ulm university universität **u**ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Konzeption und Entwicklung einer auf Smartphones optimierten mobilen Anwendung für kollaboratives Checklisten-Management

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Sabrina Geiger
sabrina.geiger@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Nicolas Mundbrod

2013

Fassung 18. November 2013

proCollab

Satz: PDF-LaTeX 2ε

Kurzfassung

Durch die zunehmende Globalisierung gewinnt die sogenannte kooperative Wissensarbeit in hochentwickelten Ländern immer mehr an Bedeutung. In diesen Ländern steht nicht mehr die Produktion materieller Güter im Vordergrund, sondern der Erwerb und die Schaffung neuen Wissens in Bereichen wie z.B. Forschung und Entwicklung. Offensichtlich ist es im Interesse der Wirtschaft wie auch der Forschung, kooperative Wissensarbeit systematisch besser unterstützen zu können. Jedoch entstehen durch kooperative Wissensarbeit einige Probleme, für die ein hoher Kommunikations- und Koordinationsaufwand für die beteiligten Wissensarbeiter ein stellvertretendes Beispiel darstellt. Mit dem Ansatz des Checklisten-Managements soll überprüft werden, ob die Zusammenarbeit der Wissensarbeiter verbessert werden kann und dabei die Durchführung von wissensintensiven Prozessen besser unterstützt werden. Da die Präsenz der mobilen Endgeräte, insbesondere Smartphones, immer mehr an Bedeutung gewinnt, ist es vorteilhaft das Szenario des Checklisten-Managements anhand einer mobilen Anwendung abzudecken. Aufgrund dessen wird in der vorliegenden Arbeit eine auf Smartphones optimierte mobile Anwendung konzipiert und entwickelt, die der Optimierung des kollaborativen Checklisten-Managements dient.

Inhaltsverzeichnis

1

Einleitung

Die zunehmende Globalisierung und vor allem die hohe Präsenz der Unternehmen auf den Auslandsmärkten wirkt sich auf die Industriegesellschaft aus [KS09]. Derzeit liegt der Fokus der Industrie in hochentwickelten Ländern nicht mehr auf der Produktion materieller Güter, sondern auf der Entwicklung und Forschung von neuen Innovationen, sowie auf der Optimierung der Entwicklungsprozesse. Durch den strukturellen Wandel der Industriegesellschaft hin zu einer Informationsgesellschaft verändern sich die Wettbewerbsbedingungen der Unternehmen. Die Internationalisierung bietet den Unternehmen einen Kostenvorteil durch bspw. Billiglöhne, was oft der Grund für die Auslagerung der Produktionsstätten ist. Außerdem wächst die Anzahl der Anbieter auf internationalen Märkten, was den Wettbewerbsdruck erhöht. Nun ist nicht nur der Preiskampf ein Aspekt der sich auf den Wettbewerb auswirkt, sondern auch der Aspekt der Qualität. Die Vielzahl an verschiedenen Angeboten lässt die Kundenerwartungen steigen und neue Ideen, Technologien und Geschäftsmodelle führen zu schnelleren Entwicklungen, die die bisher

erfolgreichen Produkte und Arbeitsprozesse ablösen. Insgesamt steigt dadurch der Innovationsdruck und die Innovationszyklen werden gleichzeitig kürzer. Um unter diesen Bedingungen zu bestehen, sind Unternehmen in der heutigen Zeit gezwungen, ihre Arbeitsmethoden und Entwicklungsvorgänge konstant neu zu strukturieren [Paw98]. Auch folgt daraus, dass die Gesellschaft einen konsequenten Wandel zur wissensintensiven Gesellschaft erfährt. Des Weiteren konzentrieren sich die Unternehmen zunehmend auf die Optimierung der Geschäftsprozesse, indem sie die Koordination der menschlichen Aktivitäten ausarbeiten und verbessern. Als Resultat wird in vielen Branchen der Fokus auf die effektive Kooperation zwischen Wissensarbeitern gelegt, um das Wissen der Individuen zu teilen und die wissensintensiven Tätigkeiten systematisch zu unterstützen. Diese Zusammenarbeit mehrerer sogenannter Wissensarbeiter wird als kollaborative Wissensarbeit bezeichnet. Jedoch treten in der Realität eine Vielzahl an Problemen bei der Zusammenarbeit von kollaborativen Wissensarbeitern auf, welche im Folgenden erläutert werden.

1.1 Problemstellung

Um im Rahmen dieser Arbeit grundsätzliche Probleme der kollaborativen Wissensarbeit zu lösen, ist es notwendig diese zu charakterisieren [MKR12]. Durch die Zusammenarbeit mehrerer Wissensarbeiter ergibt sich ein hoher Kommunikations- und Kooperationsaufwand, der das Arbeiten untereinander erschwert. Eine weitere Herausforderung stellt die Unterstützung jeglicher wissensintensiven Geschäftsprozesse dar. Da diese im Detail unvorhersehbar sind, resultiert daraus offensichtlich der Aspekt, dass wissensintensive Prozesse nicht planbar sind. Herkömmliche prozessunterstützende Entwicklungsmethoden können demnach nicht angewandt werden, da sie auf vordefinierten Arbeitsabläufen oder zumindest auf vordefinierten Aufgaben basieren. Daher sind neue Ansätze erforderlich, die wissensintensive Prozesse und vor allem die daran beteiligten Wissensarbeiter systematisch unterstützen. Momentan existiert keine informationstechnische Lösung, insbesondere keine mobile Applikation, die Wissensarbeiter bei ihrer Arbeit systematisch unterstützt. Diese fehlende Existenz fordert eine Neuentwicklung ohne bisherige Erkenntnisse oder Erfahrungen. Der Bedarf der Wissensarbeiter nach einer solchen

Lösung steigt aber stetig, was durch die genannten negativen Aspekte zu begründen ist [MKR12].

1.2 Zielsetzung

Ziel dieser Bachelorarbeit ist die Konzeption und Entwicklung einer mobilen Anwendung für das Smartphone, die die Kollaboration von Wissensarbeitern unterstützt. Die Anwendung bietet Hilfe für wissensintensive Prozesse, indem sie die Arbeitsprozesse, in Form verschiedener Checklisten und deren Verwaltung widerspiegelt. Dabei soll die Koordination wissensintensiver Tätigkeiten durch das Checklisten-Management systematisch unterstützt werden und insbesondere sollen die bereits existenten Arbeitsprozesse optimiert werden. Die Anwendung sichert hierzu die Minimierung des Kommunikations- und Kooperationsaufwandes und fördert zusätzlich effizientes Arbeiten. In diesem Zusammenhang wird ein mächtiges Datenmodell genutzt, das sich für verschiedene Aspekte des Checklisten-Managements positiv auswirkt. Ein wichtiger Aspekt ist es zum Beispiel, die Wiederverwendbarkeit der Checklisten zu sichern, indem alle bearbeiteten Checklisten archiviert werden. Weiterhin wird durch die Wiederverwendbarkeit ein gewisser Lerneffekt gefördert, da anfängliche Probleme bei den jeweiligen Checklisten verbessert und angepasst werden können. Die Bereitstellung von domänenspezifischen Checklistenvorlagen gewährleistet diesen Aspekt. Ebenso sollen die Checklisten und Einträge durch die Zugehörigkeit zu verschiedenen Domänen die Kontextintegration fördern. Des Weiteren soll die Anwendung den Vorteil der einfachen Handhabung bieten, der durch gelungene Usability Aspekte gesichert wird. Um diese Anforderungen zu gewährleisten, wurde eine Vorgehenssystematik erarbeitet, die sich an den rot gekennzeichneten Prozessschritten aus dem *Referenzmodell Usability-Engineering* in Abbildung 1.1 orientiert. Basierend auf dem fundamentalen Themenverständnis, erfolgt die Bearbeitung der genannten Problemstellung in vier weiteren Abschnitten. Auf Basis der Resultate der Anforderungsanalyse und des Entwurfs wird die Anwendung daraufhin entwickelt. Im Anschluss wird ein inhaltlicher Überblick über den Aufbau der Arbeit und die insgesamt sechs Kapitel gegeben.

Abbildung 1.1: Das Referenzmodell Usability-Engineering [Off12]

1.3 Aufbau der Arbeit

Insgesamt umfasst diese Arbeit sechs Kapitel. Einen visuellen Überblick über die Kapitel, bietet die Abbildung 1.2.

Das **Kapitel 2** dient der Sensibilisierung und Einführung in die Thematik des Checklisten-Managements. In diesem Kapitel wird das fundamentale Basiswissen für die folgenden Kapitel erläutert.

Im darauf folgenden **Kapitel 3** werden die Anforderungen vorgestellt und spezifiziert. Die Analyse der Anforderungen orientiert sich in der Vorgehensweise am Referenzmodell Usability-Engineering [Off12].

Die bisher dokumentierten Kapitel unterstützen das **Kapitel 4**. Inhaltlich beschäftigt sich der vierte Abschnitt mit dem technischen Entwurf und dem User-Interface Entwurf.

Des Weiteren werden in **Kapitel 5** die Erkenntnisse und Ergebnisse der Entwurfsphase umgesetzt. Hier werden Aspekte der Implementierung konkretisiert.

Das **Kapitel 6** enthält eine Zusammenfassung der gesamten Bachelorarbeit, in welcher mögliche Erweiterungen der mobilen Anwendung aufgegriffen werden.

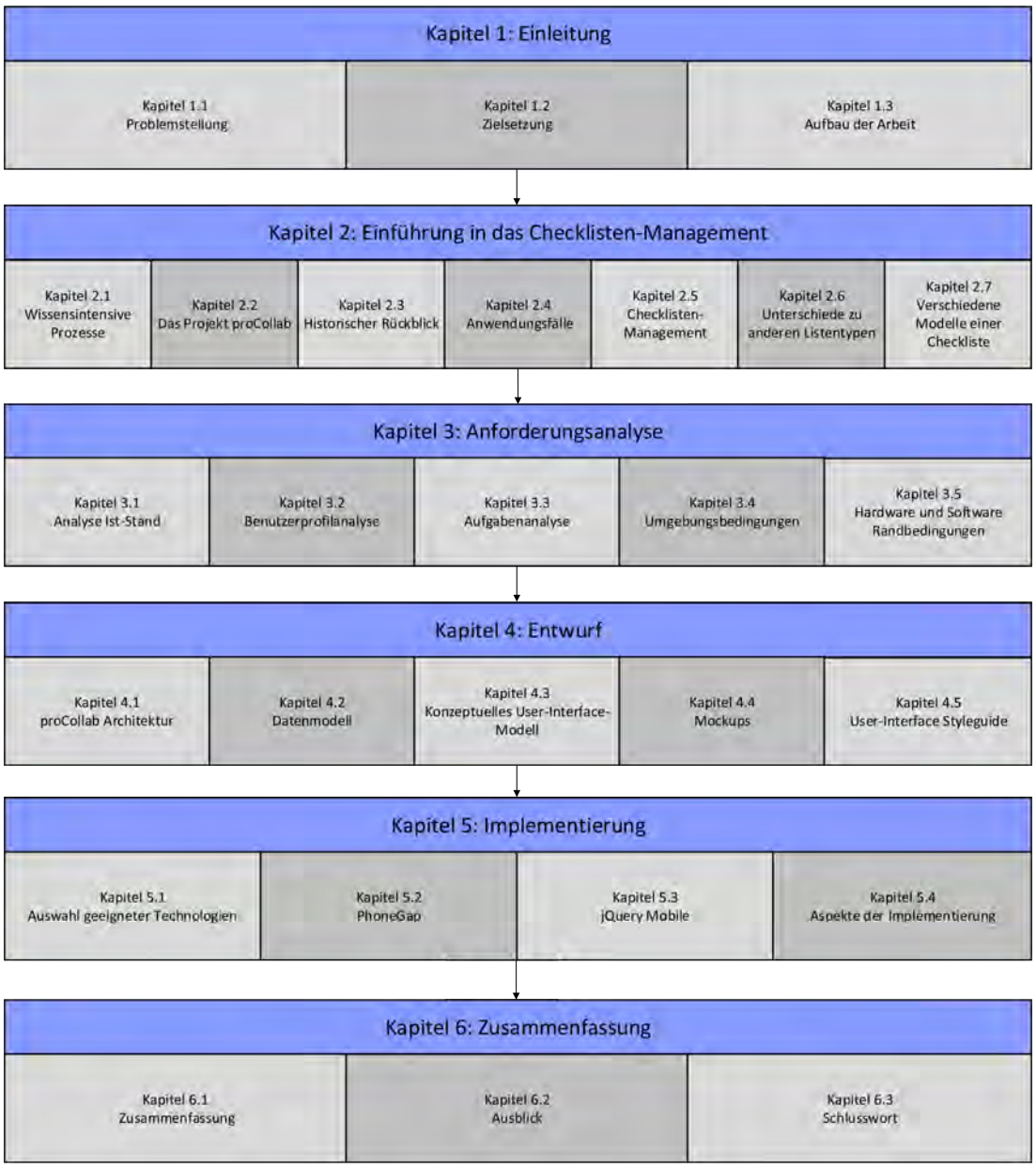


Abbildung 1.2: Aufbau der Arbeit

2

Einführung in das Checklisten-Management

Als Grundlage für die folgenden Kapitel schafft Kapitel 2 die notwendige Wissensbasis. Es werden die relevanten Begriffe definiert und die Grundlagen der Thematik erläutert. Anfangs erfolgt in Kapitel 2.1 eine Einführung in die Thematik der wissensintensiven Prozesse. Darüber hinaus wird im Kapitel 2.2 das Projekt *proCollab* vorgestellt, da die Thematik dieser Bachelorarbeit ein Bestandteil eines Gesamtprojekts ist. Des Weiteren gibt Kapitel 2.3 einen kurzen Einblick in die Geschichte des Checklisten-Managements. Im anschließenden Kapitel 2.4 werden verschiedene Anwendungsfälle vorgestellt, um den komplexen Sachverhalt zu vereinfachen. Die inhaltlich relevanten Begriffe werden anhand von Definitionen im darauffolgenden Kapitel 2.5 erläutert. Anschließend wird in Kapitel 2.6 die Abgrenzung zu anderen Listen dokumentiert. Hier wird vor allem die To-do-Liste als explizites Beispiel herangezogen. Im darauffolgenden Kapitel 2.7 werden

die verschiedenen Modelle einer Checkliste unterschieden. Insgesamt handelt es sich um zwei verschiedene Varianten, das *Zahlenmodell* und das *Kästchenmodell*. Einen Überblick über alle Kapitel bietet die Abbildung 2.1.

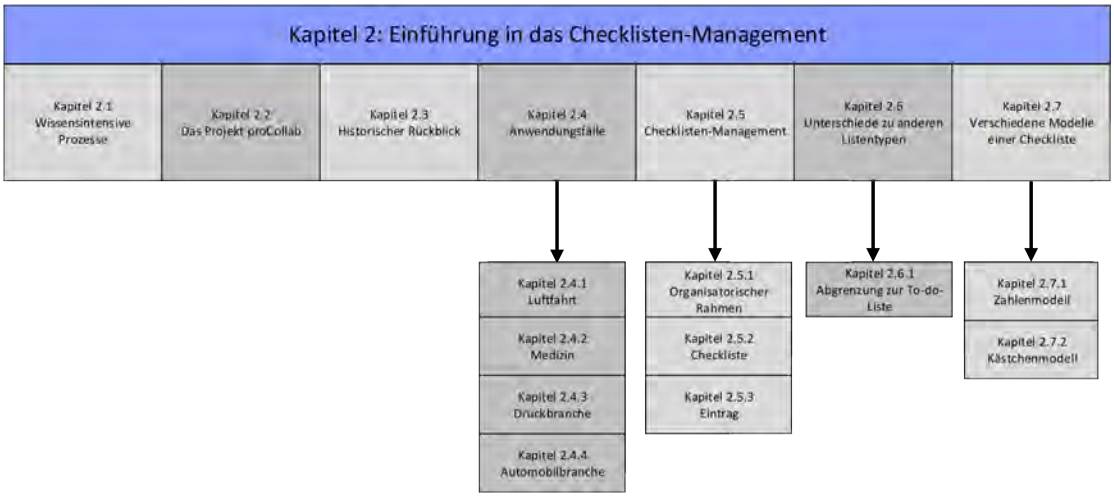


Abbildung 2.1: Aufbau des Kapitels Einführung in das Checklisten-Management

2.1 Wissensintensive Prozesse

Durch die zunehmende Globalisierung werden Entwicklungsprozesse ins Ausland verlagert, um Vorteile wie den Billiglohn zu nutzen. Dadurch werden die automatisierten und standardisierten Entwicklungsprozesse in die Niedriglohnländer ausgelagert [Mun12]. In den hoch entwickelnden Ländern wandelt sich somit die Industriegesellschaft zu einer Informationsgesellschaft, bei der die Determinanten Wissen und Information im Fokus stehen [Hub05]. Der Teil der anspruchsvollen Arbeitsaufgaben und der Führungsaufgaben steigt kontinuierlich an. Daraus resultiert, dass die Wissensarbeit in den Fokus der hoch entwickelten Länder rückt [Hub05]. Es etablieren sich Arbeiten die schwer vorhersehbar sind und einen hohen Grad an Expertise benötigen. Im Mittelpunkt steht die Steigerung und Erhaltung der Leistungsfähigkeit von Wissensarbeiter die als Produzent, Anbieter und Überträger von Informationen dienen. Die Steigerung der Produktivität wird erreicht, indem Lösungen für wissensintensive Prozesse entwickelt werden. Bei

wissensintensiven Prozessen muss man jedoch von der Einzigartigkeit der Endprodukte ausgehen. Weiterhin ist der Entwicklungsprozess nicht vorhersehbar, da er individuellen und dynamischen Einflüssen ausgesetzt ist [Hub05]. Um die Effizienz und die Effektivität der Wissensarbeiter in einem wissensintensiven Prozess zu erhöhen, wird die Zusammenarbeit von Wissensarbeitern fokussiert. Die Zusammenarbeit verschiedener Personen im Bereich Wissensarbeit, wird als kollaborative Wissensarbeit bezeichnet. Im Folgenden werden Charakteristika (C) der kollaborativen Wissensarbeit vorgestellt, um den komplexen Sachverhalt zu vereinfachen [MKR12].

C 1: Unsicherheit
Aufgrund der Komplexität von Wissensarbeit ist es nicht möglich verschiedenste Einflussfaktoren zu kalkulieren. Dies führt dazu, dass wissensintensive Prozesse unvorhersehbar und nicht planbar sind. Ausgeführte und geplante Arbeitsschritte müssen ständig durch die involvierten Wissensarbeiter neu bewertet werden. Zusätzlich steigt die Unsicherheit durch die Kooperation von Wissensarbeitern, da die Arbeitsteilung zwischenmenschliche Abhängigkeiten und Beeinträchtigungen umfasst.

C 2: Zielorientierung
Der antreibende Faktor der Wissensarbeiter ist ein gemeinsames, einheitliches Ziel. Im Optimalfall sind die individuellen Ziele der Wissensarbeiter dem Kontext eines gemeinsamen Ziels untergeordnet. Die Ableitung von Teilzielen, die in kurzen Zeitspannen erreicht werden können, unterstützt die komplexe und unvorhersehbare Abwicklung der Wissensarbeit. Diese können im Gegenteil zu dem gemeinsamen Ziel geändert oder entfernt werden.

C 3: Prozessentstehung
Um die verschiedenen Zwischenziele zu erreichen, passen Wissensarbeiter ständig ihre Aktivitäten an. Die jeweilige Planung von Teilaktivitäten und die dynamische Durchführung der Arbeit führt zu einer schrittweisen Entwicklung der kollaborativen Wissensarbeit. Unter Berücksichtigung des aktuellen Wissensstandes und den beeinflussenden Faktoren, wägen Wissensarbeiter laufend ihre Optionen bezüglich der Aktivitäten ab.

C 4: Wachsende Wissensbasis
Ein wesentlicher Aspekt der kooperativen Wissensarbeit ist die Kommunikation zwischen

den Wissensarbeitern, damit das heterogene, individuelle Wissen geteilt wird. Hierfür ist eine entsprechende Organisation der Informationen mittels Dokumenten, E-Mails oder Ähnlichem vorgesehen.

Unter Beachtung der aufgeführten Charakteristika soll eine mobile Anwendung entwickelt werden, die die kollaborative Wissensarbeit und vor allem den wissensintensiven Prozess mit Hilfe des Checklisten-Management als Lösungsansatz unterstützt. Diese Entwicklung erfolgt im Rahmen des Projektes proCollab, das im Folgenden erläutert wird.

2.2 Das Projekt proCollab

ProCollab (Process-aware Support for Collaborative Knowledge Workers) ist ein internes Projekt des Instituts für Datenbanken und Informationssysteme der Universität Ulm [Pro13]. Inhaltlich widmet sich dieses Projekt der Optimierung und Unterstützung der kollaborativen Wissensarbeit. Verschiedene Einflussfaktoren, wie Zeitaspekte, steigende Kosten oder der Mangel an Ressourcen beeinträchtigen diese Form der Arbeit und die beteiligten Wissensarbeiter elementar. Zusätzlich entsteht durch viele Einflussfaktoren ein großer Kommunikations- und Kooperationsaufwand zwischen den jeweiligen Wissensarbeitern. Aufgrund dieser Aspekte zielt das Projekt proCollab, unter anderem, auf die Entwicklung eines Systems zur systematischen Unterstützung von Wissensarbeitern und ihrer Prozesse. Hierfür wird der Ansatz des Checklisten-Managements als Möglichkeit betrachtet, der den Kooperations- und Kommunikationsaufwand zwischen Wissensarbeitern reduzieren soll. In diesem Zusammenhang ist die Entwicklung einer mobilen Anwendung im Rahmen dieser Bachelorarbeit zu sehen. Die Aspekte und vor allem der Kommunikations- und Kooperationsaufwand soll durch diese verbessert werden. Im Anschluss wird die Thematik des Checklisten-Managements, beginnend mit einem historischen Rückblick, genauer betrachtet [Pro13].

2.3 Historischer Rückblick

„*Im Jahre 1918 wurden bei einer Explosion des amerikanischen Sprengstoffherstellers Du Pont 40 Arbeiter getötet, eine noch wesentlich größere Anzahl wurde schwer verletzt.*“ [EPK07]

Dieses traurige Ereignis führte dazu, dass ein neues Sicherheitskonzept basierend auf Checklisten entwickelt wurde. Dieses setzte sich letztendlich nicht nur in der Sprengstoffbranche durch, sondern auch in weiteren risikoreichen Bereichen, wie z.B. der Luftfahrt, der Kernkraft oder der Medizin [EPK07]. Auf verschiedenen Gebieten wurden schrittweise Checklisten eingesetzt, da diese sich gut eigneten, das Fehlerrisiko von Arbeitsprozessen zu minimieren. Dies waren die fundamentalen Ideen und Anreize des Checklisten-Managements. Jedoch führte das Streben der Arbeitswelt nach hoher Qualität und Fehlertoleranz zu einer Umstrukturierung des Checklisten-Managements. Heutzutage werden in vielen Unternehmensbereichen papierbasierte Checklisten für die Qualitätssicherung verwendet, die von Individuen noch händisch bearbeitet werden [Gaw11]. Diese traditionelle Variante ist oft aufwändig, zeitintensiv, und fehleranfällig, was zusätzlich zum Aspekt, der Unterstützung von Wissensarbeitern, die Konzipierung eines optimierten Checklisten-Managements in Form einer mobilen Anwendung erklärt, welche im Rahmen dieser Arbeit erfolgen soll. Im Folgenden werden verschiedene Anwendungsfälle des Checklisten-Managements erläutert, um den komplexen Sachverhalt zu vereinfachen.

2.4 Anwendungsfälle

In diesem Kapitel werden vier Anwendungsfälle vorgestellt, die einerseits als Einleitung in die Thematik des Checklisten-Managements dienen und andererseits das bessere Verständnis für den komplexen Sachverhalt ermöglichen. Die Anwendungsfälle beschränken sich auf die Bereiche Luftfahrt, Medizin, Druckbranche und die Automobilbranche.

2.4.1 Luftfahrt

Im Zeitraum von 1978 bis 1990 waren Verkehrsflugzeuge in 37 Unfällen involviert, woraufhin die Unfalluntersuchungsbehörde *National Transportation Safety Board* (NTSB) eine Analyse der Unfallberichte durchführte [SE11]. Die Ergebnisse der Studie belegen, dass überwiegend die Piloten und deren Besatzung in den Unfallberichten als auslösende Faktoren genannt wurden. Die Hauptfehler wurden im Bereich der Überwachung und Entscheidungsfindung ausgemacht. Des Weiteren konnte durch die Studie nachgewiesen werden, dass über 50% der Besatzung zu den Unfallzeiten durch Übermüdung geschwächt waren. Eine ermüdete Besatzung ist weniger konzentriert und begeht deutlich mehr Fehler. Eine zusätzliche Gemeinsamkeit der Unfälle war, dass die Besatzung oft unter Zeitdruck stand. Bei Verspätungen hetzten die verantwortlichen Personen vor allem bei der Flugvorbereitung, was zu signifikanten Arbeitsfehlern und unpräziser Arbeitsweise führte [SE11]. Diese Erkenntnisse führten zu weiteren Forschungen im Bereich *Crew Resource Management* (CRM). Das CRM ist ein Trainingsprogramm, das durch einen *National Aeronautics and Space Administration*-Workshop entstand und auf die Verbesserung der Flugsicherheit zielt. In diesem Zusammenhang wurde bestätigt, dass die primäre Ursache der meisten Flugunfälle auf menschliches Versagen zurückzuführen ist [KHA10]. Die NTBS-Studie und das CRM forderten demnach den Einsatz von Checklisten, sowie ein Trainingsprogramm, das den Umgang mit diesen verdeutlicht. Die Hauptfunktion der Checklisten bestand darin, die Flugsicherheit zu gewährleisten. Während dem kompletten Flug, sowie der Vor- und Nachbereitung kommen verschiedene Checklisten zum Einsatz. Diese enthalten die Standardgrundlagen für die Überprüfung von Flugzeug-Konfigurationen. Des Weiteren wird die Reihenfolge der Bearbeitung von betrieblichen Anforderungen festgelegt. Die Koordination und Verteilung der Arbeitsaufgaben unter den Besatzungsmitgliedern wird durch die Zuweisung von Checklisten an die jeweiligen Arbeiter vereinfacht [DW93]. Ein repräsentatives Beispiel für solch eine Checkliste ist in Abbildung 2.2 dargestellt. Die solide Umsetzung der Checklisten führten in der Luftfahrt dazu, dass die möglichen Risiken und Fehler minimiert wurden. Aufgrund dieser positiven Ergebnisse wurden weitere Branchen wie die Medizin darauf aufmerksam [Gaw11].

BOEING 747-400 NORMAL PROCEDURES CHECKLIST

PARKING CHECK

First Officer — Captain

FLIGHT ATTENDANT ADVISORY…..
……...NON-STERILE
TAXI LIGHT……OFF
TIMER/STOPWATCH……..……STOP/RESET
PARKING BRAKES……SET
ELECTRICAL SYSTEM……SET / APU AVAIL ON
FUEL CONTROL SWITCHES……CUTOFF
NO SMOKING/SEAT BELTS SIGNS……OFF
HYDRAULIC DEMAND PUMPS……OFF
FUEL SYSTEM……OFF
IRS……OFF
BEACON LIGHTS……..……OFF
ANTI-ICE SYSTEMS……....OFF
AFT CARGO HEAT……OFF
WINDOW HEAT……OFF
YAW DAMPERS……OFF
AIRCONDITIONING……OFF
FMC……..……..CLEAR ROUTES
EXTERIOR LIGHTS (WING/LOGO)……..OFF

SECURING CHECK

INTEGRAL LIGHTS……...……OFF

TERMINATION CHECK
(After Last Passenger Has Deplaned)

EVAC SIGNAL……OFF
EMERGENCY LIGHTS……OFF
PACKS……OFF
ELECTRICAL SYSTEM……SET / APU AVAIL OFF
APU……OFF
STANDBY POWER……OFF
BATTERY……OFF

Abbildung 2.2: Boeing 747-400 normal procedures checklist [Boe13]

2.4.2 Medizin

Die medizinische Versorgung ist ein weiteres Beispiel für den strukturellen Wandel, der sich in den letzten Jahren in vielen Bereichen vollzieht. Hierbei spielt der Kostenfaktor eine große Rolle. Steigende Kosten sind in industrialisierten Ländern einerseits mit der wachsenden Nachfrage nach Gesundheitsdienstleistungen zu begründen und andererseits mit der rasanten technologischen Entwicklung im Bereich der Gesundheit und Medizin [Har05]. Hinzu kommt, dass diese rasche Entwicklung der Technologien zu einer Flut von Informationen in Form von wissenschaftlichen Publikationen führt. Dieser Aspekt erschwert praktizierenden Ärzten den Transfer von relevanten Informationen der Forschung in die Praxis [Har05]. Des Weiteren besteht der Wunsch seitens der Patienten nach einer optimalen und innovativen Versorgung und seitens der Ärzte nach qualitativ hochwertigen Leistungen [OKF01]. Um diesen Anforderungen gerecht zu werden begann die *Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaf-*

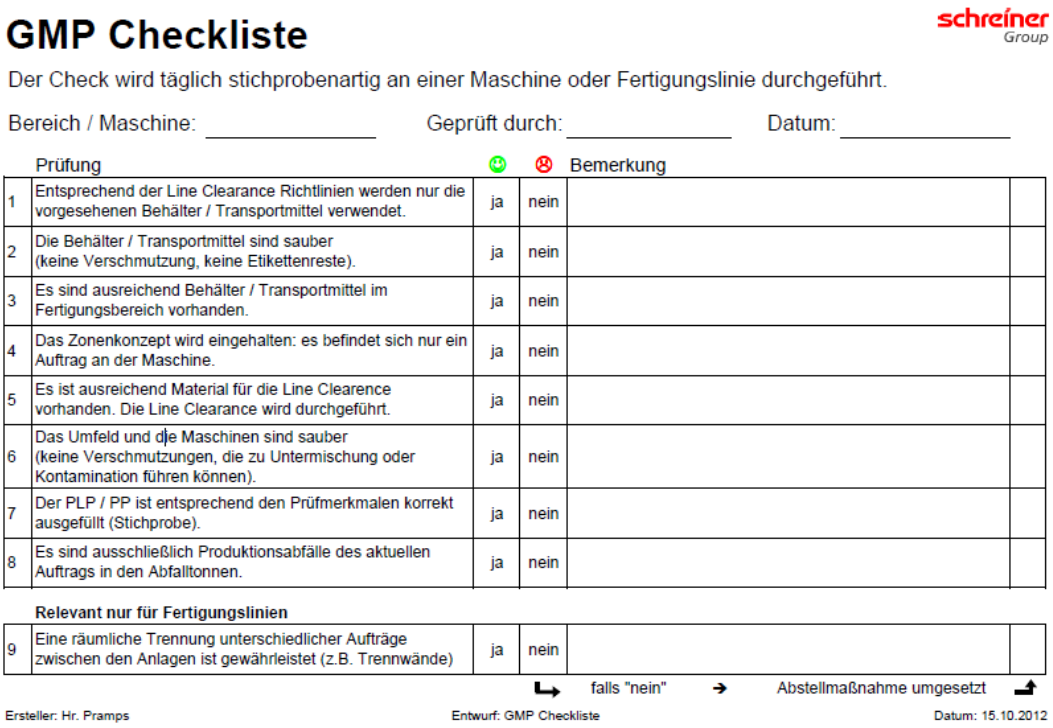
ten 1995 Leitlinien und Richtlinien zu sammeln [Har05]. Ziel dieser Leitlinien war, die medizinische Versorgung der Bevölkerung zu optimieren und überholte gesundheitliche Maßnahmen zu vermeiden. Vor allem sollte der Kostenfaktor durch die Vermeidung von überholten Maßnahmen reduziert werden. Außerdem besteht die Aufgabe der Leitlinien darin, das bisher erlangte Wissen über Krankheiten und Diagnosen festzuhalten und dieses in einer krankheitsbedingten Standardvorgehensweise zu etablieren. Im Jahr 2008 publizierte die *World Health Organization* (WHO) eine solche Leitlinie, die sich inhaltlich mit Praktiken für die Chirurgie beschäftigt (siehe Abbildung 2.3). Die Leitlinie *WHO Surgical Safety Checklist* wurde in der Praxis getestet. Insgesamt wurde sie in acht verschiedenen Krankenhäusern für eine gewisse Zeitspanne eingesetzt. Alle Standorte wiesen nach der Einführung der Leitlinie erhebliche Verbesserungen auf. Beispielsweise sank die Rate der postoperativen Komplikationen durchschnittlich um 36% [HWB+09]. Die positiven Resultate zeigen, dass diese Leitlinien im Gesundheitssystem von hoher Bedeutung sind, da Ärzte oder Krankenpfleger eine weitgehende Verantwortung tragen für die Gesundheit der Patienten. Oftmals werden banale Arbeitsschritte in einer stressigen Situation schlicht vergessen, oder nicht korrekt ausgeführt. Um den Folgen vorzubeugen, eignen sich die Leitlinien in Form von Checklisten.

