.2.4.2)

Tabelle 21: Datenbanktabelle User

7.8.1.2 Tabelle Tree

Die Tabelle „Tree" beinhaltet den Namen des jeweiligen Projektes eines Wissensproduzenten. Sämtliche zugehörigen Daten, wie beispielsweise Wissenseinträge und Baumstrukturen, referenzieren jeweils auf das zugehörige Projekt beziehungsweise diese Tabelle.

Feld	Typ	Beispielinhalt und Beschreibung
Id	INT	13
		Primärschlüssel der Entität
creation_time	DATETIME	2014-05-28T11:52:19Z
		Datum und Zeit der Erstellung
last_modification_time	DATETIME	2014-06-05T14:59:14Z
		Datum und Zeit des letzten Aktualisierungszeitpunktes
title	VARCHAR(45)	Einfamilienhaus
		Titel des Wissensprojektes

Tabelle 22: Datenbanktabelle Tree

7.8.1.3 Tabelle TreeMapping

Die Zwischentabelle TreeMapping dient zur Zuweisung von Bäumen an Benutzern. Es ist möglich, mehreren Benutzern denselben Baum zuzuweisen und ihn somit mit diesen zu teilen.

Feld	Typ	Beispielinhalt und Beschreibung
uid	INT	1
		Verweis auf einen Benutzer
ktrid	INT	13
		Verweis auf Tree-Tabelle

Tabelle 23: Datenbanktabelle TreeMapping

Um Doppeleinträge zu verhindern, sind beide Identitätsattribute dieser Tabelle Bestandteil eines zusammengesetzten Primärschlüssels.

7.8.1.4 Tabelle Directory

Die Verzeichnisse der Entscheidungsverwaltung sind in der Tabelle Directory abgelegt. Sämtliche Elternverzeichnisse sind mit dem Attribut „parentid“ verlink, lediglich das Hauptverzeichnis eines Baumprojektes, beinhaltet in diesem Feld den Wert „0“.

Feld	Typ	Beschreibung
Id	INT	Primärschlüssel der Entität
creation_time	DATETIME	Datum und Zeit der Erstellung
last_modification_time	DATETIME	Datum und Zeit des letzten Aktualisierungszeitpunktes
parentid	INT	Fremdschlüsselattribut, verweist auf das Vaterverzeichnis
ktrid	INT	Verweis auf den Baum, welcher das Verzeichnis beinhaltet.
title	VARCHAR(45)	Titel des Verzeichnisses

Tabelle 24: Datenbanktabelle Directory

Nachfolgende Tabellen enthalten die Daten aus dem beschriebenen Beispiel.

Feld	Wert
Id	21
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
parentid	0
ktrid	13
title	Einfamilienhaus

Tabelle 25: Datenbanktabelle Directory Beispielinhalt 1

Feld	Wert
Id	22
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
parentid	21
ktrid	13
title	Anfangsphase

Tabelle 26: Datenbanktabelle Directory Beispielinhalt 2

Feld	Wert
Id	23
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
parentid	22
ktrid	13
title	Baufirmen

Tabelle 27: Datenbanktabelle Directory Beispielinhalt 3

7.8.1.5 Tabelle Node

In der Tabelle „Node“ speichert der Server die angelegten Wissenseinträge eines Benutzers.

Feld	Typ	Beschreibung
Id	INT	Primärschlüssel der Entität
creation_time	DATETIME	Datum und Zeit der Erstellung
last_modification_time	DATETIME	Datum und Zeit des letzten Aktualisierungszeitpunktes
ktrid	INT	Verweis auf den Baum, welcher den Wissenseintrag beinhaltet.
title	VARCHAR(45)	Titel des Wissenseintrages
wikititle	VARCHAR(45)	Referenziert die jeweilige Wiki-Seite im MediaWiki
dynamictreeflag	INT	Boolsches Attribute, wenn Wert auf 1, ist Wissenseintrag im Entscheidungsführungsgraph sichtbar.

Tabelle 28: Datenbanktabelle Node

Nachfolgende Tabellen enthalten die Daten aus dem beschriebenen Beispiel.

Feld	Wert
Id	31
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
ktrid	13
title	Fundament auswählen
wikititle	NODE_31
dynamictreeflag	1

Tabelle 29: Datenbanktabelle Node Beispielinhalt 1

Feld	Wert
Id	32
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
ktrid	13
title	Baufirma X
wikititle	NODE_32
dynamictreeflag	1

Tabelle 30: Datenbanktabelle Node Beispielinhalt 2

Feld	Wert
Id	33
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
ktrid	13
title	Baufirma Y
wikititle	NODE_33
dynamictreeflag	1

Tabelle 31: Datenbanktabelle Node Beispielinhalt 3

7.8.1.6 Tabelle Subnode

Die einem Wissenseintrag hinzugefügten Options werden in der Tabelle Subnode gespeichert.

Feld	Typ	Beschreibung
Id	INT	Primärschlüssel der Entität
creation_time	DATETIME	Datum und Zeit der Erstellung
last_modification_time	DATETIME	Datum und Zeit des letzten Aktualisierungszeitpunktes
knid	INT	Fremdschlüsselattribut auf den zugehörigen Wissenseintrag bzw. Node
title	VARCHAR(45)	Titel des Subnode's bzw. der Option
wikititle	VARCHAR(45)	Referenziert die jeweilige Wiki-Seite im MediaWiki
position	INT	Position der Option in der Tabelle, um peristierte Sortierung zu ermöglichen.

Tabelle 32: Datenbanktabelle Subnode

Nachfolgende Tabellen enthalten die Daten aus dem beschriebenen Beispiel.

Feld	Wert
Id	41
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
knid	31
title	Einzelfundament
wikititle	SUBNODE_41
position	1

Tabelle 33: Datenbanktabelle Subnode Beispielinhalt 1

Feld	Wert
Id	42
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
knid	31
title	Streifenfundament
wikititle	SUBNODE_42
position	2

Tabelle 34: Datenbanktabelle Subnode Beispielinhalt 2

7.8.1.7 Tabelle NodeLink

Die Verbindungen zwischen den einzelnen Wissenseinträgen im Entscheidungsführungsgraphen werden in der Tabelle „NodeLink“ gespeichert.

Feld	Typ	Beschreibung
Id	INT	Primärschlüssel der Entität
creation_time	DATETIME	Datum und Zeit der Erstellung
last_modification_time	DATETIME	Datum und Zeit des letzten Aktualisierungszeitpunktes
sourceid	INT	Fremdschlüsselattribut auf den Start-Knoten der Verbindung
targetid	INT	Fremdschlüsselattribut auf den End-Knoten der Verbindung
ksnid	INT	Fremdschlüsselattribut auf den Subnode, welcher der Verbindung zugewiesen wurde. Enthält den Wert 0, falls keine Option selektiert wurde.
ktrid	INT	Verweis auf den Baum, welcher die Verbindung beinhaltet.

Tabelle 35: Datenbanktabelle NodeLink

Nachfolgende Tabellen enthalten die Daten aus dem beschriebenen Beispiel.

Feld	Wert
Id	51
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
sourceid	31
targetid	32
ksnid	41
ktrid	13

Tabelle 36: Datenbanktabelle NodeLink Beispielinhalt 1

Feld	Wert
Id	52
creation_time	2014-05-28T11:52:19Z
last_modification_time	2014-06-05T14:59:14Z
sourceid	31
targetid	33
ksnid	42
ktrid	13

Tabelle 37: Datenbanktabelle NodeLink Beispielinhalt 2

7.8.1.8 Tabelle NodeMapping

Die Tabelle NodeMapping beschreibt, in welchem Verzeichnis sich der jeweilige Wissenseintrag befindet.

Feld	Typ	Beschreibung
knid	INT	Verweis auf einen Wissenseintrag
did	INT	Verweis auf ein Verzeichnis

Tabelle 38: Datenbanktabelle NodeMapping

Nachfolgende Tabellen enthalten die Daten aus dem beschriebenen Beispiel.

Feld	Wert
knid	31
did	22

Tabelle 39: Datenbanktabelle NodeMapping Beispielinhalt 1

Feld	Wert
knid	32
did	23

Tabelle 40: Datenbanktabelle NodeMapping Beispielinhalt 2

Feld	Wert
knid	33
did	23

Tabelle 41: Datenbanktabelle NodeMapping Beispielinhalt 3

8 Beschreibung der RESTful Server WebAPI

Die Definition der Ressourcen und Methoden der HTTP-Api orientiert sich in erster Linie an den Regeln des Buches REST API Design Rulebook aus dem O'Reilly Verlag. Die Serviceimplementierung entspricht dem Level 2 des REST Maturity Model (REST-Reifegradmodell) von Leonard Richardson [Mar].
Die nachfolgende Tabelle zeigt sämtliche Levels des Reifegradmodells [Wik1].

Level	Eigenschaften
0	• verwendet XML-RPC oder SOAP • der Service wird über eine einzelne URI adressiert • verwendet eine einzelne HTTP-Methode (POST)
1	• verwendet verschiedene URIs und Ressourcen • verwendet eine einzelne HTTP-Methode (POST)
2	• verwendet verschiedene URIs und Ressourcen • verwendet mehrere HTTP-Verben
3	• basiert auf HATEOAS; verwendet Hypermedia für Navigation • verwendet verschiedene URIs und Ressourcen • verwendet mehrere HTTP-Verben

Tabelle 42: RESTful Level Beschreibungen

Auf die Anwendung von PUT und DELETE verzichtet die Implementierung bewusst. In vielen Firmen ist es üblich, dass diese Aufrufe durch Firewall Einschränkungen verhindert werden. Der Aufruf von nötigen Editier- sowie Lösch-Operationen erfolgen durch Funktions-deklarationen im Aufrufpfad der Ressource [Leo07].

8.1 Übersicht der Schnittstelle

Der Platzhalter {id} bezieht sich jeweils auf die vorhergehende Ressource. Bei den Date-Variablen erzeugt die Schnittstelle ein Datum im Format nach ISO 8601 [Mis]. Bei sämtlichen Update Aufrufen ist das Identitätsattribut im JSON Objekt optional, da die Schnittstelle das Feld des Objektes durch den angegebenen Wert im Aufrufpfad überschreibt.

8.1.1 User Ressourcen

Nachfolgende Beschreibungen betreffen die Ressource User. Sämtliche Aufrufe sind ohne vorheriges Einloggen aufrufbar und somit nicht geschützt.

GET	/users
	Liefert sämtliche Users
POST	/users
	Erstellt neuen Benutzer
POST	/users/{id}
	Bearbeitet Benutzer
GET	/users/login?username={name}?password={pw}
	Login des Benutzers
POST	/users/delete
	Löscht einen Benutzer

8.1.2 Property File Ressourcen

Die „Description“ Ressource liefert die im Property File konfigurierten Konstanten.

GET	/description
	Liefert sämtliche Konstanten des Property Files

8.1.3 Allgemeine Ressourcen

Die hier genannten Aufrufe existieren sowohl für Wissensproduzenten, als auch für Wissenskonsumenten. Der Platzhalter „[BAUM]“ kann im Falle des Wissensproduzenten durch „ktrees“ (Abk. für Knowledgetree) und im Falle des Wissenskonsumenten durch „ptrees“ (Abk. für Projecttree) ersetzt werden.

GET	/[BAUM]
	Liefert alle Wissensbäume
POST	/[BAUM]
	Erstellt neuen Wissensbaum
GET	/[BAUM]/{id}
	Liefert einen Wissensbaum
POST	/[BAUM]/{id}
	Bearbeitet Wissensbaum
GET	/[BAUM]/{id}/users
	Liefert sämtliche Benutzer mit ihren jeweiligen Rechten zum jeweiligen Baum
POST	/[BAUM]/{id}/users/{id}
	Ändert die Benutzerrechte eines Benutzers auf dem jeweiligen Baum
POST	/[BAUM]/delete
	Löscht spezifizierten Wissensbaum
GET	/[BAUM]/{id}/exports
	Liefert vorhandene exportierte Wissensbäume
POST	/[BAUM]/{id}/exports
	Erstellt neuen Wissensbaumexport
GET	/[BAUM]/{id}/exports/{id}
	Liefert exportierten Wissensbaum
POST	/[BAUM]/{id}/exports/{id}
	Bearbeitet einen Wissensbaumexport
GET	/[BAUM]/{id}/exports/{id}/set
	Importiert den exportierten Wissensbaum
POST	/[BAUM]/{id}/exports/delete
	Löscht exportierten Wissensbaum
GET	/[BAUM]/{id}/directories
	Liefert die Verzeichnisse des jeweiligen Baumes
POST	/[BAUM]/{id}/directories
	Erstellt neues Verzeichnis
GET	/[BAUM]/{id}/directories/{id}
	Liefert ein spezifisches Verzeichnis
POST	/[BAUM]/{id}/directories/{id}
	Bearbeitet Verzeichnis
POST	/[BAUM]/{id}/directories/delete
	Löscht ein Verzeichnis

GET	/[BAUM]/{id}/nodes
	Liefert sämtliche Wissensnodes des Baumes
POST	/[BAUM]/{id}/nodes
	Erstellt einen neuen Wissensnode
GET	/[BAUM]/{id}/nodes/{id}
	Liefert spezifischen Wissensnode
POST	/[BAUM]/{id}/nodes/{id}
	Bearbeitet einen Wissensnode
GET	/[BAUM]/{id}/nodes/{id}/wiki
	Liefert den Wiki-Eintrag des Wissensnodes
POST	/[BAUM]/{id}/nodes/{id}/wiki
	Bearbeitet den Wiki-Eintrag des Wissensnodes
POST	/[BAUM]/{id}/nodes/{id}/rename
	Bearbeitet lediglich den Titel von einem Node
POST	/[BAUM]/{id}/nodes/{id}/copy
	Kopiert einen Wissensnode mit Subnode und Wikieintrag
POST	/[BAUM]/{id}/nodes/delete
	Löscht einen Wissensnode
GET	/[BAUM]/{id}/nodes/{id}/drillup
	Baumtraversierung nach oben
GET	/[BAUM]/{id}/nodes/{id}/drilldown
	Baumtraversierung nach unten
GET	/[BAUM]/{id}/subnodes
	Liefert sämtliche Subnodes aller Nodes eines Baumes
GET	/[BAUM]/{id}/nodes/{id}/subnodes
	Liefert sämtliche Subnodes eines Nodes
POST	/[BAUM]/{id}/nodes/{id}/subnodes
	Fügt einem Node einen neuen Subnode hinzu
GET	/[BAUM]/{id}/nodes/{id}/subnodes/{id}
	Liefert einen Subnode
POST	/[BAUM]/{id}/nodes/{id}/subnodes/{id}
	Bearbeitet einen Subnode
POST	/[BAUM]/{id}/nodes/{id}/subnodes/delete
	Löscht einen Subnode
GET	/[BAUM]/{id}/nodes/{id}/subnodes/{id}/wiki
	Liefert den Wiki-Eintrag des Wissensnodes
POST	/[BAUM]/{id}/nodes/{id}/subnodes/{id}/wiki
	Bearbeitet den Wiki-Eintrag des Wissensnodes
GET	/[BAUM]/{id}/nodes/{id}/subnodes/{id}/drillup
	Baumtraversierung nach oben
GET	/[BAUM]/{id}/nodes/{id}/subnodes/{id}/drilldown
	Baumtraversierung nach oben

GET	/[BAUM]/{id}/links
	Liefert sämtliche Verbindungen eines Baumes
POST	/[BAUM]/{id}/links
	Erstellt eine neue Verbindung
POST	/[BAUM]/{id}/links/delete
	Löscht eine Verbindung
GET	/[BAUM]/{id}/links/nodes/{id}/drillup
	Baumtraversierung nach oben
GET	/[BAUM]/{id}/links/nodes{id}/drilldown
	Baumtraversierung nach unten

8.1.4 Wissensproduzent Ressourcen

Nachfolgend beschriebene Ressourcen stehen lediglich der Benutzerrolle „Wissensproduzent“ zur Verfügung.

POST	/ktrees/{id}/nodes/{id}/subnodes/{id}/rename
	Umbenennen eines Subnodes
GET	/ktrees/{id}/templates
	Liefert sämtliche Templates eines Baumes
POST	/ktrees/{id}/templates
	Erstellt neues Template
GET	/ktrees/{id}/templates/{id}
	Liefert spezifisches Template
POST	/ktrees/{id}/templates/{id}
	Bearbeitet ein Template
POST	/ktrees/{id}/templates/delete
	Löscht ein Template

8.1.5 Wissenskonsument Ressourcen

Nachfolgend beschriebene Ressourcen stehen lediglich der Benutzerrolle „Wissenskonsument“ zur Verfügung.