Abbildung 2.3: Surgical Safety Checklist [Sur09]

2.4.3 Druckbranche

In der Druckbranche werden ebenfalls Checklisten verwendet. Ein stellvertretendes Beispiel ist die Schreiner Group GmbH & Co. KG, die Checklisten zur Qualitätssicherung verwendet. Abbildung 2.4 zeigt eine sogenannte *Good Manufacturing Practice* (GMP) Checkliste. Unter *Good Manufacturing Practice* versteht man Richtlinien der Arzneimittelproduktion, die der Qualitätssicherung dienen. Kommt es zu falschen Angaben auf den Etiketten, hat dies möglicherweise schwerwiegende Auswirkungen auf den Verbraucher. Aufgrund dessen hat die Qualitätssicherung einen hohen Stellenwert. Diese Sicherung wird durch verschiedene Checklisten erreicht, die unterschiedlichen Qualitätsaspekten gewidmet sind. Die Checkliste ist ein erweitertes Kästchenmodell (siehe Kapitel 2.7.2), bei der die Einträge mit Ja oder Nein beantwortet werden können. Falls ein Eintrag mit Nein beantwortet wird, soll der Nutzer in einem neuen Kästchen eintragen, ob die

Abstellmaßnahme umgesetzt wurde. Ebenso können noch Bemerkungen eingetragen werden. Diese Informationen wurden durch ein persönliches Interview mit dem Mitarbeiter *Lorenz Geiger*, der *Schreiner Group GmbH & Co. KG* in *85764 Oberschleißheim*, in Erfahrung gebracht.



GMP Checkliste

schreiner Group

Der Check wird täglich stichprobenartig an einer Maschine oder Fertigungslinie durchgeführt.

Bereich / Maschine: _____ Geprüft durch: _____ Datum: _____

	Prüfung	☺	☹	Bemerkung	
1	Entsprechend der Line Clearance Richtlinien werden nur die vorgesehenen Behälter / Transportmittel verwendet.	ja	nein		
2	Die Behälter / Transportmittel sind sauber (keine Verschmutzung, keine Etikettenreste).	ja	nein		
3	Es sind ausreichend Behälter / Transportmittel im Fertigungsbereich vorhanden.	ja	nein		
4	Das Zonenkonzept wird eingehalten: es befindet sich nur ein Auftrag an der Maschine.	ja	nein		
5	Es ist ausreichend Material für die Line Clearence vorhanden. Die Line Clearance wird durchgeführt.	ja	nein		
6	Das Umfeld und die Maschinen sind sauber (keine Verschmutzungen, die zu Untermischung oder Kontamination führen können).	ja	nein		
7	Der PLP / PP ist entsprechend den Prüfmerkmalen korrekt ausgefüllt (Stichprobe).	ja	nein		
8	Es sind ausschließlich Produktionsabfälle des aktuellen Auftrags in den Abfalltonnen.	ja	nein		
	Relevant nur für Fertigungslinien				
9	Eine räumliche Trennung unterschiedlicher Aufträge zwischen den Anlagen ist gewährleistet (z.B. Trennwände)	ja	nein		

falls "nein" → Abstellmaßnahme umgesetzt

Ersteller: Hr. Pramps Entwurf: GMP Checkliste Datum: 15.10.2012

Abbildung 2.4: GMP Checkliste

2.4.4 Automobilbranche

Ein weiterer Anwendungsfall ist in der Betrachtung eines mechatronischen Entwicklungsprozesses in der Automobilbranche zu finden [Mun13]. Die Entwickler mechatronischer Systeme legen großen Wert auf eine systematische Vorgehensweise, konkret über das *V-Modell* [Mun13]. Da ein Entwicklungsprojekt jedoch mehrere Phasen beinhalten kann, wird das V-Modell sogar auf die verschiedenen Phasen jeweils in angepasster Form an-

gewendet. Des Weiteren werden den einzelnen Phasen bestimmte Meilensteine zugeordnet, wobei diese Meilensteine wiederum in verschiedene parallele Entwicklungsprozesse unterteilt sind. Um die Qualität des gesamten Entwicklungsprozesses zu gewährleisten, wurde eine projektspezifische Checkliste erstellt, die von einem Qualitätssicherheitsbeauftragten verwaltet wird. Diese Checkliste enthält eine Vielzahl an Einträgen, die verschiedenen hierarchischen Kategorien zugeordnet werden. Die Einträge beinhalten zusätzlich unterschiedliche Attribute, die ein Qualitätssicherheitsbeauftragter und die Entwickler als notwendig ansehen. In einem regelmäßigen Treffen mit den Entwicklern wird über die Relevanz bestimmter Einträge entschieden. Die Einträge werden in einer Anwendung für Anforderungsmanagement, IBM Rational DOORs [Doo13], verwaltet und in einer hierarchischen Baumstruktur organisiert. Sie werden in zwei Gruppen unterteilt, die sich durch ihre logische Verwendung auszeichnen. Entweder werden die Einträge der Basisgruppe zugeordnet, was bedeutet, dass diese für standardisierte Projekte verwendet werden können, oder sie werden einer speziellen Gruppe zugeordnet, um die Einträge nur in bestimmten Projektphasen einzusetzen. Um nun eine neue projektspezifische Checkliste zu erstellen, fügen die Verantwortlichen die Basisgruppe der Einträge hinzu und konfigurieren diese. Zusätzlich werden spezielle Einträge für dieses individuelle Projekt ausgewählt. Schließlich wird die fertiggestellte Liste in eine EXCEL-Datei exportiert und auf einem Laufwerk für die zuständigen Entwickler zur Verfügung gestellt. Die manuelle Bearbeitung der Checklisten durch den Entwickler ist nicht optimal und zeitintensiv [Mun13].

Wie schon beschrieben, werden in diesem Anwendungsfall die Einträge der Basisgruppe entweder als standardisierte Vorlagen verwendet oder in einem weiteren Schritt zusätzlich spezialisiert. Dieser Aspekt unterstützt die Wiederverwendbarkeit, was einen Vorteil für das Checklisten-Management darstellt. Im folgenden Kapitel 2.5 werden die zwei Begriffe *Typ* und *Instanz* erläutert, die exakt dieses Vorgehen beschreiben und im folgenden Kontext relevant sind.

2.5 Checklisten-Management

Die im vorherigen Kapitel 2.4 beschriebenen Anwendungsfälle verwenden Begriffe wie z.B. Checkliste oder Eintrag. Bisher wurden diese Begriffe noch nicht explizit definiert, weshalb im Folgenden die Begriffe organisatorischer Rahmen, Checkliste und Eintrag erläutert werden. Diese sind notwendig um den weiteren Kontext der Arbeit zu verstehen. Beginnend beim organisatorischen Rahmen wird jede dieser Bezeichnungen zunächst definiert und daraufhin in einer Beschreibung explizit erläutert.

2.5.1 Organisatorischer Rahmen

Beginnend mit der folgenden *Definiton 2.1* wird der Begriff organisatorischer Rahmen erläutert und der Bezug zu den folgenden *Definitionen 2.2* und *2.3* wird hergestellt.

Definition 2.1 Organisatorischer Rahmen:
Ein *organisatorischer Rahmen (OR)* wird über die kollaborative Wissensarbeit charakterisiert. Die Zusammenarbeit der Wissensarbeiter erfolgt stets in einem OR, durch den bestimmte Randbedingungen und insbesondere Koordinationsaspekte, wie beispielsweise Verantwortlichkeiten für die kollaborative Wissensarbeit festgelegt werden. Er fungiert als Kontext für untergeordneten Checklisten und Einträge und kann selbst weitere ORs enthalten. Im Sprachgebrauch ist bei einem OR oft die Rede von einem Projekt oder einem Fall.

Es bedarf einer genaueren Definition des Begriffes OR, da er durch verschiedene Bedeutungen differenziert werden kann. In der Literatur finden sich wenige Definitionen für den Begriff OR, die in diesem Kontext anwendbar sind. U.a. definiert der Duden den OR als „*etwas, was einer Sache ein bestimmtes [äußeres] Gepräge gibt* “ [Dud13]. Er kann möglicherweise als Projekt oder als Fall fungieren und legt für jede der beiden Varianten den Kontext fest. Ihm können Checklisten, Einträge oder weitere OR untergeordnet werden. Besitzt ein OR weitere untergeordnete ORs, handelt es sich meist um ein Großprojekt, welches weitere Teilprojekte beinhaltet. Ein weiterer wichtiger Aspekt wird schon im Anwendungsfall in Kapitel 2.4.4 beschrieben. Hier werden Checklisteneinträ-

ge der Basisgruppe entweder als standardisierte Vorlagen verwendet oder in einem weiteren Schritt zusätzlich spezialisiert. Diese Vorgehensweise kann ebenso auf den organisatorischen Rahmen übertragen werden, wobei dieser Aspekt die Wiederverwendbarkeit unterstützt, was einen Vorteil für das Checklisten-Management darstellt. Im Folgenden werden die zwei Begriffe *organisatorischer Rahmentyp (ORT)* und *organisatorische Rahmeninstanz (ORI)* erläutert, die exakt dieses Vorgehen beschreiben und im folgenden Kontext relevant sind.

Organisatorischer Rahmentyp:
Der *organisatorische Rahmentyp (ORT)* entspricht einer Vorlage, die Standardangaben für den OR anbietet. Diese Angaben sind generisch, allgemeingültig, entsprechen bestimmten Parametern und können beliebig geändert werden. Die Existenz von ORTs bietet einige Vorteile, wie beispielsweise die Wiederverwendbarkeit oder die Zeitminimierung bei der Erstellung von den jeweiligen organisatorischen Rahmeninstanzen. Basierend auf einem ORT können die Nutzer eine organisatorische Rahmeninstanz erstellen.

Organisatorische Rahmeninstanz:
Die *organisatorische Rahmeninstanz (ORI)* ist entweder eine individuelle und auf den Kontext passende Spezifizierung der Daten, die sich von den bestimmten Parametern des ORT unterscheiden, oder eine individuelle Angabe der Parameter, ohne auf einen ORT zurückzugreifen. Wird eine ORI erzeugt, die nicht auf einem ORT basiert, kann der Nutzer beliebige Parameter angeben, die exakt auf seine Anforderungen passen.

2.5.2 Checkliste

Die *Checkliste* wird schon in den Anwendungsfällen (siehe Kapitel 2.4) als Werkzeug für die Unterstützung der Fähigkeiten von Fachkräften, wie z.B. Piloten oder Ärzten, beschrieben. In diesem Abschnitt wird der Begriff der Checkliste in *Definiton 2.2* vorgestellt und im Folgenden näher beschrieben. Des Weiteren wird der Bezug zu *Definition 2.3* hergestellt.

Definition 2.2 Checkliste:
Die *Checkliste (CL)* ist eine Sonderform einer Liste. Sie ist stets einem OR zugeordnet und besteht aus mindestens einem Eintrag. Anhand von Checklisten werden Aktivitäten in Prozessen koordiniert sowie auf Vollständigkeit und Reihenfolge geprüft. Dies erleichtert systematisches und fehlerfreies Arbeiten, erhöht die Aufmerksamkeit der Beteiligten und ermöglicht eine frühzeitige Fehleranalyse.

Es existiert eine große Menge an Definitionen für den Begriff CL, aber keine allgemeingültige. Jedoch bringen die Definitionen überwiegend dieselbe Verwendung und denselben funktionalen Hintergrund zum Ausdruck. Oftmals werden CLs in der Qualitätssicherung verwendet, um Arbeitsabläufe und deren Qualität zu kontrollieren. Durch ihre Anwendung werden zusätzlich Schwachstellen und Fehler entdeckt, die zu einer Optimierung der bestimmten Arbeitsabläufe führen. Das Lexikon der Informatik und Datenverarbeitung definiert eine CL wie folgt: „*In der Schwachstellenanalyse oft angewendet, womit mögliche Schwachstellen mit Hilfe von weitergehenden detaillierten Einzelfragen relativ rasch aufgefunden werden können*“ [Sch91]. Dieses Zitat bringt ebenfalls das Ziel der Schwachstellenminimierung und Qualitätssicherung durch CLs zum Ausdruck. Die Überprüfung der Abläufe erfolgt durch eine Auflistung von Teilaspekten, der sogenannten Einträge, die in Kapitel 2.5.3 definiert werden. Dies impliziert, dass CLs ein oder mehrere Einträge besitzen. Des Weiteren sind sie Bestandteil des ORs, der im vorherigen Kapitel 2.5.1 erläutert wurde. Ebenso wie die ORs, unterteilt sich die Checkliste in zwei Varianten, den *Checklistentyp (CLT)* und die *Checklisteninstanz (CLI)*. Diese Begriffe werden im Folgenden beschrieben.

Checklistentyp:
Der *Checklistentyp (CLT)* entspricht einer Vorlage, die Standardangaben für eine CL anbietet. Allgemeingültige CLs können zu einem CLT zusammengefasst werden, bei dem verschiedene Parameter und möglicherweise auch Einträge (siehe Kapitel 2.5.3) definiert sind. Diese Spezifikationen können nach der Ableitung eines CLTs individuell in einer Checklisteninstanz erweitert oder angepasst werden.

Checklisteninstanz:
Die *Checklisteninstanz (CLI)* ist entweder von einem CLT abgeleitet und enthält dessen allgemeingültige Parameter, oder sie wurde manuell erstellt, ohne dass sie Vorgaben besitzt.

2.5.3 Eintrag

Der *Eintrag* wird erstmals im Anwendungsfall der Automobilbranche (siehe Kapitel 2.4.4) verwendet. Um den Begriff einheitlich und explizit für den folgenden Kontext zu definieren, erfolgt in diesem Kapitel eine Definierung des Begriffes Eintrag (siehe *Definiton 2.3*), welche daraufhin ausführlicher beschrieben wird.

Definition 2.3 Eintrag:
Der *Eintrag* ist eine ausformulierte Aussage oder Frage, welche Bestandteil einer Checkliste ist und von einer oder mehreren Personen bewertet werden soll. Wird ein Eintrag als erledigt gekennzeichnet, bedeutet dies, dass die Aussage des Eintrags, meist ein Prozessschritt, in der Vergangenheit erfüllt oder korrekt ausgeführt wurde.

Der Eintrag ist ein Element einer CL (siehe Kapitel 2.5.2), sowie indirekt eines ORs (siehe Kapitel 2.5.1). Er ist typischerweise ein Teilaspekt eines Arbeitsablaufes, der überprüft werden soll. Insgesamt gliedert sich der Aufbau eines Eintrags in eine Aussage oder Frage und in eine Antwortvorgabe. Die Aussage oder Frage spiegelt inhaltlich den zu kontrollierenden Prozess wider, der durch die Antwortvorgabe bewertet wird. Wurde diese Aussage oder Frage erledigt, kann der Eintrag, durch die Antwortvorgabe gekennzeichnet werden. Ist das Gegenteil der Fall, wird die Antwortvorgabe nicht bearbeitet, was somit bedeutet, dass die Aussage oder Frage in der Vergangenheit noch nicht korrekt erledigt wurde. Darüber hinaus werden mit diesem Prinzip der Kontrolle effizient Fehlerquellen definiert und Arbeitsprozesse verbessert. Schon im Anwendungsfall der Automobilbranche in Kapitel 2.4.4, wurde beschrieben, dass bei einem Eintrag zwischen einer sogenannten Basisgruppe und einer spezialisierten Gruppe von Einträgen unterschieden wurde. Dieses Prinzip der Unterscheidung von zwei Varianten eines Eintrags wird in dieser Arbeit weiterverfolgt. Aufgrund dessen werden im Folgenden die Begriffe

Eintragstyp (ET) und *Eintragsinstanz (EI)* erläutert.

Eintragstyp:
Der *Eintragstyp (ET)* entspricht einer Vorlage, die Standardangaben für einen Eintrag anbietet. Diese ETs können für mehrere Anwendungsfälle verwendet werden, da sie allgemeingültig sind und meist Basisangaben enthalten, die auf mehrere EIs zutreffen.

Eintragsinstanz:
Die *Eintragsinstanz (EI)* ist entweder von einem ET abgeleitet und enthält dessen allgemeingültige Parameter, oder sie wurde manuell erstellt, ohne dass sie Vorgaben besitzt. Die EI ist exakt und detailliert für den bestimmten Einsatz bzw. Anwendungsfall spezifiziert. Sie enthält keine allgemeingültigen Parameter, sondern auf den Kontext individuell passende Angaben.

2.6 Unterschiede zu anderen Listentypen

Anhand der eingeführten Definitionen 2.1 bis 2.3 und den Anwendungsfällen aus Kapitel 2.4 kann man erkennen, dass der Begriff Checkliste auf verschiedene Arten interpretiert wird. Eine vollständige und einheitliche Definition, bei der alle Gesichtspunkte betrachtet werden, existiert nicht. Stattdessen gibt es oft domänenspezifischen Unterschiede. Im deutschen Sprachgebrauch wird der Begriff Checkliste fälschlicherweise als Synonym für andere Formen einer Liste verwendet, welche sich jedoch in einigen Aspekten stark von der CL unterscheiden. Um Missverständnissen vorzubeugen, werden im Folgenden die Unterschiede zwischen einer To-do-Liste und einer CL erläutert.

2.6.1 Abgrenzung zur To-do-Liste

Die To-do-Liste, auch Aufgabenliste genannt, hat das Ziel die Aufgaben einer Person zu organisieren und festzuhalten. Die Frage die hinter einer solchen Liste steckt ist offensichtlich „Wer macht was bis wann?“. Aus diesem Grund wird sie oft als sogenannter *Reminder* eingesetzt. Es werden Aufgaben, die in der Zukunft noch zu erledigen sind, in

einer Liste festgehalten. Der wesentliche Unterschied zur CL, die zur Überprüfung dient, liegt folglich in der Zielsetzung der Listen. Im zeitlichen Bezug weist die CL darüber hinaus einen retrospektiven und die To-do-Liste, im Unterschied, einen prospektiven Charakter auf. Dies bedeutet, dass die Einträge, die in einer CL aufgeführt sind, bereits in der Vergangenheit abgearbeitet werden sollten. Über die CL wird geprüft, ob dies wirklich erfolgte bzw. in der angestrebten Qualität erfolgte. In einer To-do-Liste sind die aufgeführten Punkte in der Zukunft noch zu erledigen. Ein weiterer elementarer Aspekt einer CL ist, dass die aufgelisteten Einträge nach einer bestimmten Reihenfolge geprüft werden. Bei einer To-do-Liste hingegen können Zeitlimits bei einem oder mehreren Aufgaben gesetzt werden, wobei diese jedoch nicht immer benötigt werden. Diese Zeitlimits betreffen meistens nur einen Eintrag und nicht die komplette Liste. Daraus folgt, dass die To-do-Liste bezüglich der Reihenfolge variabler ist. Neben der Differenzierung zur To-do-Liste, ist es nötig einige Unterscheidungen zwischen verschiedenen Modellen einer CL zu charakterisieren. Im folgenden Kapitel 2.7 werden diese vorgestellt.

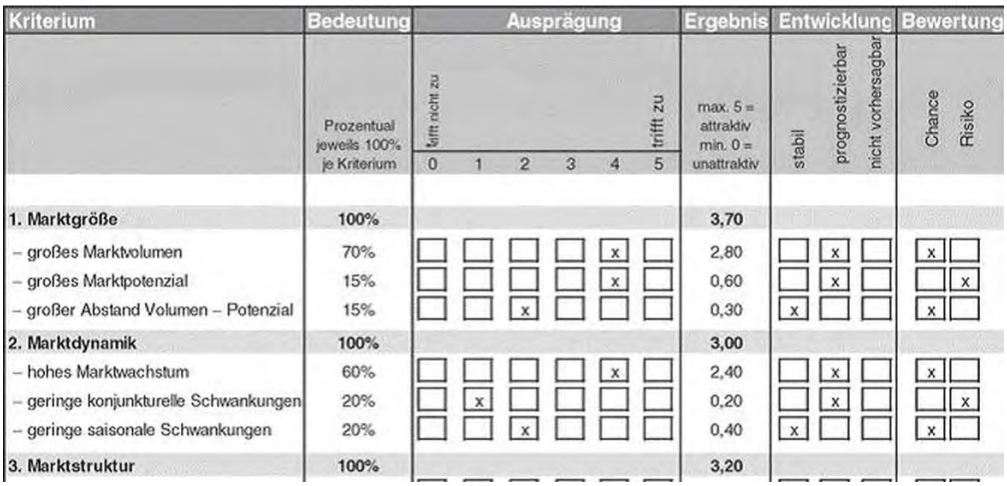
2.7 Verschiedene Modelle einer Checkliste

Oftmals unterscheidet sich die Struktur einer CL im Aufbau ihrer Einträge (siehe Kapitel 2.5). Die Einträge wiederum, bestehen aus einer Frage und einer Antwortvorgabe, wobei hier der Aufbau und die Struktur abhängig von den verschiedenen Modellen sind. Die Modelle unterscheiden sich größtenteils in der Formulierung der Fragestellung und den Antwortvorgaben. Die zwei Varianten Zahlenmodell und Kästchenmodell werden in den weiteren Kapiteln vorgestellt, da diese am weitesten verbreitet sind und am meisten genutzt werden [Ska06].

2.7.1 Zahlenmodell

Das Zahlenmodell beschränkt sich nicht auf Fragen, die mit *Ja* oder *Nein* beantwortet werden können, sondern auf Schätzfragen. Diese Schätzfragen brauchen als Antwortvorgaben ein Intervall von mehreren Antwortmöglichkeiten, die bestimmte Wertigkeiten besitzen. Bei diesem Modell sollte klar sein, welche Größe das Antwortintervall besitzt

und welche Zahlen als Wertigkeiten erlaubt sind. In den meisten Fällen sind nur natürliche Zahlen zugelassen. Um fehlerhafte Antworten zu vermeiden, sollte am Beginn einer CL die Bedeutung der verschiedenen Zahlen erwähnt sein [Ska06]. Ein Beispiel für ein Zahlenmodell wird in Abbildung 2.5 grafisch dargestellt.



Kriterium	Bedeutung	Ausprägung						Ergebnis	Entwicklung			Bewertung	
	Prozentual jeweils 100% je Kriterium	trifft nicht zu					trifft zu	max. 5 = attraktiv min. 0 = unattraktiv	stabil	prognostizierbar	nicht vorhersagbar	Chance	Risiko
		0	1	2	3	4	5						
1. Marktgröße	**100%**							**3,70**					
– großes Marktvolumen	70%					x		2,80		x		x	
– großes Marktpotenzial	15%					x		0,60		x			x
– großer Abstand Volumen – Potenzial	15%			x				0,30	x			x	
2. Marktdynamik	**100%**							**3,00**					
– hohes Marktwachstum	60%					x		2,40		x		x	
– geringe konjunkturelle Schwankungen	20%		x					0,20		x			x
– geringe saisonale Schwankungen	20%			x				0,40	x			x	
3. Marktstruktur	**100%**							**3,20**					

Abbildung 2.5: Beispiel eines Zahlenmodells [Mar13]

2.7.2 Kästchenmodell

Das am häufigsten verwendete Modell für CLs ist das Kästchenmodell, welches schon im Anwendungsfall in Kapitel 2.4.2 verwendet wird. Die Einträge sind bei diesem Modell Aussagen oder Entscheidungsfragen. Diese Variante einer CL bildet als Antwortvorgabe ein Quadrat ab, welches gekennzeichnet wird, falls die dazugehörige Frage erledigt wurde (siehe Abbildung 2.3, Kapitel 2.4.2). Diese Form der Antwortvorgabe ist geschlossen und kann nur mit einem Haken als geprüft gekennzeichnet werden.

Eine Variante des Kästchenmodells ist das Ja-Nein-Modell (siehe Kapitel 2.4.3). Die Fragen sind gleicher Art, jedoch ändert sich die Antwortvorgabe. Nun wird nicht nur ein Kästchen verwendet, sondern zwei, wobei diese für *Ja* und *Nein* stehen. Somit ist es leichter zu erkennen, ob Einträge vergessen wurden oder der bezügliche Eintrag nicht

erledigt wurde. Wird ein solches Quadrat mit *Ja* gekennzeichnet, bedeutet dies, dass die Aufgabe vollständig und gemäß den Vorgaben erledigt wurde. Ist dies nicht der Fall, wird das *Nein-Feld* angekreuzt (siehe Abbildung 2.4, 2.4.3). Durch die Nominalskalierung können negative Antworten sofort als Fehler erkannt werden. Möglicherweise ist ein Eintrag in bestimmten Situationen nicht relevant, weshalb manche CLs noch ein *Nicht zutreffend Feld* (N.Z.-Feld) enthalten. Tritt der genannte Fall ein, dass für einen konkreten Rahmen eine Frage nicht passend ist, so wird das *N.Z.-Feld* gekennzeichnet. Dies hilft bei der Differenzierung zwischen einem Fehler und einer nicht passenden Frage [Ska06].

3

Anforderungsanalyse

Aufbauend auf den fundamentalen Erkenntnissen werden in diesem Kapitel die Anforderungen für die Applikation systematisch vorgestellt und analysiert, da sie das Fundament für Kapitel 4 bilden. Es werden relevante Informationen für die spätere Entwicklung und Gestaltung erarbeitet. Zunächst wird in Kapitel 3.1 anhand von verschiedenen Konkurrenzprodukten der Ist-Stand analysiert, um die erlangten Informationen in die Entwicklung einfließen zu lassen. Daraufhin wird eine Benutzerprofilanalyse durchgeführt, die in Kapitel 3.2 erläutert wird. Diese Analyse beschränkt sich auf die späteren Benutzer und ihre Eigenschaften. Anschließend wird im Kapitel 3.3 auf die verschiedenen Aufgaben eingegangen, die das System erfüllen muss. Diese Aufgaben werden durch Flussdiagramme visualisiert. In den letzten Kapiteln 3.4 und 3.5 werden die Umgebungsbedingungen und die Hardware und Software Randbedingungen dokumentiert. Einen Überblick über alle Kapitel bietet Abbildung 3.1.

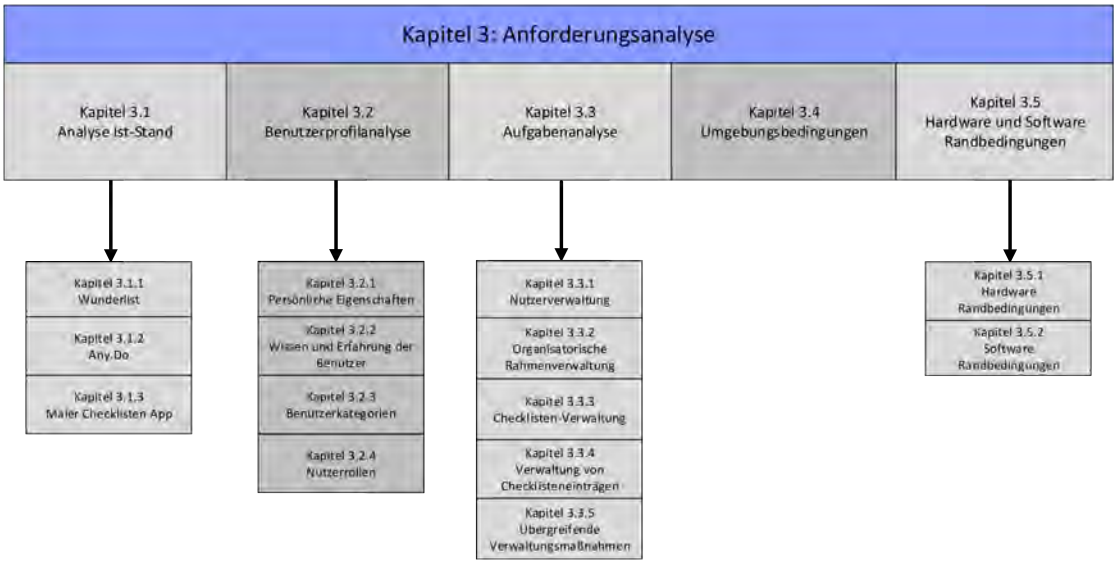


Abbildung 3.1: Aufbau des Kapitels Anforderungsanalyse

3.1 Analyse Ist-Stand

Die Analyse Ist-Stand beschäftigt sich mit dem aktuellen Zustand von Konkurrenz- oder Altprodukten. Es ist sinnvoll verschiedene Konkurrenzsysteme zu analysieren, um wertvolle Erkenntnisse und Informationen zu gewinnen, die dann in die Neuentwicklung einfließen können [Off12]. Obwohl fast alle der folgenden Konkurrenzprodukte Applikationen sind, welche To-do-Listen verwalten (siehe Kapitel 2.6.1), werden diese analysiert. Grund für die Analyse ist, trotz semantischer Unterschiede, die Ähnlichkeit der zu entwickelnden Software, sowie das Fehlen sonstiger, vergleichbarer Checklisten-Management Applikationen. Im Folgenden werden die Konkurrenzprodukte beschrieben, wobei der primäre Fokus der Beschreibung auf den visuellen Aspekten und den damit verbundenen Funktionen der Applikation liegt.

3.1.1 Wunderlist

Das erste Konkurrenzprodukt ist Wunderlist und stellt eine kostenlose Anwendung dar, mit der Nutzer ihre Aufgaben und Notizen managen können [Wun13b]. Zu Beginn wird

vom Nutzer eine Anmeldung über E-Mail und Passwort verlangt. Nach dieser stehen dem Anwender die eigentlichen Funktionen zur Verfügung. In einem Seitenfenster werden die Namen bereits erstellter Aufgabenlisten angezeigt und es können beliebig viele neue Aufgabenlisten hinzugefügt werden (siehe Markierung 1, Abbildung 3.2). Dies erfolgt indem man unterhalb der letzten Aufgabenliste auf die Zeile „*Liste hinzufügen...*“ tippt (siehe Markierung 2, Abbildung 3.2).

Abbildung 3.2: Beispieldialoge der Anwendung *Wunderlist* [Wun13a]

Diese Operation ist durch ein Plus versehen, was dem Nutzer suggerieren soll, dass an dieser Stelle etwas hinzugefügt werden kann. Jede der angezeigten Aufgabenlisten besitzt außerdem ein kleines Icon, auf welchem eine Büste mit einem Plus zu sehen ist. Durch berühren des Icons kann man Aufgabenlisten mit Personen teilen, falls verschiedene Personen die Einträge bearbeiten sollen. Mit einer Fingerbewegung von rechts nach links, verschwindet das Seitenfenster und die ausgewählte Aufgabenliste wird angezeigt (siehe Markierung 3, Abbildung 3.2). Hier kann ein Nutzer seine zugehörigen Einträge festlegen, indem er diese in ein Textfeld einträgt, das sich oberhalb befindet (siehe Markierung 4, Abbildung 3.2). Aufgabenlisten können per Facebook, E-Mail oder

beispielsweise Google+ geteilt werden. Des Weiteren kann man in dieser Ansicht ebenfalls befreundete Nutzer einladen. Möchte der Nutzer nun eine weitere Aufgabenliste bearbeiten, sollte er mit einer Fingerbewegung von links nach rechts das Seitenfenster wieder sichtbar machen. Nun kann eine andere beliebige Aufgabenliste ausgewählt werden, welche ebenfalls dieselben Funktionen enthält. Die Einträge einer Aufgabenliste besitzen ein Kästchen als Antwortvorgabe (siehe Kapitel 2.7.2). Wird dieses aktiviert, markiert Wunderlist den Eintrag als erledigt, visualisiert diesen als durchgestrichen und ordnet ihn abgegrenzt unterhalb an (siehe Markierung 5, Abbildung 3.3). Um den Vorgang rückgängig zu machen, genügt die wiederholte Berührung des Kästchens. Hält man den Finger etwas länger gedrückt auf einen Eintrag, so wird in der rechten Ecke ein Mülleimer dargestellt (siehe Markierung 6, Abbildung 3.3). Die Kästchen verändern in diesem Schritt ihre Form und werden nun rund angezeigt. Betätigt man die Radiobuttons, werden sie blau gefärbt und können mit dem Mülleimersymbol entfernt werden (siehe Markierung 7, Abbildung 3.3). Wird ein Haken im linken oberen Eck angetippt, kommt man zurück und die ausgewählten Elemente sind gelöscht (siehe Markierung 8, Abbildung 3.3). Zusätzlich zu den Namen der Einträge, können Nutzer Teilaufgaben hinzufügen, Notizen schreiben und Erinnerungen einstellen. Mit einem einfachen Klick auf einen Eintrag, stehen diese Optionen zur Verfügung und der Name der Aufgabenliste kann geändert werden. Schließlich haben Nutzer die Möglichkeit ihren Eintrag mit einer höheren Priorität zu belegen, indem sie einen kleinen Stern aktivieren, der an jedem Eintrag angeheftet ist (siehe Markierung 9, Abbildung 3.3). Einträge mit höherem Rang werden an den Anfang der Aufgabenliste gesetzt und mit einem roten Stern versehen, um ihre Wichtigkeit zu verdeutlichen (siehe Markierung 10, Abbildung 3.3) [Wun13b].

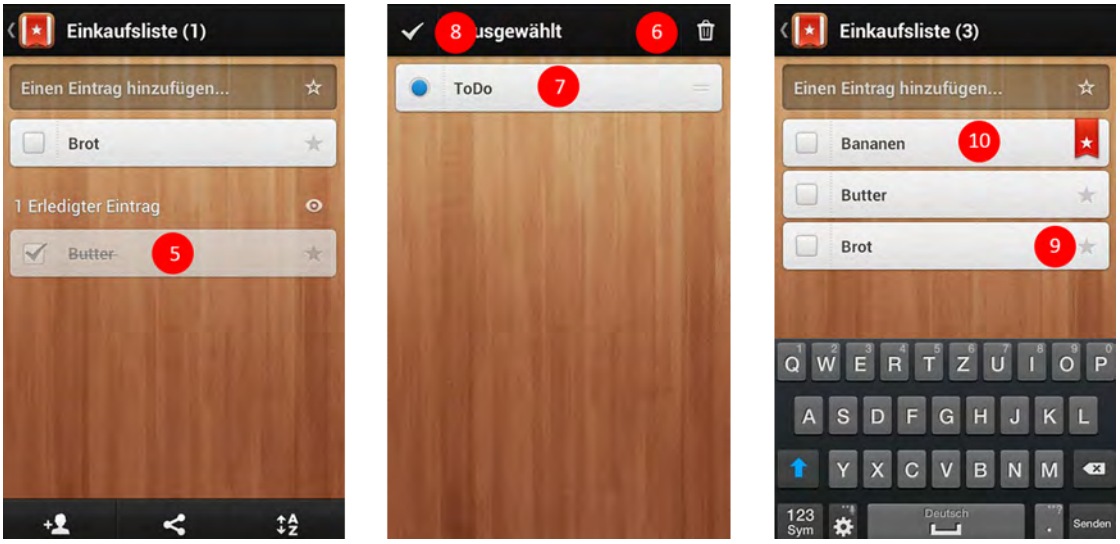


Abbildung 3.3: Weitere Beispieldialoge der Anwendung *Wunderlist*

3.1.2 Any.DO

Any.DO ist eine weitere App, mit der verschiedene Aufgaben verwaltet werden können [Any13b]. Um die App nutzen zu können, erfolgt eine Registrierung mit dem Namen, der E-Mail-Adresse und einem Passwort. Ist dieser Vorgang abgeschlossen, kann man in ein Textfeld Aufgaben eintragen (siehe Markierung 1, Abbildung 3.4). Diese können jeweils in die vier zeitliche Kategorien, „*Heute*“, „*Morgen*“, „*In Kürze*“ und „*Irgendwann*“ eingeteilt werden (siehe Markierung 2, Abbildung 3.4). Des Weiteren kann man von Termindaten zu Ordner wechseln, wobei sich die dynamischen Listen „*Heute*“, „*Morgen*“, „*In Kürze*“ und „*Irgendwann*“ zu „*Persönlich*“ und „*Geschäftlich*“ ändern (siehe Markierung 3, Abbildung 3.4). Möchte man Einträge einer Kategorie unterordnen, sollte auf das Plus getippt werden (siehe Markierung 4, Abbildung 3.4). Nun wird die Anzahl der bereits eingetragenen Einträge angezeigt. Über das Textfeld können nun weitere Einträge angelegt werden. Wurde eine Aufgabe erledigt, muss der Nutzer nur von links nach rechts seinen Finger über den Eintrag ziehen. Dabei wird dieser durchgestrichen und somit als erledigt gekennzeichnet. Jedoch werden diese nicht visuell von den noch zu erledigenden Einträgen getrennt (vgl. Kapitel 3.1.1).

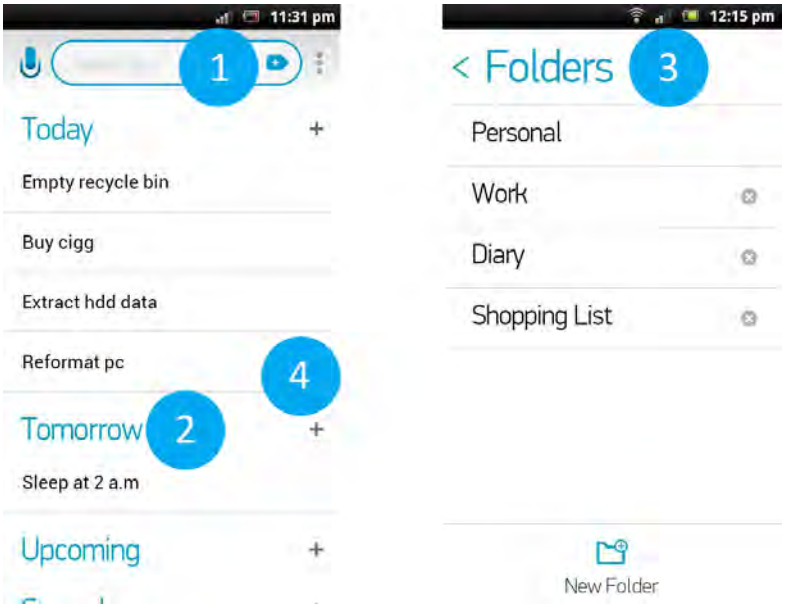


Abbildung 3.4: Beispieldialoge der Anwendung *Any.Do* [Any13a]

Soll der erledigte Eintrag gelöscht werden, muss das kleine „*X-Icon*“ am rechten Rand des zugehörigen Eintrags betätigt werden. Um Einträge zu bearbeiten reicht ein berühren des Punktes und ein kleines Menü wird unterhalb angezeigt. Hier bestehen die Möglichkeiten den Namen zu ändern (siehe Markierung 5, Abbildung 3.5), Prioritäten zu setzen (siehe Markierung 6, Abbildung 3.5), Erinnerungen festzulegen (siehe Markierung 7, Abbildung 3.5) oder Notizen anzulegen (siehe Markierung 8, Abbildung 3.5) [Any13b].

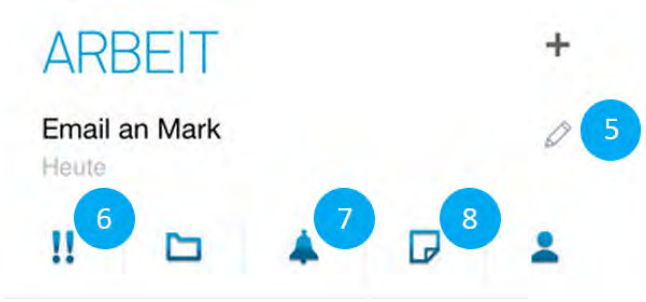


Abbildung 3.5: Weitere Beispieldialoge der Anwendung *Any.Do* [Any13a]

3.1.3 Maler Checklisten App

Die Maler Checklisten App wurde als Unterstützung für Personen des Maler Berufs konzipiert. Nach der Installation werden Nutzer nach ihren Stammdaten gefragt. Anschließend kann man im rechten oberen Eck auf den Button „*Speichern* “ klicken und sofort mit der Bearbeitung von CLs beginnen. Das Hauptmenü bietet nun vier Optionen, die ausgewählt werden können, „*Projekte* “, „*Checklisten* “, „*Stammdaten* “ und „*Mehr*“ (siehe Abbildung 3.6). Das Hauptmenü bietet den Nutzer die Möglichkeit einen OR zu erstellen (siehe 2.5.1), was durch die Option „*Projekte* “ umgesetzt wird. Um ein Projekt anzulegen, sollte der Nutzer den Button „*Projekte* “ im Hauptmenü wählen. Hier kann man einen Namen hinzufügen, sowie die Kundendaten, verschiedene Bemerkungen und im letzten Abschnitt kann man Mitarbeiter projektspezifisch zuordnen. Diese Informationen trägt der Benutzer in Textfelder ein. Letztendlich werden in diesem Schritt die CLs für das angewählte Projekt angelegt. Drückt der Ersteller den Button „*Checklisten hinzufügen*“, werden ihm Checklistentypen angezeigt (siehe Kapitel 2.5.2). Hier ist es möglich, mehrere vorgefertigte Checklistentypen zu verwenden und diese in einer CL zusammenzufassen.

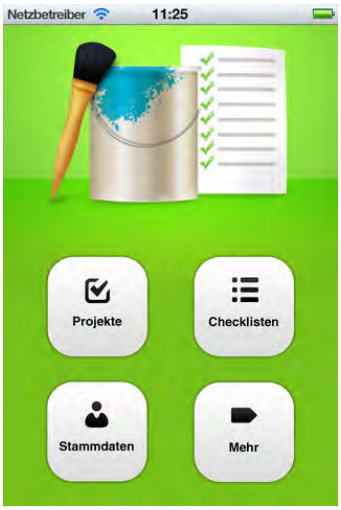


Abbildung 3.6: Startbildschirm der Anwendung *Maler Checklisten App* [Mal13a]

Möchte man noch nicht vorhandene Einträge einfügen, kann man in der Beschreibung der Einträge auf ein Plus klicken. Das Klicken öffnet ein Textfeld, in welches man den erforderlichen Eintrag einträgt. Zu jeder dieser Einträge kann der Nutzer außerdem - falls nötig - eine Anmerkung schreiben. Um den Prozess zu beenden, muss der Button „*Übernehmen*“ gedrückt werden. Daraufhin zeigt die App die ausgewählten CLs und gibt dem Nutzer die Möglichkeit, diese CLs zu ändern. Wird der Button „*Speichern*“ gedrückt, kann das erstellte Projekt in PDF-Format angezeigt werden. Des Weiteren kann im Hauptmenü der Button Checklisten gewählt werden. Hier werden alle gespeicherten vorgefertigten CLs verwaltet. Um eine neue CL zu erstellen sollte man den Button Neu antippen. Nun können Nutzer den Namen und die Einträge angeben. In demselben Modus können sie nun bearbeitet oder gelöscht werden [Mal13b].

Insgesamt wurden in diesem Kapitel drei verschiedene Konkurrenzprodukte vorgestellt. Beginnend mit der Applikation *Wunderlist*, die sich inhaltlich mit To-do-Listen beschäftigt, wurden die visuellen Aspekte und die Funktionen betrachtet. Daraufhin wurde die mobile Anwendung *Any.DO* vorgestellt, deren Funktionen der Applikation *Wunderlist* ähneln. Im Unterschied zu diesen beiden Applikation wurde das letzte Konkurrenzprodukt, die *Maler Checklisten App* beschrieben, welche sich mit Checklisten beschäftigt und somit weitere mögliche Funktionen bietet, die in diesem Kontext relevant sind. Diese visuellen Aspekte und Funktionen bilden die Basis, für die Analyse der Aufgaben (siehe Kapitel 3.3), der zu entwickelnden Anwendung.

3.2 Benutzerprofilanalyse

Im Rahmen einer Benutzerprofilanalyse werden die späteren Nutzer des Systems analysiert, um ihre Eigenschaften zu identifizieren. Diese Analyse liefert relevanten Mehrwert für die Aufgabenanalyse (siehe Kapitel 3.3) und die Entwurfsphase (siehe Kapitel 4), indem sie neue Erkenntnisse im Bezug auf die Gestaltung der Benutzeroberflächen liefert. Persönliche Eigenschaften, Wissen und Erfahrung der Benutzer, sowie die Unterscheidung verschiedener Benutzerkategorien spielen hier eine große Rolle [Nie93]. Des

Weiteren werden die verschiedenen Nutzergruppen und Nutzerrollen erläutert und mit den Benutzerkategorien assoziiert.

3.2.1 Persönliche Eigenschaften

Die persönlichen Eigenschaften der Nutzer sind im Bezug auf die spätere Gestaltung der Oberflächen relevant. Sie liefern Informationen, die die Benutzerfreundlichkeit und Anforderungen sicherstellen. Zu Beginn ist es wichtig, das Alter der späteren Nutzer zu kennen, um diese Information in die folgende Gestaltung der Benutzeroberflächen fließen zu lassen. Es wird davon ausgegangen, dass überwiegend Personen mittleren Alters die Applikation später nutzen, was zur Folge hat, dass Bedienelemente in Standardgrößen angezeigt werden können ohne das Menschen Probleme mit dem Sehen oder Tasten dieser Elemente haben.

3.2.2 Wissen und Erfahrung der Benutzer

Grundsätzlich ist es sinnvoll den Benutzer in Bezug auf seine Erfahrungen und sein Wissen zu kategorisieren. Die Analyse verschiedener Aspekte des späteren Benutzers, wie die Joberfahrung, die Smartphone Kenntnisse und die Benutzung ähnlicher Systeme, können relevanten Mehrwert für die Entwicklung liefern. Die Joberfahrung der Nutzer, sowie die Benutzung ähnlicher Systeme spielt hier jedoch eine untergeordnete Rolle, da die Eigenschaften nicht klar definiert und spezifiziert werden können. Es handelt sich um eine komplette Neuentwicklung der Applikation, die bisher nicht in digitalen Varianten existierte, weshalb der Aspekt Benutzung ähnlicher Systeme keine Informationen liefert. Bei den zukünftigen Benutzern kann jedoch davon ausgegangen werden, dass es sich um keine Laien im Umgang mit einem Smartphone handelt. Weiter wird angenommen, dass sie ihr Smartphone nicht sporadisch verwenden, was auf eine regelmäßige Nutzung schließen lässt. Daher erschließt sich, dass die späteren Benutzer vertraut mit den typischen Kontroll- bzw. Bedienelementen und den spezifischen Styleguides der Betriebssysteme sind, was dazu führt, dass für die Benutzer der Umgang mit einem Smartphone oder einer bestimmten Applikation sowie die Nutzung verschiedener Funk-

tionen der Applikation weitgehend selbstverständlich sind. Aufgrund der verschiedenen Aufgaben, die in Kapitel 3.3 erarbeitet werden, ergeben sich für die späteren Nutzer verschiedene Benutzerkategorien.

3.2.3 Benutzerkategorien

Um die Applikation bezüglich verschiedener Aspekte, wie z.B. Usability zu optimieren, ist es wichtig, die Benutzerkategorien zu analysieren. Aufgrund der verschiedenen Aufgaben und Ziele der Nutzer, weisen diese typischerweise jeweils unterschiedliches Benutzerverhalten auf. Dieses Verhalten äußert sich in der Nutzungshäufigkeit und ändert oftmals den Grad der Erfahrung der späteren Nutzer [Nie93]. Insgesamt werden die Nutzer damit in die vier nachstehenden Kategorien eingeteilt [Off12].

- Ungeübter und sporadischer Nutzer
- Ungeübter und regelmäßiger Nutzer
- Geübt und sporadischer Nutzer
- Geübt und regelmäßiger Nutzer

Für jede dieser Kategorien werden die Kriterien der Benutzerfreundlichkeit (BFK) unterschiedlich gewichtet. Diese erbringen wertvolle Informationen für die komfortable und benutzerfreundliche Gestaltung der Applikation. Im Anschluss werden die Kriterien der Benutzerfreundlichkeit erläutert und den jeweiligen Benutzerkategorien zugeordnet und bewertet [Off12]. Eine Übersicht über die Kriterien und ihre Wertung ist in Abbildung 3.1 dargestellt.

BFK 1: Lernförderlichkeit
Ein System ist erlernbar, wenn es die Nutzer bei allen zu erledigenden Aufgaben unterstützt. Hilfestellungen jeglicher Art leiten die Personen durch das System. Ebenfalls tragen sie dazu bei, dass die Nutzer sich mit dem System auseinandersetzen und einen gewissen Lerneffekt erfahren, welcher den Grad der Erfahrung steigert [ENI06].

BFK 2: Fehlertoleranz
Das Kriterium der Fehlertoleranz bedeutet, dass das erwünschte Ziel der Nutzer erreicht wird, trotz möglicher Fehleingaben seitens der Benutzer. Möglicherweise ist ein geringer Aufwand nötig, der die Eingaben korrigiert [ENI06].

BFK 3: Individualisierbarkeit
Anwendungen sind individualisierbar, wenn sie Anpassungen an die Vorlieben und Bedürfnisse der Benutzer zulassen [ENI06].

BFK 4: Komfort
Hoher Komfort gilt als Sekundäraufgabe eines Systems. Ein komfortables System unterstützt den Nutzer bei der Erledigung seiner Aufgaben und bietet ihm alle nötigen Informationen und Funktionen komfortabel an. Der Nutzer sollte während dem gesamten Zeitraum, in welchem er mit dem System arbeitet, nicht durch Bedienungsprobleme belastet werden [Off12].

BFK 5: Nützlichkeit
Das Kriterium der Nützlichkeit ist die Primäraufgabe der Anwendung. Eine Anwendung ist dann nützlich, wenn dem Nutzer alle Funktionen und Informationen angeboten werden, mit denen er seine Ziele erreichen kann [Off12].

BFK 6: Übersichtlichkeit
Eine Anwendung ist übersichtlich, wenn die Informationen strukturiert und unterscheidbar angeordnet sind. Die Orientierung und Lesbarkeit der Nutzer soll durch dieses Kriterium gefördert werden [Off12].

BFK 7: Verfügbarkeit
Ein System wird als verfügbar bezeichnet, wenn der Aspekt der Stabilität unter dem Kriterium der Systemausfälle maximiert wird. Ein weiterer Aspekt der Verfügbarkeit ist das Antwortzeitverhalten. Dieses sollte im Idealfall minimiert werden [Off12].

Der **ungeübte und sporadische Nutzer** wird als ein Nutzer beschrieben, der weniger als einen Monat Erfahrung im Bereich der elektronischen Datenverarbeitung hat und die Anwendung weniger als zwei Stunden täglich nutzt [Off12]. Als ungeübter und sporadischer Nutzer ist es wichtig, das Kriterium der Lernförderlichkeit und der Fehlertoleranz

als hoch zu gewichten. Die Erklärung für diese Gewichtung ist, dass ein unerfahrener Nutzer ein höheres Risiko von Fehleingaben aufweist, wobei die Anwendung auf diese angemessen reagieren und dem Nutzer einen Lösungsansatz für den Fehler anbieten muss. Im Gegenzug zu den genannten Kriterien, ist die Individualisierbarkeit gering bewertet. Da Nutzer aufgrund der sporadischen Nutzung noch keine individuellen Bedürfnisse entwickelt haben, richten sie deshalb ihren primären Fokus auf das Erreichen des gewünschten Zieles. Jeder Nutzer verfolgt das Ziel der komfortablen Aufgabenerledigung, unabhängig von der bestimmten Benutzerkategorie, was die Kriterien Nützlichkeit und Komfort widerspiegelt. Diese beiden Aspekte werden in allen Kategorien hoch gewichtet, da sie die Primär- und Sekundäraufgabe einer Anwendung sind. Im Rahmen dieser Benutzerkategorie ist es ebenfalls notwendig das Kriterium der Übersichtlichkeit hoch zu bewerten. Ein strukturierter und klarer Aufbau der Dialoge unterstützt die intuitive Bedienung durch den Nutzer. Dies wirkt sich positiv auf das Nutzerverhalten aus und reduziert die mentale Belastung des Nutzers. Das letzte Kriterium die Verfügbarkeit, wird als mittel eingestuft [Off12].

Besitzt ein Nutzer weniger als einen Monat Erfahrung im Bereich der elektronischen Datenverarbeitung und nutzt die Anwendung jedoch mehr als drei Stunden am Tag, wird er als **ungeübter und regelmäßiger Nutzer** kategorisiert. Da der Nutzer wenig Erfahrung hat, jedoch regelmäßig am System arbeitet, ist die Lernförderlichkeit und Fehlertoleranz mittel bewertet. Die regelmäßige Nutzung reduziert das Fehlerrisiko trotz der geringen Erfahrung, da der Nutzer bestimmte Fehleingaben bereits getätigt hat und diese mit geringerer Wahrscheinlichkeit wiederholt. Da der Nutzer durch seine regelmäßige Nutzung bereits Überblick über die Struktur und Anordnung der Dialogelemente hat, wird das Kriterium der Übersichtlichkeit als mittel eingestuft. Im Rahmen dieser Benutzerkategorie ist es ebenfalls notwendig das Kriterium der Verfügbarkeit hoch zu bewerten. Der Nutzer arbeitet regelmäßig mit der Anwendung, weshalb er mögliche Systemausfälle oder lange Antwortzeiten nicht als Zufall interpretiert. Treten Systemausfälle und lange Wartezeiten des öfteren auf, werden diese vom Nutzer nicht mehr toleriert und tragen zu einer negativen Bewertung der Anwendung, seitens der Nutzer bei [Off12].

Ein erfahrener Nutzer, der mehr als einen Monat Erfahrung im Bereich der elektronischen Datenverarbeitung besitzt und weniger als zwei Stunden am Tag mit der Anwendung arbeitet, wird in diesem als **geübter und sporadischer Nutzer** definiert. Diese Benutzerkategorie orientiert sich an der Kategorie **ungeübter und regelmäßiger Nutzer**. Außer der Lernförderlichkeit und der Verfügbarkeit, werden alle Kriterien der Benutzerfreundlichkeit gleich bewertet. Das Kriterium der Lernförderlichkeit wird hier als hoch bewertet und die Verfügbarkeit mittel. Die sporadische Nutzung führt dazu, das die Toleranzgrenze der Nutzer höher ist [Off12].

Der **geübte und regelmäßige Nutzer** besitzt mit mehr als einem Jahr Erfahrung in der elektronischen Datenverarbeitung das nötige Fachwissen, um die Anwendung effektiv und effizient zu nutzen. Dieser Nutzer ist ein Experte im Umgang mit der Anwendung und er besitzt zusätzlich noch das Fachwissen. Dies erklärt, weshalb das Kriterium der Lernförderlichkeit als gering bewertet wird. Bei dieser Nutzerkategorie befindet sich der Fokus nicht nur auf der Aufgabenerledigung, sondern auch darauf, diese schnell und einfach zu erledigen. Der Nutzer hat bereits Vorlieben und Bedürfnisse entwickelt, welche er für gewöhnlich anwenden möchte. Um diesen Aspekt zu erfüllen, wird das Kriterium der Individualisierbarkeit hoch gewichtet. Die weiteren Kriterien besitzen dieselbe Bewertung, wie die Kategorie des **ungeübten und regelmäßigen Nutzers** [Off12].

Kriterien der BFK:	Ungeübter und regelmäßiger Nutzer:	Ungeübter und sporadischer Nutzer:	Geübter und sporadischer Nutzer:	Geübter und regelmäßiger Nutzer:
Lernförderlichkeit	mittel	hoch	hoch	gering
Fehlertoleranz	mittel	hoch	mittel	mittel
Individualisierbarkeit	gering	gering	gering	hoch
Komfort	hoch	hoch	hoch	hoch
Nützlichkeit	hoch	hoch	hoch	hoch
Übersichtlichkeit	mittel	hoch	mittel	mittel
Verfügbarkeit	hoch	mittel	mittel	hoch

Tabelle 3.1: Gewichtung der BFKs im Bezug auf die Benutzerkategorien

3.2.4 Nutzerrollen

Auf Basis der vorgestellten Anwendungsfälle (siehe Kapitel 2.4) und der beschriebenen Konkurrenzprodukte in Kapitel 3.1 wird momentan vereinfacht angenommen, dass es insgesamt vier verschiedene Nutzerrollen gibt, die die Personen einnehmen können. Dazu zählen der Administrator, der Verwalter, der Nutzer und der Gast. Zwischen diesen Rollen existieren Abhängigkeiten, die in Abbildung 3.7 dargestellt sind. Das Schaubild erläutert die Vererbung der Funktionen und Aufgaben, die den Nutzern zur Verfügung stehen. Beginnend beim Administrator der alle Rechte besitzt, besitzen die weiteren Personen weniger Funktionenvielfalt und Rechte. Im Folgenden werden die erwähnten Nutzerrollen (NR) erläutert, um die theoretische Grundlage für die folgenden *User Stories* (siehe Kapitel 3.3) zu schaffen und insbesondere um die Dialoggestaltung den Bedürfnissen der jeweiligen Nutzerrollen anzupassen.

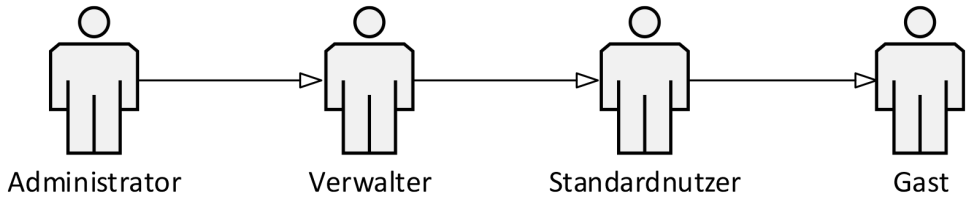


Abbildung 3.7: Abhängigkeiten der Nutzerrollen

NR 1: Administrator
Der Administrator verwaltet die Anwendung. Ihm stehen alle Funktionen zur Verfügung und er besitzt alle Rechte. Zusätzlich besitzt ein Administrator technisches Hintergrundwissen und ist mit allen möglichen Ausnahmesituationen vertraut. Da er täglich am System arbeitet und ein geschulter Nutzer ist, fällt er unter die Kategorie **geübter und regelmäßiger Nutzer**.

NR 2: Verwalter
Ein Nutzer nimmt automatisch die Rolle eines Verwalters ein, wenn er einen OR erzeugt oder instanziiert. Er ist für die Überwachung, Änderungen und die Zuordnung der Bearbeiter zuständig. Im schlechtesten Fall hat er keine Erfahrungswerte, was ihn als ungeübten Nutzer identifiziert. Nimmt ein Nutzer die Rolle der Verwalters des Öfteren

ein, kann man ihn als regelmäßigen und geübten Verwalter kategorisieren. Insgesamt kann man keine schlüssige Aussage bezüglich der Benutzerkategorie eines Verwalters treffen. Diese Rolle ist demnach abhängig von der Person, die sie einnimmt.

NR 3: Standardnutzer
Die Rolle des Standardnutzers nimmt ein Gast dann ein, wenn er sich erfolgreich angemeldet hat. Er ist zuständig für die Bearbeitung der ORs, CLs und Einträge. Da der Verwalter dem Standardnutzer verschiedenen Aufgaben delegiert, ist dieser mit seinem Wissen auf den jeweiligen Aufgabenbereich beschränkt. Ziel des Standardnutzers ist es, die zugeordneten Aufgaben schnell und effizient zu erledigen, was auf eine regelmäßige Nutzung hinweist. Domänenspezifisch ist der Standardnutzer in die Kategorie der geschulten Personen einzuordnen, weshalb er der Kategorie **geübt und regelmäßiger Nutzer** zugewiesen wird.

NR 4: Gast
Eine Person erhält die Rolle des Gastes, wenn er sich weder registriert noch angemeldet hat. Außer der Anmeldung oder Registrierung stehen ihm keinerlei Funktionen zur Verfügung.

3.3 Aufgabenanalyse

Die kontextuelle Aufgabenanalyse bildet die Grundlage zur erfolgreichen Entwicklung eines aufgabenangemessenen Systems. Um die Nützlichkeit einer Applikation zu steigern, sollten die Aufgaben klar definiert werden. Aufgrund dessen werden im folgenden Abschnitt die späteren Aufgaben mit Hilfe von Kapitel 3.1, anhand von ausformulierten *User Stories* (dt. Anwendererzählungen) erläutert. Diese User Stories werden sowohl textuell, als auch grafisch, in einem Flussdiagramm dargestellt. Die Flussdiagramme wurden mit der Notation aus Abbildung 3.8 erstellt.

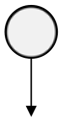
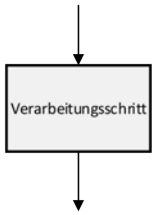
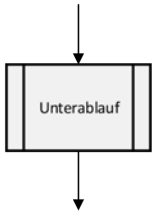
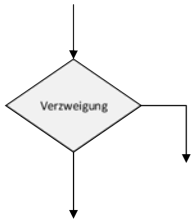
Symbol	Bezeichnung	Bedeutung
	Anfangsmarke	Startet einen Ablauf
	Endmarke	Beendet einen Ablauf
	Aktor	Ein Nutzer der auf das System einwirkt
	Verarbeitungsschritt	Legt die Reihenfolge des Ablaufs fest
	Unterablauf	Bezeichnet einen eigenen Ablauf, der wiederrum aus verschiedenen Verarbeitungsschritten besteht
	Verzweigung	Bezeichnet eine Spaltung des Kontrollflusses aufgrund einer Entscheidung

Abbildung 3.8: Definition der Notation für Flussdiagramme

Zusätzlich beantworten die textuell erfassten *User Stories*, die sogenannten Kontextfragen. Die Kontextfragen werden in einem Systementwicklungsprozess häufig verwendet, um die späteren Aufgaben zu analysieren [Off12]. Ziel ist es, die Kontextfragen, jeweils in Bezug auf eine Aufgabe des späteren Systems zu beantworten, um einen Gewinn an Mehrwert für die Entwurfsphase zu erzielen. Bezogen auf die Entwicklung der Applikation, sind die folgenden fünf Kontextfragen eine sinnvolle Auswahl, um die Aufgaben angemessen zu bewerten.

- Von wem wird die Aufgabe durchgeführt?
- Warum wird die Aufgabe durchgeführt?
- Was ist bei der Durchführung im Einzelnen zu tun?
- Wie oft wird die Aufgabe durchgeführt?
- Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Schließlich decken sich die *User Stories* inhaltlich mit den Kontextfragen, woraufhin diese in den folgenden Abschnitten integriert wurden. Die Kontextfragen *Von wem wird die Aufgabe durchgeführt?*, *Warum wird die Aufgabe durchgeführt?* und *Wie oft wird die Aufgabe durchgeführt?* sind jeweils in einer Tabelle grafisch zusammengefasst. Des Weiteren ist noch zu erwähnen, dass die Kontextfrage *Wie oft wird die Aufgabe durchgeführt?* unter einer Annahme beantwortet wird. Die folgenden *User Stories* gliedern sich in fünf Abschnitte, die Nutzerverwaltung, organisatorische Rahmenverwaltung, Checklisten-Verwaltung, Verwaltung von Checklisteneinträgen und die übergreifenden Verwaltungsmaßnahmen, welche jeweils inhaltlich sortiert sind.