GET	/ptrees/{id}/nodes/{id}/comments
	Liefert alle Kommentare eines Wissensnodes
POST	/ptrees/{id}/nodes/{id}/comments
	Erstellt einen neuen Kommentar
GET	/ptrees/{id}/nodes/{id}/comments/{id}
	Liefert einen spezifischen Kommentar
POST	/ptrees/{id}/nodes/{id}/comments/{id}
	Bearbeitet einen Kommentar
POST	/ptrees/{id}/nodes/{id}/comments/delete
	Löscht einen Kommentar

8.2 HTTP Statuscodes

Die Beantwortung einer Anfrage an die CDAR WebAPI erfolgt durch einen HTTP-Statuscode. Während die erfolgreichen Antworten im Payload des Bodys die gewünschte Ressource enthalten, generiert der Server bei einer Anfrage welche eine Exception auslöst, mit einem Fehler-Statuscode [W3].

Statuscode	Beschreibung
200 OK	Anfrage war erfolgreich. Gewünschte Ressource im Payload der Antwort.
201 Created	Erstellung der Ressource war erfolgreich. Gewünschte Ressource im Payload der Antwort.
400 Bad Request	Gewünschte Anfrage konnte vom Server nicht beantwortet werden.
401 Unauthorized	Benutzer hat falsche oder keine Authentifizierungsdaten im Header der Anfrage mitgesendet.
409 Conflict	Benutzer hat eine Anfrage unter falscher Annahme gestellt. Das Projekt ist in Bearbeitung.

Tabelle 43: Statuscodes

8.3 Detailbeschreibung der Ressourcen

8.3.1 Property File Konstanten

Request URL	/descriptions	
Methods	GET : liefert eine Liste sämtlicher im Property File für den Client relevanten Beschreibungen	
JSON Response	[{ "directoryDescription" : String "nodeDescription" : String "subnodeDescription" : String "wikiUrl" : String "expandedLevel" : String }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.2 Benutzer

8.3.2.1 Benutzer erstellen

Request URL	/users	
Methods	POST : erstellt Benutzer	
JSON Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "username" : String "password" : String "accesstoken" : String "treeaccess" : bool "drillHierarchy" : int }]	
HTTP Status Code	**Code**	**Begründung**
	201	Erstellung der Ressource erfolgreich
	409	Benutzername existiert bereits

8.3.2.2 Benutzer editieren

Request URL	/users/{id}	
Methods	POST : editiert Benutzer	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"username" : String
	"password" : String
	"accesstoken" : String
	"treeaccess" : bool
	"drillHierarchy" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.2.3 Benutzer einloggen

Request URL	/users/login?username=“name“?passsword=“pw“	
Methods	GET : versucht Benutzer einzuloggen	

JSON Response	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"username" : String
	"password" : null
	"accesstoken" : String
	"treeaccess" : bool
	"drillHierarchy" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	401	Login nicht erfolgreich

8.3.2.4 Benutzer löschen

Request URL	/users/delete	
Methods	POST : löscht Benutzer	

JSON Request	{
	"id" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.3 Baum

8.3.3.1 Baum Erstellung und Abfrage

Request URL	/ktrees /ptrees	
Methods	GET : liefert Liste aller Bäume POST : erwartet einzelne Ressource	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "title" : String "uid" : int }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	201	Erstellung der Ressource erfolgreich
	400	Fehlerhafte Requestdaten

8.3.3.2 Spezifische Baumabfrage und Bearbeitung

Request URL	/ktrees/{id} /ptrees/{id}	
Methods	GET : liefert Baum POST : bearbeitet Baum	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"uid" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.3.3 Baum löschen

Request URL	/ktrees/delete /ptrees/delete	
Methods	POST : löscht Baum	

JSON Request	{
	"id" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.4 Export/Import

8.3.4.1 Export Erstellung und Abfrage

Request URL	/ktrees/{id}/export /ptrees/{id}/export	
Methods	GET : liefert sämtliche erstellten Exports POST : erwartet einzelne Ressource	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "title" : String "xmlString" : String "isFull" : bool }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	201	Erstellung der Ressource erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.4.2 Spezifischer Exportabfrage und Bearbeitung

Request URL	/ktrees/{id}/export/{id} /ptrees/{id}/export/{id}	
Methods	GET : liefert einen spezifischen Export POST : erwartet einzelne Ressource	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"xmlString" : String
	"isFull" : bool
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.4.3 Exportdaten in den Baum aufnehmen

Request URL	/ktrees/{id}/export/{id}/set /ptrees/{id}/export/{id}/set	
Methods	GET : importiert den jeweiligen Export in den Baum	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.4.4 Export löschen

Request URL	/ktrees/{id}/export/delete /ptrees/{id}/export/delete	
Methods	POST : löscht den spezifischen Export	

JSON Response/Request	{
	"id" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.5 Verzeichnisse

8.3.5.1 Verzeichnis Erstellung und Abfrage

Request URL	/ktrees/{id}/directories /ptrees/{id}/directores	
Methods	GET : liefert sämtliche Verzeichnisse POST : erstellt Verzeichnis	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "title" : String "treeId" : int "parentId" : int }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	201	Erstellung der Ressource erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.5.2 Spezifische Verzeichnisabfrage und Bearbeitung

Request URL	/ktrees/{id}/directories/{id} /ptrees/{id}/directores/{id}	
Methods	GET : liefert sämtliche Verzeichnisse POST : erstellt Verzeichnis	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"treeId" : int
	"parentId" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.5.3 Verzeichnis löschen

Request URL	/ktrees/{id}/directories/delete /ptrees/{id}/directories/delete	
Methods	POST : löscht Verzeichnis	

JSON Request	{
	"id" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.6 Nodes

8.3.6.1 Node Erstellung und Abfrage

Request URL	/ktrees/{id}/nodes /ptrees/{id}/nodes	
Methods	GET : liefert sämtliche Nodes POST : erstellt neuen Node	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "title" : String "wikititle" : String "treeId" : int "directoryId" : int "dynamicTreeFlag" : int "status" : int //nur in der Rolle als Konsument }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	201	Erstellung der Ressource erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.6.2 Spezifische Node Abfrage und Bearbeitung

Request URL	/ktrees/{id}/nodes/{id} /ptrees/{id}/nodes/{id}	
Methods	GET : liefert Node POST : bearbeitet Node	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"wikititle" : String
	"treeId" : int
	"directoryId" : int
	"dynamicTreeFlag" : int
	"status" : int //nur in der Rolle als Konsument
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.6.3 Node umbenennen

Request URL	/ktrees/{id}/nodes/{id}/rename	
Methods	POST : ändert Node Name	

JSON Request/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"wikititle" : String
	"treeId" : int
	"directoryId" : int
	"dynamicTreeFlag" : int
	"status" : int //nur in der Rolle als Konsument
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.6.4 Node kopieren

Request URL	/ktrees/{id}/nodes/{id}/copy	
Methods	POST : kopiert einen Node mit Subnode und Wikieintrag	

JSON Request/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"wikititle" : String
	"treeId" : int
	"directoryId" : int
	"dynamicTreeFlag" : int
	"status" : int //nur in der Rolle als Konsument
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.6.5 Wiki-Eintrag

Request URL	/ktrees/{id}/nodes/{id}/wiki /ptrees/{id}/nodes/{id}/wiki	
Methods	GET : liefert Wiki-Eintrag zum definierten Node POST : bearbeitet Wiki-Eintrag zum definierten Node	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"wikititle" : String
	"wikiContentPlain" : String
	"wikiContentHtml" : String
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.6.6 Node-Traversierung

Request URL	/ktrees/{id}/nodes/{id}/drillup /ptrees/{id}/nodes/{id}/drillup /ktrees/{id}/nodes/{id}/drilldown /ptrees/{id}/nodes/{id}/drilldown	
Methods	GET : liefert eine Liste von Kinds- bzw. Elternnodes ab definiertem Knoten	
JSON Response	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "title" : String "wikititle" : String "treeId" : int "directoryId" : int "dynamicTreeFlag" : int "status" : int //nur in der Rolle als Konsument }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.6.7 Node löschen

Request URL	/ktrees/{id}/nodes/delete /ptrees/{id}/nodes/delete	
Methods	POST : löscht den spezifischen Node	

JSON Request	{
	"id" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.7 Subnodes

8.3.7.1 Subnode Erstellung und Abfrage

Request URL	/ktrees/{id}/subnodes /ptrees/{id}/subnodes	
Methods	GET : liefert sämtliche Subnodes POST : erstellt neuen Subnode	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "title" : String "wikititle" : String "nodeId" : int "position" : int "status" : int //nur in der Rolle als Konsument }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	201	Erstellung der Ressource erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.7.2 Spezifische Subnode Abfrage und Bearbeitung

Request URL	/ktrees/{id}/nodes/{id}/subnodes/{id} /ptrees/{id}/nodes/{id}/subnodes/{id}	
Methods	GET : liefert Subnode POST : bearbeitet Subnode	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"wikititle" : String
	"nodeId" : int
	"position" : int
	"status" : int //nur in der Rolle als Konsument
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.7.3 Subnode umbenennen

Request URL	/ktrees/{id}/nodes/{id}/subnodes/rename	
Methods	POST : ändert Subnode Name	

JSON Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"wikititle" : String
	"nodeId" : int
	"position" : int
	}	

JSON Response	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "sourceId" : int "targetId" : int "treeId" : int "subnodeId" : int }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.7.4 Wiki-Eintrag

Request URL	/ktrees/{id}/nodes/{id}/subnodes/{id}/wiki /ptrees/{id}/nodes/{id}/subnodes/{id}/wiki	
Methods	GET : liefert Wiki-Eintrag zum definierten Subnode POST : bearbeitet Wiki-Eintrag zum definierten Subnode	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"title" : String
	"wikititle" : String
	"wikiContentPlain" : String
	"wikiContentHtml" : String
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.7.5 Subnode Traversierung

Request URL	/ktrees/{id}/nodes/{id}/subnodes/drillup /ptrees/{id}/nodes/{id}/subnodes/drillup /ktrees/{id}/nodes/{id}/subnodes/drilldown /ptrees/{id}/nodes/{id}/subnodes/drilldown	
Methods	GET : liefert eine Liste von Kinds- bzw. Elternsubnodes ab definiertem Knoten	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "title" : String "wikititle" : String "nodeId" : int "position" : int "status" : int //nur in der Rolle als Konsument }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.7.6 Subnode löschen

Request URL	/ktrees/{id}/nodes/{id}/subnodes/delete /ptrees/{id}/nodes/{id}/subnodes/delete	
Methods	POST	

JSON Request	{
	"id" : int
	}	

JSON Response	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "sourceId" : int "targetId" : int "treeId" : int "subnodeId" : int }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.8 Verbindungen

8.3.8.1 Verbindungserstellung und Abfrage

Request URL	/ktrees/{id}/links /ptrees/{id}/links	
Methods	GET : liefert sämtliche Verbindungen POST : erstellt neue Verbindung	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "sourceId" : int "targetId" : int "treeId" : int "subnodeId" : int }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	201	Erstellung der Ressource erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.8.2 Bearbeitung einer Verbindung

Request URL	/ktrees/{id}/links/{id} /ptrees/{id}/links/{id}	
Methods	POST : bearbeitet Verbindung	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"sourceId" : int
	"targetId" : int
	"treeId" : int
	"subnodeId" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.8.3 Verbindungstraversierung

Request URL	/ktrees/{id}/links/{id}/nodes/{id}/drillup /ptrees/{id}/links/{id}/nodes/{id}/drillup /ptrees/{id}/links/{id}/nodes/{id}/drilldown /ktrees/{id}/links/{id}/nodes/{id}/drilldown	
Methods	GET : liefert eine Liste von Kinds- bzw. Eltern-verbindungen ab definiertem Node	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "sourceId" : int "targetId" : int "treeId" : int "subnodeId" : int }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.8.4 Verbindung löschen

Request URL	/ktrees/{id}/links/delete /ptrees/{id}/links/delete	
Methods	POST : löscht eine Verbindung	

JSON Request	{
	"id" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten
	409	Lock auf Baum

8.3.9 Templates

8.3.9.1 Erstellung Template und Abfrage

Request URL	/ktrees/{id}/templates	
Methods	GET : liefert sämtliche Templates POST : erstellt neues Template	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "templatetext" : String "templatetexthtml" : String "isDefault" : bool "decisionMade" : bool "isSubnode" : bool }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	201	Erstellung der Ressource erfolgreich
	400	Fehlerhafte Requestdaten

8.3.9.2 Spezifische Abfrage eines Templates und Bearbeitung

Request URL	/ktrees/{id}/templates/{id}	
Methods	GET : liefert Template POST : bearbeitet Template	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"templatetext" : String
	"templatetexthtml" : String
	"isDefault" : bool
	"decisionMade" : bool
	"isSubnode" : bool
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.9.3 Template löschen

Request URL	/ktrees/{id}/templates/delete	
Methods	POST : löscht spezifisches Template	

JSON Request	{
	"id" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.10 Kommentare

8.3.10.1 Erstellung Kommentar und Abfrage

Request URL	/ptrees/{id}/comments	
Methods	GET : liefert sämtliche Kommentare POST : erstellt neuen Kommentar	
JSON Response/Request	[{ "id" : int "creationTime" : Date "lastModificationTime" : Date "userId" : int "nodeId" : int "comment" : String }]	
HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	201	Erstellung der Ressource erfolgreich
	400	Fehlerhafte Requestdaten

8.3.10.2 Spezifische Abfrage eines Kommentars und Bearbeitung

Request URL	/ptrees/{id}/comments/{id}	
Methods	GET : liefert Kommentar POST : bearbeitet Kommentar	

JSON Response/Request	{
	"id" : int
	"creationTime" : Date
	"lastModificationTime" : Date
	"userId" : int
	"nodeId" : int
	"comment" : String
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

8.3.10.3 Kommentar löschen

Request URL	/ptrees/{id}/comments/delete	
Methods	POST : löscht spezifischen Kommentar	

JSON Request	{
	"id" : int
	}	

HTTP Status Code	**Code**	**Begründung**
	200	Abfrage erfolgreich
	400	Fehlerhafte Requestdaten

9 Schlussfolgerung

9.1 Zusammenfassung

Die im Rahmen dieser Bachelorarbeit implementierten Funktionalitäten deckten alle geplanten UseCases ab. Der Einsatz von aktuellen Web Application Frameworks sichern die Wartbarkeit des Produktes. Die Verlinkung der Wiki-Seiten ermöglicht auch zusätzliche Software zur Integrierung von weiteren Wissensinhalten. Dies könnten beispielsweise Wikis mit bereits eingepflegten Informationen sein.
Die Inbetriebnahme und Nutzung des Wissensmanagementwerkzeuges ist gemäss einer Benutzeranleitung beschrieben.
Die Serverapplikation besteht aus einer Java 7 Webanwendung auf Basis von Tomcat 7. Der Rich Client basiert auf dem AngularJS JavaScript-Framework. Die Datenhaltung erfolgt in einer relationalen MySQL Datenbank sowie MediaWiki.

Umgesetzte geplante Funktionen
Benutzer erstellen
Wissensprojekte erstellen
Entscheidungsverwaltung erstellen, lesen, aktualisieren und löschen
Wissenseintrage einpflegen
Vorlage für Wissenseinträge erstellen und aktualisieren
Entscheidungsführungsgraph erstellen, lesen, aktualisieren und löschen
Entscheidung kommentieren
Zustand von Architekturentscheidungen festlegen
Inhalte exportieren und importieren
Reports erstellen

Umgesetzte zusätzliche Funktionen
Single Sign-on
Einträge kopieren (Deep-Copy)
Wissenseinträgen bereits bestehende Wiki-Seite zuweisen
Erstellte Projekte an andere Benutzer freigeben
Behandlung von konkurrierenden Zugriffen
Namensgebung konfigurierbar

Wenige Funktionen wie das Sortieren des Reports und die Statusgebung bei einer Verbindung im Entscheidungsführungsgraphen sind aufgrund veränderter Prioritäten nicht umgesetzt worden.

9.2 Ausblick

Das in dieser Bachelorarbeit entwickelte prototypisierte Wissensmanagementwerkzeug dient als Grundlage für Weiterentwicklungen. Bereits im Herbstsemester dieses Jahres sind Erweiterungen für die Applikation in Form einer weiteren Bachelorarbeit geplant.

Darüber hinaus haben wir im Laufe der Arbeit folgende Erweiterungen identifiziert:

- Verbesserung der Anwendungssicherheit, bspw. mächtigerer Zugriffskontroll-mechanismen
- Entwicklung eines Tablet- oder Mobile-Clients, unter Einsatz von Responsive-Design-Prinzipien
- Einbindung von weiteren Wiki-Engines, wie beispielsweise Confluence
- Die Applikation ist auf Instanz von Entscheidungen und Optionen aufgebaut, als Erweiterung könnten aber auch Refactorings mit Umbauaufgaben hinzugefügt und unterschieden werden.
- Einbindung eines UML-Tools wie Enterprise Architect, um die Architekturentscheidung nahe an andere Architekturdokumentation Artefakte zu bringen

Die Software wurde bewusst sehr generisch gehalten, dies ermöglicht den Einsatz auch in anderen Bereichen als der Informatik. Es ist geplant, den Prototypen Industriepartnern vorzustellen und ebenfalls in dessen Projekten zu integrieren.