3.3.1 Nutzerverwaltung

Die *Nutzerverwaltung (NV)* analysiert alle späteren Aufgaben, bei denen der Nutzer im Fokus steht. Die Abbildung 3.9 bildet ein Use-Case-Diagramm ab, welches einen Überblick über die darauffolgenden ausformulierten *User Stories* zeigt.

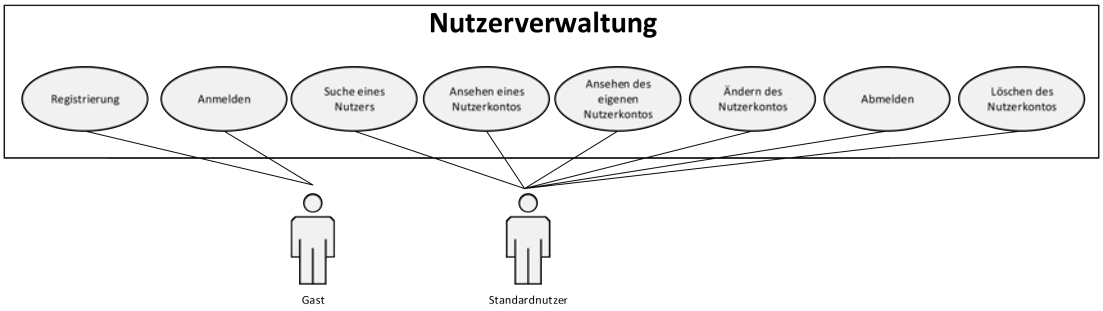


Abbildung 3.9: Use-Case-Diagramm der Nutzerverwaltung

NV 1: Registrierung

Der komplette Ablauf der Registrierung ist in Abbildung 3.10 dargestellt.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Gast	Um die Applikation und alle ihre Funktionalitäten zu benutzen	Einmal

Tabelle 3.2: Registrierung-NV1

Was ist bei der Durchführung im Einzelnen zu tun?

Der Gast muss auf den Button Registrieren klicken. Daraufhin wird dem Gast ein Dialog angezeigt, bei dem er seinen Nachnamen, Vornamen, das Passwort, welches einmal wiederholt werden muss, und die E-Mail-Adresse in die Textfelder eingeben muss. Des Weiteren wird vom Gast die Angabe, welcher Organisation er angehört, sowie deren URL verlangt. Aus Sicherheitsgründen, wird nach dem vollständigen Ausfüllen des Formulars dem Gast eine E-Mail geschickt, um seine Daten zu verifizieren.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Gast nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Gast auf die jeweiligen fehlenden Angaben hin. Hat der Gast ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Gast auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

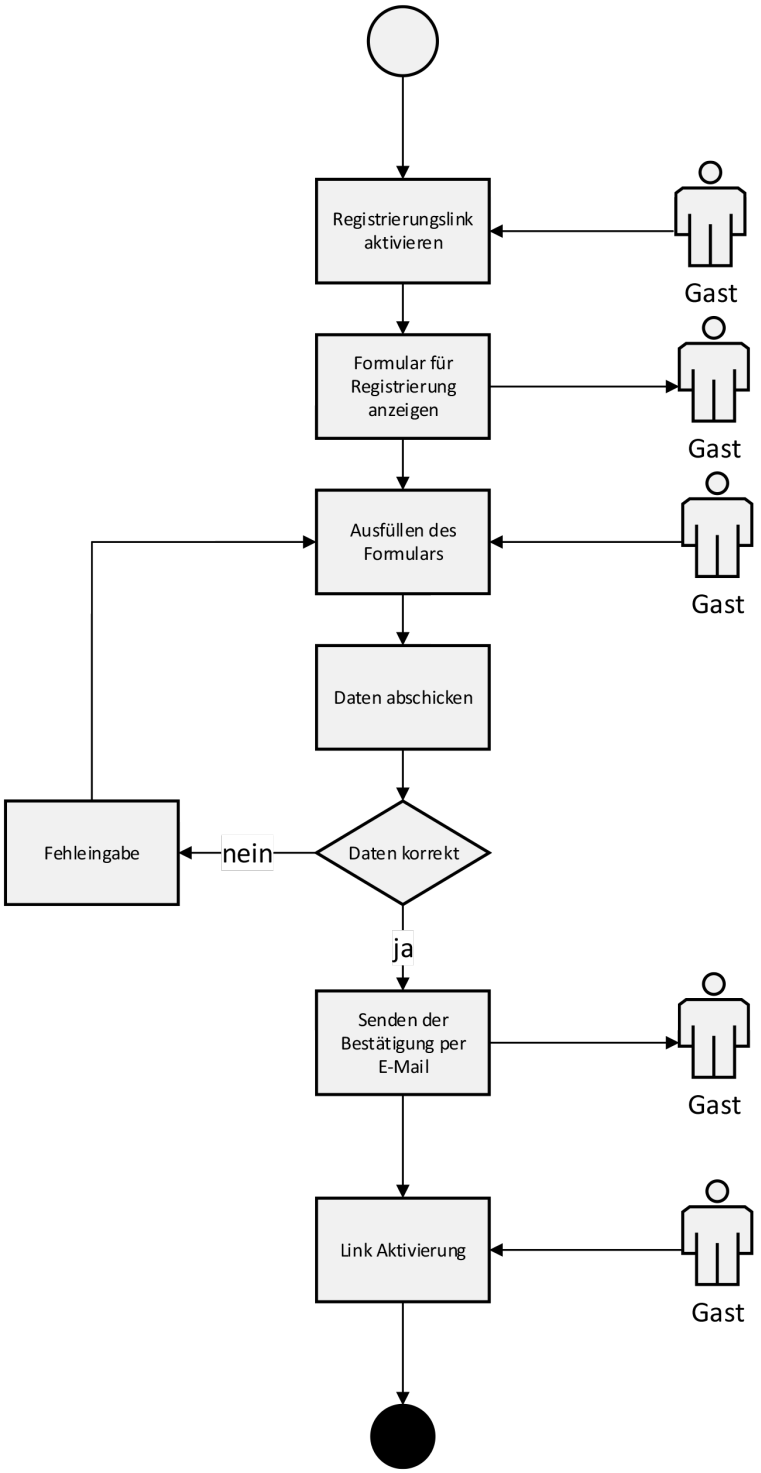


Abbildung 3.10: Registrierung

NV 2: Anmelden

Der komplette Ablauf der Anmeldung ist in Abbildung 3.11 dargestellt.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Gast	Verwendung der Applikation, ohne nochmalige Eingabe der Benutzerdaten	Einmal

Tabelle 3.3: Anmelden-NV2

Was ist bei der Durchführung im Einzelnen zu tun?

Für die Nutzung ist eine einmalige Anmeldung mit E-Mail-Adresse und Passwort notwendig. Die Nutzerdaten werden gespeichert, woraufhin der Nutzer kein weiteres Mal - außer er meldet sich ab oder ändert sein Nutzerkonto - die Anmeldedaten eingeben muss.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Gast nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Gast auf die jeweiligen fehlenden Angaben hin. Hat der Gast ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Gast auf bestimmte Fehleingaben hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen. Hat der Gast sein Passwort vergessen, sollte er über einen Link die Möglichkeit haben, sein Passwort, über eine E-Mail zu erhalten.

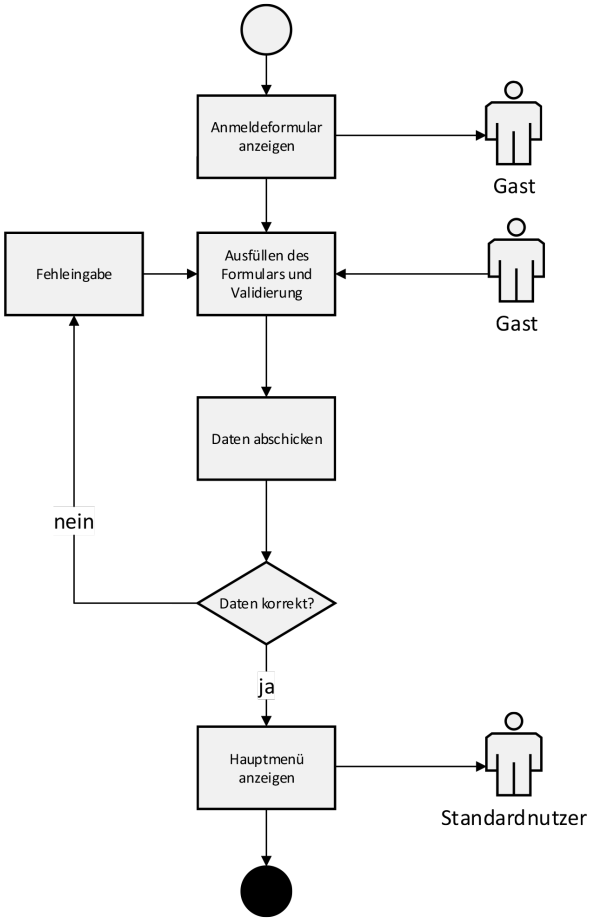


Abbildung 3.11: Anmelden

NV 3: Suche eines Nutzers

Die komplette Suche nach einem Nutzer ist in Abbildung 3.12 dargestellt.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Erlangen von Informationen über den Nutzer	Regelmäßig bis selten

Tabelle 3.4: Suche eines Nutzers-NV3

Was ist bei der Durchführung im Einzelnen zu tun?

Die Attribute Nachname, Vorname, E-Mail-Adresse und die Organisation dienen als Suchkriterien und werden in ein Textfeld eingetragen. Passend zu den angegebenen Suchkriterien, werden nun alle Nutzer, inklusive aller weiteren Parameter, aufgelistet.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Tritt der Fall ein, dass kein Nutzer auf die angegebenen Suchkriterien passt, wird dem Standardnutzer durch einen Warnhinweis darauf hingewiesen.

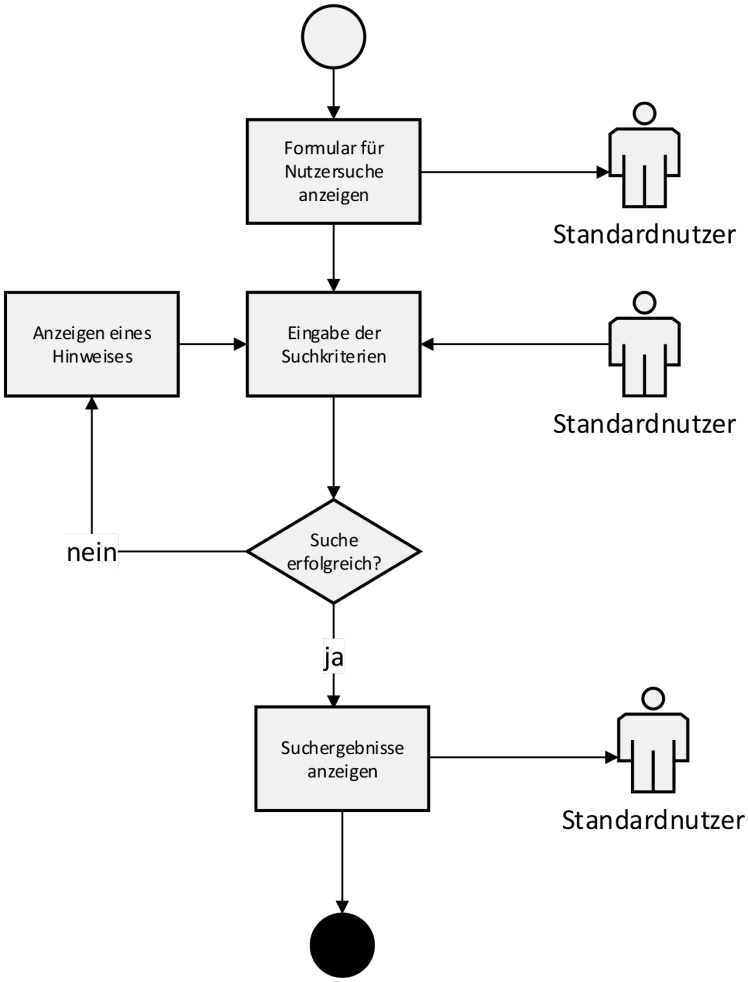


Abbildung 3.12: Suche eines Nutzers

NV 4: Ansehen eines Nutzerkontos

Die Abbildung 3.13 zeigt den durchzuführenden Vorgang, um ein Nutzerkonto anzusehen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Erlangen von Informationen über weitere Nutzer	Regelmäßig

Tabelle 3.5: Ansehen eines Nutzerkontos-NV4

Was ist bei der Durchführung im Einzelnen zu tun?

Nach der erfolgreichen Nutzersuche, kann der Standardnutzer auf ein in der Liste aufgeführtes Ergebnis klicken. Daraufhin werden ihm alle Details zu der ausgewählten Person angezeigt.

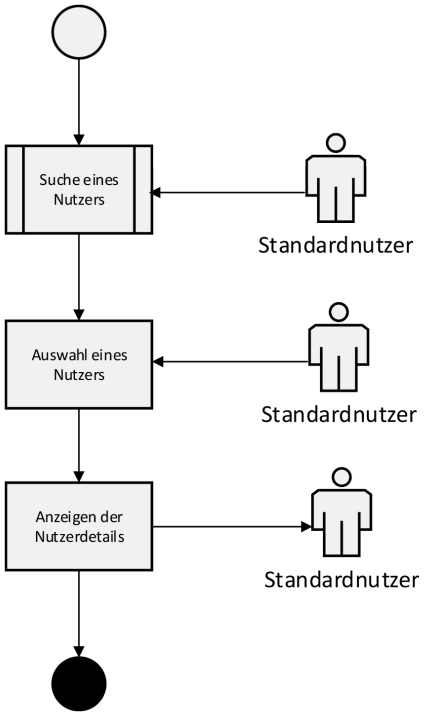


Abbildung 3.13: Ansehen eines Nutzerkontos

NV 5: Ansehen des eigenen Nutzerkontos

Die Abbildung 3.14 zeigt den durchzuführenden Vorgang, um sein eigenes Nutzerkonto anzusehen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Ansehen der eigenen Stammdaten	Regelmäßig bis selten

Tabelle 3.6: Ansehen des eigenen Nutzerkontos-NV5

Was ist bei der Durchführung im Einzelnen zu tun?

Entweder sucht der Standardnutzer sein eigenes Konto oder er betätigt im Hauptmenü einen Button. Es werden relevante persönliche Daten angezeigt.

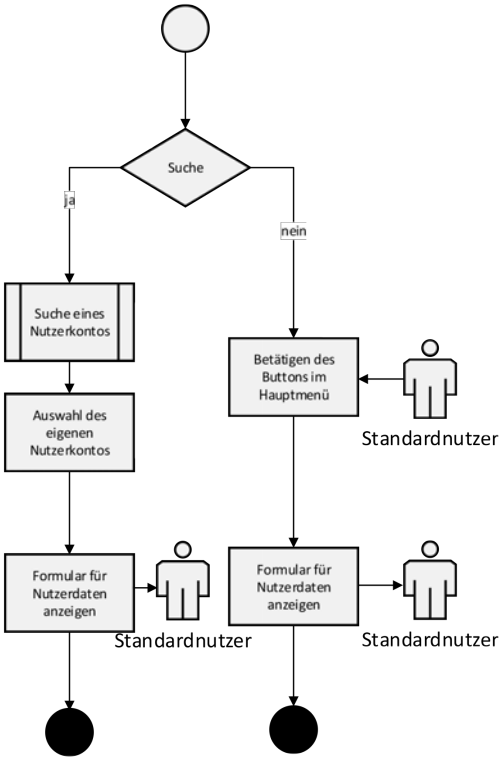


Abbildung 3.14: Ansehen des eigenen Nutzerkontos

NV 6: Ändern des Nutzerkontos

Die Abbildung 3.15 zeigt den durchzuführenden Vorgang, um sein eigenes Nutzerkonto zu ändern.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Nutzer möchte seine Stammdaten ändern	Regelmäßig bis selten

Tabelle 3.7: Ändern des Nutzerkontos-NV6

Was ist bei der Durchführung im Einzelnen zu tun?

Hier kann der Nachname, Vorname, das Passwort, die E-Mail-Adresse und die Organisation geändert werden. Nach Änderung der Eingaben werden diese auf Richtigkeit geprüft. Wird die E-Mail-Adresse geändert, muss der Standardnutzer diese ein weiteres Mal verifizieren.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Standardnutzer nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die jeweiligen fehlenden Angaben hin. Hat der Standardnutzer ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die bestimmte Falscheingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

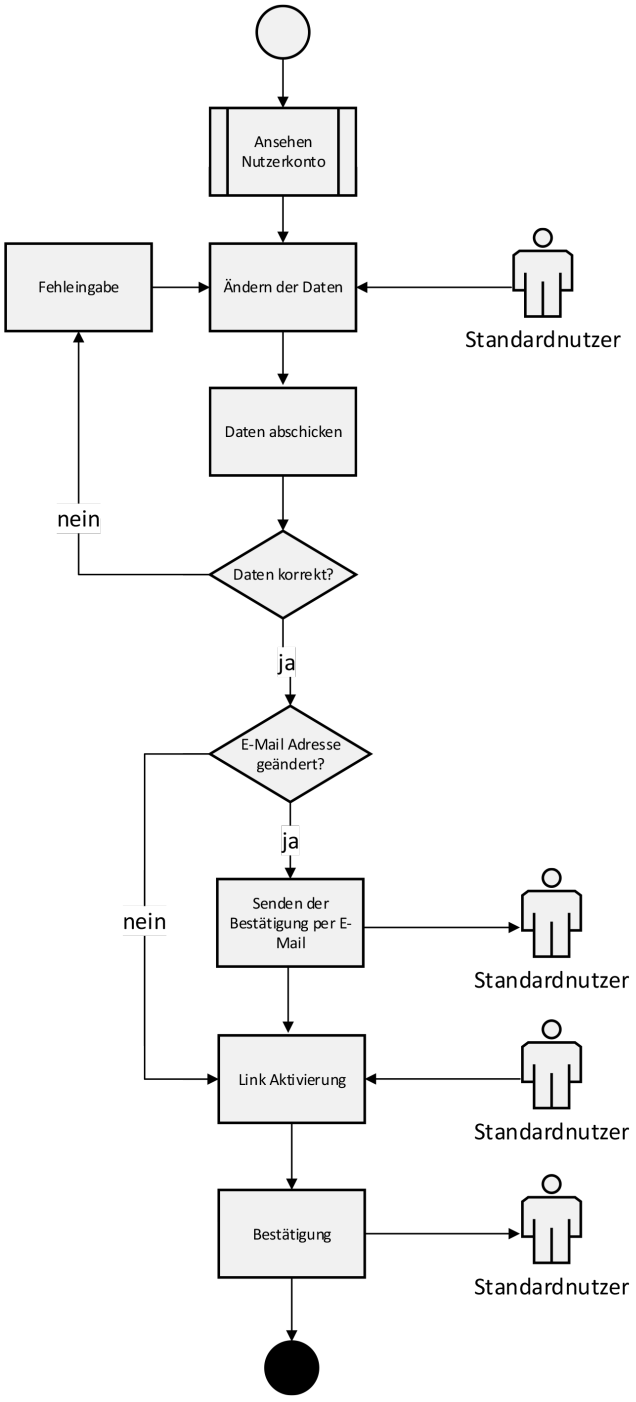


Abbildung 3.15: Ändern des Nutzerkontos

NV 7: Abmelden

Der komplette Ablauf der Abmeldung ist in Abbildung 3.16 dargestellt.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Möchte für einen gewissen Zeitraum nicht mit der Applikation arbeiten	Regelmäßig bis selten

Tabelle 3.8: Abmelden-NV7

Was ist bei der Durchführung im Einzelnen zu tun?

Der Standardnutzer kann sich abmelden, indem er auf den entsprechenden Button klickt. Letztendlich wird ihm nach dem Prozess des Abmeldens, der Dialog für die Anmeldung angezeigt.

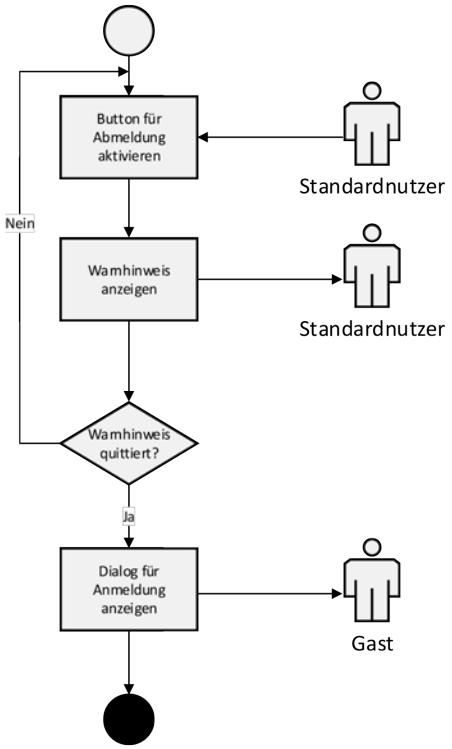


Abbildung 3.16: Abmelden

NV 8: Löschen des Nutzerkontos

Die Abbildung 3.17 zeigt den durchzuführenden Vorgang, um sein eigenes Nutzerkonto zu löschen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Nutzer möchte Applikation nicht mehr verwenden	Einmal

Tabelle 3.9: Löschen des Nutzerkontos-NV8

Was ist bei der Durchführung im Einzelnen zu tun?

Klickt der Standardnutzer auf den entsprechenden Link, wird ihm ein Dialog angezeigt, bei dem der Standardnutzer gefragt wird, ob er sein Konto wirklich löschen möchte. Der Standardnutzer muss diese Frage bestätigen und einen Link aktivieren, welchen er per E-Mail zugesandt bekommen hat. Schließlich ist sein Konto gelöscht.

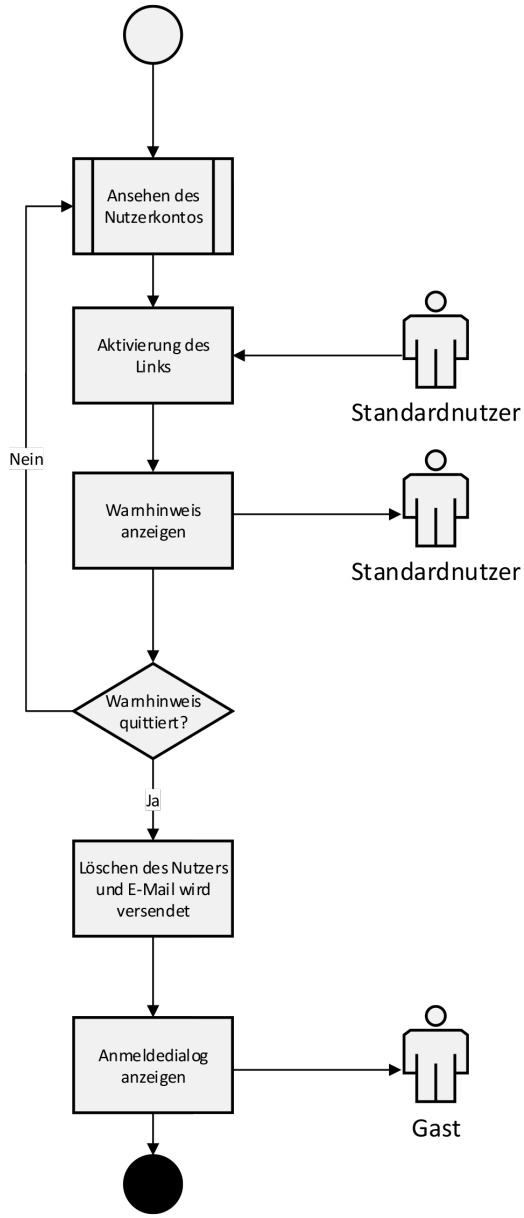


Abbildung 3.17: Löschen des Nutzerkontos

3.3.2 Organisatorische Rahmenverwaltung

Die *organisatorische Rahmenverwaltung (ORV)* beschäftigt sich mit allen User-Stories, die in Abbildung 3.18 dargestellt sind. Diese beziehen sich alle auf die selbe Domäne. Es werden alle User-Stories beschrieben, die mit dem Verwalten von OR (siehe Kapitel 2.5.1) in Verbindung stehen.

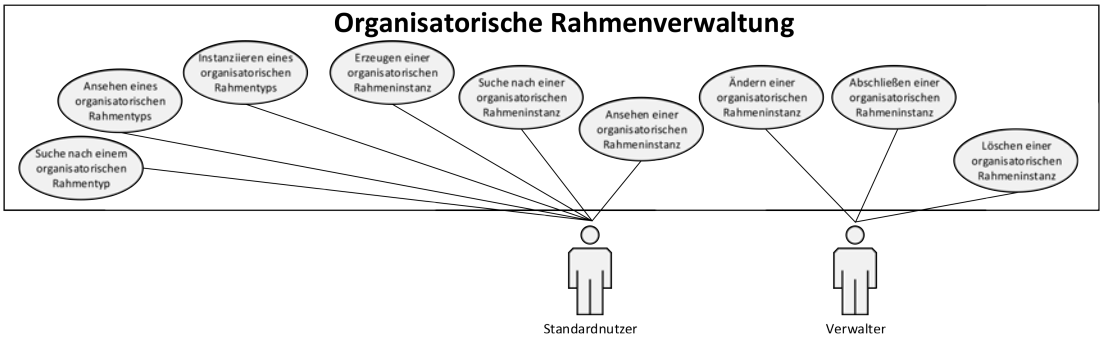


Abbildung 3.18: Use-Case-Diagramm der organisatorischen Rahmenverwaltung

ORV 1: Suche nach einem organisatorischen Rahmentyp

Die Abbildung 3.19 zeigt den durchzuführenden Vorgang, um nach einem *organisatorischen Rahmentyp (ORT)* (siehe Kapitel 2.5.1) zu suchen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Nutzer will ORT für OR verwenden	Regelmäßig bis selten

Tabelle 3.10: Suche nach einem organisatorischen Rahmentyp-ORV1

Was ist bei der Durchführung im Einzelnen zu tun?

Bevor ein Standardnutzer einen neuen OR erstellen, kann der Standardnutzer die existierenden ORTs ansehen. Diese ORTs basieren auf dem Namen, einer Beschreibung

und dem zugehörigen Ziel. Diese Parameter dienen als Suchkriterien, welche vom Standardnutzer in ein Textfeld eingetragen werden. Passen ein oder mehrere ORTs zu den angegebenen Parametern, wird eine Liste angezeigt, welche den Namen, eine kurze Beschreibung und die abhängigen Subtypen der Ergebnisse anzeigt.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Passt kein ORT zu den angegebenen Suchkriterien, wird dies dem Standardnutzer über einen Warnhinweis mitgeteilt. Hat der Standardnutzer nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die jeweiligen fehlenden Angaben hin.

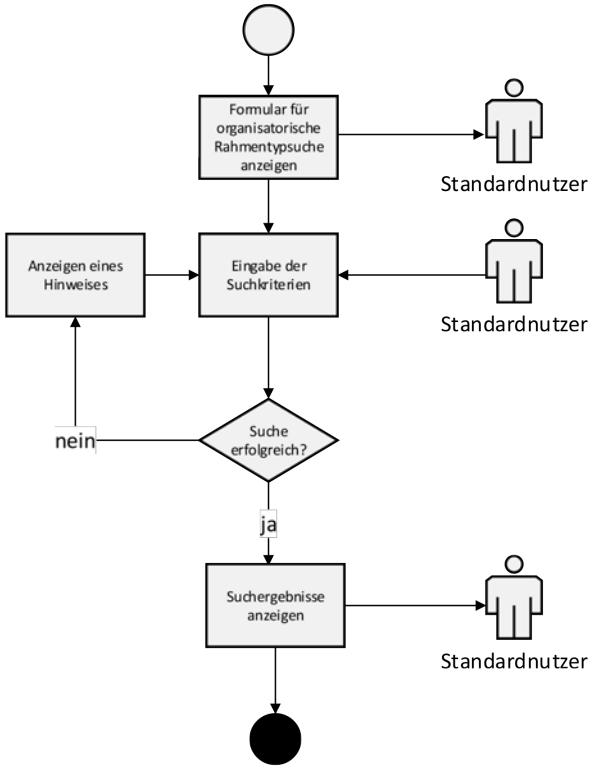


Abbildung 3.19: Suche nach einem organisatorischen Rahmentyp

ORV 2: Ansehen eines organisatorischen Rahmentyps

Die Abbildung 3.20 zeigt den durchzuführenden Vorgang, um einen ORT anzusehen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Nutzer will ORT für OR verwenden	Regelmäßig bis selten

Tabelle 3.11: Ansehen eines organisatorischen Rahmentyps-ORV2

Was ist bei der Durchführung im Einzelnen zu tun?

Nach der erfolgreichen Suche nach einem ORT, kann der Standardnutzer auf einen bestimmten ORT klicken, der nach einer Suche aufgelistet wird. Nun werden ihm alle zugehörigen Details angezeigt.

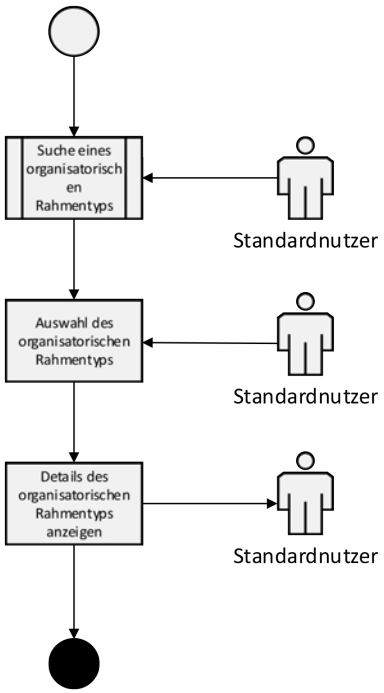


Abbildung 3.20: Ansehen eines organisatorischen Rahmentyps

ORV 3: Instanziieren eines organisatorischen Rahmentyps

Die Abbildung 3.21 zeigt den durchzuführenden Vorgang, um einen ORT zu instanziieren.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Nutzer will ORT für OR verwenden	Regelmäßig bis selten

Tabelle 3.12: Instanziieren eines organisatorischen Rahmentyps-ORV3

Was ist bei der Durchführung im Einzelnen zu tun?

Jeder Standardnutzer kann einen neuen OR erstellen, welcher auf den existierenden ORTs basiert. Hat der Standardnutzer einen bestimmten ORT ausgewählt und befindet sich in der zugehörigen Detailansicht, kann er auf den Link für die Instanziierung klicken. Nun kann der Standardnutzer verschiedene Details spezifizieren und die Abhängigkeiten zwischen anderen Instanzen festlegen. Der Standardnutzer nimmt für diese Instanz nun automatisch die Rolle des Verwalters ein.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Standardnutzer ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

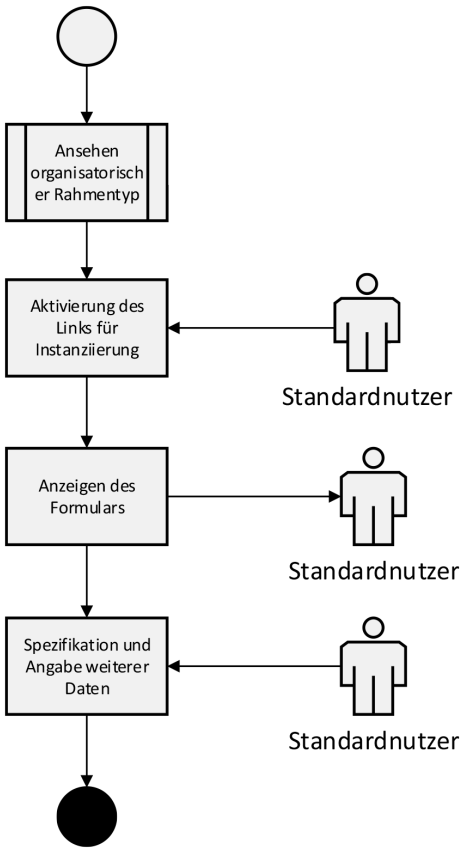


Abbildung 3.21: Instanziieren eines organisatorischen Rahmentyps

ORV 4: Erzeugen einer organisatorischen Rahmeninstanz

Die Abbildung 3.22 zeigt den durchzuführenden Vorgang, um eine organisatorische Rahmeninstanz (ORI) (siehe Kapitel 2.5.1) zu erzeugen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Nutzer findet keinen passenden ORT oder möchte eine individuelle ORI erstellen	Regelmäßig bis selten

Tabelle 3.13: Erzeugen einer organisatorischen Rahmeninstanz-ORV4

Was ist bei der Durchführung im Einzelnen zu tun?