A Qualitätssicherungsmassnahmen

A.1 Kennzahlen

Um die Codequalität besser zu beschreiben, beinhalten nachfolgende Abschnitte einige Kennzahlen, welche einen Überblick über die Applikation und ihre Schichten geben.

Übersicht

Kennzahl	Anzahl
Test-Cases	143
ELOC-Integrationtests	3287
Prepared-Statements	106
REST-Calls	61
ELOC-Server	9158
LOC-Client	5295

Tabelle 44: Kennzahlen

A.1.1 Anzahl Codezeilen

Um eine Übersicht auf den Projektumfang zu erlangen, zeigt folgende Grafik die Anzahl Codezeilen . Das ganze Projekt enthält 17‘740 Zeilen Code. Dabei sind die HTML-Files sowie CSS-Files nicht eingerechnet. Serverseitig beschreiben die Zahlen die effektiven Codezeilen (ELOC), clientseitig sind auch Kommentarzeilen mitgezählt

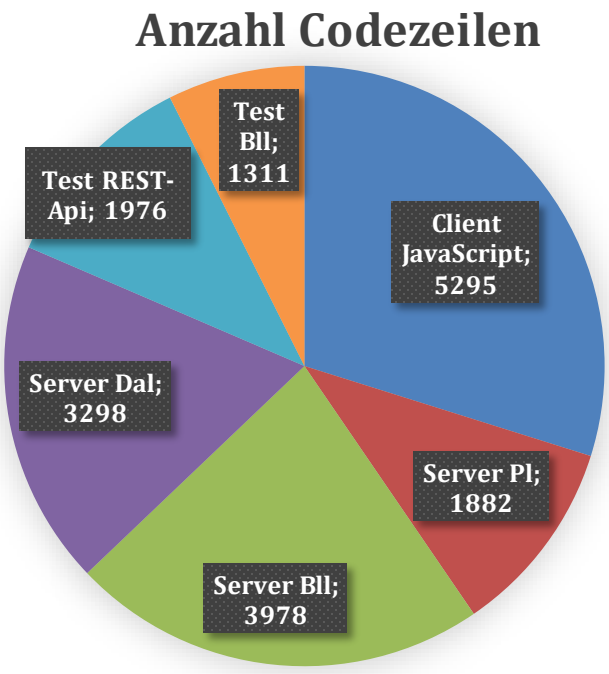


Abbildung 31: Anzahl Codezeilen

A.1.2 Aufteilung der Klassen der Server-Anwendung

Folgendes Diagramm zeigt die Verteilung sowie die Anzahl des CDAR-Servers. Insgesamt belaufen sich die Anzahl geschriebener Klassen auf 116.

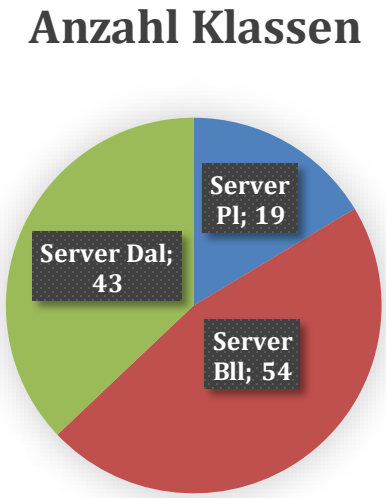


Abbildung 32: Anzahl Klassen

A.1.3 Anzahl Datenbank-Tabellen

Die Applikation verwendet für die Persistierung der Daten 19 Tabellen. Dabei sind jene, von MediaWiki verwaltete nicht eingerechnet.

A.1.4 Testing

Um stets eine funktionsfähige Applikation zu verwenden, nutzen wir mehrere Testverfahren.

A.1.4.1 Integration Test

Wir erstellten Integration Tests um die korrekte Funktionalität zu gewährleisten. Dabei fokussierten wir uns auf die REST-Schnittstelle sowie den Business Logic Layer.

A.1.4.2 Usability Test

Wir entschieden uns Client-Seitig kein Testframework zu verwenden. Um dennoch die Funktionalität und Benutzerfreundlichkeit zu überprüfen, setzten wir auf Usability Tests. Diese wurden vor allem im zweiten Teil der Projektzeit, in Zusammenarbeit mit dem Betreuer, durgeführt. Ergebnisse der Usability Tests ist im Folgekapitel „**Fehler! Verweisquelle konnte icht gefunden werden. Fehler! Verweisquelle konnte nicht gefunden werden.**“ zu finden.

A.1.4.3 Drone.io

Drone.io ist ein Continuous Integrations Service. Dabei testet die Plattform automatisch jeden Commit auf dessen Funktionalität mittels der Integration Tests. Um eine hohe Softwarequalität gewährleisten zu können, wurde dieser Service in der Entwicklungsphase der Applikation genutzt.

A.2 Usability Tests

In der Entwicklungphase wurden mehrfach Usability Tests durchgeführt, um die Benutzerfreundlichkeit und die korrekte Funktionalität der Applikation zu testen.

A.2.1 Test 1

Ziel: Erkennen von Benutzerschwierigkeiten, Bugs, Einschränkungen und Benutzererwartungen

Test setup:

- Firefox 29
- Durchtesten aller Use Cases

Erkenntnis:

- Entscheidungsverwaltung sollte bis zu 12 Hierarchien ausklappbar sein
- Aktuell noch keine Textvalidierung
- Bei grösseren Aktionen wie z.B. Löschen, keine Benutzerbestätigung
- Fehlermeldungen sind zu wenig spezifisch
- SQL-Injection möglich
- Doppelklick auf Knoten löscht den Knoten ist nicht Intuitiv
- Benutzer kann von der Registration nicht auf die Login-Seite zurück
- Aktuelle Benutzerrolle wird nicht angezeigt
- Kein Benutzerhinweis bei Passwortwechsel, dass MediaWiki-Passwort ebenfalls zu ändern ist.
- Namensgebung z.T. nicht passend
- Der Benutzer kann Optionen nicht effizient hinzufügen

Änderungen:

- Entscheidungsverwaltung breiter darstellen, zur Last des Entscheidungsführungsgraphen
- Textvalidierung einbauen
- Benutzerbestätigungen einbauen
- Fehlermeldungen spezifizieren
- SQL-Injection durch PreparedStatements vermeiden
- Doppelklick löscht zeigt nun dessen Optionen an. Löschen nur noch über expliziten Buttonclick möglich
- Bei Registration, Zurück-Button zum Login
- Anzeigen der Benutzerrolle auf jeder Seite beim rechten oberen Rand
- Hinweis bei Passwortwechsel hinzugefügt
- Anpassen der Namensgebungen
- Option können effizient per Eingabe und Enterbestätigung wiederholt hinzugefügt werden

A.2.2 Test 2

Ziel: Erkennen von Benutzerschwierigkeiten, Bugs, Einschränkungen und Benutzererwartungen

Test setup:

- Firefox 29
- Durchtesten aller Use Cases

Erkenntnis:

- Änderung an den Wiki-Seiten nimmt viel Zeit in Anspruch
- Konsument hat nicht die gleiche Funktionalität wie der Produzent
- Design der Tabs für Read/Write bei Wikifeld anpassen
- Drill-Aktion nicht nur über den Knoten ermöglichen
- Buttons oberhalb des Entscheidungsführungsgraphen irritieren
- Parallele Bearbeitung des gleichen Projektes kann zu Inkonsistenzen führen

Änderungen:

- Parallelisierung für die Bearbeitung des Wikis, sowie permanente Verbindung zu MediaWiki
- Konsumenten die gleiche Funktionalität wie dem Produzenten anbieten
- Tabs für Read/Write kleiner und anderes Design
- Drill-Aktion zusätzlich über Buttons möglich
- Buttons sind nun rechtsbündig
- Sperrung eines ganzen Projektes falls bereits ein Benutzer diesen bearbeitet

A.2.3 Test 3

Ziel: Erkennen von Benutzerschwierigkeiten, Bugs, Einschränkungen und Benutzererwartungen

Test setup:

- Durchtesten aller Use Cases
- Firefox 29.0.1

Erkenntnis:

- Umbenennen Top Level in der Projekverwaltung nicht möglich.
- Grammatik überprüfen
- Benutzer findet Reporting Feature zuerst nicht
- Full Import/Export erzeugt Fehlermeldung
- Im Entscheidungsführungsgraph kann keine Connection erzeugt werden
- Entscheidungsführungsgraph ist inkonsistent
- Timestamp bei den Kommentaren fehlt
- Refresh Button möglicherweise auf rechter Seite besser platziert
- Property-File Problem im Client, falls nicht geladen erscheint „undefined“
- Umbenennen mehreren Namesgebungen

Änderungen:

- Umbenennen nur in der Projekteverwaltung möglich
- Reporting nun in der Projektübersicht und in der Projekteverwaltung möglich
- Bugfix Export/Import
- Connection anpassen, dass auch in IE und Firefox möglich
- Inkonstanten bestehen weiterhin, nach Problemlösung bzw. Workaround suchen
- Design und Namesanpassungen

A.2.4 Usablity Test 4

Ziel: Erkennen von Benutzerschwierigkeiten, Bugs, Einschränkungen, Benutzererwartungen

Erkenntnis:

- Kopieren der Entscheidungen funktioniert, direkte Bearbeitung danach schlägt fehl
- Nach dem Kopieren kann der Knoten nicht in den Entscheidungsführungsgraphen gezogen werden
- Umbenennen von „Show Report“ und „Access Rights“
- Knowledge Consumer und Knowledge Producer sollte anpassbar sein

Änderungen:

- Erwähnen in der Benutzeranleitung, dass der Kopiervorgang einige Zeit in Anspruch nimmt und die kopierten Entscheidungen nicht direkt verändert werden sollen. Code ist bereits parallelisiert.
- Anpassungen beim Kopiervorgang (DynamicTreeFlag und Status werden auf Standard zurückgesetzt)
- Diverse Umbenennungen
- Knowledge Consumer und Knowledge Producer über Property File konfigurierbar

B Verzeichnis der Abbildungen und Tabellen

B.1 Abbildungsverzeichnis

Abbildung 1: Rollenaufteilung in der Applikation 12
Abbildung 2: Architekturübersicht 13
Abbildung 3: Ist-Zustand Bau Architekt 15
Abbildung 4: Projektablauf Einfamilienhaus 16
Abbildung 5: Einfamilienhaus mit CDAR-Applikation 17
Abbildung 6: Use Case Diagramm 24
Abbildung 7: Domainmodel 30
Abbildung 8: Softwarearchitektur 34
Abbildung 9: Mockup Login und Registrierung 35
Abbildung 10: Mockup Hauptseite Wissensproduzent 36
Abbildung 11: Mockup Baumansicht 37
Abbildung 12: Mockup Hauptseite Wissenskonsument 38
Abbildung 13: Mockup Projektansicht 39
Abbildung 14: Datenhaltung 45
Abbildung 15: Lockingmeldung 46
Abbildung 16: Kommunkation 47
Abbildung 17: Kommunikation Node erstellen 48
Abbildung 18: Kommunikation Entscheidung kopieren 49
Abbildung 19: Beispielkommunikation Entscheidung löschen 50
Abbildung 20: Beispielkommunikation Entscheidung Drag & Drop 51
Abbildung 21: Beispielkommunikation Abfrage einer Wiki-Seite 52
Abbildung 22: Kommunikation Sicherheit und Session Handling 53
Abbildung 23: Kommunikation Single-Sing-On 54
Abbildung 24: Lösungsvorschlag Single-Sign-On 55
Abbildung 25: Entscheidungsverwaltung 56
Abbildung 26: Entscheidungsführungsgraphen 57
Abbildung 27: Layerübersicht Serverteil 58
Abbildung 28: Aufbau des Clients 59
Abbildung 29: Beispielinhalt Entscheidungsverwaltung 65
Abbildung 30: Beispielinhalt Entscheidungsführungs-graphen 65
Abbildung 31: Anzahl Codezeilen I
Abbildung 32: Anzahl Klassen II
Abbildung 33: Projektaufwand XIX
Abbildung 34: Zeitaufwand pro Iteration XX
Abbildung 35: Registration XXVIII
Abbildung 36: Login XXVIII
Abbildung 37: Angemeldeter Rollenwechsel XXIX
Abbildung 38: Ändern des Passwortes XXX
Abbildung 39: Ebenen hierarchischer Navigation XXX
Abbildung 40: Projekteverwaltung Konsument XXXI
Abbildung 41: Projektübersicht Produzent XXXI
Abbildung 42: Templates XXXII

Abbildung 43: Entscheidungsvorlageverwaltung XXXII
Abbildung 44: Entscheidungsvorlage Wiki XXXII
Abbildung 45: Optionenvorlagenverwaltung XXXIII
Abbildung 46: Optionenvorlage Wiki XXXIII
Abbildung 47: Konsumentenvorlagenverwaltung XXXIV
Abbildung 48: Konsumentenvorlagen Wiki XXXIV
Abbildung 49: Import/Export XXXV
Abbildung 50: Zugriffskontrolle XXXV
Abbildung 51: Report XXXVI
Abbildung 52: Entscheidungsverwaltung XXXVII
Abbildung 53: Entscheidungsführungsgraphen XXXVIII
Abbildung 54: Knoten XXXIX
Abbildung 55: Popup XXXIX
Abbildung 56: Wiki-Eintrag für Entscheidungen XL
Abbildung 57: Optionen XLI
Abbildung 58: Wiki-Eintrag für Optionen XLI
Abbildung 59: Kommentare XLI
Abbildung 60: Status verändern XLII
Abbildung 61: Lockingmeldung XLIII

B.2 Tabellenverzeichnis

Tabelle 1: Beispiel eines Wissenseintrages 18
Tabelle 2: CDAR-Terminologie 19
Tabelle 3: Allgemeine Produkt Funktionen 20
Tabelle 4: Produkt Funktionen des Wissensproduzenten 21
Tabelle 5: Produkt Funktionen des Wissenskonsumenten 22
Tabelle 6: Weitere Anforderungen 28
Tabelle 7: Testumgebungssetup 29
Tabelle 8: Attribute Entscheidungsprojekt 31
Tabelle 9: Attribute Verzeichnis 31
Tabelle 10: Attribute Wissensnode 31
Tabelle 11: Attribute Wissenssubnode 32
Tabelle 12: Attribute Wissenslink 32
Tabelle 13: Beziehung der Klassen 33
Tabelle 14: Lizenzen Clientteil 44
Tabelle 15: Lizenzen Serverteil 44
Tabelle 16: Layerbeschreibung 58
Tabelle 17: Zusammenhang Views und Controllers 61
Tabelle 18: Authentification Service 62
Tabelle 19: Description Service 62
Tabelle 20: Entity Relationshop Model 64
Tabelle 21: Datenbanktabelle User 66
Tabelle 22: Datenbanktabelle Tree 66
Tabelle 23: Datenbanktabelle TreeMapping 67
Tabelle 24: Datenbanktabelle Directory 68
Tabelle 25: Datenbanktabelle Directory Beispielinhalt 1 68

Tabelle 26: Datenbanktabelle Directory Beispielinhalt 2 68
Tabelle 27: Datenbanktabelle Directory Beispielinhalt 3 68
Tabelle 28: Datenbanktabelle Node 69
Tabelle 29: Datenbanktabelle Node Beispielinhalt 1 69
Tabelle 30: Datenbanktabelle Node Beispielinhalt 2 69
Tabelle 31: Datenbanktabelle Node Beispielinhalt 3 69
Tabelle 32: Datenbanktabelle Subnode 70
Tabelle 33: Datenbanktabelle Subnode Beispielinhalt 1 70
Tabelle 34: Datenbanktabelle Subnode Beispielinhalt 2 70
Tabelle 35: Datenbanktabelle NodeLink 71
Tabelle 36: Datenbanktabelle NodeLink Beispielinhalt 1 71
Tabelle 37: Datenbanktabelle NodeLink Beispielinhalt 2 71
Tabelle 38: Datenbanktabelle NodeMapping 72
Tabelle 39: Datenbanktabelle NodeMapping Beispielinhalt 1 72
Tabelle 40: Datenbanktabelle NodeMapping Beispielinhalt 2 72
Tabelle 41: Datenbanktabelle NodeMapping Beispielinhalt 3 72
Tabelle 42: RESTful Level Beschreibungen 73
Tabelle 43: Statuscodes 78
Tabelle 44: Kennzahlen I
Tabelle 45: Projektiterationsbeschreibung XIII
Tabelle 46: Meilensteine XVII
Tabelle 47: Property-File Konfiguration XXVI
Tabelle 48: Softwareversionen XXVII
Tabelle 49: Statusübersicht XLII
Tabelle 50: Property-File XLIV

C Glossar

AngularJS
: AngularJS ist ein Open-Source-Framework zur Erstellung von browserbasierten Single-page-Anwendungen.