Jeder Standardnutzer kann eine ORI erzeugen, ohne dass ein ORT verwendet wird. Der Standardnutzer kann nun verschiedene Details spezifizieren und Abhängigkeiten zwischen anderen Instanzen festlegen. Der Standardnutzer nimmt für diese ORI nun automatisch die Rolle des Verwalters ein.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Standardnutzer nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die jeweiligen fehlenden Angaben hin. Hat der Standardnutzer ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

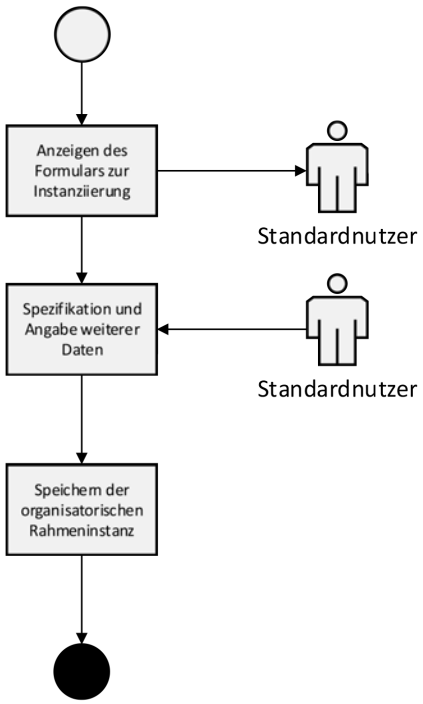


Abbildung 3.22: Erzeugen einer organisatorischen Rahmeninstanz

ORV 5: Suche nach einer organisatorischen Rahmeninstanz

Die Abbildung 3.23 zeigt den durchzuführenden Vorgang, um nach einer ORI zu suchen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Nutzer möchte verschiedene ORIs ansehen	Regelmäßig bis selten

Tabelle 3.14: Suche nach einer organisatorischen Rahmeninstanz-ORV5

Was ist bei der Durchführung im Einzelnen zu tun?

Der Verwalter verschiedener ORIs kann nach diesen suchen, indem er nach den Parametern Name, Beschreibung oder Ziel sucht. Wenn passende ORs existieren, werden diese in einer Liste angezeigt. Diese Liste beinhaltet Informationen, wie den Namen, die Beschreibung und die abhängigen Subinstanzen.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Findet das System keine passenden Ergebnisse, wird dies dem Standardnutzer über einen Dialog mitgeteilt. Hat der Standardnutzer ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

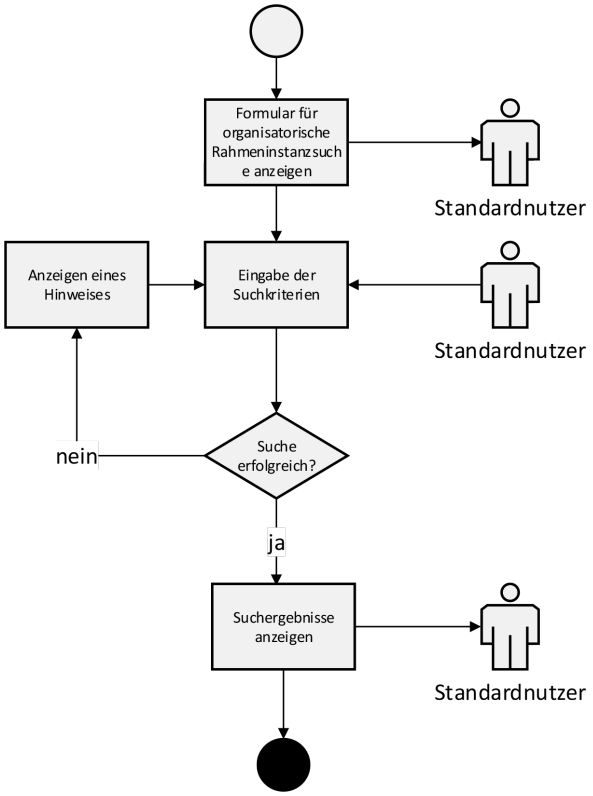


Abbildung 3.23: Suche nach einer organisatorischen Rahmeninstanz

ORV 6: Ansehen einer organisatorischen Rahmeninstanz

Die Abbildung 3.24 zeigt den durchzuführenden Vorgang, um eine ORI anzusehen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Nutzer will Details über ORI erfahren	Regelmäßig

Tabelle 3.15: Ansehen einer organisatorischen Rahmeninstanz-ORV6

Was ist bei der Durchführung im Einzelnen zu tun?

Nachdem eine Suche nach bestimmten ORIs abgeschlossen ist, kann der Standardnutzer die jeweiligen Details verschiedener Instanzen ansehen. Hierfür muss der Standardnutzer eine ORI der aufgelisteten Ergebnisse anklicken und alle zugehörigen Details werden ihm angezeigt.

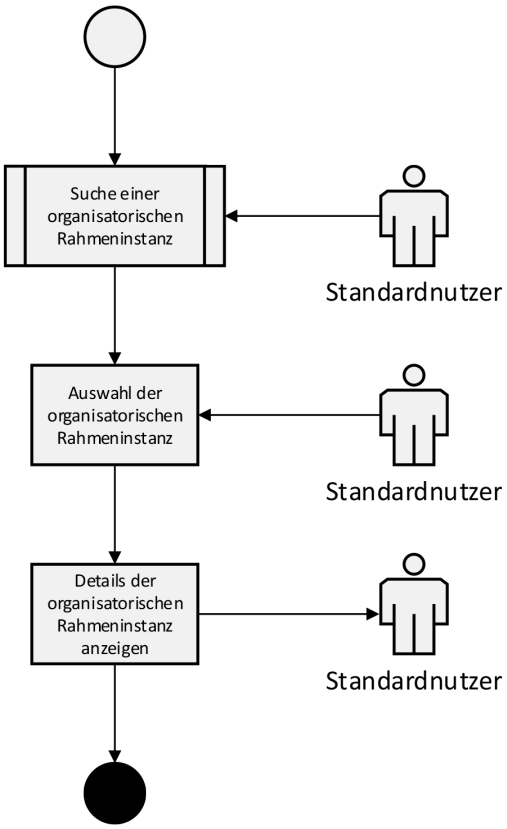


Abbildung 3.24: Ansehen einer organisatorischen Rahmeninstanz

ORV 7: Ändern einer organisatorischen Rahmeninstanz

Die Abbildung 3.25 zeigt den durchzuführenden Vorgang, um eine ORI zu ändern.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will Änderungen am ORI vornehmen	Regelmäßig bis selten

Tabelle 3.16: Ändern einer organisatorischen Rahmeninstanz-ORV7

Was ist bei der Durchführung im Einzelnen zu tun?

Ein Nutzer, dem die Rolle des Verwalters zugeteilt ist, kann eine ORI ändern und bearbeiten. In einem bestimmten Dialog, kann er den Namen, das Ziel, Startdatum, Enddatum und das Ablaufdatum ändern und anschließend speichern. Das Ablaufdatum bestimmt die Zeit, nachdem der OR erledigt sein muss.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Verwalter nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die jeweiligen fehlenden Angaben hin. Hat der Verwalter ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

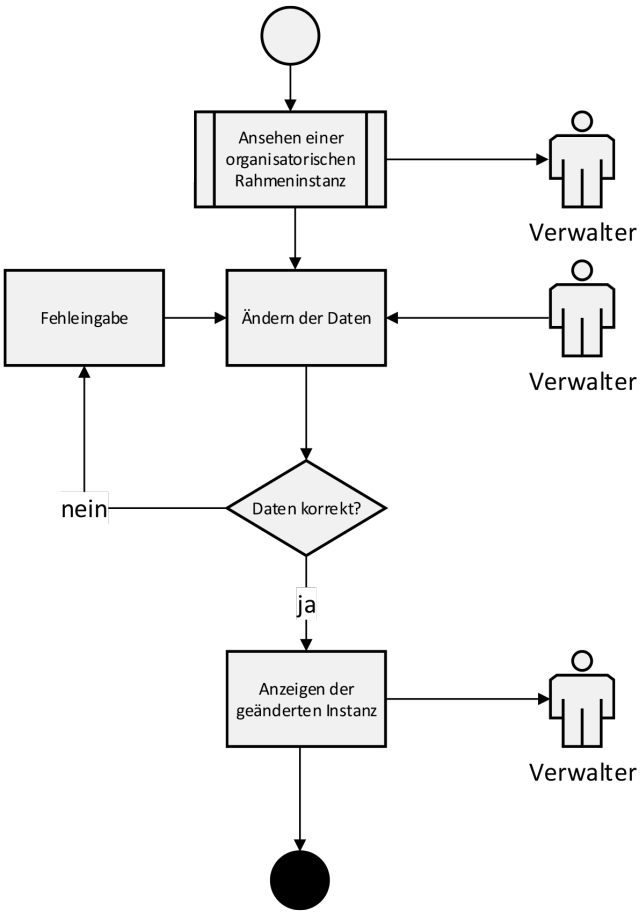


Abbildung 3.25: Ändern einer organisatorischen Rahmeninstanz

ORV 8: Abschließen einer organisatorischen Rahmeninstanz

Die Abbildung 3.26 zeigt den durchzuführenden Vorgang, um eine ORI abzuschließen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will den OR als abgeschlossen kennzeichnen	Selten

Tabelle 3.17: Abschließen einer organisatorischen Rahmeninstanz-ORV8

Was ist bei der Durchführung im Einzelnen zu tun?

Ein Nutzer, dem die Rolle des Verwalters zugeteilt ist, kann eine ORI abschließen. Klickt der Verwalter auf den Link für den Abschluss in der zugehörigen ORI, wird ihm ein Dialog angezeigt, welcher ihn fragt ob er diese abschließen möchte. Quittiert der Verwalter diesen Dialog wird die ORI fertiggestellt und archiviert.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Enthält eine ORI untergeordnete ORs, CLs oder Einträge, welche nicht abgeschlossen sind, wird dem Verwalter ein Warnhinweis angezeigt, welcher quittiert werden muss.

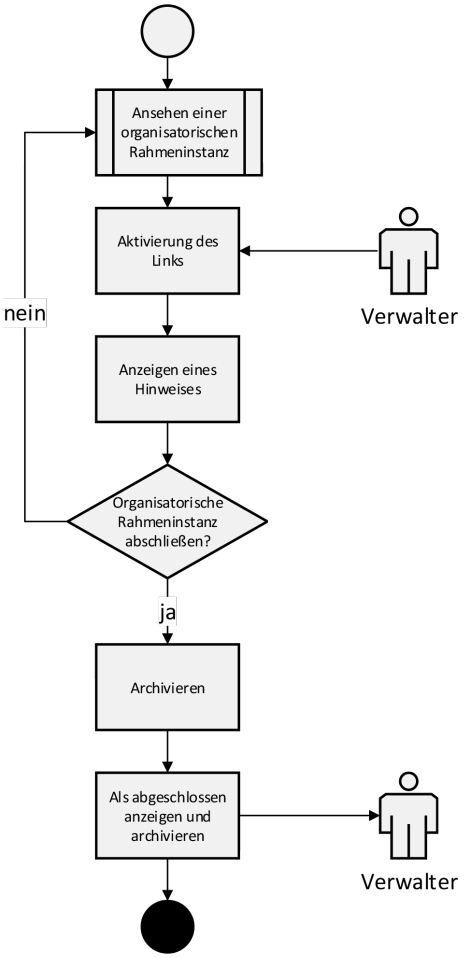


Abbildung 3.26: Abschließen einer organisatorischen Rahmeninstanz

ORV 9: Löschen einer organisatorischen Rahmeninstanz

Die Abbildung 3.27 zeigt den durchzuführenden Vorgang, um eine ORI zu löschen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	ORI wird nicht mehr benötigt	Regelmäßig bis selten

Tabelle 3.18: Löschen einer organisatorischen Rahmeninstanz-ORV9

Was ist bei der Durchführung im Einzelnen zu tun?

Ein Verwalter kann eine ORI löschen, hierzu muss er auf einen Link klicken, welcher den Löschvorgang einleitet. Um den OR und seine Subinstanzen zu löschen, muss der folgende Dialog quittiert werden. Dieser Dialog zeigt eine Nachricht mit dem Inhalt, ob der OR wirklich gelöscht werden soll. Letztendlich wird dem Verwalter eine Bestätigung über den Löschvorgang angezeigt.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Enthält eine ORI untergeordnete Instanzen werden diese ebenfalls gelöscht.

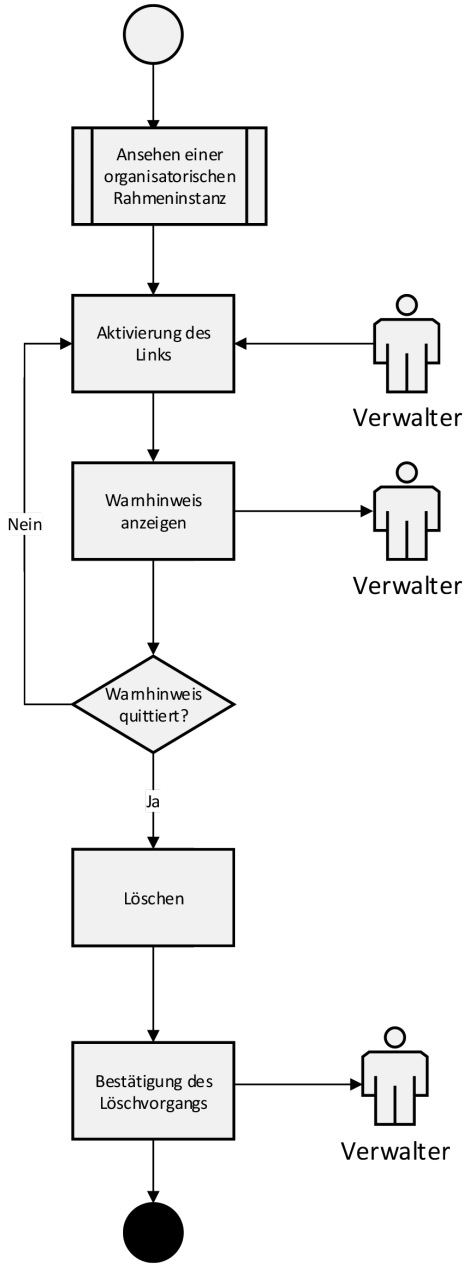


Abbildung 3.27: Löschen einer organisatorischen Rahmeninstanz

3.3.3 Checklisten-Verwaltung

Die *Checklisten-Verwaltung (CLV)* bezieht sich auf diejenigen User-Stories, die den Umgang mit CLs (siehe Kapitel 2.5.2) beschreiben. In Abbildung 3.28 sind alle später ausformulierten User-Stories visualisiert.

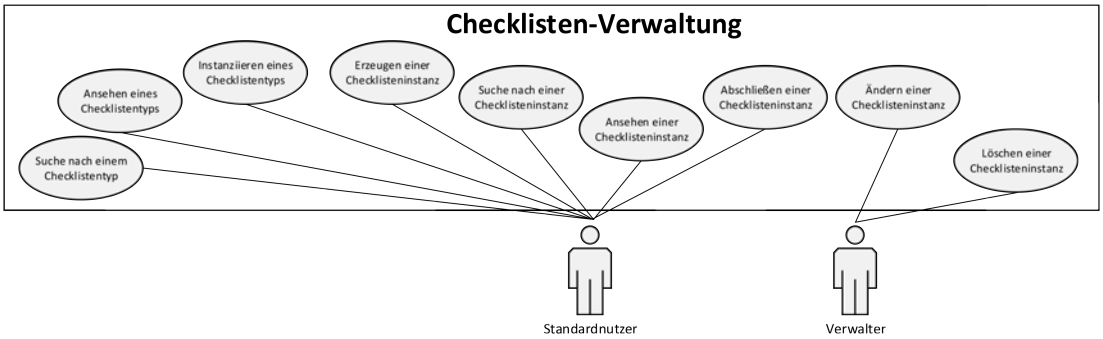


Abbildung 3.28: Use-Case-Diagramm der Checklisten-Verwaltung

CLV 1: Suche nach einem Checklistentyp

Die Abbildung 3.29 zeigt den durchzuführenden Vorgang, um einen Checklistentyp (CLT) (siehe Kapitel 2.5.2) zu suchen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will CLT ansehen und möglicherweise instanziieren	Regelmäßig bis selten

Tabelle 3.19: Suche nach einem Checklistentyp-CLV1

Was ist bei der Durchführung im Einzelnen zu tun?

Um eine neue CL in einem existierenden OR zu erstellen, kann der Verwalter eines ORs nach einem CLT suchen. Der Name und der Text fungieren in diesem Anwendungsfall als Suchkriterien. Die passenden Ergebnisse werden dem Verwalter angezeigt, inklusive der Parameter Name und Text des CLTs.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Findet das System keine CLTs mit den Suchkriterien, wird dies dem Verwalter über einen Warnhinweis mitgeteilt. Hat der Verwalter nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die jeweiligen fehlenden Angaben hin. Hat der Verwalter ein oder mehrere Textfelder falsch ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese weist ihn auf die bestimmte Falscheingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

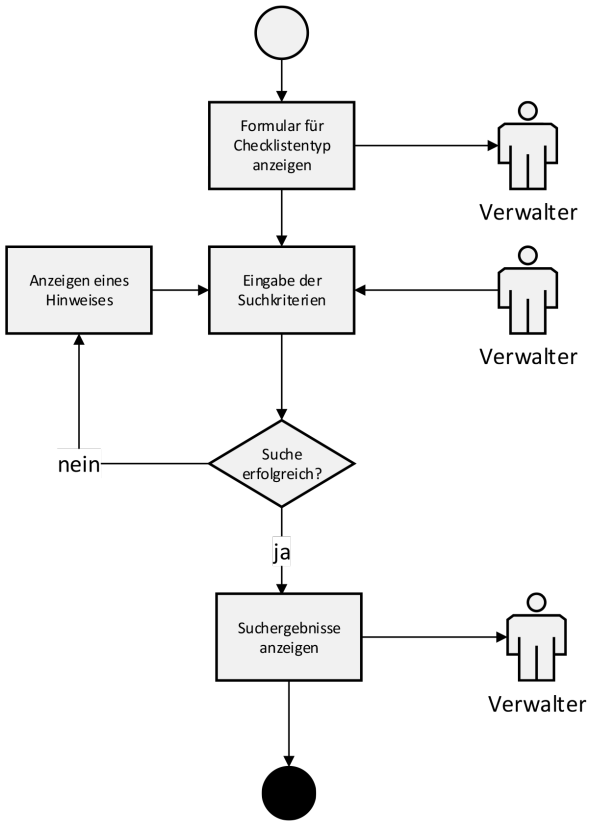


Abbildung 3.29: Suche nach einem Checklistentyp

CLV 2: Ansehen eines Checklistentyps

Die Abbildung 3.30 zeigt den durchzuführenden Vorgang, um einen CLT anzusehen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will Details über CLT erfahren	Regelmäßig

Tabelle 3.20: Ansehen eines Checklistentyps-CLV2

Was ist bei der Durchführung im Einzelnen zu tun?

Nachdem die Suche eines CLTs abgeschlossen wurde, kann der Verwalter auf die Ergebnisse klicken und es werden ihm die Details zu diesem CLT angezeigt.

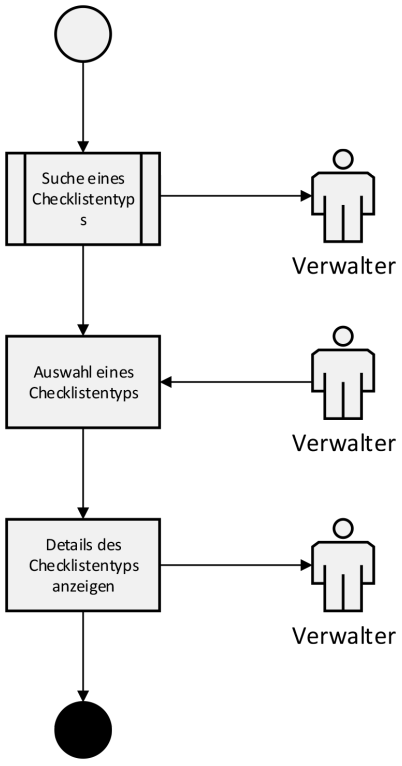


Abbildung 3.30: Ansehen eines Checklistentyps

CLV 3: Instanziieren eines Checklistentyps

Die Abbildung 3.31 zeigt den durchzuführenden Vorgang, um einen CLT zu instanziieren.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will den CLT instanziieren	Regelmäßig bis selten

Tabelle 3.21: Instanziieren eines Checklistentyps-CLV3

Was ist bei der Durchführung im Einzelnen zu tun?

Um eine CLI in einer ORI zu erstellen, kann ein Verwalter diese basierend auf einem CLT erstellen. Der Verwalter sucht sich einen bestimmten CLT aus und klickt auf diesen, woraufhin die Details angezeigt werden. Nun muss der Verwalter auf den Link für die Instanziierung klicken, so wird der CLT instanziiert und der ORI zugeordnet. Letztendlich kann der Verwalter verschiedene Details spezifizieren und die Abhängigkeiten zwischen anderen Instanzen festlegen.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Verwalter ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

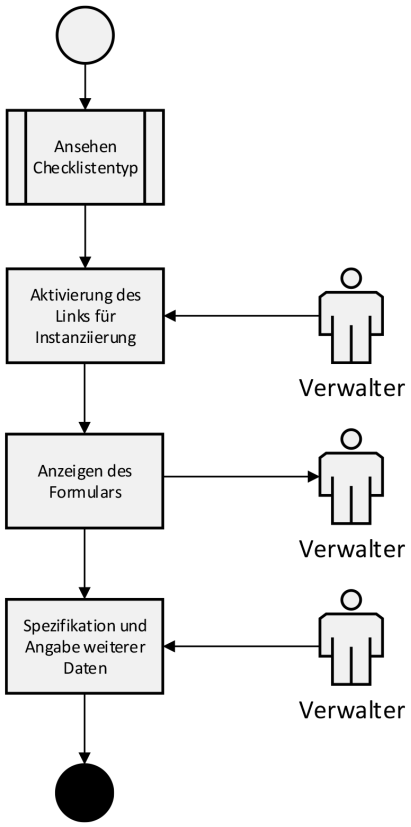


Abbildung 3.31: Instanziieren eines Checklistentyps

CLV 4: Erzeugen einer Checklisteninstanz

Der komplette Ablauf der Erzeugung einer Checklisteninstanz (CLI) (siehe Kapitel 2.5.2) ist in Abbildung 3.32 dargestellt.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will selbst eine CLI erstellen	Regelmäßig bis selten

Tabelle 3.22: Erzeugen einer Checklisteninstanz-CLV4

Was ist bei der Durchführung im Einzelnen zu tun?

Das Erstellen einer CLI in einer bestimmten ORI ist ohne einen CLT möglich. Hierfür werden die Daten und Parameter der CLI vom Verwalter spezifiziert.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Verwalter nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die jeweiligen fehlenden Angaben hin.

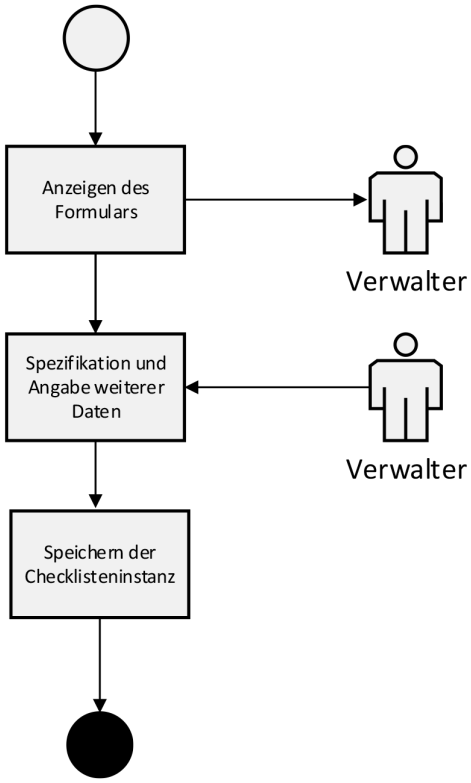


Abbildung 3.32: Erzeugen einer Checklisteninstanz

CLV 5: Suche nach einer Checklisteninstanz

Der komplette Ablauf der Suche nach einer CLI ist in Abbildung 3.33 dargestellt.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Standardnutzer will CLIs ansehen	Regelmäßig bis selten

Tabelle 3.23: Suche nach einer Checklisteninstanz-CLV5

Was ist bei der Durchführung im Einzelnen zu tun?

Ein Verwalter kann nach CLIs in den zugehörigen ORIs suchen, indem er verschiedene Suchkriterien, wie Name, Text und Zustand angibt. Darüber hinaus kann jeder Standardnutzer nach CLIs suchen, die für ihn relevant sind. Die passenden Ergebnisse werden dem Standardnutzer in einer Liste angezeigt, inklusive der Parameter Name und Text der bestimmten CLI.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Findet das System keine CLIs mit den angegebenen Suchkriterien, wird dies dem Standardnutzer über einen Warnhinweis mitgeteilt. Hat der Standardnutzer nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die jeweiligen fehlenden Angaben hin. Hat der Standardnutzer ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

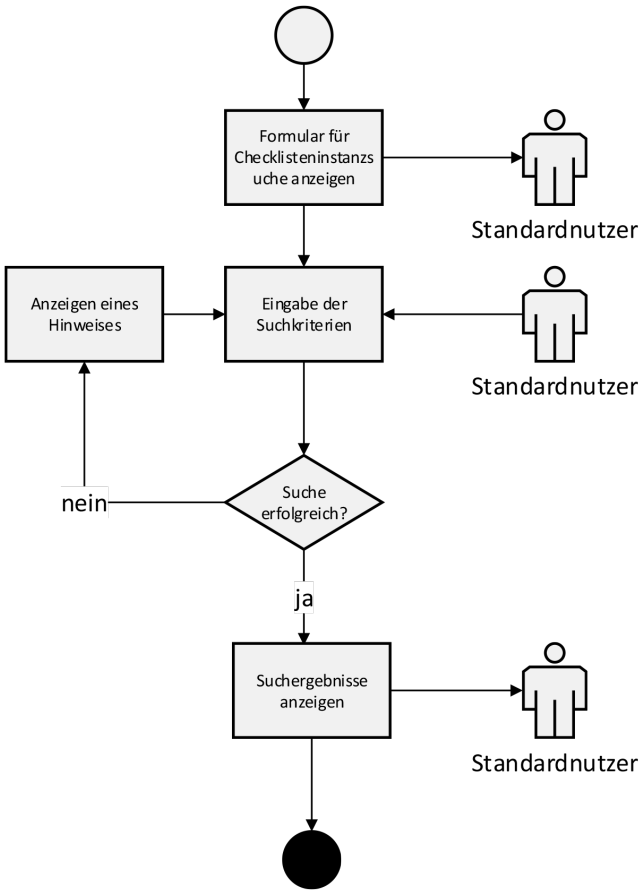


Abbildung 3.33: Suche nach einer Checklisteninstanz

CLV 6: Ansehen einer Checklisteninstanz

Die Abbildung 3.34 zeigt den durchzuführenden Vorgang, um eine CLI anzusehen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Standardnutzer will Details einer CLI ansehen	Regelmäßig

Tabelle 3.24: Ansehen einer Checklisteninstanz-CLV6

Was ist bei der Durchführung im Einzelnen zu tun?

Nachdem die Suche einer CLI erfolgreich abgeschlossen wurde, kann der Standardnutzer auf die Ergebnisse klicken. Klickt der Standardnutzer auf ein bestimmtes Suchergebnis, werden ihm alle Details zu dieser CLI angezeigt.

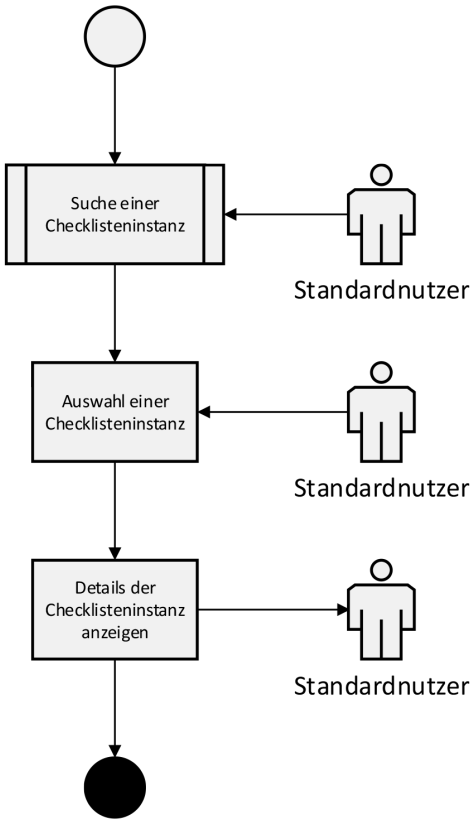


Abbildung 3.34: Ansehen einer Checklisteninstanz

CLV 7: Ändern einer Checklisteninstanz

Die Abbildung 3.35 zeigt den durchzuführenden Vorgang, um eine CLI zu ändern.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will Änderungen an CLI vornehmen	Regelmäßig bis selten

Tabelle 3.25: Ändern einer Checklisteninstanz-CLV7

Was ist bei der Durchführung im Einzelnen zu tun?

Wird in einer ORI nach verschiedenen CLIs gesucht und daraufhin eine ausgewählt, kann der Verwalter diese ändern. Die Parameter Name, Text und der Zustand können geändert werden.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Verwalter nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die jeweiligen fehlenden Angaben hin. Hat der Verwalter ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

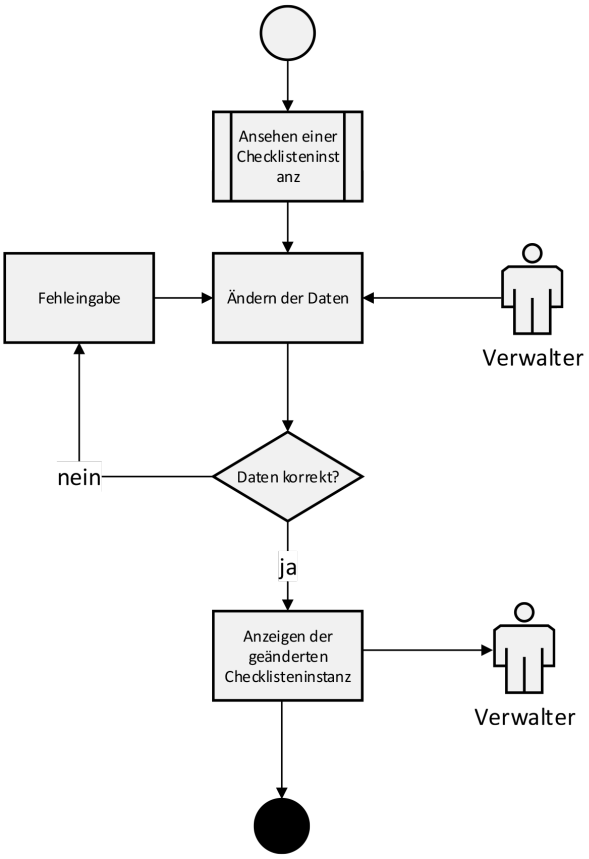


Abbildung 3.35: Ändern einer Checklisteninstanz

CLV 8: Abschließen einer Checklisteninstanz

Die Abbildung 3.36 zeigt den durchzuführenden Vorgang, um eine CLI abzuschließen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Standardnutzer möchte die CLI als abgearbeitet kennzeichnet	Regelmäßig bis selten

Tabelle 3.26: Abschließen einer Checklisteninstanz-CLV8

Was ist bei der Durchführung im Einzelnen zu tun?