Apache Maven
: Apache Maven ist ein Build-Management-Tool basierend auf Java [Mav].

API
: Englisch für "application programming interface", wortwörtlich "Anwendungs-programmierschnittstelle" [Pro].

Balsamiq Mockups
: Balsamiq Mockups ist eine Software zum Erstellen von Mockups.

Bliki
: Die Java Wikipedia API (Bliki Engine) ist eine Parser Bibliothek für Java zur Umwandlung von Wikipedia Markup-Text Notation zu HTML [Gwt].

Bootstrap
: Bootstrap ist eine Sammlung von Hilfsmitteln zur Gestaltung von Webapplikationen [Boo].

CSS
: Cascading Style Sheets ist eine Anwendungssprache für Stilvorlagen von Dokumenten [Css].

ELOC
: Effective Lines of Code. Anzahl Codezeilen ohne Leerzeilen und Kommentare.

Genson
: Open Source Bibliothek für die Serialisierung von Java- zu JSON-Objekten und umgekehrt [Gen].

Git
: Git ist eine freie Software zur verteilten Versionsveraltung von Quellcode.

HTML
: Hypertext Markup Language

HTTP
: Hypertext Transfer Protocol

Jersey
: Java Framework für die Entwicklung von RESTful Web Services.

Joda-Time
: Erweiterung der Java *date* und *time* Klassen [Jod].

JQuery
: Jquery ist eine freie JavaScript-Library, welche Funktionen zur DOM-Navigation und -Manipulation zur Verfügung stellt [Jqu].

JSON
: Kompaktes Datenformat zum Zweck des Datenaustauschs zwischen Anwendungen [JSO].

JsPlumb
: JavaScript-Bibliothek zur Darstellung von interaktiven Graph Strukturen im Browser

JSTL
: Implementierung der Standard Tag Library für JavaServer Pages [JST].

JsTree
: JsTree ist ein JQuery Plugin zur Darstellung von interaktiven Baumdarstellungen im Browser.

JUnit
: Java-Framework für automatisiertes Testen von Applikationen [Jun].

Lock / Locking
: Englisch für Sperre, steht für eine Sperrung des Zugriffes bei paralleler Bearbeitung [Loc].

MediaWiki
: Freie Wiki-Engine welche beispielsweise auch von Wikipedia verwendet wird.

MySQL
: MySQL ist ein relationales Datenbankverwaltungssystem [Mys].

REST
: Representational State Transfer bezeichnet ein Programmierparadigma für Webanwendungen

RUP
: Rational Unified Process; Iteratives Projektvorgehen

Tomcat
: Web-Application Server für Java-basierte Applikationen

URI
: Uniform Resource Identifier

URL
: Uniform Resource Locator

X-editable
: X-editable ist eine Open Source JavaScript-Bibliothek zum Erstellen von editierbaren Elementen im Browser [Xed].

XML
: Extensible Markup Language

D Literatur

[Ang] *AngularJS, Google*. https://angularjs.org/,

[Apa] *Apache Tomcat*. http://tomcat.apache.org/,

[Bal] *Balsamiq*. http://balsamiq.com/products/mockups/,

[Boo] *Bootstrap*. http://getbootstrap.com/,

[Css] *Wikipedia*. http://de.wikipedia.org/wiki/CSS,

[Dro] *drone.io*. https://drone.io/,

[Gen] *Genson*. https://code.google.com/p/genson/,

[Git] *GitHub*. https://github.com/,

[Gwt] *Gwtwiki*. https://code.google.com/p/gwtwiki/,

[Ifs] IFS: *Architectural Refactoring for Cloud (ARC)*. http://www.ifs.hsr.ch/Architectural-Refactoring-for.12044.0.html?&L=4,

[Jav] *Java Wikipedia API (Bliki engine)*. https://code.google.com/p/gwtwiki/,

[Jer] *Jersey - RESTful Web Services in Java.* https://jersey.java.net/,

[Jod] *Joda Time*. http://www.joda.org/joda-time/,

[Jqu] *JQuery*. http://jquery.com/,

[JSO] *JSON*. http://de.wikipedia.org/wiki/JavaScript_Object_Notation,

[JsP] *jsPlumb, GitHub*. https://github.com/sporritt/jsPlumb,

[JsT] *jsTree*. http://www.jstree.com/,

[JST] *JSTL*. https://jstl.java.net/,

[Jun] *JUnit*. http://junit.org/,

[Leo07] Leonard Richardson S.R. (2007): RESTful Web Services., ISBN 978-0-596-52926-0. S. 251.

[Loc] *Wikipedia*. http://de.wikipedia.org/wiki/Lock,

[Mar] Masse M.: *REST API Design Rulebook*.

[Mav] *Maven*. http://maven.apache.org/,

[Med] *MediaWiki*. http://www.mediawiki.org/wiki/MediaWiki,

[Mis] Misha Wolf C.W.: *Date and Time Formats*. http://www.w3.org/TR/NOTE-datetime,

[Mys] *MySQL*. http://www.mysql.de/,

[Nel] Nelly Schuster O.Z. (2007): *Web 2.0 Collaboration System for Architectural Decision Engineering*. IBM Zurich Research Laboratory. http://design.inf.unisi.ch/sites/default/files/biblio/adkwik-seke2007.pdf,

[Ola] Zimmermann O.: *Architectural Refactoring - agile Umsetzung von Modernisierungsentscheidungen*. http://www.oop-konferenz.de/nc/oop2014/konferenz/konferenzprogramm/conference-detail/architectural-refactoring-agile-umsetzung-von-modernisierungsentscheidungen.html,

[Ola14] Zimmermann O.: *Link und AD-Beispiele*. zuletzt geprüft am 31.03.2014.

[Pro] *Wikipedia*. http://de.wikipedia.org/wiki/Programmierschnittstelle,

[The] *The JSON Data*. ecma. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf,

[W3] *W3*. http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html,

[Wik] *wiki-java*. https://code.google.com/p/wiki-java/,

[Wik1] *Wikipedia*. http://de.wikipedia.org/wiki/Representational_State_Transfer#REST_Maturity_Model,

[Xed] *GitHub*. http://vitalets.github.io/x-editable/,

E Projektplanung

E.1 Iterationsplanung

Die Iterationsplanung orientiert sich grob am Rational Unified Process (RUP) und ist unterteilt in eine *Analyse*-, *Realisierung*s- sowie *Auswertungsphase*.

Iteration	*Dauer*	*Beschreibung*
Analyse	3 Wochen	Projektsetup, Vorbereitungen & Planungen, Aufgabenstellung formulieren
Realisierung 1	2 Wochen	Grundstruktur (Client und Server) erstellen, Einrichtung Data Access Layer, Tests mit verschiedenen Baum Frameworks
Realisierung 2	2 Wochen	Drag & Drop Funktionalität, Grundfunktionalität der Baum-Libraries, Datenbankfunktionalität
Realisierung 3	2 Wochen	Template Funktion, Hinzufügen von Options-Funktionalität, erste Qualitätssicherungsmassnahmen
Realisierung 4	2 Wochen	Abschliessen der Hauptfunktionalitäten des Wissenskonsumenten
Realisierung 5	2 Wochen	Multi-User Funktionalität, Export/Inhalt Implementierung
Realisierung 6	2 Wochen	Allgemeine Fehlerbehebungen, Refactorings, Dokumentation, weitere User Tests
Auswertung	2 Wochen	Finalisierung der Dokumentation sowie Erstellung der HSR Artefakte.

Tabelle 45: Projektiterationsbeschreibung

E.2 Arbeitsplanung

Die Arbeitsplanung beinhaltet detailiertere Ausführungen der zu realisierenden Arbeitspakete, aufgeteilt in die jeweiligen Projektphasen.

E.2.1 Analyse

- Meetings
- Arbeitsplanung grob
- Arbeitsplanung fertigstellen
- Aufgabenstellung BA formulieren
- Dokumentation
- Reserve
- Entscheidung Technologie
- Meilenstein – Technologie Entscheidung
- **Meilenstein - Ende Analyse**

E.2.2 Realisierung

E.2.3 Iteration 1

- Entscheidungsverwaltung Dummy Daten
- Entscheidungsführungsgraphen Dummy Daten
- Einrichtung Datenbank (MySQL)
- Einrichtung Hibernate
- Grundstruktur AngularJS
- Dokumentation
- Reserve
- Meetings
- Meilenstein – Anzeige Testdaten im Client
- **Meilenstein – Ende Realisierung Iteration 1**

E.2.4 Iteration 2

- Baumansichten Drag&Drop
- Grundfunktionalität des Entscheidungsführungsgraphen
- Grundfunktionalität Entscheidungsverwaltung
- Authentifizierung
- Datenbankfunktionalität Hibernate
- Datenbankfunktionalität JDBC
- Qualitätssicherungsmassnahmen Data Access Layer
- Refactorings
- Zwischenpräsentation
- Node Anzeige mit Options (verschoben)
- Reserve
- Meetings
- Meilenstein – Erster Prototyp
- Meilenstein – Zwischenpräsentation
- Meilenstein – Erstes Codereview durch Betreuer
- **Meilenstein – Ende Realisierung Iteration 2**

E.2.5 Iteration 3

- Node Anzeige mit Options
- Einpflege Funktionalität für Options
- Template-Funktion
- Qualitätssicherungsmassnahmen Business Logic Layer
- Qualitätssicherungsmassnahmen REST-Api Calls
- Hauptfunktionalität Knowledge Producer
- Anzeige Entscheidungsführungsgraph: Teilbaum
- Reserve
- Meetings
- Meilenstein – Funktionalität Knowledge Producer
- Meilenstein - Teilbaumanzeige
- **Meilenstein – Ende Realisierung Iteration 3**

E.2.6 Iteration 4

- Hauptfunktionalität Knowledge Consumer
- Usability Tests
- Sample Daten einfügen
- Refactorings gemäss Codereview L. Wegmann und O. Zimmermann
- Dokumentation
- Reserve
- Meetings
- Meilenstein – Funktionalität Knowledge Producer
- **Meilenstein – Ende Realisierung Iteration 4**

E.2.7 Iteration 5

- Multi-User Funktionalität
- Export strukturierter Inhalt
- Import/Export Inhalt
- Reporting (HTML, PDF, XML)
- Reserve
- Meetings
- **Meilenstein – Ende Realisierung Iteration 5**

E.2.8 Iteration 6

- Refactoring
- Bug Fixing
- User Test und Feedback Implementierung
- Dokumentation
- Reserve
- Meetings
- Meilenstein – Ende Code-Refactoring
- Meilenstein – Ende Dokumentations-Inhalte
- **Meilenstein – Ende Realisierung Iteration 6**

E.2.9 Auswertung

- Abstract für Diplomarbeitsbroschüre an Betreuer (4.6.2014)
- A0-Poster zur Prüfung an Betreuer (6.6.2014)
- A0-Poster Fertigstellung (13.6.2014)
- Abgabe Bericht (13.6.2014)
- Präsentation Bachelorarbeiten (13.6.2014)
- Reserve
- Meetings
- **Meilenstein - Abgabe HSR Artefakte**
- **Meilenstein - Abgabe Bachelorarbeit**

E.3 Meilensteine

Meilenstein	*Termin*	*Beschreibung*
Ende Analyse	11.3.2014	Die Aufgabenstellung wurde gem. Auftrag klar definiert und die Projektinfrastruktur ist aufgesetzt. Erste Einarbeitungen wurden vorgenommen und eine grobe Projektplanung besteht.
Ende Realisierung Iteration 1	25.3.2014	Erste JavaScript Prototypen mit Baum-Libraries erstellt und mit Dummy-Daten gefüllt. Die Datenbank wurde angebunden und erste Tabellen integriert.
Ende Realisierung Iteration 2	8.4.2014	Die Grundfunktionalität der beiden Bäume funktioniert. Erste Nodes können im Client angezeigt werden. Eine einfache Abfrage durch alle Schichten bis in die Datenbank konnte erstellt werden.
Ende Realisierung Iteration 3	22.4.2014	Erste Qualitätssicherungsmassnahmen erfolgt. Die Template-Funktion wurde in die Applikation integriert.
Ende Realisierung Iteration 4	6.5.2014	Erste Usability-Tests mit Testusern durchgeführt. Refactorings und Änderungen nach Codereview wurden vorgenommen.
Ende Realisierung Iteration 5	20.5.2014	Eine Locking-Funktion zwecks Multi-User Support in die Applikation eingebaut. Reporting sowie Import- und Export-Funktionalität implementiert.
Ende Realisierung Iteration 6	3.6.2014	Die grössten Bugs und Smells wurden durch Refactorings und Bugfixes aus der Applikation entfernt. Dokumentationsaufbau sowie erste Kapitel wurden geschrieben.
Abgabe HSR Artefakte	13.6.2014	Das A0-Poster sowie die Kurzfassung der Studienarbeit sind dem Betreuer zugestellt.
Abgabe Bachelorarbeit	13.6.2014	Alle abzugebenden Artefakte sind dem Betreuer zugestellt worden.

Tabelle 46: Meilensteine

E.4 Meetings

Während der gesamten Projektdauer findet jeweils am Dienstag um 17 Uhr ein wöchentliches Statusmeeting statt. Die Sitzungen werden jeweils von einem Studenten geführt sowie von Beiden protokolliert.
Das Projektteam stellt die Agenda der jeweiligen Sitzung bis spätestens am vorangehenden Montagabend bereit. Das Sitzungsprotokoll wird dem Betreuer jeweils bis zum folgenden Mittwochabend zugestellt.
Zusätzlich können weitere Sitzungen durch den Betreuer oder die Studenten angefordert werden. Bei Klärung von Fachfragen, speziell JavaScript und AngularJS, kann Herr Lukas Wegmann den Sitzungen beigezogen werden.

E.5 Tools und Unterstützungssoftware

E.5.1 Projektverwaltung

Für die komplette Projektplanung, die Zeitrapportierung sowie das Issue-Management wird Redmine eingesetzt.
Der aktuelle Stand der Arbeiten am Projekt kann durch den Betreuer jederzeit eingesehen werden und wird vom Projektteam aktiv aktualisiert.

E.5.2 Entwicklungsumgebung

Zur Entwicklung der Applikation wurde Eclipse eingesetzt.

E.5.3 Versionsverwaltung

Zur Versionsverwaltung wurde ein öffentliches GitHub Repository erstellt.

E.5.4 Kontinuierliche Integration

Um eine hohe Softwarequalität gewährleisten zu können, wurde während der Arbeit auf Kontinuierliche Integration gesetzt. Durch kontinuierliche Ausführung der JUnit Integration Tests der CDAR-Applikation, waren fehlerhafte Implementierung nach jedem Commit sofort sichtbar.
Der Webdienst Drone.io ermöglicht kostenfreie Kontinuierliche Integration mit dem öffentlichen GitHub-Repository [Dro] [Git].