Nachdem ein Standardnutzer eine CLI in einer ORI gewählt hat, kann er die CLI als erledigt kennzeichnen, wenn er auf den zugehörigen Link klickt. Daraufhin wird dem Standardnutzer ein Warnhinweis angezeigt, den er quittieren muss, um die CLI abzuschließen und archivieren zu können.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Enthält eine Rahmeninstanz untergeordnete CLs oder Einträge, welche nicht abgeschlossen sind, wird dem Standardnutzer ein Warnhinweis angezeigt, welcher quittiert werden muss.

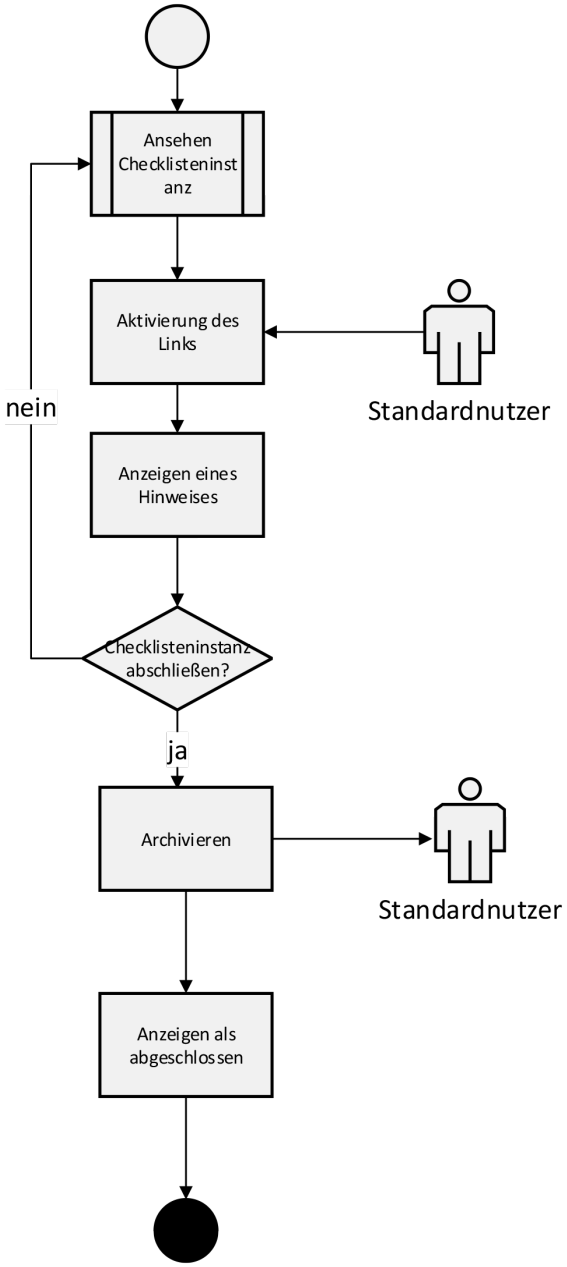


Abbildung 3.36: Abschließen einer Checklisteninstanz

CLV 9: Löschen einer Checklisteninstanz

Die Abbildung 3.37 zeigt den durchzuführenden Vorgang, um eine CLI zu löschen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter benötigt CLI nicht mehr	Regelmäßig bis selten

Tabelle 3.27: Löschen einer Checklisteninstanz-CLV9

Was ist bei der Durchführung im Einzelnen zu tun?

Nachdem der Rahmenverwalter sich eine CLI ausgewählt hat, kann er diese löschen. Hierzu muss er nur auf den zugehörigen Link klicken und den darauf folgenden Warnhinweis quittieren.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Enthält eine Rahmeninstanz untergeordnete Instanzen werden diese ebenfalls gelöscht.

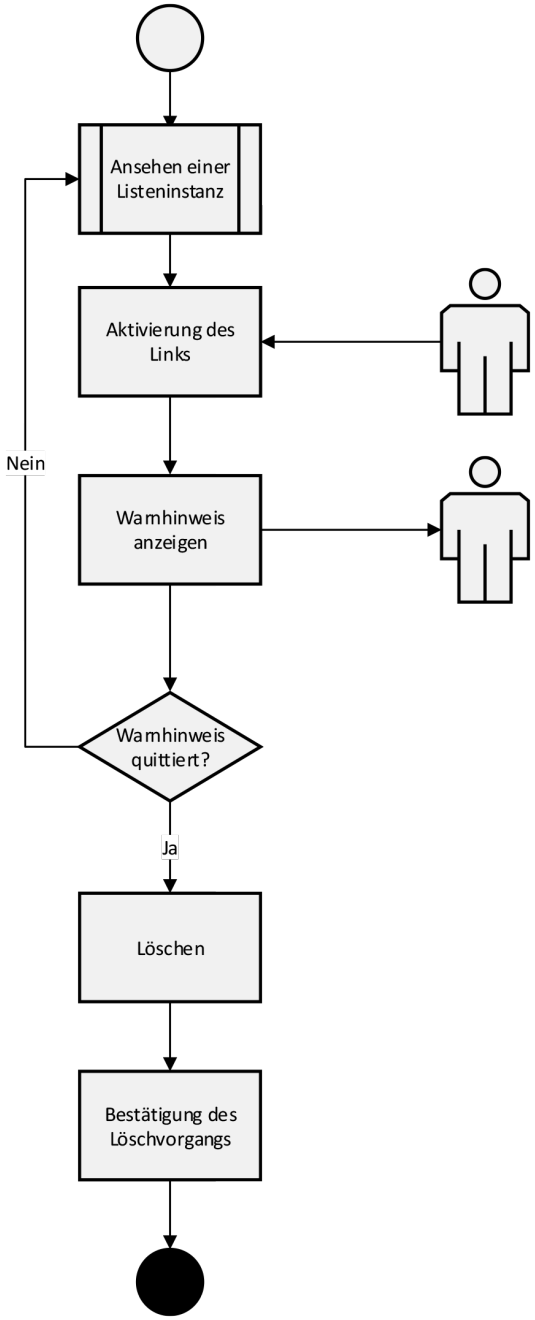


Abbildung 3.37: Löschen einer Checklisteninstanz

3.3.4 Verwaltung von Checklisteneinträgen

Die *Verwaltung von Checklisteneinträgen (VCE)* spezifiziert alle Aufgaben, welche in Abbildung 3.38 beschrieben sind. Insgesamt werden alle User-Stories erläutert, die eine Rolle im Umgang mit Einträgen (siehe Kapitel 2.5.3) spielen.

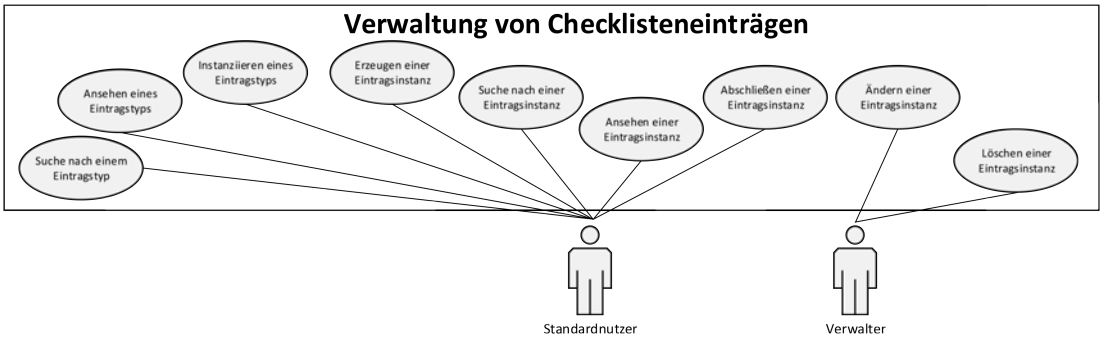


Abbildung 3.38: Use-Case-Diagramm der Verwaltung von Checklisteneinträgen

VCE 1: Suche nach einem Eintragstyp

Die Abbildung 3.39 zeigt den durchzuführenden Vorgang, um einen Eintragstyp (ET) (siehe Kapitel 2.5.3) zu suchen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will ET als Vorlage verwenden	Regelmäßig bis selten

Tabelle 3.28: Suche nach einem Eintragstyp-VCE1

Was ist bei der Durchführung im Einzelnen zu tun?

Um eine bestehende CL zu erweitern, kann der Verwalter eines ORs nach einem existierendem ET suchen. Der Name und der Text fungieren in diesem Anwendungsfall als Suchkriterien. Die passenden Ergebnisse werden dem Verwalter in einer Liste angezeigt, inklusive der Parameter Name und Text des bestimmten ETs.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Findet das System keine ETs, wird dies dem Verwalter über einen Warnhinweis mitgeteilt. Hat der Verwalter nicht alle Textfelder ausgefüllt bzw. falsch ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die jeweiligen fehlenden und falschen Angaben hin.

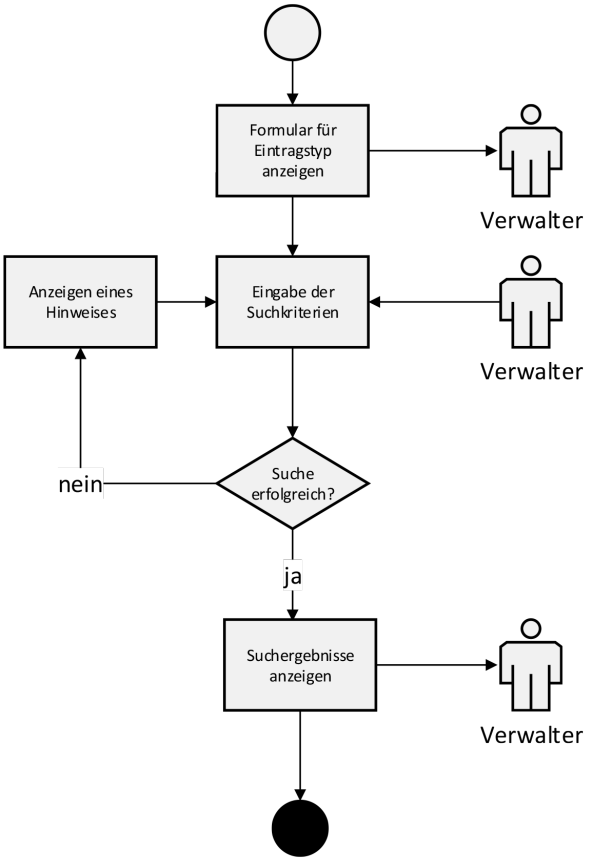


Abbildung 3.39: Suche nach einem Eintragstyp

VCE 2: Ansehen eines Eintragstyps

Die Abbildung 3.40 zeigt den durchzuführenden Vorgang, um einen ET anzusehen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will den ET möglicherweise instanziieren und möchte die Details ansehen	Regelmäßig

Tabelle 3.29: Ansehen eines Eintragstyps-VCE2

Was ist bei der Durchführung im Einzelnen zu tun?

Nachdem die Suche eines ETs abgeschlossen wurde, kann der Verwalter auf die Ergebnisse klicken und ihm werden alle Details zu diesem ET angezeigt.

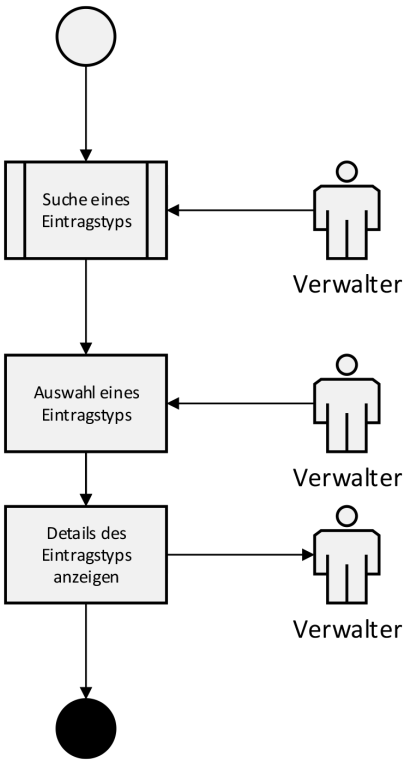


Abbildung 3.40: Ansehen eines Eintragstyps

VCE 3: Instanziieren eines Eintragstyps

Die Abbildung 3.41 zeigt den durchzuführenden Vorgang, um einen ET zu instanziieren.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will den ET instanziieren	Regelmäßig bis selten

Tabelle 3.30: Instanziieren eines Eintragstyps-VCE3

Was ist bei der Durchführung im Einzelnen zu tun?

Um eine existierende CLI in einer ORI zu erweitern, kann ein Verwalter eine Eintragsinstanz erstellen, basierend auf einem ET. Der Verwalter sucht sich einen bestimmten ET aus und klickt auf diesen, woraufhin die Details angezeigt werden. Nun muss der Verwalter auf den Link für die Instanziierung klicken und der ET wird instanziiert und der CL zugeordnet. Zusätzlich kann der Verwalter jedoch den Eintrag spezifizieren, indem er beispielsweise den Text ändert oder hinzufügt.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Verwalter ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

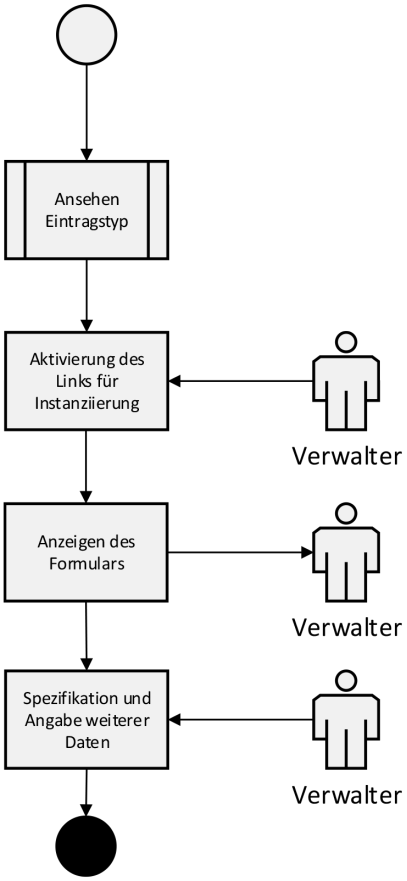


Abbildung 3.41: Instanziieren eines Eintragstyps

VCE 4: Erzeugen einer Eintragsinstanz

Die Abbildung 3.42 zeigt den durchzuführenden Vorgang, um eine Eintragsinstanz (EI) (siehe Kapitel 2.5.3) zu erzeugen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will den EI ohne ET erstellen	Regelmäßig bis selten

Tabelle 3.31: Erzeugen einer Eintragsinstanz-VCE4

Was ist bei der Durchführung im Einzelnen zu tun?

Um eine existierende CLI in einer ORI zu erweitern, kann ein Verwalter eine EI erstellen, welche nicht auf einem ET basiert. Diese EI wird spezifiziert durch verschiedene Parameter wie Name, Text und den Zustand im Zusammenhang mit in einer CLI.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Verwalter nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die jeweiligen fehlenden Angaben hin. Hat der Verwalter ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

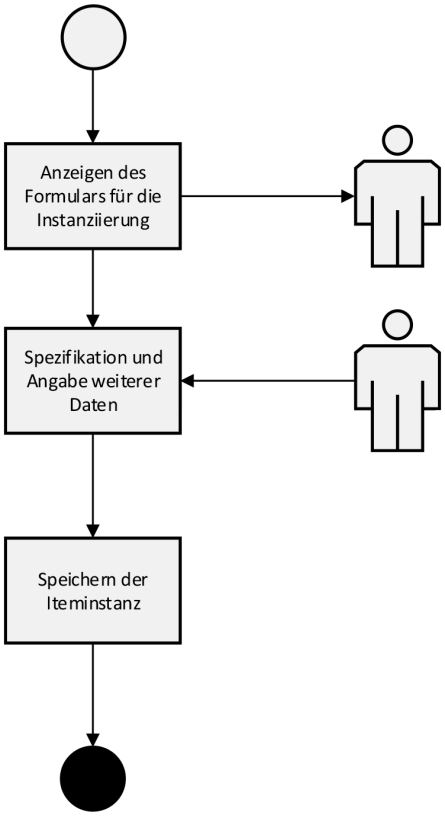


Abbildung 3.42: Erzeugen einer Eintragsinstanz

VCE 5: Suche nach einer Eintragsinstanz

Die Abbildung 3.43 zeigt den durchzuführenden Vorgang, um eine EI zu suchen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Standardnutzer möchte Details einer EI ansehen	Regelmäßig bis selten

Tabelle 3.32: Suche nach einer Eintragsinstanz-VCE5

Was ist bei der Durchführung im Einzelnen zu tun?

Ein Verwalter kann nach EIs in den zugehörigen CLIs suchen, indem er verschiedene Suchkriterien, wie Name, Text und Zustand angibt. Darüber hinaus kann jeder Standardnutzer nach EIs in den zugehörigen CLs suchen, die für ihn relevant sind. Die passenden Ergebnisse werden dem Standardnutzer in einer Liste angezeigt, inklusive der Parameter Name und Text der bestimmten EI.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Findet das System keine EIs, wird dies dem Standardnutzer über einen Warnhinweis mitgeteilt. Hat der Standardnutzer nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die jeweiligen fehlenden Angaben hin. Hat der Standardnutzer ein oder mehrere Textfelder falsch ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Standardnutzer auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

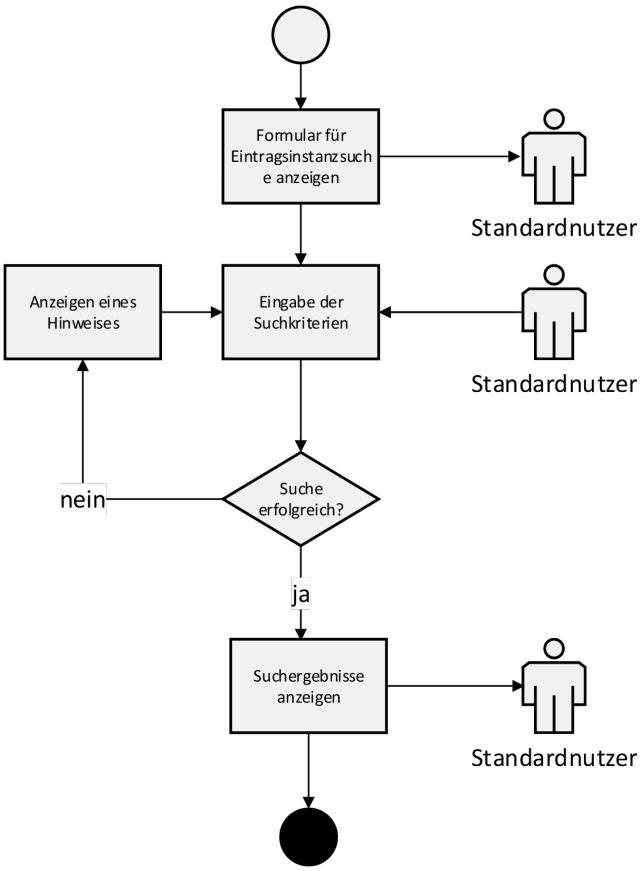


Abbildung 3.43: Suchen nach einer Eintragsinstanz

VCE 6: Ansehen einer Eintragsinstanz

Die Abbildung 3.44 zeigt den durchzuführenden Vorgang, um eine EI anzusehen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Standardnutzer möchte Details einer EI ansehen um sie eventuell zu ändern	Regelmäßig

Tabelle 3.33: Ansehen einer Eintragsinstanz-VCE6

Was ist bei der Durchführung im Einzelnen zu tun?

Nachdem die Suche einer EI erfolgreich abgeschlossen wurde, kann der Standardnutzer auf die Ergebnisse klicken und ihm werden alle Details zu dieser EI angezeigt.

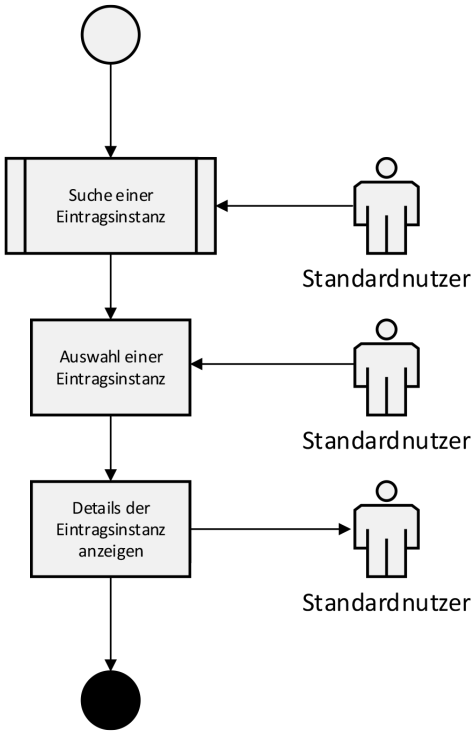


Abbildung 3.44: Ansehen einer Eintragsinstanz

VCE 7: Ändern einer Eintragsinstanz

Die Abbildung 3.45 zeigt den durchzuführenden Vorgang, um eine EI zu ändern.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter will Änderungen an einer EI vornehmen	Regelmäßig bis selten

Tabelle 3.34: Ändern einer Eintragsinstanz-VCE7

Was ist bei der Durchführung im Einzelnen zu tun?

Wird in einer CLI nach verschiedenen EIs gesucht und eine ausgewählt, kann der Verwalter diese ändern. Die Parameter Name, Text und der Zustand im Zusammenhang mit einer CLI können geändert werden. Letztendlich können Verwalter auch den Zustand eines Eintrags auf erledigt setzen.

Welche Ausnahmefälle/Sonderfälle zur Normalvorgehensweise gibt es?

Hat der Verwalter nicht alle Textfelder ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die jeweiligen fehlenden Angaben hin. Hat der Verwalter ein oder mehrere Textfelder nicht korrekt ausgefüllt, wird ihm eine Fehlermeldung angezeigt. Diese Fehlermeldung weist den Verwalter auf die bestimmte Fehleingabe hin und suggeriert ihm, welche Änderungen vorgenommen werden müssen.

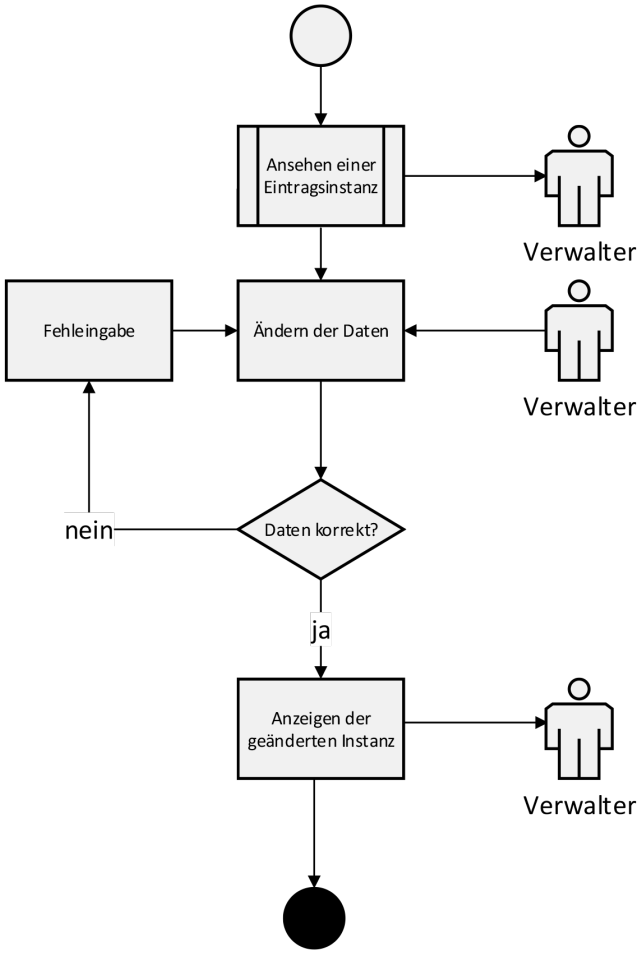


Abbildung 3.45: Ändern einer Eintragsinstanz

VCE 8: Abschließen einer Eintragsinstanz

Die Abbildung 3.46 zeigt den durchzuführenden Vorgang, um eine EI abzuschließen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Standardnutzer	Standardnutzer will den Eintrag als erledigt kennzeichnen	Regelmäßig

Tabelle 3.35: Abschließen einer Eintragsinstanz-VCE8

Was ist bei der Durchführung im Einzelnen zu tun?

Nachdem der Standardnutzer eine EI in einer CLI gewählt hat, kann er die EI als erledigt kennzeichnen, wenn er auf den zugehörigen Link klickt.

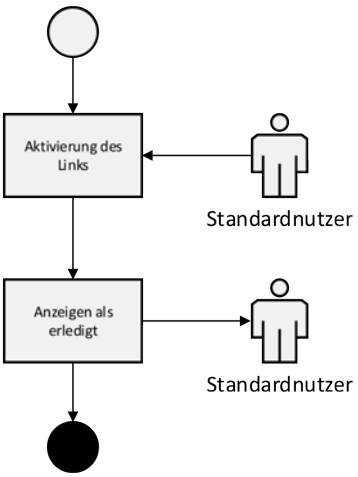


Abbildung 3.46: Abschließen einer Eintragsinstanz

VCE 9: Löschen einer Eintragsinstanz

Die Abbildung 3.47 zeigt den durchzuführenden Vorgang, um eine EI zu löschen.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter benötigt Eintrag nicht mehr oder er ist abgearbeitet	Regelmäßig bis selten

Tabelle 3.36: Löschen einer Eintragsinstanz-VCE9

Was ist bei der Durchführung im Einzelnen zu tun?

Nachdem der Verwalter sich eine EI ausgewählt hat, kann er diese löschen. Hierzu muss er nur auf den zugehörigen Link klicken und den darauf folgenden Warnhinweis quittieren.

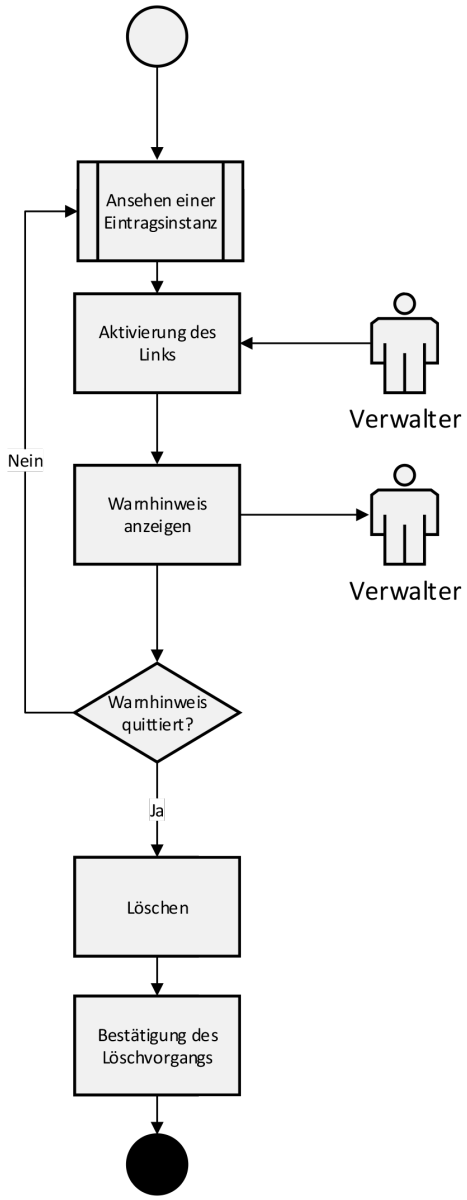


Abbildung 3.47: Löschen einer Eintragsinstanz

3.3.5 Übergreifende Verwaltungsmaßnahmen

Die *übergreifenden Verwaltungsmaßnahmen (UEV)* sind diejenigen Aufgaben des Systems, die sich nicht spezifisch auf eine Kategorie der bisher erläuterten User-Stories beziehen. Das untenstehende Use-Case-Diagramm zeigt alle Aufgaben, die im Anschluss erläutert werden.

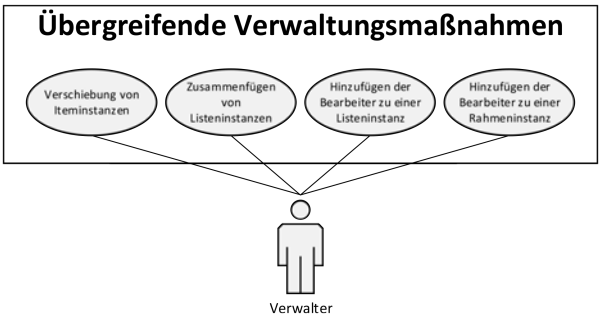


Abbildung 3.48: Use-Case-Diagramm der übergreifenden Verwaltungsmaßnahmen

UEV 1: Verschieben von Eintragsinstanzen

Diese User-Story beschreibt das Verschieben von EIs, durch den zuständigen Verwalter.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter möchte die Anordnung der EIs ändern	Regelmäßig bis selten

Tabelle 3.37: Verschieben von Eintragsinstanzen-UEV1

Was ist bei der Durchführung im Einzelnen zu tun?

Der Verwalter kann EIs untereinander positionieren. Er wählt eine bestimmte EI und fügt diese an einen beliebigen Ort. Beispielsweise kann er Eintragsinstanzen anderer unterordnen oder überordnen.

UEV 2: Zusammenfügen von Checklisteninstanzen

Diese User-Story beschreibt das Zusammenfügen von CLIs, durch den zuständigen Verwalter.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter möchte mehrere CLs zu einer CL verschmelzen	Regelmäßig bis selten

Tabelle 3.38: Zusammenfügen von Checklisteninstanzen-UEV2

Was ist bei der Durchführung im Einzelnen zu tun?

Der Verwalter kann CLIs zusammenfügen.

UEV 3: Hinzufügen der Verwalter zu einer Checklisteninstanz

Diese User-Story beschreibt das Hinzufügen von zusätzlichen Verwaltern zu einer CLI durch den zuständigen Verwalter.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter möchte Aufgaben an Standardnutzer delegieren	Regelmäßig

Tabelle 3.39: Hinzufügen der Verwalter zu einer Checklisteninstanz-UEV3

Was ist bei der Durchführung im Einzelnen zu tun?

Der Verwalter, der für eine CLI zuständig ist, kann mehrere zusätzliche Verwalter zu einer CLI hinzufügen. Hier muss er für jeden einzelnen Verwalter die Nutzerrollen festlegen.

UEV 4: Hinzufügen der Verwalter zu einer organisatorischen Rahmeninstanz

Diese User-Story beschreibt das Hinzufügen von zusätzlichen Verwaltern zu einer ORI, durch den zuständigen Verwalter.

Von wem wird die Aufgabe durchgeführt?	Warum wird die Aufgabe durchgeführt?	Wie oft wird die Aufgabe durchgeführt?
Verwalter	Verwalter möchte Aufgaben an Standardnutzer delegieren	Regelmäßig

Tabelle 3.40: Hinzufügen der Verwalter zu einer organisatorischen Rahmeninstanz-UEV4

Was ist bei der Durchführung im Einzelnen zu tun?

Der Verwalter einer ORI ist befugt, ein oder mehrere zusätzliche Verwalter zu der ORI hinzuzufügen. Er muss beliebig viele Verwalter über ein Eingabefeld hinzufügen. Letztendlich muss er für jeden einzelnen Verwalter die Nutzerrolle festlegen.

3.4 Umgebungsbedingungen

Die Analyse der Umgebungsbedingungen ist ein wichtiger Bestandteil der Anforderungsanalyse, wobei Einflüsse von außen und der Einsatzort identifiziert werden. Diese Aspekte können die Gestaltung des Systems stark beeinflussen [Off12]. Der Einsatzort der zu entwickelnden Applikation ist der Arbeitsplatz der Benutzer. Der Arbeitsplatz ist unter Annahme nicht öffentlich, was bedeutet, dass die Applikation innerhalb eines Gebäudes genutzt wird. Aufgrund dessen sind Aspekte, wie Lärm, Klima oder Lichteinwirkung nicht weiter relevant, da diese Faktoren relativ konstant bleiben.

3.5 Hardware und Software Randbedingungen

Die Analyse der Hardware und Software Randbedingungen ist von entscheidender Bedeutung für die Entwurfs- und Implementierungsphase. Die frühzeitige Analyse unterstützt die Auswahl einer passenden Entwicklungsumgebung und die Entscheidung einer geeigneten Technologie für die Implementierung [Off12].

3.5.1 Hardware Randbedingungen

Da im Kontext der Arbeit eine mobile Anwendung entwickelt werden soll, impliziert dies die Verwendung eines mobilen Geräts als Hardwarekomponente. Laut Jakob Nielsen werden mobile Geräte in mehrere Kategorien eingeteilt [NB12]. Die nachfolgende vorgestellte Kategorie *Full-screen phone* eines mobilen Geräts betrifft die Hardwarekomponente, die im Rahmen dieser Arbeit verwendet wird.

„***Full-screen phones:*** *(such as iPhone, Android and Windows Phone) with nearly device-sized touch screen and a true GUI (graphical user interface) driven by direct manipulation and touch gestures. These phones offer 3G or better Internet connectivity and even faster speeds when connecting through WiFi. They offer the best usability among all different types of phones, but the experience is still often suboptimal. Typically users are successful when using a search engine to land on a deep page in a full site, or when they are on well designed sites or apps that are optimized for mobile [NB12].*“

Wenn in dieser Arbeit von einem *Smartphone* geschrieben wird, ist immer das definierte *Full-screen phone* gemeint. Die Definition zeigt das Smartphones viel Platz für die Präsentation von Informationen auf dem Bildschirm bieten. Über direkte Manipulation des Bildschirms durch Berührungsgesten kann das Smartphone bedient werden. Anders als bei einem Tablet ist das Smartphone in der Größe und im Gewicht kleiner als Tablets und somit handlicher.

3.5.2 Software Randbedingungen

Historisch erfuhr der Mobilsektor erstmals im Jahr 2007 einen regelrechten Wirtschaftsboom. Dieser Verkaufsanstieg ist aufgrund der Einführung des *IPhones* zu verzeichnen [FI10]. Seither entwickelte sich der Smartphone Markt stetig weiter und erfuhr einen rasanten Anstieg an Verkaufszahlen. Mehr als 300 Millionen Menschen besitzen derzeit ein Smartphone, mit jeweils unterschiedlichen Betriebssystemen. In der Abbildung 3.49 wird die Verteilung des Marktanteils, der verschiedenen Plattformen dargestellt. Mit über 50% Anteil ist *Android* der Marktführer unter den Betriebssystemen. Ein Grund dafür ist, dass Google ihre Software *Android* bei mehreren Hardwareherstellern zum Einsatz bringt. Verschiedene Hersteller, wie beispielsweise *HTC*, *LG*, *Motorola* oder *Samsung* verwenden diese Software. Neben dem Marktführer ist erkennbar, dass *iOS* von *Apple* mit einer geringeren Prozentzahl von 23,9% beteiligt ist. Die zwei weiteren Plattformen *Symbian* und *Blackberry* besitzen einen wesentlich geringeren Marktanteil. Aufgrund der Verteilung der beiden Riesen *Android* und *iOS*, ist es sinnvoll eine plattformunabhängige Applikation zu entwickeln, so dass mit geringem Aufwand eine Realisierung auf mehreren Betriebssystemen möglich ist [FI10].

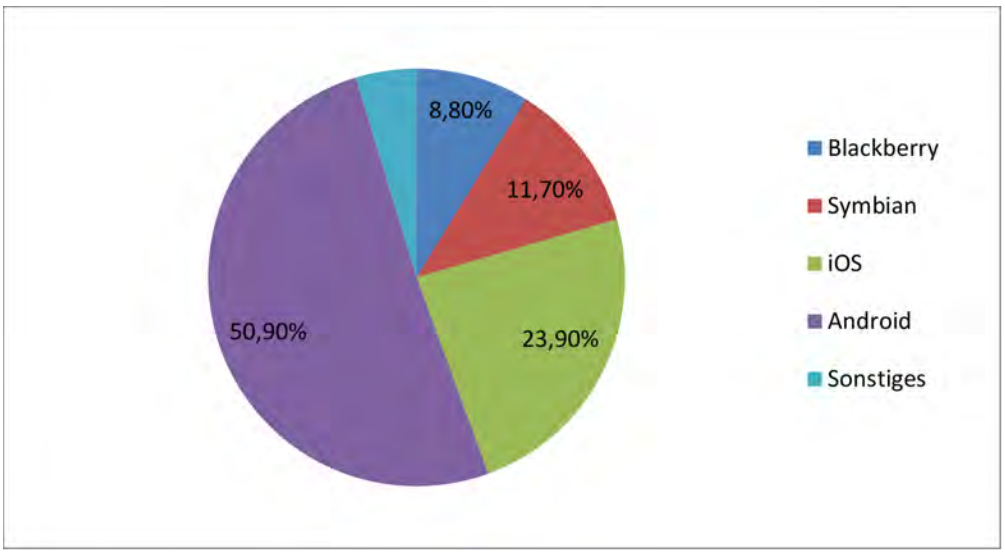


Abbildung 3.49: Marktanteile [FI10]

4

Entwurf

Die Entwurfsphase ist eine präzise und vollständige Beschreibung der Lösungskonzeption. Hier wird das Modell bis auf das technische Konzept konkretisiert [Bal09]. Diese Beschreibung der Lösungskonzeption basiert auf den Resultaten der Anforderungsanalyse. Die gewonnenen Anforderungen werden in diesem Kapitel verfeinert und konkretisiert. Zu Beginn wird in Kapitel 4.1 die ganzheitliche Architektur des Projektes proCollab beschrieben, um eine Übersicht über das Zusammenspiel verschiedener Schichten und Komponenten zu bekommen. Das Datenmodell wird in Kapitel 4.2 detailliert beschrieben, welches Informationen und Mehrwert für die Implementierungsphase liefert. Zusätzlich werden in diesem Kapitel die Anforderungen aus dem Kapitel 3 in konkrete Benutzeroberflächen umgesetzt. Beginnend mit dem konzeptuellen User-Interface-Modell in Kapitel 4.3 wird das Fundament für die Dialogstrukturen festgelegt. Daraufhin werden im folgenden Kapitel 4.4, auf Basis des konzeptuellen Modells, verschiedene Mockups defi-

niert. Um all diese Dialoge einheitlich und konsistent zu halten, werden im letzten Kapitel 4.5 die Styleguideregeln definiert. Einen Überblick über alle Kapitel bietet Abbildung 4.1.

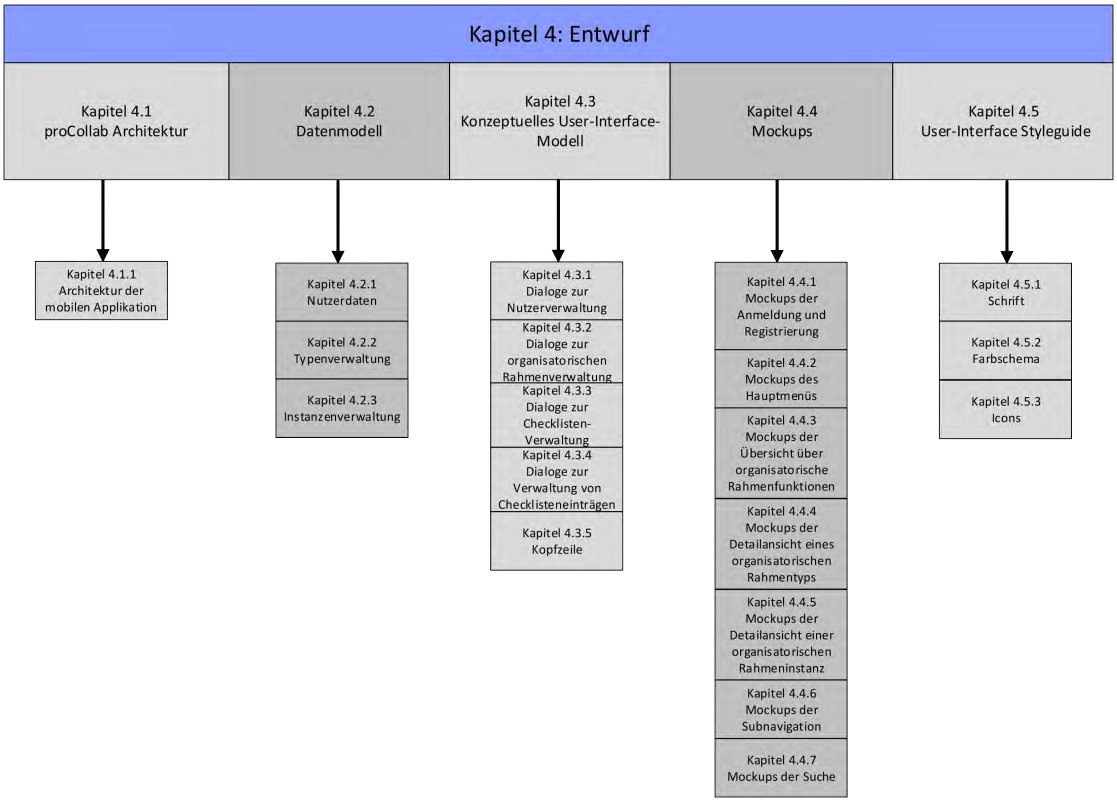


Abbildung 4.1: Aufbau des Kapitels Entwurf

4.1 proCollab Architektur

Das Projekt proCollab (siehe Kapitel 2.2) besitzt ein komplexes Architekturmodell mit mehreren Schichten (Layer). Diese vier Schichten unterteilen sich in weitere Komponenten. Eine Übersicht über alle Schichten und die zugehörigen Komponenten bildet das Architekturmodell in Abbildung 4.2 ab. Die Einteilung in verschiedene Schichten und Komponenten bietet den Vorteil, dass die Entwicklung, die Komplexität und die Änderungen an einzelnen Komponenten vereinfacht werden. Diese Aspekte werden besonders durch die geringe Kopplung und die hohe Bindung gewährleistet, die als

Qualitätsmerkmale gelten. Der Zusammenhang zwischen den beiden Merkmale beschreibt eine qualitativ hochwertige Architektur [Bal09]. Diese Aufteilung ist ein häufig angewandtes Strukturprinzip für die Architektur von Systemen. Im Anschluss werden die vorhandenen Schichten beschrieben:

Persistence Layer: Die Basis der Architektur bildet der *Persistence Layer*. Diese Schicht ist für die persistente Datenspeicherung zuständig, was mit der Anbindung einer oder mehrerer Datenbanken realisiert wird. Der gesamte Austausch von Daten und deren Speicherung bzw. Haltung wird in dieser Schicht bewerkstelligt.

Application Layer: Auf den Persistence Layer folgt die Schicht des *Application Layers*. Der Application Layer umfasst die notwendige Applikationslogik auf Server-Seite, mit den Komponenten Basic Checklist Management, Basic User Management, Basic Frame Management und Basic Repository Management. Hier wird die Logik der einzelnen Komponenten implementiert.

Delivery Layer: Die Schicht des *Delivery Layers* besteht aus den Schnittstellen Rest Interface und Java API, sowie den Komponenten PhoneGap Frontend Applikationen und Vaadin Backend. Hauptsächlich dient diese Schicht zum Datenaustausch und vor allem, zur Zustellung der Daten an den Client Layer. Hier steht vor allem der Transport der Daten vom Back-End zum Front-End im Fokus.

Client Layer: Der *Client Layer* setzt sich aus den Komponenten Smartphone, Tablet und Browser zusammen. Er symbolisiert die verschiedenen Klassen an Endgeräten und Plattformen, wie beispielsweise *Android* und *iOS*. Hier wird dem Benutzer eine grafische Oberfläche zur Verfügung gestellt.

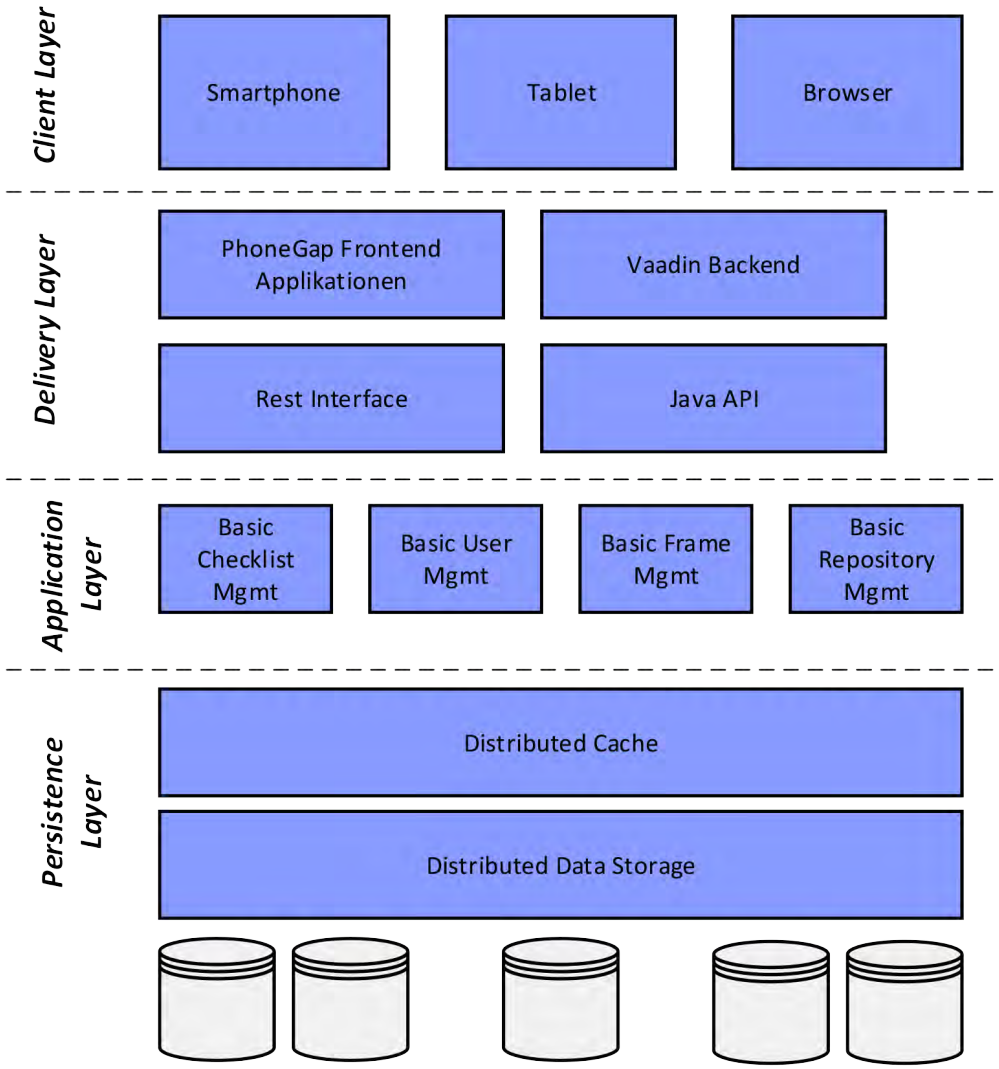


Abbildung 4.2: proCollab Architektur

Im Kontext dieser Arbeit stehen besonders die zwei Schichten Delivery Layer und Client Layer mit den zugehörigen Komponenten PhoneGap Frontend Applikationen und Smartphone im Vordergrund. Die nachstehende Abbildung 4.3 visualisiert das Zusammenspiel der Schichten und Komponenten. Beginnend mit einem Datenaustausch zwischen dem Persistence Layer und dem Application Layer, werden die Daten vom Delivery Layer entgegen genommen. Aufgrund der Zielanforderungen ist dieser Datenaustausch zwi-

schen dem Delivery Layer und dem Application Layer notwendig. Dieser Austausch wird durch ein Zusammenspiel zwischen dem Rest Interface und der zu implementierenden Anwendung erreicht. Die Architektur der zu implementierenden Smartphone Applikation wird im folgenden Kapitel beschrieben.

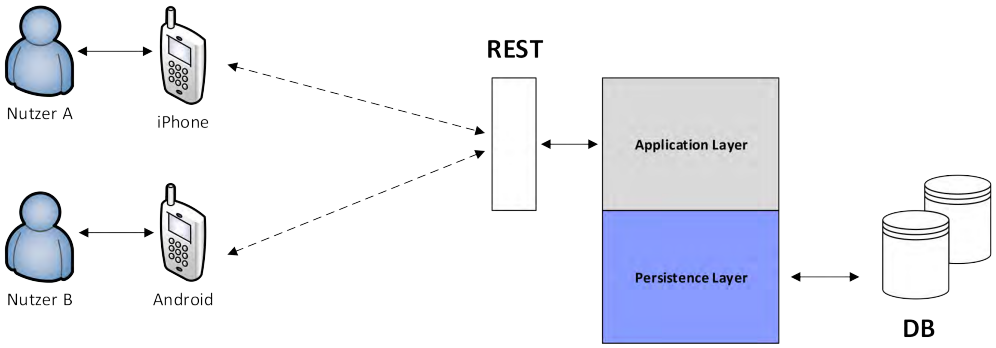


Abbildung 4.3: Zusammenspiel der einzelnen Schichten und Komponenten

4.1.1 Architektur der mobilen Applikation

Die Architektur der zu implementierenden Smartphone Applikation ist in Abbildung 4.4 grafisch dargestellt. Sie unterteilt sich in drei Komponenten: die *View*, den *Controller* und die Komponente *Core*. Durch die Trennung in verschiedene Komponenten, wird die Gesamtkomplexität der Architektur reduziert. Des Weiteren sind die Zuständigkeiten und Aufgaben der einzelnen Komponenten klar definiert. Zudem kann die Applikation, durch die Trennung von *View*-Komponente, *Controller*-Komponente und *Core*-Komponente, einfach erweitert werden. Anschließend werden die verschiedenen Komponenten beschrieben.

View: Die *View* ist für die Präsentation der Daten und Informationen auf dem Bildschirm des Smartphones zuständig.

Controller: Der *Controller* verwaltet die Präsentationen der *View*. Er ist für die Manipulation und Steuerung der Daten und Informationen zuständig.

Core: Der *Core* beinhaltet weitere Komponenten, die vom *Controller*, zur Unterstützung der Manipulation und Steuerung von Daten, verwendet werden. Zudem wird von dieser Komponente auf die serverseitigen *Rest-Schnittstellen* zugegriffen.

Darüber hinaus funktioniert das Zusammenspiel der *Core*-Komponente und der serverseitigen *Rest-Schnittstellen* nur, wenn die auszutauschenden Daten einheitlich definiert werden. Aufgrund dessen wurde ein Datenmodell erstellt, welches im Anschluss beschrieben wird.

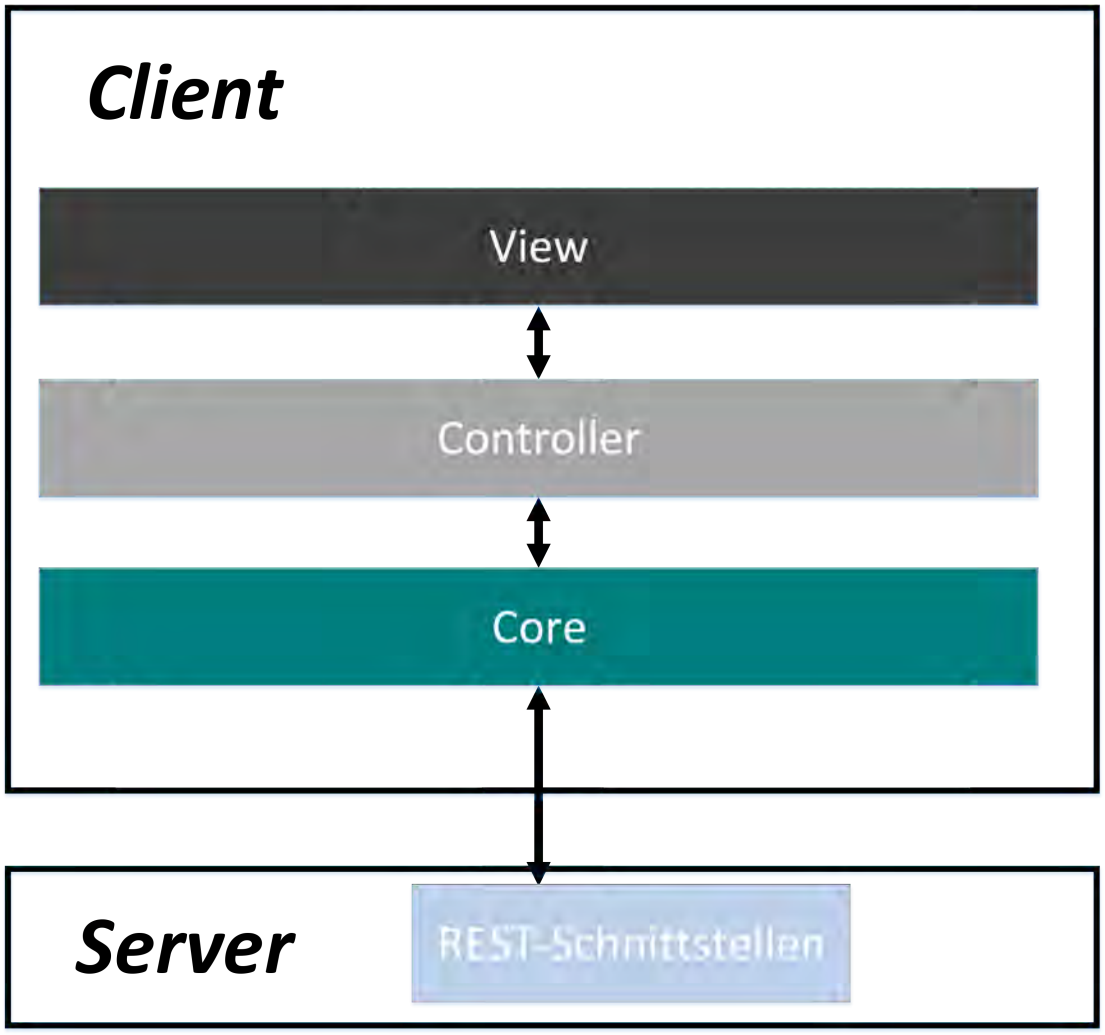


Abbildung 4.4: Architektur der mobilen Applikation

4.2 Datenmodell

Ein Datenmodell beschreibt die zu verarbeitenden Daten des Delivery Layers und ihre Abhängigkeiten untereinander. Des Weiteren gewährleistet ein klar definiertes und spezifiziertes Datenmodell die reibungslose Kommunikation zwischen dem Delivery Layer und dem Application Layer. Das konzipierte Datenmodell wurde in der grafischen Modellierungssprache *Unified Modeling Language* (UML) erstellt. Abbildung 4.5 zeigt das Datenmodell, welches in drei verschiedene Bereiche unterteilt wird: die Nutzerdaten, die Typenverwaltung und die Instanzenverwaltung. In den nachfolgenden Abschnitten werden diese unterschiedliche Bereiche beschrieben.

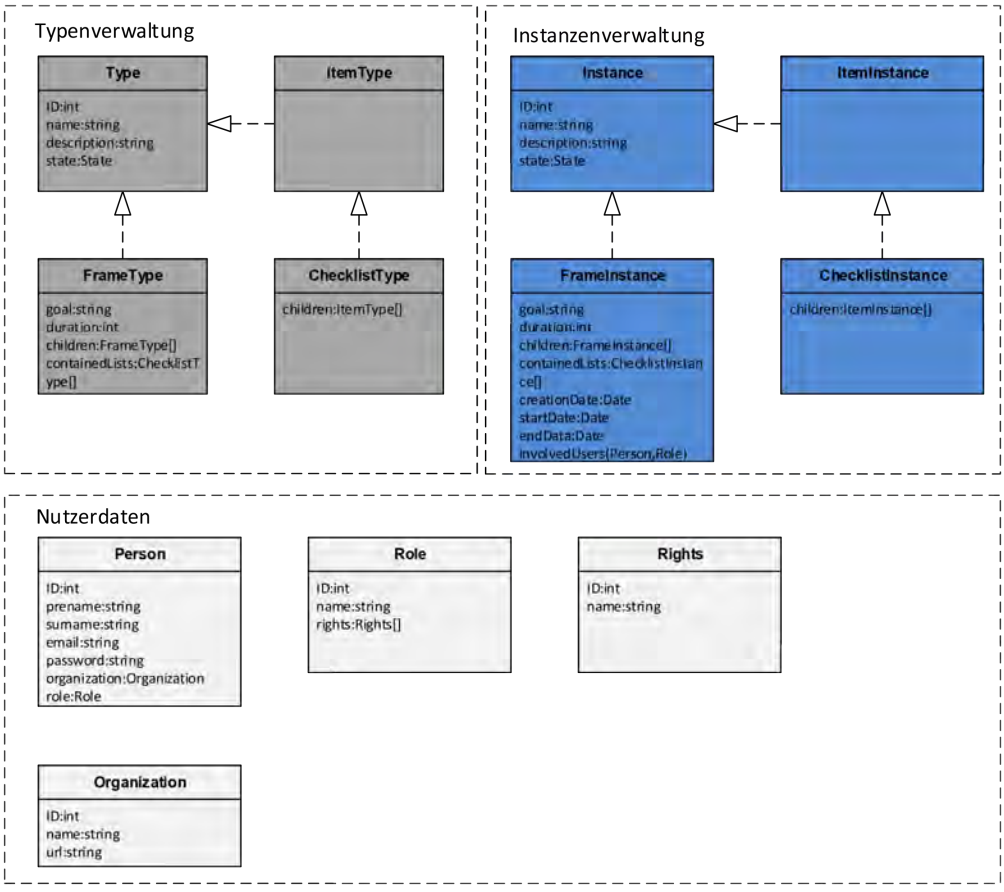


Abbildung 4.5: Datenmodell in UML

4.2.1 Nutzerdaten

Die Nutzerdaten sind im obigen Datenmodell hellgrau markiert. Ein Nutzer besitzt immer eine Identifikationsnummer (ID), einen Vor- und Nachnamen, eine E-Mail-Adresse, ein Passwort und eine Organisation. Die Organisation ist unterteilt in eine ID, den Namen und eine URL. Somit können Beziehungen und Abhängigkeiten zwischen verschiedenen Organisationen und Nutzern realisiert werden. Des Weiteren ist einem Nutzer immer eine bestimmte Rolle zugeordnet, der jeweils mehrere Rechte untergeordnet sind.

4.2.2 Typenverwaltung

Der Bereich der Typenverwaltung ist in der Abbildung 4.5 dunkelgrau markiert. *Type* bildet hierbei die Superklasse, von welcher die zwei Subklassen *FrameType* (siehe Kapitel 2.5.1) und *ItemType* (siehe Kapitel 2.5.3) erben. Die Superklasse enthält alle Attribute, die in den Subklassen ebenfalls vorhanden sind. Ein *Type* jeglicher Art, also ein organisatorischer Rahmentyp, ein Checklistentyp oder ein Eintragstyp, enthält die Attribute Identifikationsnummer (ID), Name, Beschreibung und Status. In der Subklasse *FrameType* werden drei weitere Attribute verwendet: das Ziel, die Dauer, eine Liste von untergeordneten *FrameType*-Objekten (*children[]*) und eine Liste von *ChecklistType*-Objekten (*containedLists[]*). Weiterhin existiert eine Abhängigkeit zwischen *ItemType* und deren Subklasse *ChecklistType* (siehe Kapitel 2.5.2). Ebenso besitzt *ChecklistType* untergeordnete Listen, die allerdings vom Typ *ItemType* sind. Ähnliche Verwendung findet dieses Vererbungsprinzip im nachfolgendem Bereich der Instanzenverwaltung.

4.2.3 Instanzenverwaltung

Die blau markierten Bereiche in der Darstellung 4.5 bilden die Instanzenverwaltung. *FrameInstance* (siehe Kapitel 2.5.1) und *ItemInstance* (siehe Kapitel 2.5.3) erben die Attribute ID, Name, Beschreibung und Status von der Superklasse *Instance*. Darüber hinaus wird *FrameInstance* um die Attribute Ziel, Dauer, Erstelldatum, Startdatum, Enddatum und einem Tupel von beteiligten Personen und ihrer zugehörigen Rolle erweitert.

Die Assoziation zur *Instance*-Klasse erfolgt durch eine Liste von weiteren *FrameInstance*-Objekten (*children[]*) und *ChecklistInstance*-Objekten (*containedLists[]*). Des Weiteren wird die *ChecklistInstance* (siehe Kapitel 2.5.2), welche eine Subklasse der *ItemInstance* ist, um eine Liste von *ItemInstance*-Objekten erweitert.

Um die beschriebenen Daten auf den mobilen Endgeräten und explizit in verschiedenen Dialogen der Anwendung darzustellen, ist es notwendig diese Dialoge und Navigationspfade der Anwendung zu charakterisieren. Im folgenden Kapitel 4.3 werden diese, beginnend mit dem konzeptuellen User-Interface-Modell erläutert.

4.3 Konzeptuelles User-Interface-Modell

Das konzeptuelle User-Interface-Modell ist die übergeordnete Sicht auf die Gesamtheit aller Dialoge. Dieser Abschnitt ist der erste Schritt des User-Interface-Entwurfs. Es werden grundsätzliche Dialogstrukturen und Interaktionsmöglichkeiten festgelegt, welche das Fundament von Mockups (siehe Kapitel 4.4) bilden [Off12]. In Abbildung 4.6 werden alle Dialoge und ihre Übergänge dargestellt. Auf Basis der User Stories (siehe Kapitel 3.3) wurden Aufgaben identifiziert, die Ähnlichkeiten in den Funktionsabläufen aufweisen. Demnach wurden aus diesen Aufgaben generelle Sichten abgeleitet, die den selben Aufbau der Benutzeroberfläche besitzen. Die farblich identisch markierten Dialoge besitzen den gleichen Oberflächenaufbau und die selben fundamentalen Sichten.