F Auswertung Zeitrapportierung

F.1 Projektaufwand

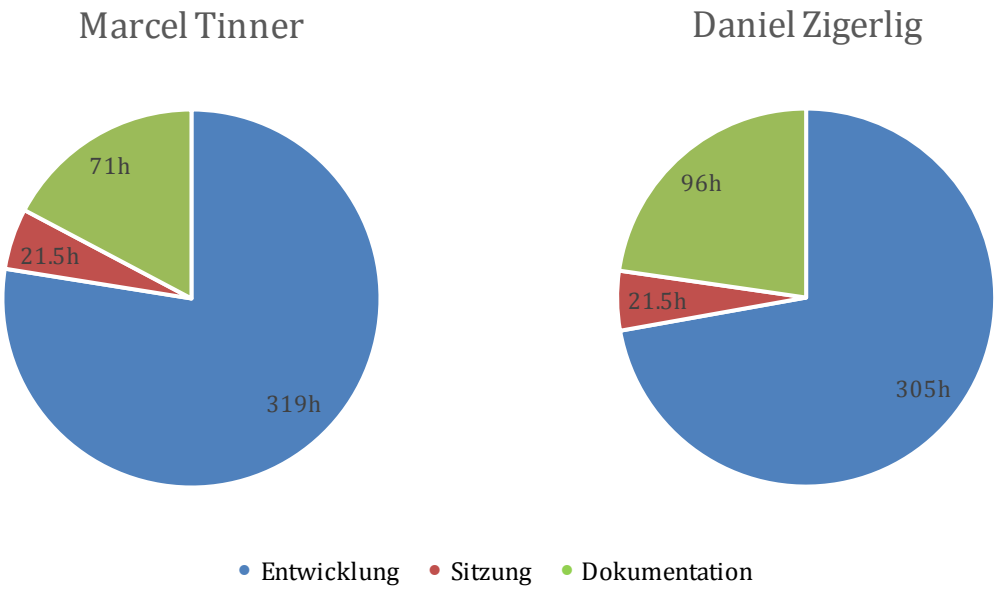


Abbildung 33: Projektaufwand

F.2 Zeitaufwand pro Iteration

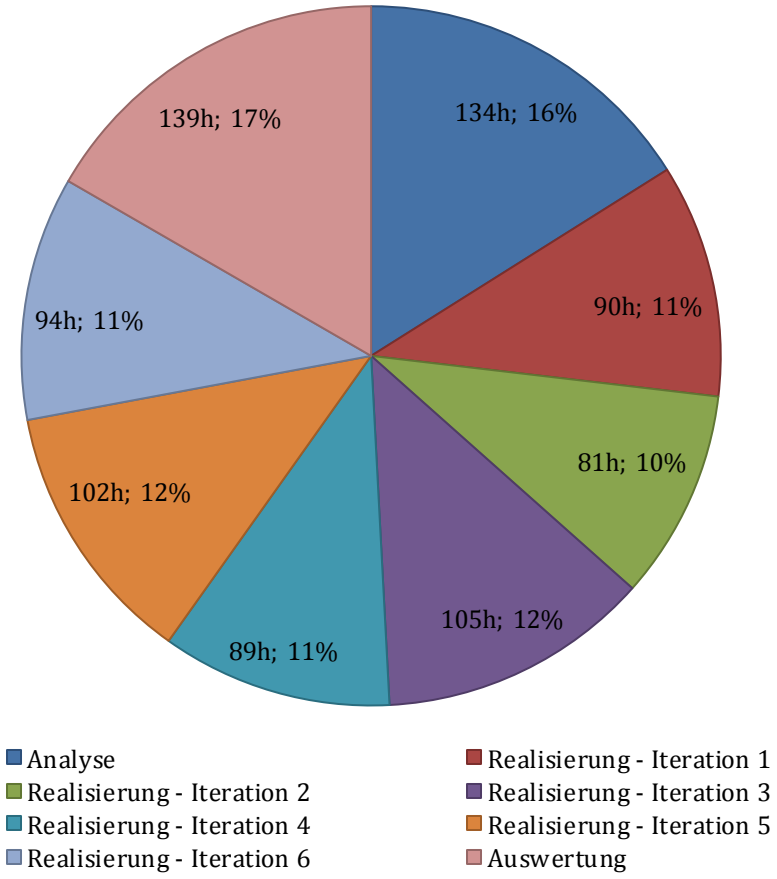


Abbildung 34: Zeitaufwand pro Iteration

G Aufgabenstellung

Die folgenden vier Seiten enthalten die offizielle Aufgabenstellung dieser Bachelorarbeit.

G.1 Scan Aufgabenstellung

Aufgabenstellung Bachelorarbeit
Marcel Tinner und Daniel Zigerlig

Collaborative Decision Management and Architectural Refactoring (CDAR) Tool

1. Auftraggeber und Betreuer

Diese Studienarbeit wird in Vorbereitung einer Zusammenarbeit mit Industriepartnern durchgeführt.

Betreuer HSR:

Prof. Dr. Olaf Zimmermann, Institut für Software ozimmerm@hsr.ch

Unterstützung Industriepartner:

Mehrere Firmen haben Interesse an einer Evaluation des CDAR-Tools geäussert; die konkreten Ansprechpartner werden im Projektverlauf durch den Betreuer benannt.

2. Ausgangslage

Die Bachelorarbeit unterstützt das IFS-Institutsprojekt ARC in folgendem Forschungskontext:

Capturing decision knowledge in text documents is bound to fail in the long run as document-oriented decision logs are hard to maintain and to process once they reach a certain size (say, 50-70 decisions captured). With such approach, sequential reading or full text searches are the main processing options. Decision documentation models promise to improve the situation, but also have to be created, organized and maintained over time.

Furthermore, textual decision logs are unable to steer the initial design and review work on a project. The existing decision capturing methods and tools do not make the need for decisions explicit in the design process; they only allow decision makers to record (log) their decisions once they have identified and made them on their own. As a consequence, the preparation for design workshops and reviews is time consuming and error prone, involving a lot of copy-paste and other manual content assembly work as well as tacit knowledge.

To overcome the limitations of current practices, novel concepts are needed:

1. How to model decisions so that a) they are useful for multiple projects and b) do not age fast due to technology evolution?

2. How to integrate decision reuse concepts into existing modeling and decision capturing tools in the light of technical and organizational diversity?

An example of a recurring design issue when moving a Web application to a cloud is session state management (e.g., think of a shopping session in an online store). The three top-level design options are Client Session state, Server Session State and Database Session State. Client Session State scales well, but has security and possibly performance problems; this does not change when moving to a cloud platform. Server Session State uses main memory or proprietary data stores in an application

server (e.g. HTTP session in JEE servlet container); this approach is no longer recommended when deploying in to a cloud due to scalability and reliability concerns. Finally, Database Session State is well supported in many clouds, e.g. via highly scalable key-value storage (a type of NoSQL database). This decision has to be made/revisited when taking any Web-based business application to the cloud; outcomes may vary, but issues and options stay the same (i.e., they recur).

3. Ziele und Aufgabenstellung

Das Ziel der Arbeit ist die Konzeption und Entwicklung eines webbasierten Wissensmanagement- und Entscheidungsführungstools insbesondere für die beiden Wissensinhalte Architekturentscheidungen und Architectural Refactorings. Die Benutzbarkeit des Tools soll im Context von Cloud Computing gezeigt werden (die Cloud-Wissensinhalte für die Evaluation werden vom Betreuer zugeliefert).

Die konkreten Aufgaben für die Bachelorarbeit sind:

- Auswahl einer geeigneten Toolpattform für die Entscheidungsmodellierung und -findung (z.B. Semantic Wiki Engine oder Enterprise Architect API; Visualisierungskomponenten) unter Berücksichtigung des Domain Models und Architecture Overview Diagrams aus dem Institutsprojekt (wird vom Betreuer zugeliefert).
- Design, Implementierung und Test des Tools auf Basis der gewählten Toolplattform (gemäss SE2-Standards).

Wichtige Erfolgsfaktoren für das Tool und damit Kriterien für die Arbeit sind:

- Gestaltung der Benutzeroberfläche und allgemeine Benutzerfreundlichkeit (Usability) bei a) Tool-Installation, b) Erstellen (Einpflegen) von Wissensinhalten (inkl. leichter Änderbarkeit, Bsp. Umbenennen von Wissensinhalten und Umhängen im Baum) sowie c) Projekt-Initialisierung der Wissensinhalte und laufende Entscheidungsführung und -dokumentation.
- Konzepte für statische und dynamische Beziehungen (Decision Dependencies) und deren Umsetzung im Tool.
- Generelle Wartbarkeit und Erweiterbarkeit, insbesondere Vorbereitungen für die zukünftige Unterstützung weiterer Typen von Wissensinhalten, Schichtenbildung und lose Kopplung bei der Integration externer Software (Bsp. Wiki-Engine, Datenbank).

4. Unterstützung

Die erwartete und effektiv erhaltene Unterstützung wird durch die Studenten in Sitzungsprotokollen definiert und im BA Bericht dokumentiert.

5. Zur Durchführung

Mit dem HSR-Betreuer finden in der Regel wöchentlich Besprechungen statt. Neben Herrn Zimmermann kann auf den IFS-Mitarbeiter Herrn Lukas Wegmann für technische Fragen zugegriffen werden. Zusätzliche Besprechungen sind nach Bedarf zu veranlassen.

Alle Besprechungen, bei denen eine Vorbereitung durch den Betreuer nötig ist, sind von den Studenten mit einer Traktandenliste vorzubereiten. Beschlüsse sind in einem Protokoll zu dokumentieren.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. Arbeitszeiten sind zu dokumentieren.

Die Spezifikation der Anforderungen geschieht durch die Studenten in Absprache mit dem Betreuer. Bei Disputen entscheidet der Betreuer in Rücksprache mit den Studenten über die definitiv für die Bachelorarbeit relevanten Anforderungen.

Vorstudie, Anforderungsdokumentation und Architekturdokumentation sollten im Laufe des Projektes mittels Milestone von dem Betreuer in einem stabilen Zustand abgenommen werden. Zu den abgegebenen Arbeitsresultaten wird ein vorläufiges Feedback abgegeben. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

6. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in zwei Exemplaren abzugeben. Bei der Projektdokumentation sind die Anleitungen des Studienganges inklusive Anhängen zu beachten, siehe https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html. Zudem ist eine kurze Projektresultatdokumentation für das Wiki von Prof. Zimmermann erwünscht (ggfs. kurzes Video).

7. Termine

Die generellen Termine sind der HSR-internen Webseite https://www.hsr.ch/Termine-Bachelor-und-Studiena.5142.0.html zu entnehmen. Dazu kommen die folgenden Termine:

- Der Abstract muss abweichend von den generellen Terminen bereits am 3.6. 2014 bereit zum Review sein.
- Die mündliche BA-Prüfung ist vorläufig für den 8. 8. 2014 geplant; die endgültige Terminbestätigung erfolgt, wenn der Gegenleser benannt ist.

Allfällige weitere Termine sind bereits per Email vom Sekretariat der Abteilung Informatik mitgeteilt worden bzw. sind dort zu erfragen.

Änderungen und Ergänzungen sollten in einem Sitzungsprotokoll dokumentiert werden.

8. Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS-Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Siehe auch Modulbeschreibung der Bachelorarbeiten, http://studien.hsr.ch/allModules/24809_M_BAI14.html.

Für die Beurteilung ist der HSR-Betreuer verantwortlich ggfs. unter Einbezug des Feedbacks der Tool-Evaluatoren.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/6
2. Berichte (Abstract, Mgmt Summary, techn. u. persönliche Berichte) sowie Gliederung, Darstellung und Sprache der gesamten Dokumentation.	1/6
3. Inhalt*)	3/6
4. Mündliche Prüfung zur Bachelorarbeit	1/6

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit festgelegt.

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Studienarbeiten.

Rapperswil, den 12. März 2014

Prof. Dr. Olaf Zimmermann
Institut für Software
Hochschule für Technik Rapperswil

H Benutzerhandbuch

Diese Benutzeranleitung bezieht sich auf das CDAR-Programm, welches während der Bachelorarbeit 2014 entstand. Sie beinhaltet zwei Hauptteile, eine Installationsanleitung und eine Anleitung für den Gebrauch der Applikation.

H.1 Installationsanleitung

Dieser Teil beschreibt die Installation der CDAR-Applikation und zeigt wichtige Informationen auf.

H.1.1 SQL-Script (MySQL)

Um die Applikation zu verwenden, ist das beiliegende SQL-Skript (SQLScript.sql) in der Datenbank auszuführen. Es enthält alle notwendigen Tabellen sowie deren Eigenschaften. Falls Tabellennamen umbenannt werden, sind diese auch in der Klasse „DBTableHelper“ anzupassen.

H.1.2 Installation MediaWiki

Da MediaWiki bereits gute Anleitungen und Informationen zu Installation und Betrieb der Software bereitstellt, verweisen wir an dieser Stelle auf diese.
Vorbedingungen: http://www.mediawiki.org/wiki/Manual:Installation_requirements
Download von MediaWiki unter: http://www.mediawiki.org/wiki/Download/de
Installation: http://www.mediawiki.org/wiki/Installation/de

H.1.3 Installation Tomcat

Tomcat erlaubt es, die Web-Anwendung auszuführen. Die Installation von Tomcat ist unter folgendem Link beschrieben.
Link: http://tomcat.apache.org/tomcat-7.0-doc/deployer-howto.html

H.1.4 Property-File

Im Property-File sind Konfigurationen hinterlegt. Diese sind bei folgenden Fällen anzupassen:

Fall	Betroffene Properties
Inbetriebnahme Applikation	• CDAR_LOCAL_DB_CONNECTION • CDAR_LOCAL_DB_USER • CDAR_LOCAL_DB_PASSWORD • CDAR_MEDIAWIKI_CONNECTION
Änderungen der Datenbankverbindung	• CDAR_LOCAL_DB_CONNECTION
Änderungen des Datenbankbenutzers	• CDAR_LOCAL_DB_USER • CDAR_LOCAL_DB_PASSWORD
Änderungen der MediaWiki-Verbindung	• CDAR_LOCAL_DB_CONNECTION

Tabelle 47: Property-File Konfiguration

H.1.5 Softwareversionen

Folgende Softwareversionen wurden beim Erstellen dieser Applikation verwendet.

Produkt	Version
Datenbank MySQL	5.6.14
MediaWiki	1.22.7
Tomcat	7.0.53

Tabelle 48: Softwareversionen

H.2 Betriebsanleitung

Dieses Kapitel zeigt, wie die Applikation zu verwenden ist, und beschreibt wichtige Eigenschaften sowie Einschränkungen. Die Applikation bietet zwei Benutzerrollen, welche sich in der Funktionalität teilweise nicht oder nur geringfügig unterscheiden. Deshalb kann es sein, dass das jeweilige Bildschirmfoto nicht der Rolle entspricht, in der Sie sich gerade befinden. Jedoch ist die erklärte Funktion in beiden Rollen vorhanden, ansonsten wird explizit darauf hingewiesen.

H.2.1 Applikationsaufruf

Die veröffentlichte Anwendung wird über „http://Domäne:Port/CDAR/cdarclient/#/login“ im Browser aufgerufen.

H.2.2 Registration und Login

Um die Applikation nutzen zu können, müssen Sie sich zuerst registrieren. Falls Sie sich bereits vorgängig im MediaWiki registriert haben und diesen Benutzernamen behalten wollen, registrieren Sie sich hier ebenfalls mit den gleichen Benutzerdaten wie im MediaWiki. Falls Sie jedoch noch kein Login bei MediaWiki verfügen, wird bei der Registration ein MediaWiki-Account für Sie erstellt. Falls die Registration nicht erfolgreich war, könnte es an den Richtlinien für die Registration liegen, dazu mehr im Kapitel H.2.2.1.

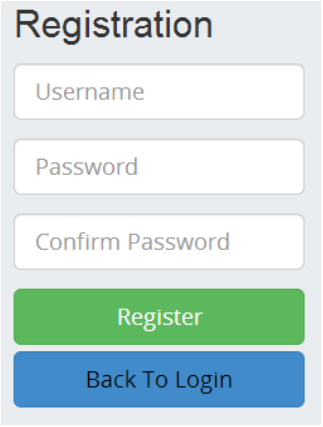


Abbildung 35: Registration

Nach der erfolgreichen Registration verfügt die Applikation nun über ihre Benutzerdaten und Sie können sich über unten stehende Ansicht anmelden.

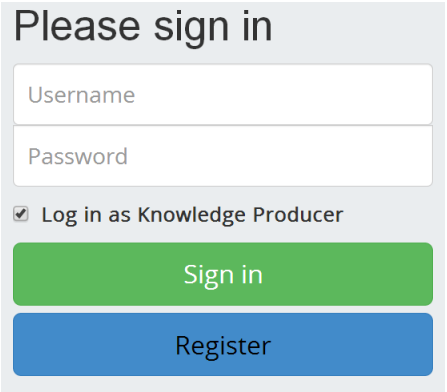


Abbildung 36: Login

H.2.2.1 Richtlinien

Folgende Richtlinien gelten für die Registration:

- Benutzername und Passwort müssen unterschiedlich sein
- Benutzername muss einmalig sein
- Benutzername oder Passwort dürfen nicht leer sein
- Benutzername darf nicht länger als 20 Zeichen sein
- Passwort darf nicht länger als 40 Zeichen lang sein

H.2.3 Benutzerrollen

In der Applikation gibt es zwei Benutzerrollen. Der Produzent fügt sein Wissen projektspezifisch anhand von Entscheidungen ein. Der Konsument hingegen kann vom Wissen des Produzenten profitieren und die Entscheidungen akzeptieren und übernehmen. Die Benutzerrollen können jederzeit gewechselt werden, ohne sich abmelden zu müssen. Dies können Sie über klicken auf „Current Role“ im rechten oberen Ecken vollziehen (Falls Sie den Firefox nutzen, lesen Sie bitte den Abschnitt „H.2.9 Kompatibilität und Einschränkungen“).

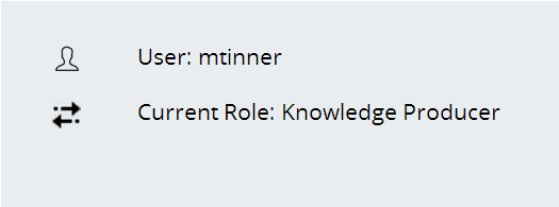


Abbildung 37: Angemeldeter Rollenwechsel

Alternativ können Sie dies auch bereits beim Anmelden vornehmen wie in Abbildung 36: Login zu sehen ist.

H.2.4 Account

Ihr Account können Sie nur bearbeiten, wenn Sie angemeldet sind. Dazu klicken Sie auf der rechten Seite auf „Account“.

H.2.4.1 Passwortwechsel

In folgendem Feld können Sie Ihr Passwort ändern. Dabei ist zu beachten, dass der Wechsel nur Ihr Passwort in der Applikation ändert, nicht aber das von MediaWiki. Um jenes von MediaWiki zu ändern, müssen Sie dies in den Einstellungen von MediaWiki vornehmen.
Es können Anmeldeschwierigkeiten entstehen, wenn die Passwörter Ihres Accounts unterschiedlich sind. Achten Sie deshalb darauf, dass wenn Sie das Passwort in der Applikation ändern, jenes von MediaWiki ebenfalls ändern. Beachten Sie ebenfalls die Richtlinien, welche für die gesamte Applikation gelten, diese sind im Kapitel H.2.2.1 zu finden.

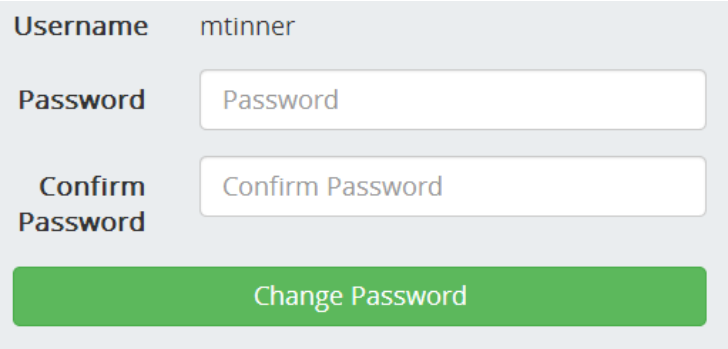


Abbildung 38: Ändern des Passwortes

H.2.4.2 Einstellen hierarchischer Navigation (Drill)

Über unten stehendes Feld kann die Standardtiefe der hierarchischen Navigation eingestellt werden. Die Tiefe sollte so gewählt werden, dass nicht mehr als 50 Entscheidungen gleichzeitig angezeigt werden. Mehr über die Navigation ist im Kapitel H.2.6.6.2 zu finden.

Abbildung 39: Ebenen hierarchischer Navigation

H.2.5 Projekteverwaltung

Es gibt zwei verschiedene Projekteverwaltungen. Eine für die Konsumenten-Projekte und eine für die Produzenten-Projekte. In die Ansicht gelangt man über die rechte Seite bei „Tree Overview“. Beim Konsumenten ist dabei zu beachten, dass bereits ein Produzenten-Projekt bestehen muss, ansonsten kann keine Duplizierung eines Masterprojektes gemacht werden. Dies entfällt beim Produzenten. Bei der Duplizierung entsteht eine gesamte Kopie des Masterprojektes. Anschliessende Änderungen am Masterprojekt bewirken keine Änderungen am Konsumentenprojekt. Ausgenommen ist dabei der Wiki-Eintrag, welche erst nach einer Statusänderung kopiert wird und dadurch unabhängig ist.

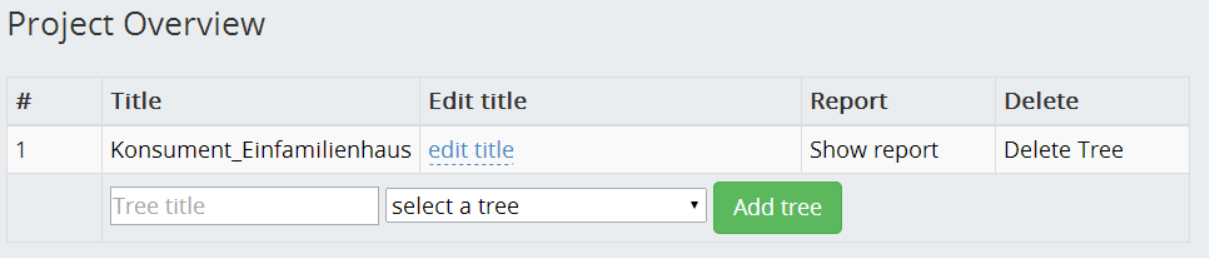


Abbildung 40: Projekteverwaltung Konsument

H.2.6 Projektübersicht

Sie gelangen in diese Ansicht, indem Sie auf ein erstelltes Projekt in der Projekteverwaltung klicken.
In der Projektübersicht können Sie das einzelne Projekt gestalten und verändern. Eine Funktionalität „Manage Templates“ steht nur dem Produzenten zur Verfügung, ansonsten besitzen beide Rollen dieselbe Funktionalität.

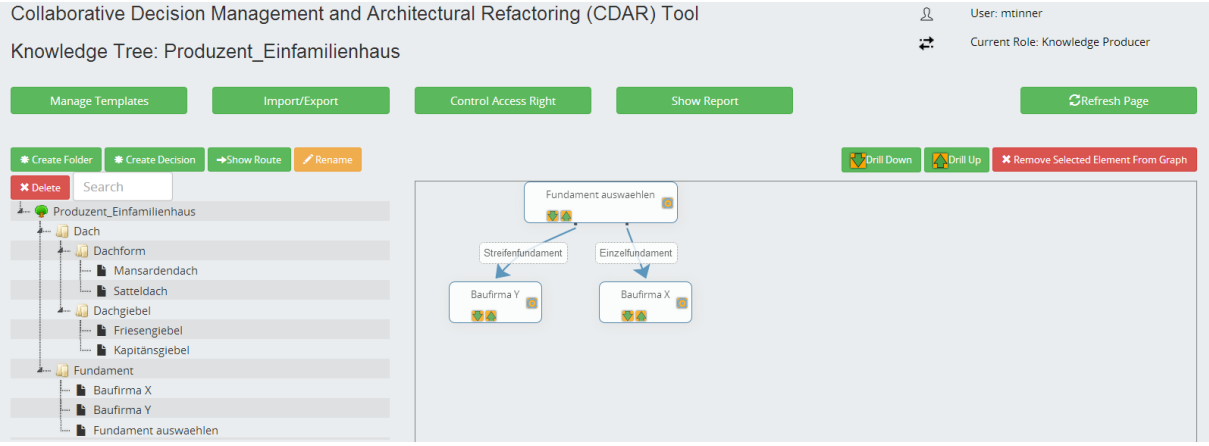


Abbildung 41: Projektübersicht Produzent

H.2.6.1Vorlagen

Diese Funktion steht nur dem Produzenten zur Auswahl. Sie gelangen in folgende Ansicht über einen Klick auf „Manage Templates“ in der Projektübersicht.

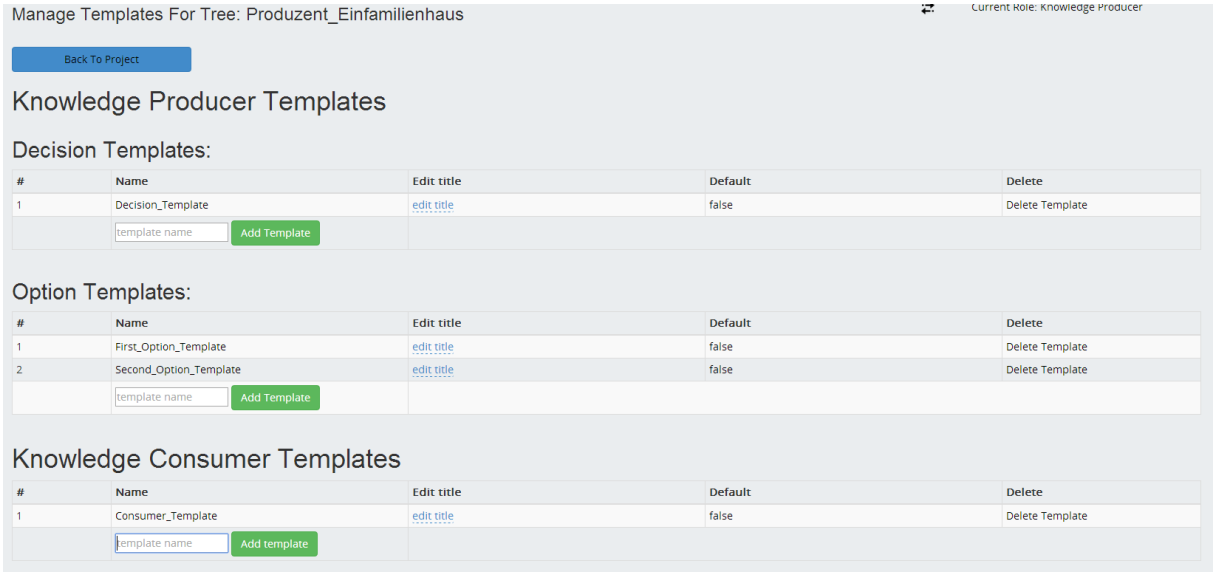


Abbildung 42: Templates

H.2.6.1.1 Entscheidungsvorlagen

Erstellen und Verwalten von Entscheidungsvorlagen:
Folgendes Bild zeigt den Abschnitt, in welchem Sie Entscheidungsvorlagen erstellen und verwalten können. Sie können jeweils nur eine erstellte Vorlage als Standardvorlage auswählen. Die ausgewählte Standardvorlage erscheint als Vorlage eines Wiki-Eintrages, welche Sie im Abschnitt „H.2.6.8 Optionen“ sehen. Den Wikitext, welchen Sie dort vorfinden, ist eine Kopie der Vorlage.

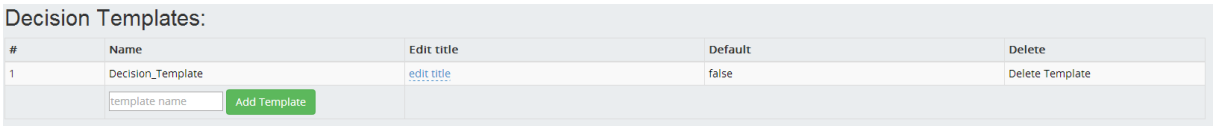


Abbildung 43: Entscheidungsvorlageverwaltung

Verfassen von Entscheidungsvorlagen:
Wenn Sie eine erstellte Vorlage selektieren, dann erscheint im unteren Fensterbereich der Wiki-Eintrag der Vorlage. Diesen können Sie über den Tab „Write“ editieren und danach speichern.

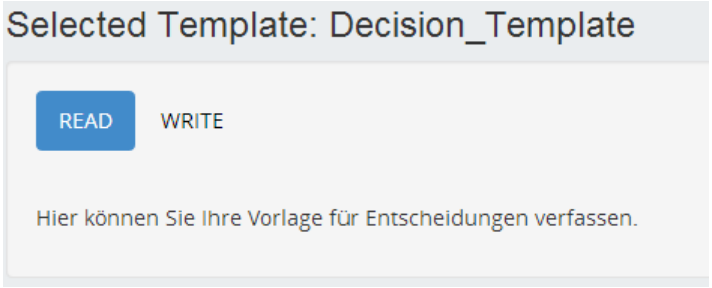


Abbildung 44: Entscheidungsvorlage Wiki

H.2.6.1.2 Optionsvorlagen

Erstellen und Verwalten von Optionsvorlagen:
Folgendes Bild zeigt den Abschnitt, in welchem Sie Vorlagen für die Optionen erstellen und verwalten können. Sie können jeweils nur eine erstellte Vorlage als Standardvorlage auswählen. Die ausgewählte Standardvorlage erscheint als Vorlage eines Wiki-Eintrages, welche Sie im Abschnitt „H.2.6.7 Verfassen von Entscheidungen“ sehen. Der Wiki-Eintrag, welchen Sie dort sehen, ist eine Kopie der Vorlage.

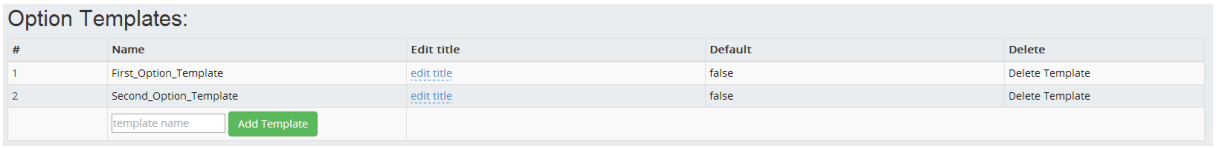


Abbildung 45: Optionenvorlagenverwaltung

Verfassen von Optionsvorlagen:
Wenn Sie eine erstellte Vorlage selektieren, dann erscheint im unteren Fensterbereich der Wiki-Eintrag der Vorlage. Diesen können Sie über den Tab „Write“ editieren und danach speichern.

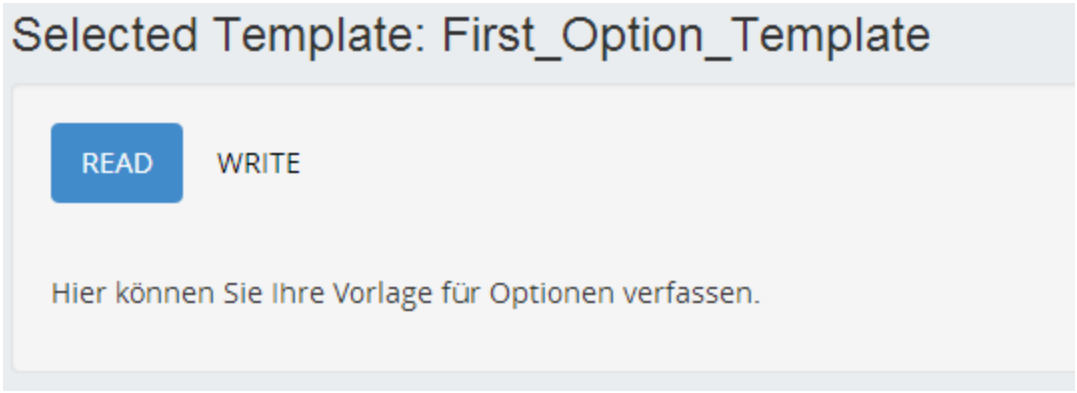


Abbildung 46: Optionenvorlage Wiki

H.2.6.1.3 Konsumentenvorlagen

Erstellen und Verwalten von Optionsvorlagen:
Folgendes Bild zeigt den Abschnitt, in welchem Sie Vorlagen für die Optionen erstellen und verwalten können. Sie können jeweils nur eine erstellte Vorlage als Standardvorlage auswählen. Die ausgewählte Standardvorlage erscheint als Vorlage für den Wiki-Eintrag, welche im Abschnitt „H.2.6.10 Status“ zu sehen ist. Der Wiki-Eintrag welchen Sie dort sehen ist eine Kopie der Vorlage und dessen Eintrag, welcher der Produzent für die Entscheidung verfasst hat. Der Vorlagetext erscheint erst, wenn Sie den Status verändert haben.

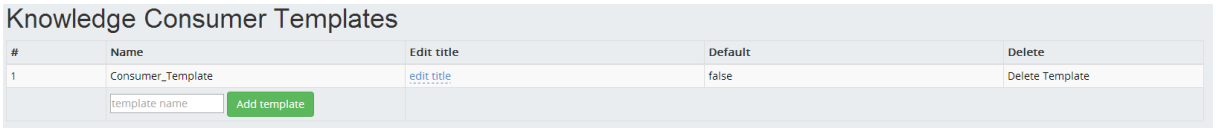


Abbildung 47: Konsumentenvorlagenverwaltung

Verfassen von Konsumentenvorlagen:
Wenn Sie eine erstellte Vorlage selektieren, dann erscheint im unteren Fensterbereich der Wiki-Eintrag der Vorlage. Diesen können Sie über den Tab „Write“ editieren und danach speichern.

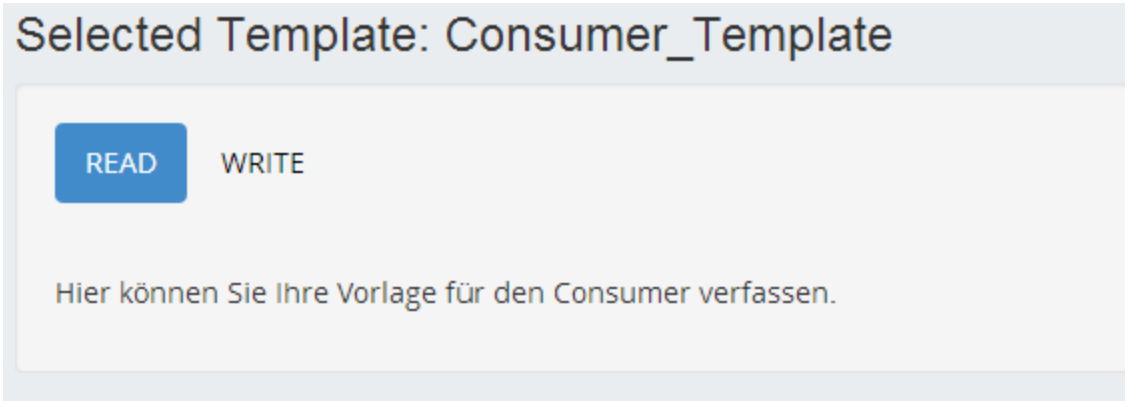


Abbildung 48: Konsumentenvorlagen Wiki

H.2.6.2Import und Export

Über den Button Import/Export gelangen Sie auf nachfolgend illustrierte Ansicht. Auf dieser Seite können Sie Ihr Projekt importieren bzw. exportieren. Bei einem Simple Export wird ein Projekt exportiert, jedoch werden die Wiki-Seiten als Verlinkung exportiert. Falls Sie den Inhalt der Wiki-Seiten ebenfalls exportieren wollen, müssen Sie einen Full Export durchführen. Dabei werden beim Import neue Wikiseiten angelegt. Um ein Projekt zu importieren, wählen Sie eine Datei aus und klicken danach auf den Button „Add Export". Wenn Sie ein Importiertes Projekt übernehmen wollen, bestätigen Sie dies mit „Import to Project". In einem anschliessenden PopUp können Sie entscheiden, ob Sie die bereits von Ihnen eingetragenen Daten behalten oder überschreiben möchten.

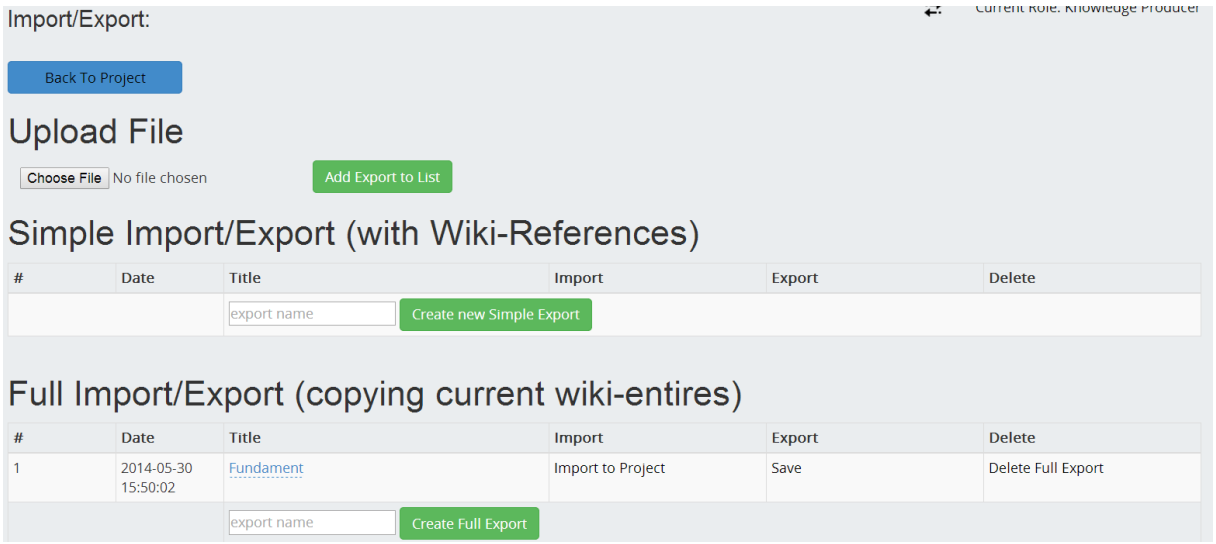


Abbildung 49: Import/Export

H.2.6.3Zugriffskontrolle

Über den Button „Access Right" gelangen Sie in folgende Ansicht.

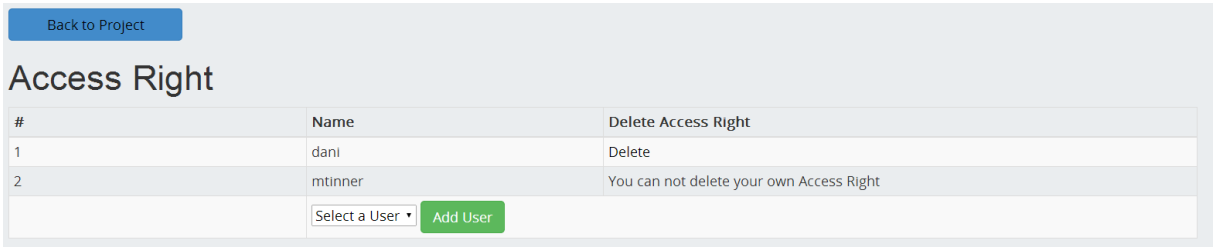
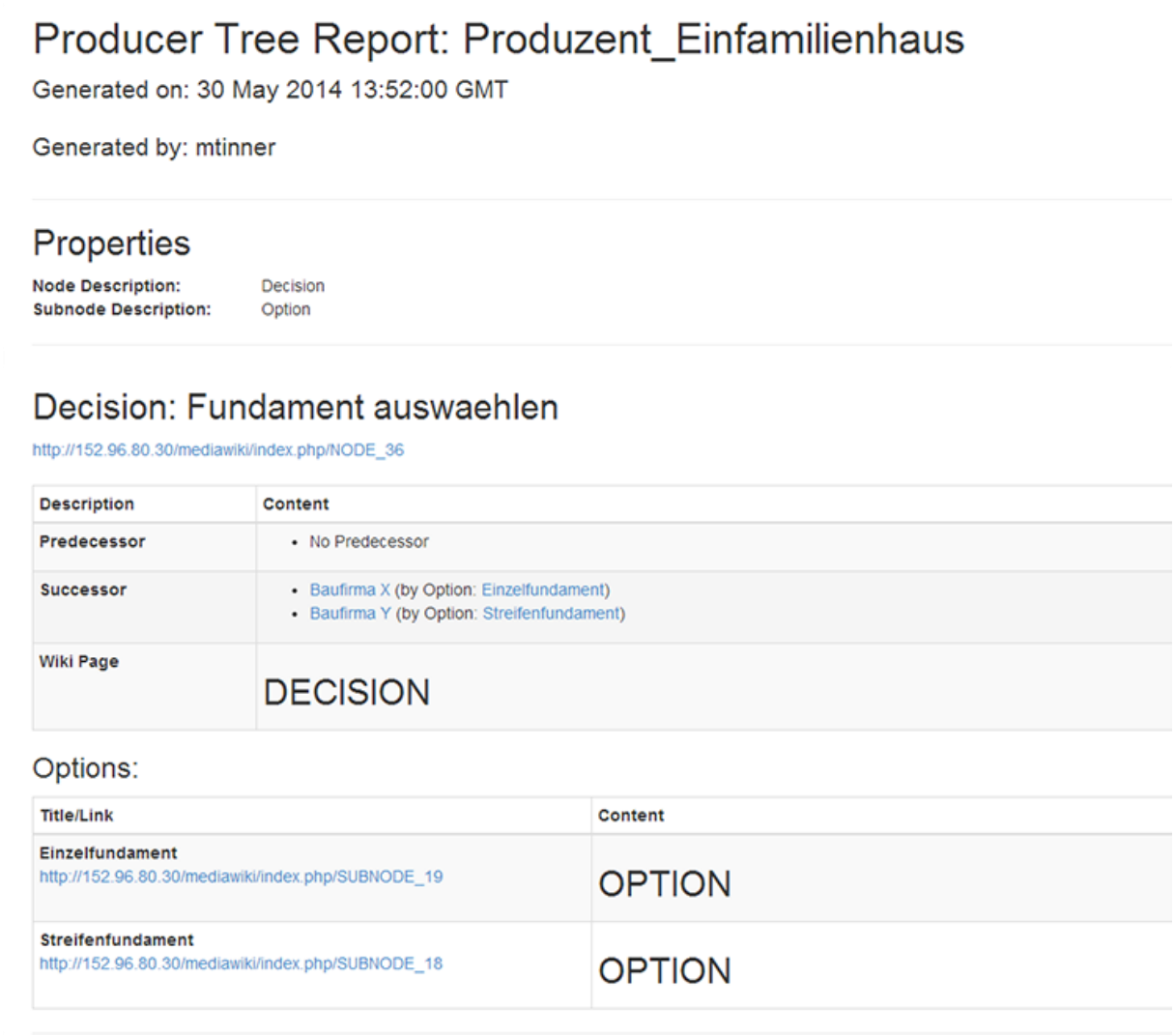


Abbildung 50: Zugriffskontrolle

Hier können Sie Ihr erstelltes Projekt anderen Benutzern freigeben. Dabei erlaubt man dem eingeladenen Benutzer alle Rechte, welche auch der Ersteller des Projektes hat.

H.2.6.4 Projekt Report

Untenstehend sehen Sie einen Report eines Projektes. Dabei wird einen vollständigen Auszug des Projektes erstellt. Mittels eines Klickes auf das blau Geschriebene im Report gelangen Sie direkt zum jeweiligen Punkt bzw. auf den Wiki-Eintrag. Ein Report kann entweder in der Projektverwaltung oder in der Projektübersicht erstellt werden.



Producer Tree Report: Produzent_Einfamilienhaus

Generated on: 30 May 2014 13:52:00 GMT

Generated by: mtinner

Properties

Node Description: Decision
Subnode Description: Option

Decision: Fundament auswaehlen

http://152.96.80.30/mediawiki/index.php/NODE_36

Description	Content
Predecessor	• No Predecessor
Successor	• Baufirma X (by Option: Einzelfundament) • Baufirma Y (by Option: Streifenfundament)
Wiki Page	DECISION

Options:

Title/Link	Content
Einzelfundament http://152.96.80.30/mediawiki/index.php/SUBNODE_19	OPTION
Streifenfundament http://152.96.80.30/mediawiki/index.php/SUBNODE_18	OPTION

Abbildung 51: Report

H.2.6.5 Entscheidungsverwaltung und Gruppierung

In der Entscheidungsverwaltung können Sie Ihre Entscheidungen beliebig gruppieren. Dabei gibt es jedoch keine Verknüpfungen wie Sie diese möglicherweise von Ihrem Betriebssystem her kennen. „Create Folder“ erstellt einen Ordner wie der Ordner „Dach“ in unten stehendem Bild. Dabei kann einen Ordner wiederum Ordner bzw. auch Entscheidungen enthalten. Der Button „Show Route“ zeigt die selektierte Entscheidung inkl. der Kindsknoten, falls im Entscheidungsführungsgraphen vorhanden. Falls keine Entscheidung im Entscheidungsführungsgraphen angezeigt wird, wurde dieser dem Graphen nicht hinzugefügt. Mehr dazu im Kapitel „**Fehler! Verweisquelle konnte nicht gefunden werden. Fehler! rweisquelle konnte nicht gefunden werden.**“.
Sie können Entscheidungen, Ordner oder ganze Verzeichnisse löschen, verschieben und kopieren.

Löschen von Elementen:
Falls Sie vorhaben einen Ordner, welcher weitere Elemente enthält zu löschen, beachten Sie dabei, dass diese ebenfalls gelöscht werden. Wenn Sie Entscheidungen löschen, löschen Sie auch dessen Optionen. Löschen von Entscheidungen bzw. Ordnern kann auch Auswirkungen auf den Entscheidungsführungsgraphen haben.

Verschieben von Elementen:
Durch einfaches Drag & Drop verschieben Sie die Elemente. Achten Sie darauf, dass Sie keine Elemente in die oberste Hierarchie verschieben. Diese ist ausschliesslich für die Gesamtgruppierung gedacht.

Kopieren von Elementen:
Kopieren können Sie durch Drag & Drop innerhalb des Baumes und zusätzliches halten der STRG-Taste bei Windows bzw. der CMD-Taste bei Mac. Dabei wird der Wikieintrag sowie die Optionen der Entscheidungen mitkopiert. Die Entscheidung erscheint sofort als Kopie, achten Sie jedoch darauf, dass Sie die neu kopierten Elemente nicht sofort bearbeiten, da deren Wiki-Einträge zuerst erstellt werden müssen. Um sicher zu stellen, dass diese mit Sicherheit vollständig kopiert wurden, warten Sie pro kopierte Entscheidung zehn Sekunden.

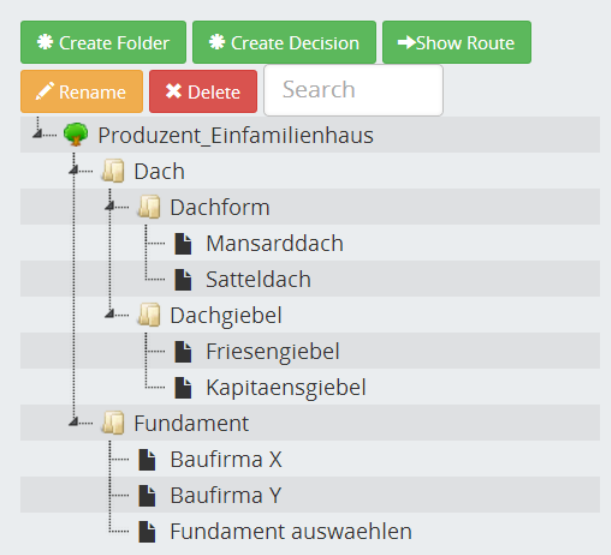


Abbildung 52: Entscheidungsverwaltung

H.2.6.6 Entscheidungsführungsgraphen

Den Entscheidungsführungsgraphen können Sie anhand der vorgängig erstellten Entscheidungen zusammenstellen. Entscheidungen bringen Sie durch Drag & Drop von der Entscheidungsverwaltung in den Graphen. Diese Entscheidungen im Entscheidungsführungsgraphen sind Verlinkungen, d.h. wenn Sie eine Entscheidung in der Verwaltung löschen, so wird der dazugehörende Knoten auch im Entscheidungsführungsgraphen gelöscht, inklusive sämtlicher Erweiterungen (siehe Kapitel H.2.6.8 Optionen) und anhängenden Verbindungen. Bevor Sie Elemente löschen können, müssen Sie diese selektieren. Die Selektion ist durch die Veränderung der Elementfarbe sichtbar. Bei Knoten werden auch anhängende Verbindungen selektiert. Anschliessen können Sie die Elemente über den Button „Remove Selected Element From Graph“ entfernen. Dabei ist zu beachten, dass Links vollständig gelöscht werden, Knoten hingegen nicht, diese werden nur vom Entscheidungsführungsgraphen entfernt. Entscheidungserweiterungen bleiben dabei erhalten.
Wenn Sie ein Projekt öffnen, führt das Programm einen Drill Down von Ihrem Root-Knoten aus. Falls Sie im Graphen eine zyklische Abhängigkeit haben, kann die Applikation den Root-Knoten nicht bestimmen und wählt stattdessen die Entscheidung, welche zuerst erstellt und in den Entscheidungsführungsgraphen gezogen wurde.

Was Sie zwingend vermeiden müssen:
Bei jeder Entscheidung, welche Sie in den Entscheidungsführungsgraphen ziehen, müssen Sie eine darauf verweisende Verbindung ziehen. Ausgenommen sind dabei der Root-Knoten des Projektes und die Root-Knoten bei einem Drill-Up oder Drill-Down. Falls Sie diesen Grundsatz nicht beachten, entstehen Teilgraphen, welche Sie nicht mehr miteinander verbinden können. Sie könnten dieses Problem beheben, indem Sie die exportiere Datei des Projektes anpassen und anschliessend wieder importieren.

Abbildung 53: Entscheidungsführungsgraphen

H.2.6.6.1 Knoten und Verbindungen

Untenstehendes Bild zeigt einen Knoten mit Optionen. Die Optionen können Sie durch einen Doppelklick auf den Knoten anzeigen bzw. ausblenden. Das linke Häkchen mit grünem Hintergrund zeigt den Status der Entscheidung an. Mehr Informationen dazu finden Sie im Kapitel „H.2.6.10 Status“.
Die Pfeile im unteren Bereich ermöglichen die Drill-Funktion, diese ist im Kapitel „H.2.6.6.2 Hierarchische Navigation (Drill)“ beschrieben. Über den Kreis auf der rechten Seite können Sie per Drag & Drop Verbindungen ziehen. Vermeiden Sie bei diesem Vorgang das Erstellen von Zyklen.

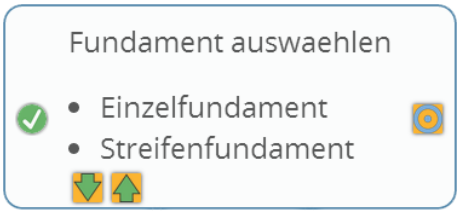


Abbildung 54: Knoten

Haben Sie eine Verbindung gezogen, erscheint folgendes Popup. Hier müssen Sie eine Option für die Verbindung anwählen. Diese wird anschliessend bei der Verbindung angezeigt. Falls Sie keine Option auswählen, führt dies zur Löschung der gezogenen Verbindung.

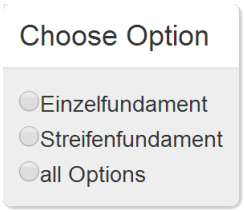


Abbildung 55: Popup

H.2.6.6.2 Hierarchische Navigation (Drill)

Über die Buttons für Drill Down und für Drill Up können Sie eine hierarchische Navigation durchführen. Die hierarchische Tiefe ist über einen Wert festgelegt, welchen Sie verändern können. Mehr dazu im Kapitel „H.2.4.2 Einstellen hierarchischer Navigation (Drill)“.

H.2.6.6.2.1 Drill Down

Bei einem Drill Down wird der gewählte Knoten, dessen Kindesknoten sowie deren Optionen und die dazwischenliegende Verbindung angezeigt. Dieser Vorgang wiederholt sich entsprechend der Standardtiefe (Einstellen hierarchischer Navigation (Drill)).

H.2.6.6.2.2 Drill Up

Bei einem Drill Up wird der gewählte Knoten dessen Geschwisterknoten, Elternknoten sowie deren Optionen und die dazwischenliegende Verbindung angezeigt. Dieser Vorgang wiederholt sich entsprechend der Standardtiefe (Einstellen hierarchischer Navigation (Drill)).

H.2.6.7Verfassen von Entscheidungen

Wenn Sie einen Knoten bzw. eine Entscheidung selektieren, sehen Sie im unteren Teil folgenden Abschnitt.

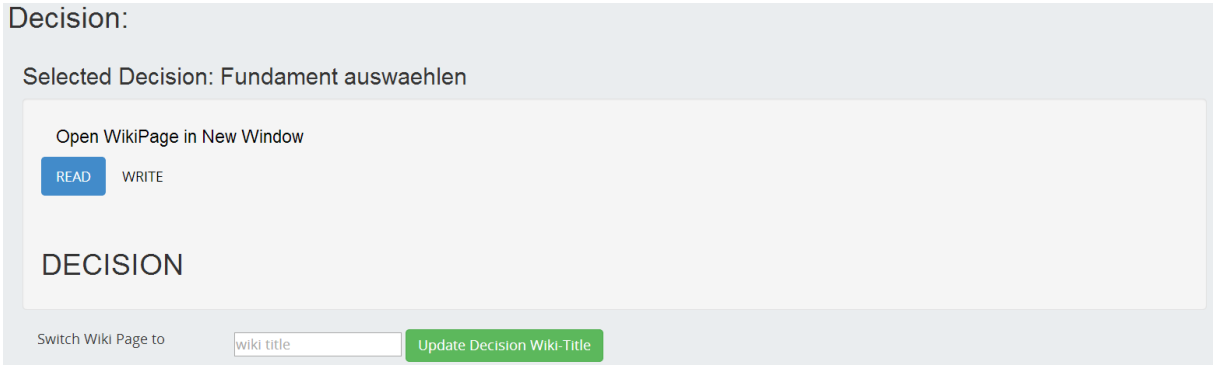


Abbildung 56: Wiki-Eintrag für Entscheidungen

Hier können Sie einen Wiki-Eintrag basierend auf der ausgewählten Entscheidung verfassen. Für Formatierungen verwenden Sie die Wiki-. Falls Sie einen bestehenden Wiki-Eintrag der Entscheidung zuordnen wollen, können Sie dessen Wiki-Titel im unteren Eingabefeld eintragen und über den Button bestätigen. „Open WikiPage in New Window“ öffnet den Wiki-Eintrag im MediaWiki in einem neuen Tab.

H.2.6.8Optionen

Indem Sie einen Knoten bzw. eine Entscheidung selektieren, erscheint im unteren Bereich ein zusätzliches Feld. Hier können Sie neue Optionen hinzufügen, löschen, umbenennen und deren Position verändern. Um effizient neue Optionen hinzuzufügen, können Sie den Text eingeben und über „Enter“ hinzufügen. Die Optionen können Sie anschliessend beim Erstellen einer neuen Verbindung auswählen. Siehe dazu Kapitel „H.2.6.6.1 Knoten und Verbindungen“. Ebenso können Sie Optionen hinzufügen, bei welchen der Wiki-Eintrag bereits besteht, hierzu verwenden Sie die unteren beiden Eingabefelder.

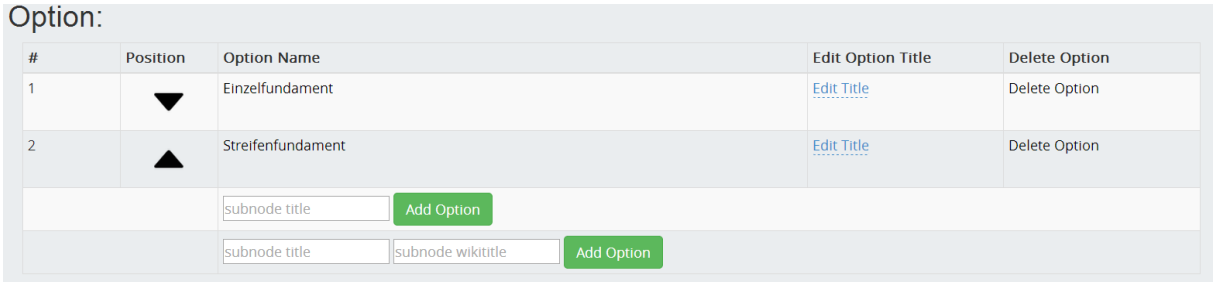


Abbildung 57: Optionen

Indem Sie eine Option über den Namen selektieren, erscheint unterhalb folgende Ansicht. Hier können Sie einen Wiki-Eintrag basierend auf der Option, die Sie gewählt haben, editieren bzw. lesen. Verwenden Sie um den Text zu Formatieren die Wiki-. Sie können den MediaWiki-Artikel auch direkt öffnen, indem sie auf „Open Wiki Page in New Window“ klicken.

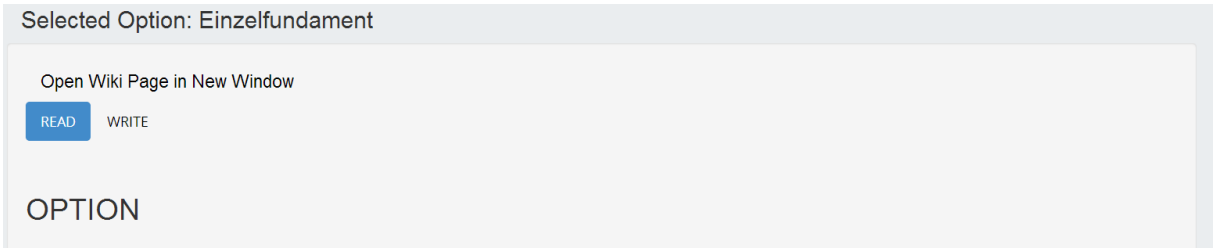


Abbildung 58: Wiki-Eintrag für Optionen

H.2.6.9Kommentare

Sie können einen Knoten bzw. eine Entscheidung auch kommentieren. Hierzu selektieren Sie diesen, danach erscheint im unteren Bereich folgendes Feld. Diese Funktion steht aber ausschliesslich der Konsumentenrolle zur Verfügung.

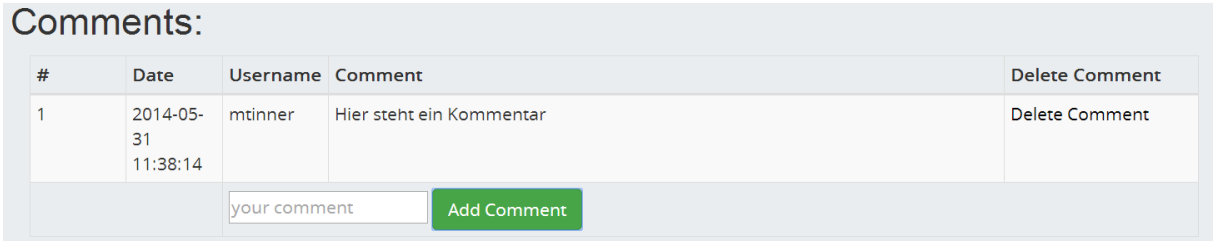


Abbildung 59: Kommentare

H.2.6.10 Status

Ein Status veranschaulicht den aktuellen Stand der Entscheidung. Im Knoten wird jeweils der Status als Bild angezeigt. Es stehen folgende Status für die Entscheidung zur Auswahl. Standardmässig erscheint Open. Falls der Status geändert wurde, kann nicht mehr auf Open gewechselt werden. Diese Funktion steht nur dem Konsumenten zur Verfügung.

Status	Bild im Knoten
Open	
Decided	
Accepted	
Rejected	
Closed	

Tabelle 49: Statusübersicht

Wenn Sie eine Entscheidung oder einen Knoten selektieren, erscheint unterhalb folgender Abschnitt. Auf der rechten Seite können Sie den Status des jeweiligen Knotens ändern.

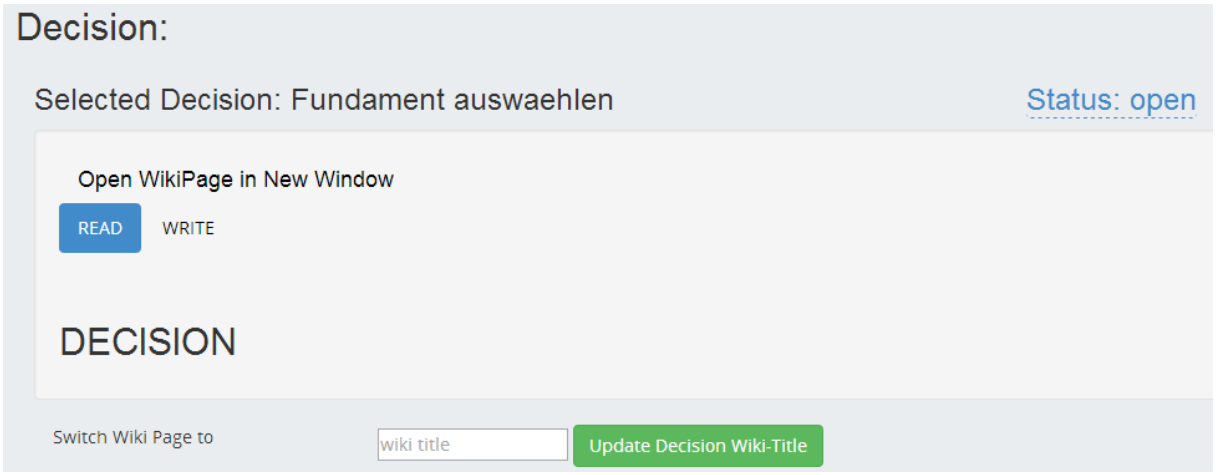


Abbildung 60: Status verändern

H.2.6.11 Wiki-Syntax

Wikipedia verwendet ebenfalls MediaWiki. Daher leiten wir Sie an dieser Stelle für valide grundlegende Syntaxelemente zur Hilfe vom Wikipedia weiter.
Link: http://de.wikipedia.org/wiki/Hilfe:Wikisyntax

H.2.7 Gemeinsame Nutzung

Falls bereits ein Benutzer Ihr ausgewähltes Projekt am Bearbeiten ist, können Sie keine Änderungen vornehmen. Falls Sie Änderungen vornehmen und das Projekt blockiert ist, erscheint ein roter Balken am oberen Fensterrand. Darin steht, wer das Projekt bearbeitet und wie lange die Sperrung aktuell noch dauert. Dies ist nur eine vorübergehende Zeitangabe, welche sich verändert. Die Sperrzeit wird bei jeder Änderung wieder erhöht.

Abbildung 61: Lockingmeldung

H.2.8 Konfigurationen der Applikation

In der Applikation ist ein Property-File eingepflegt. Die Datei ist im Projekt unter „src/main/resources“ abgelegt. Konfigurierbar sind Namensgebungen, Zieladressen und Werte.

Property	Beschreibung	Beispieleintrag
CDAR_LOCAL_DB_CONNECTION	Datenbankzieladresse	CDAR_LOCAL_DB_CONNECTION=jdbc:mysql://localhost:3306/cdar
CDAR_LOCAL_DB_USER	Datenbankbenutzer	CDAR_LOCAL_DB_USER=root
CDAR_LOCAL_DB_PASSWORD	Datenbankpasswort	CDAR_LOCAL_DB_PASSWORD=
CDAR_DIRECTORY_DESCRIPTION	Namensgebung für die Projektübersicht und den Report	CDAR_DIRECTORY_DESCRIPTION=Folder
CDAR_NODE_DESCRIPTION	Namensgebung für die Projektübersicht und den Report	CDAR_NODE_DESCRIPTION=Decision
CDAR_SUBNODE_DESCRIPTION	Namensgebung für die Projektübersicht und den Report	CDAR_SUBNODE_DESCRIPTION=Option
CDAR_MEDIAWIKI_CONNECTION	Verbindungsadresse zu MediaWiki. Ohne „http://“ angeben!	CDAR_MEDIAWIKI_CONNECTION=152.96.80.30/mediawiki
CDAR_MEDIAWIKI_PAGEURL	Erweiterung für direkten Zugang zur MediaWiki-Seite	CDAR_MEDIAWIKI_PAGEURL=index.php
CDAR_EXPANDING_LEVEL	Anzahl anfänglich ausgeklappter Stufen in der Entscheidungsverwaltung	CDAR_EXPANDING_LEVEL=4
CDAR_LOCKING_HOUR	Ausschluss gegen gemeinsame Bearbeitung in Stunden	CDAR_LOCKING_HOUR=0
CDAR_LOCKING_MINUTE	Ausschluss gegen gemeinsame Bearbeitung in Minuten	CDAR_LOCKING_MINUTE=15

Tabelle 50: Property-File

H.2.9 Kompatibilität und Einschränkungen

Gemäss Anforderungsspezifikation wurde die Applikation auf den neusten gängigsten Browsern getestet. Dazu gehören Google Chrome Version 35, Internet Explorer 11 sowie Firefox 29. Wir empfehlen Google Chrome zu verwenden.

Beim Firefox gibt es eine Einschränkung. Im angemeldeten Zustand ist der Rollenwechsel nicht möglich.

H.2.10 Frequently Asked Questions (FAQ)

Weshalb kann ich die Entscheidung bzw. die Verbindung im Entscheidungsführungs-graphen nicht selektieren?
Anstatt einen einfachen Klick, einen Doppelkick auf das gewünschte Element wird das Element selektieren.

Wie kann ich zwei Teilbäume verknüpfen?
Dies ist aktuell nicht möglich im Projekt. Einzige Möglichkeit zur Behebung des Problems ist, indem Sie die exportiere Datei des Projektes anpassen und anschliessend wieder importieren.

Weshalb erscheint das Popup nicht?
Dies kann geschehen, wenn Sie eine Entscheidung in den Entscheidungsführungsgraphen gezogen haben, diesen verbinden, daraufhin den Knoten vom Graphen entfernen und anschliessend dieselbe Entscheidung wieder in den Entscheidungsführungsgraphen ziehen und verbinden wollen.
Ein Refresh der Webseite wird das Problem lösen.

Weshalb ist meine Verbindung nicht beschriftet?
Dies hat zur Ursache, dass das Popup nicht erschienen ist und Sie keine Option für die Verbindung auswählen konnten. Lesen Sie dazu die FAQ „Weshalb erscheint das Popup nicht?“.

Weshalb kann ich meine Entscheidung nicht in den Entscheidungsführungsgraphen droppen?
Die Entscheidung kann nur einmal im Entscheidungsführungsgraphen vorkommen. Demnach wurde die Entscheidung bereits in den Graphen eingefügt. Falls Sie den Knoten aktuell im Entscheidungsführungsgraphen nicht sehen, können Sie diesen über die Entscheidungsverwaltung selektieren und über „Show Route“ anzeigen.

Wo kann ich mein Passwort ändern?
Lesen Sie dazu das Kapitel H.2.4.1 Passwortwechsel.

Weshalb kann ich mich nicht anmelden?
Falls Sie kürzlich einen Passwortwechsel in der Applikation vorgenommen haben, müssen Sie das Passwort ebenfalls bei MediaWiki ändern. Siehe Kapitel H.2.4.1 Passwortwechsel.

Wie funktioniert ein Drill Up bzw. Drill Down?
Lesen Sie hierfür das Kapitel H.2.6.6.2 Hierarchische Navigation (Drill).