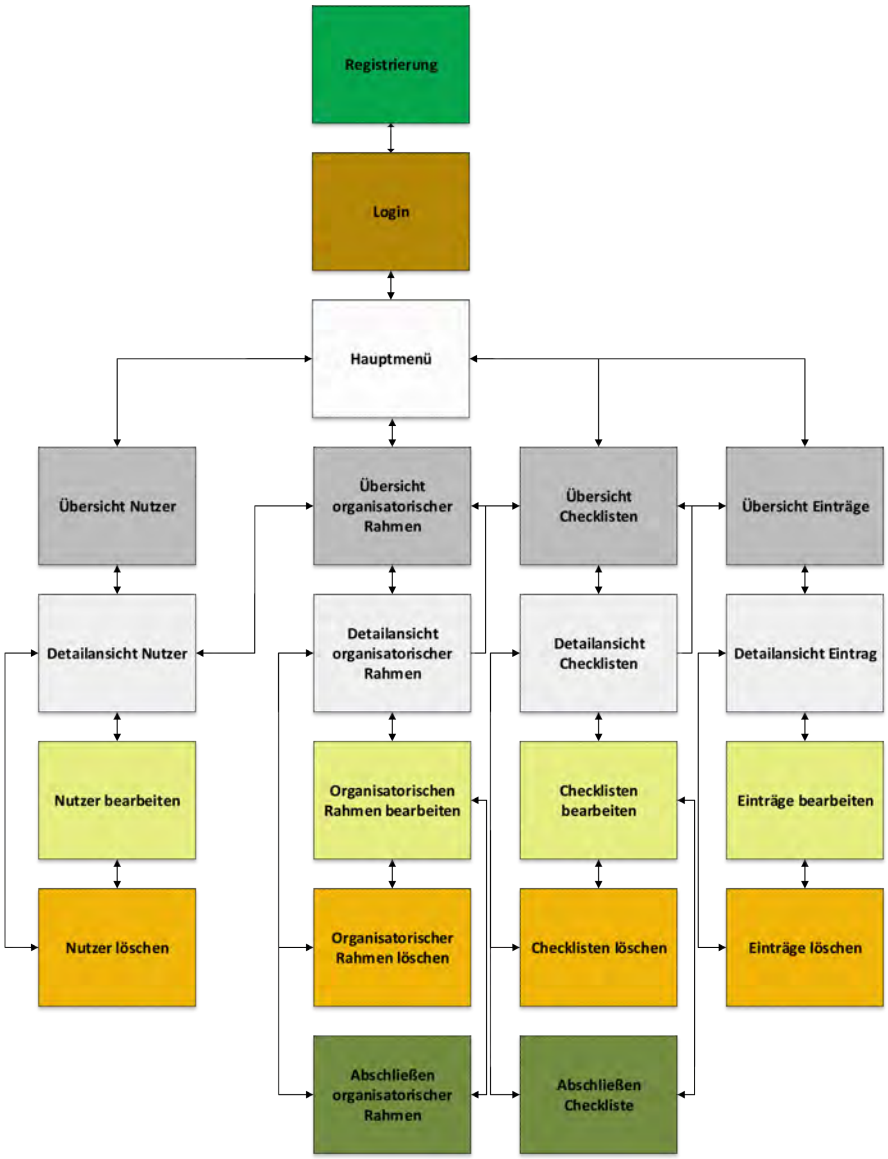


Abbildung 4.6: Ganzheitliches Dialogstrukturdiagramm

Aufbauend auf dem ganzheitlichen Dialogstrukturdiagramm (siehe Abbildung 4.6) ergeben sich detailliertere Dialogstrukturdiagramme für die einzelne Bereiche. Aus Gründen der Übersichtlichkeit werden diese in vier Kategorien unterteilt: die Nutzerverwaltung, die organisatorische Rahmenverwaltung, die Checklisten-Verwaltung und die Verwaltung von Checklisteneinträgen (siehe Kapitel 3.3). Um die Struktur und Komplexität

der Dialogstrukturdiagramme zu minimieren, wurden in den Diagrammen verschiedene Farben verwendet. In Abbildung 4.7 werden die Varianten der Farbgebung dargestellt und erläutert.

Farbe	Bedeutung
<Dialog>	Dieses Element des Dialogstrukturdiagramms beschreibt einen normalen Dialog ohne zusätzliche Bedeutung.
<Dialog> Hauptmenü	Dieses Element des Dialogstrukturdiagramms steht für das Hauptmenü. Da das Hauptmenü das Zentrum der Anwendung darstellt, wurde es mit einem auffallenden Gelbton gekennzeichnet.
	Dieses Element des Dialogstrukturdiagramms beschreibt einen Dialog, der an einer anderen Stelle schon verwendet wird und weitere Navigationspfade besitzt.

Abbildung 4.7: Farbbedeutung in den Dialogstrukturdiagrammen

4.3.1 Dialoge zur Nutzerverwaltung

Die Aufgabenanalyse aus Kapitel 3.3, im speziellen die Ergebnisse des Kapitels 3.3.1, liefern die Grundlagen, um die Dialoge passend zu konstruieren. Die Abbildung 4.8 stellt die notwendigen Dialoge und die möglichen Navigationspfade dar. Der Ablauf bei der primären Nutzung legt die Reihenfolge einiger Dialoge hierarchisch fest. Beginnend bei dem Dialog der Anmeldung, muss der Gast vorerst eine Registrierung durchführen, da er noch keine Anmeldedaten besitzt. Ist der Nutzer erfolgreich registriert, wird ihm das Hauptmenü angezeigt. Daraufhin kann der Nutzer entweder weitere Nutzer suchen oder sein eigenes Nutzerkonto ansehen. Sucht man nach weiteren Nutzern und wählt anschließend ein Suchergebnis aus, wird eine Übersicht über die Kontodaten angezeigt. Falls es das eigene Nutzerkonto ist, kann man seine Daten ändern oder sein Konto löschen. Meldet sich der Nutzer ab, wird ihm der Anmeldedialog angezeigt.

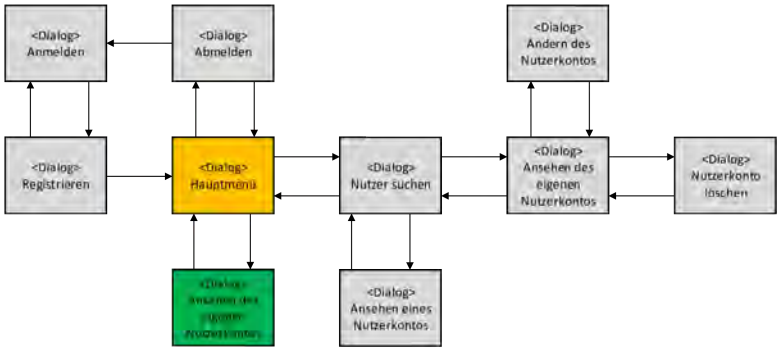


Abbildung 4.8: Dialogstrukturdiagramm der Nutzerverwaltung

4.3.2 Dialoge zur organisatorische Rahmenverwaltung

Das Kapitel 3.3.2 liefert die fundamentalen Erkenntnisse, um die Dialoge der organisatorischen Rahmenverwaltung passend zu konstruieren. Das Diagramm in Abbildung 4.9 zeigt alle Dialoge, die im Bezug auf die ORV relevant sind. Anfangs kann der Nutzer vom Hauptmenü aus in eine Übersicht navigieren, in welcher Informationen über ORTs und ORIs komprimiert dargestellt werden. Daraufhin kann der Nutzer nach ORTs und ORIs suchen oder eine eigene ORI erzeugen. Sucht der Nutzer nach einem ORT, kann er ein Ergebnis anzeigen lassen und den ORT, falls gewollt, instanziieren. Wird eine passende ORI ausgewählt, können die Details geändert werden oder die ORI abgeschlossen werden. Wird die ORI nicht mehr gebraucht, besteht die Möglichkeit die ORI zu löschen.

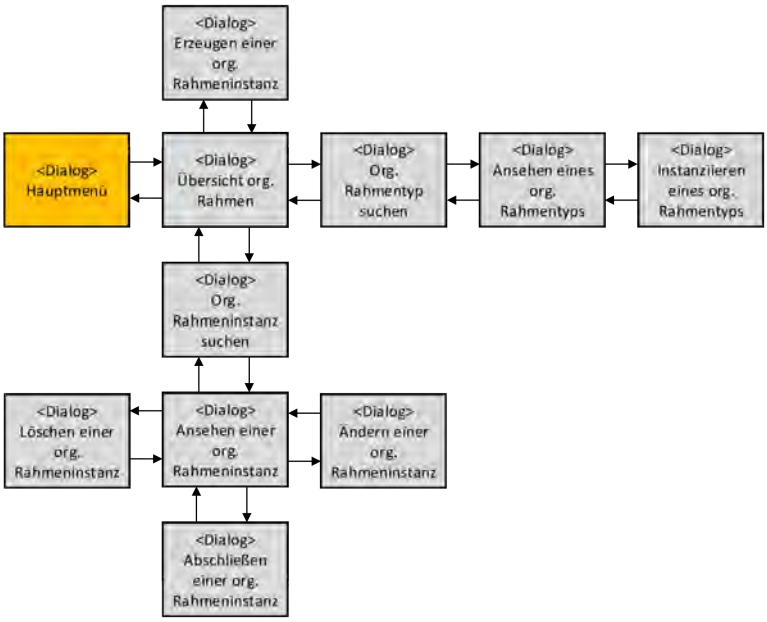


Abbildung 4.9: Dialogstrukturdiagramm der organisatorischen Rahmenverwaltung

4.3.3 Dialoge zur Checklisten-Verwaltung

Aufbauend auf Kapitel 3.3.3 wurden die Dialoge und Navigationspfade der Checklisten-Verwaltung konzipiert. Zu Beginn kann der Nutzer vom Hauptmenü zur Checklistenübersicht navigieren. Ferner kann er nach CLTs und CLIs suchen oder eine CLI erzeugen. Die weiteren Pfade ähneln denen, die im vorherigen Kapitel erläutert wurden. Wird ein CLT ausgewählt, kann dieser anschließend instanziiert werden. Wird jedoch nach einer CLI gesucht, wird dem Nutzer die Detailansicht angezeigt. Von diesem Dialog aus kann der Nutzer nun zu den üblichen Dialoge *Löschen einer CLI*, *Ändern einer CLI* und *Abschließen einer CLI* navigieren. Die beschriebenen Navigationspfade sind in der Darstellung 4.10 ersichtlich.

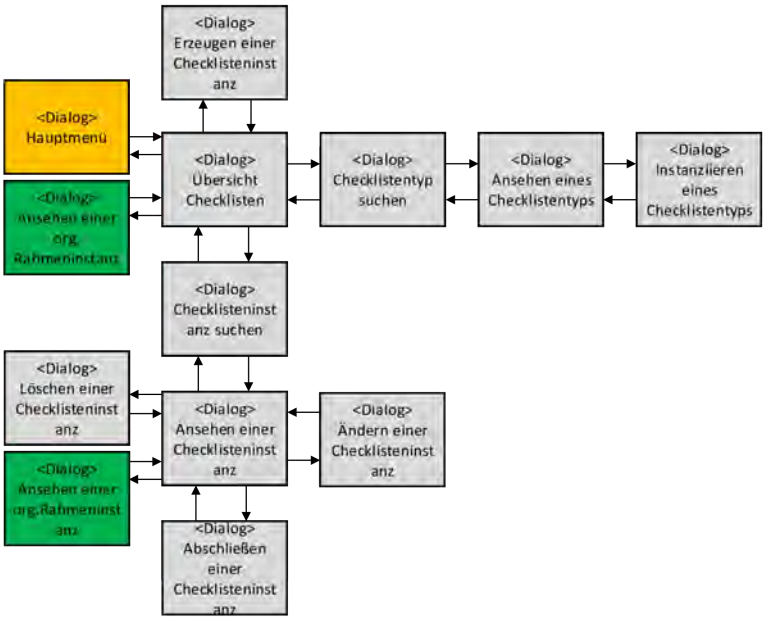


Abbildung 4.10: Dialogstrukturdiagramm der Checklisten-Verwaltung

4.3.4 Dialoge zur Verwaltung von Checklisteneinträgen

Letztendlich wird, basierend auf Kapitel 3.3.4, in diesem Abschnitt der Ablauf für die Benutzerführung der Verwaltung von Checklisteneinträgen beschrieben. Die Dialoge der VCE unterscheiden sich zu den vorherigen Navigationspfaden nur durch die Nahtstellen zur Checklisten- und organisatorischen Rahmenverwaltung. Ferner gibt es keinen Dialog, der das Abschließen der EI visualisiert. Startend beim Hauptmenü, kann der Nutzer in den Dialog *Übersicht Einträge* wechseln. Die weiteren Dialoge und Interaktionsmöglichkeiten sind konform zu denen der ORV und CLV, weshalb diese nicht wiederholt explizit erläutert werden. Die folgende Abbildung 4.11 stellt alle relevanten Sichten und Navigationspfade dar.

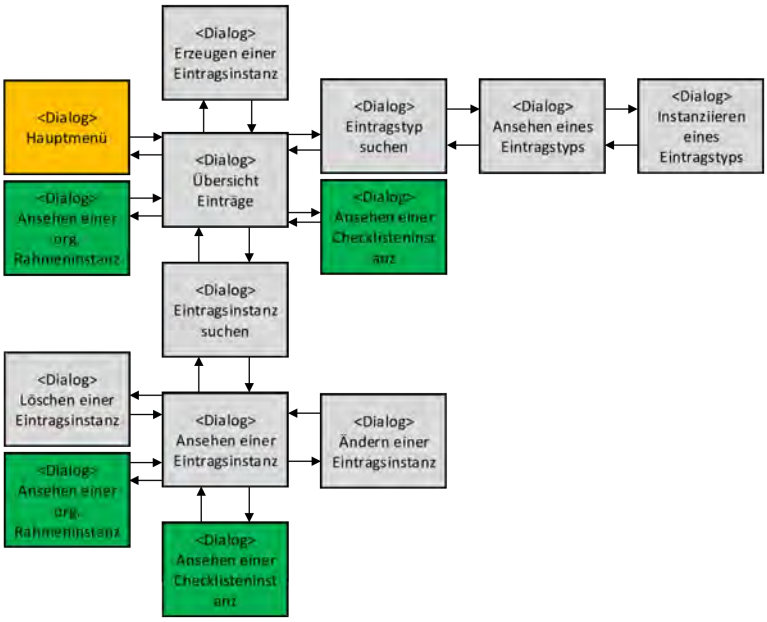


Abbildung 4.11: Dialogstrukturdiagramm der Verwaltung von Checklisteneinträgen

4.3.5 Kopfzeile

Die Kopfzeile befindet sich am oberen Rand der mobilen Anwendung. In jedem Dialog ist sie sichtbar und fixiert an der gleichen Stelle positioniert. Die Kopfzeile dient der Orientierung und Navigation zwischen den verschiedenen Dialogen. Insgesamt wurden zwei unterschiedliche Kopfzeilen entwickelt, die beide den Namen des Dialogs anzeigen, in dem man sich aktuell befindet. Die erste bietet den Nutzern die Möglichkeiten auf den vorher besuchten Dialog zu wechseln und in das Hauptmenü zu navigieren (siehe Abbildung 4.12). Die Navigationsmöglichkeit von jedem Dialog der Anwendung in das Hauptmenü, ermöglicht es den Nutzern jederzeit effizient auf andere Funktionen der mobilen Anwendung zuzugreifen. Darüber hinaus kann jeder Nutzer bestimmte Aktionen abbrechen und sich im Hauptmenü neu orientieren. Diese Kopfzeile ist in jedem Dialog, außer dem Hauptmenü, sichtbar.

Abbildung 4.12: Kopfzeile

Der Grund für die Konzipierung einer weiteren Kopfzeile ist, dass es nicht zielführend ist dem Nutzer die Möglichkeit zu bieten, über die Kopfzeile in das Hauptmenü zu navigieren, wenn er sich dort schon befindet. Deshalb wurde für das Hauptmenü eine Kopfzeile entwickelt, mit der der Nutzer einerseits wieder auf den vorherigen Dialog wechseln kann und andererseits sich abmelden kann (siehe Abbildung 4.13).

Abbildung 4.13: Kopfzeile im Hauptmenü

4.4 Mockups

Der nächste Schritt der Entwurfsphase ist die Entwicklung der Oberflächen anhand von Mockups. Hierfür liefert das im vorherigen Kapitel beschriebene konzeptuelle User-Interface Modell (siehe Kapitel 4.3) die Grundlage. Die Mockups stellen den wesentlichen Teil der zu entwickelnden Dialoge dar und evaluieren somit die konzipierten Interaktionsmöglichkeiten. Im Folgenden werden die ausgearbeiteten Mockups vorgestellt.

4.4.1 Mockups der Anmeldung und Registrierung

Die folgende Abbildung 4.14 zeigt die Dialoge der Anmeldung (siehe Markierung 1) und der Registrierung (siehe Markierung 2). Im Anmeldedialog kann der Nutzer entweder seine Anmeldedaten in die zwei Textfelder eintragen und danach den Anmeldebutton klicken, oder er wählt den Registrierungsbutton, woraufhin der Registrierungsdialog angezeigt wird. Der Dialog der Registrierung zeigt alle auszufüllenden Textfelder (Vorname,

Nachname, E-Mail etc.), die notwendig sind, um sich im System zu registrieren. Sind alle Textfelder ausgefüllt kann der Nutzer über den Registrieren Button seine Formulardaten abschicken und das Hauptmenü wird ihm angezeigt.

Der Anmelde- und Registrierungsdialog enthält alle Bedien- und Eingabeelemente um die Aufgabe des Anmeldens bzw. des Registrierens zu erfüllen (siehe *BFK: Nützlichkeit*, in Kapitel 3.2.3). Durch die räumliche Nähe der Eingabefelder und die Abgrenzung dieser zu den Buttons, ist das Benutzerfreundlichkeitskriterium der Übersichtlichkeit (siehe Kapitel 3.2.3) gewährleistet. Zusätzlich wird die Übersichtlichkeit durch die unterschiedliche Farben der Eingabefelder und der Buttons erreicht.

Abbildung 4.14: Mockups zu den Dialogen Anmeldung und Registrierung

4.4.2 Mockups des Hauptmenüs

Nach erfolgreicher Anmeldung eines Nutzers im System, wird ihm das Hauptmenü angezeigt. Abbildung 4.15 visualisiert zwei unterschiedliche Varianten. Die erste Variante

(siehe Markierung 1) zeigt im Hauptmenü die verschiedenen Funktionen, welche durch einfache Listenelemente dargestellt sind. Zusätzlich befindet sich das proCollab-Logo am Anfang des Dialogs, welches bei der zweiten Variante nicht dargestellt ist. Durch entfernen des Logos bietet die Bildschirmoberfläche mehr Platz für die Funktionen der mobilen Anwendung.

Insgesamt bietet das Hauptmenü eine übersichtliche Auflistung aller wichtigen Funktionen, die Nutzer mit der mobilen Anwendung ausführen können. Des Weiteren ist die Reihenfolge der Listenelemente relevant. Das erste Listenelement bildet die organisatorische Rahmenfunktionen ab, die wiederum die Checklisten-Verwaltung und die Verwaltung von Checklisteneinträgen beinhaltet. Aufgrund der Funktionenvielfalt der organisatorischen Rahmen, ist dieses Listenelement an erster Stelle platziert. Darüber hinaus wird in den nächsten Listenelementen erst die Checklisten-Verwaltung abgebildet und dann die Verwaltung von Checklisteneinträgen, da die Checklisten-Verwaltung die Verwaltung von Checklisteneinträgen beinhaltet. Die Weiteren Listenelemente bilden die Suche und die Nutzerverwaltung ab, welche nicht zu den Hauptfunktionen der mobilen Anwendung zählen.

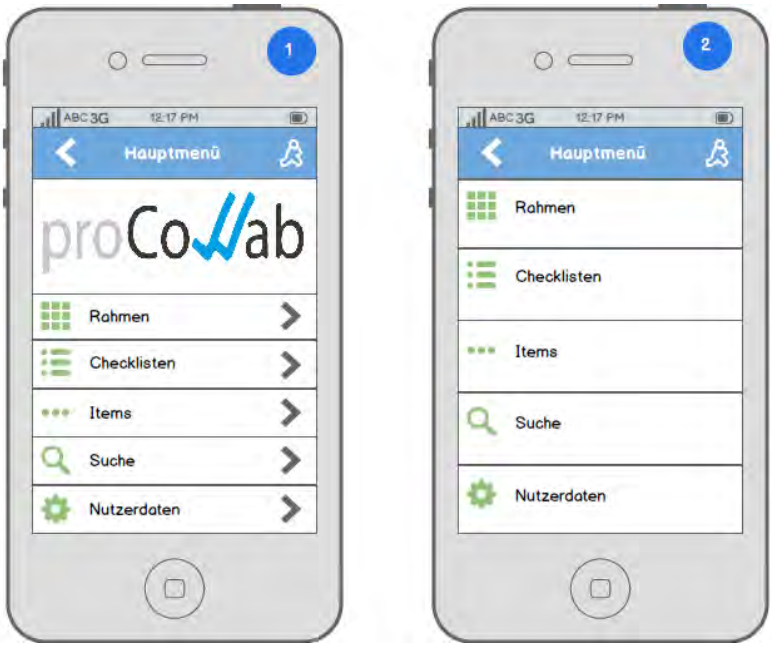


Abbildung 4.15: Mockups des Hauptmenüs

4.4.3 Mockups der Übersicht über die organisatorischen Rahmenfunktionen

Die in Abbildung 4.16 dargestellten Dialoge visualisieren die Übersicht über die organisatorische Rahmenfunktionen, zu der der Nutzer über das Hauptmenü navigieren kann. Insgesamt besitzt der Nutzer die Wahlmöglichkeit zwischen drei verschiedenen Funktionen: organisatorische Rahmentypen suchen, organisatorische Rahmeninstanzen suchen und der Erzeugung einer individuellen organisatorischen Rahmeninstanz (siehe Kapitel 2.5.1). In der ersten Variante (siehe Markierung 1) ist die Funktion der Erzeugung eines organisatorischen Rahmentyps als Button dargestellt, wobei in der zweiten Variante (siehe Markierung 2) diese Funktion als Listenelement abgebildet wird.

Alle Funktionen der organisatorischen Rahmenverwaltung werden hier zusammengefasst als Übersicht dargestellt (siehe *BFK: Nützlichkeit*, in Kapitel 3.2.3). Des Weiteren sind die Funktionen als Listenelemente untereinander angeordnet, was im Hauptmenü

ebenso der Fall ist (siehe Kapitel 4.4.2). Hiermit wird die Erwartungskonformität des Nutzers sicher gestellt und die Lernförderlichkeit, durch das Wiederkehren der Listenelemente, unterstützt (siehe *BFK: Lernförderlichkeit*, in Kapitel 3.2.3). Klickt der Nutzer auf das Listenelement der organisatorischen Rahmentypen, wird ihm eine Detailansicht angezeigt, die im Folgenden beschrieben wird.

Abbildung 4.16: Mockups der Übersicht über die organisatorischen Rahmenfunktionen

4.4.4 Mockups der Detailansicht eines organisatorischen Rahmentyps

Die nachstehende Abbildung 4.17 stellt die Detailansicht eines ausgewählten organisatorischen Rahmentyps dar. Markierung 1 zeigt die zugehörigen Subtypen des organisatorischen Rahmentyps. Um die Details der Subtypen anzusehen, muss der Nutzer nur das jeweilige Listenelement anwählen, woraufhin der Name dieses Subtypen in der Subnavigation (siehe Markierung 2) aufgelistet wird. Über die Subnavigation können die Nutzer ihre Navigationspfade nachvollziehen und zusätzlich zu den Detailansichten der vorherig gewählten Subtypen wechseln. Darüber hinaus besitzen die Nutzer die Möglichkeit die angegebenen Daten des organisatorischen Rahmentyps anzusehen (siehe Markierung

3). Hierzu müssen sie nur ihren Finger von rechts nach links ziehen und der Dialog der organisatorischen Rahmendaten wird ihnen angezeigt. Weiterhin haben die Nutzer nun die Möglichkeiten den organisatorischen Rahmentyp zu instanziieren, Bearbeiter hinzuzufügen und die genauen Daten des organisatorischen Rahmentyps anzusehen (siehe Markierung 4). Diese Dialogansicht wird für die Darstellung der organisatorischen Rahmeninstanzen wiederverwendet, unter Erweiterung einiger Oberflächenelemente.

Die Dialoge unterstützen den Nutzer bei der Erledigung seiner Aufgaben (siehe *BFK: Nützlichkeit*, in Kapitel 3.2.3). Des Weiteren wird in der Kopfzeile angegeben in welcher Sicht sich der Nutzer gerade befindet. Die Subnavigation, die sich unterhalb der Kopfzeile befindet, weist den Nutzer daraufhin, wo er sich explizit in einem ORT befindet (siehe *BFK: Übersichtlichkeit, in Kapitel 3.2.3*). Der Button *Instanziieren* ist in dieser Sicht unterhalb der Eingabeelemente angeordnet, wie auch im Anmelde- und Registrierungsdialog (siehe Kapitel 4.4.1). Dieses Ordnungsschema der Oberflächenelemente unterstützt das Benutzerfreundlichkeitskriterium der Lernförderlichkeit (siehe Kapitel 3.2.3). Darüber hinaus wurden in Markierung 3 *Collapsible*-Elemente verwendet, um die Informationen auf der Benutzeroberfläche zu reduzieren.

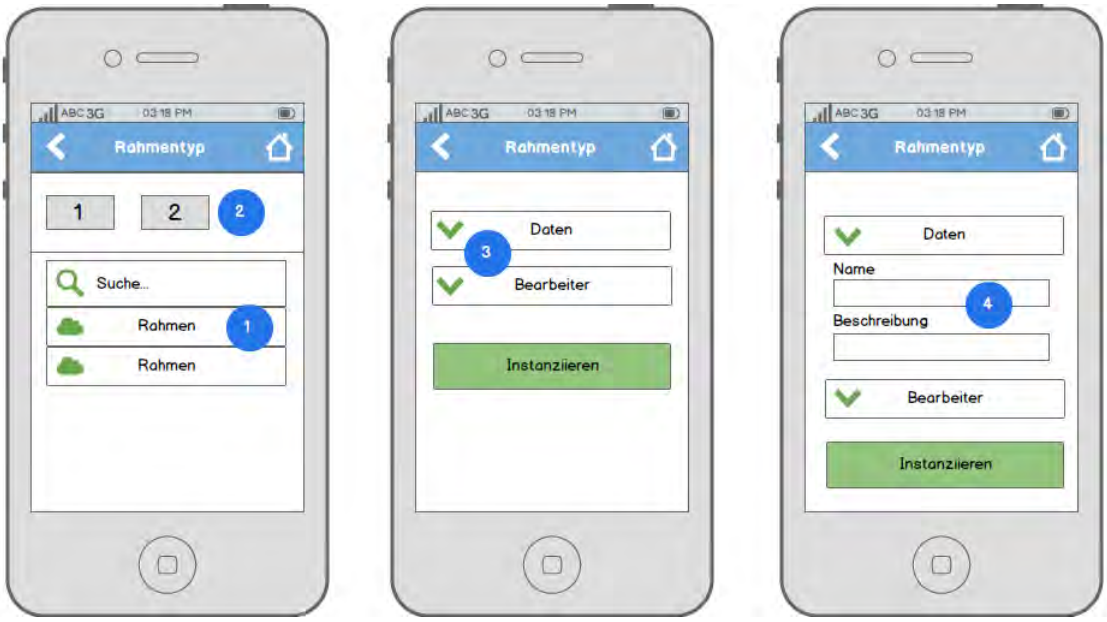


Abbildung 4.17: Mockups der Detailansicht eines organisatorischen Rahmentyps

4.4.5 Mockups der Detailansicht einer organisatorischen Rahmeninstanz

In der Abbildung 4.18 ist der Dialog der Detailansicht einer organisatorischen Rahmeninstanz visualisiert. Unterhalb einer Suchleiste werden mögliche Subinstanzen (siehe Markierung 1) der ersten Hierarchieebene angezeigt. Zusätzlich werden Buttons unter den Subinstanzen angeordnet, mit welchen die Nutzer weitere Instanzen hinzufügen können (siehe Markierung 2). Mit der Fingerbewegung von rechts nach links können die Nutzer (siehe Kapitel 4.4.4) die Daten der organisatorischen Rahmeninstanz ansehen und ändern.

Der Aufbau und die Struktur der Benutzeroberfläche gleicht dem der Detailansicht von organisatorischen Rahmentypen (siehe Kapitel 4.4.4). Durch diese erwartungskonforme Strukturierung wird die Lernförderlichkeit (siehe Kapitel 3.2.3) des Nutzers unterstützt. Des Weiteren steigert der Einsatz des konsistenten Farbschemas das Kriterium der Übersichtlichkeit (siehe Kapitel 3.2.3). Die Buttons sind immer in der gleichen Farbe

dargestellt, ebenso wie die Kopfzeile und sonstige Bedien-und Eingabeelemente (siehe Kapitel 4.5.2).

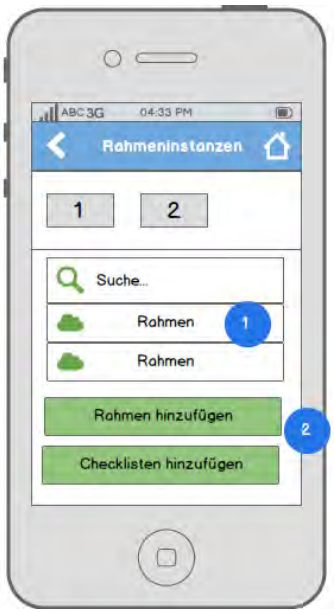


Abbildung 4.18: Mockup der Detailansicht einer organisatorischen Rahmeninstanz

4.4.6 Mockups der Subnavigation

Da die hierarchischen Navigationswege in der Verwaltung von Instanzen für den Nutzer nachvollziehbar sein sollen, wurden verschiedene Varianten der Subnavigation entwickelt (siehe Abbildung 4.19). Die erste Möglichkeit (siehe Markierung 1) zeigt die von links nach rechts ausgewählten Dialoge, die in einem Rechteck angezeigt werden. Eine weitere Möglichkeit, welche in Markierung 2 dargestellt ist, bildet je einen Navigationspfad in einem Rechteck ab. Die folgenden Navigationspfade werden in einem individuellen Rechteck platziert, wobei sich die jeweilige Farbgebung der verschiedenen Rechtecke unterscheidet. Die Subnavigation wird direkt unterhalb der Kopfzeile angeordnet (siehe Markierung 3).

Die Subnavigation steigert den Aspekt der Übersichtlichkeit (siehe *BFK:Übersichtlichkeit*, in Kapitel 3.2.3), da der Nutzer zu jedem Zeitpunkt weiß wo er sich befindet. Zudem wurde die Subnavigation unterhalb der Kopfzeile platziert. Der Aufbau der Subnavigation erfolgt waagerecht, anstatt senkrecht, von links nach rechts, wodurch Platz auf der Benutzeroberfläche eingespart wird.

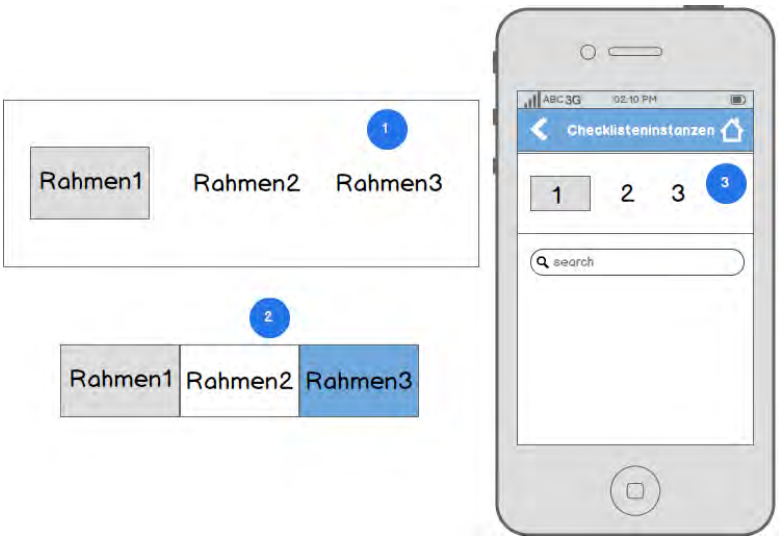


Abbildung 4.19: Mockups zu den Varianten der Subnavigation

4.4.7 Mockups der Suche

Dieser generische Dialog ermöglicht es den Nutzern nach organisatorischen Rahmeninstanzen, Checklisten, weiteren Nutzer etc. zu suchen. Dem Nutzer stehen verschiedene Filterfunktionen zur Verfügung, die ihm per klick eines Buttons angezeigt werden. Mit den verfügbaren Filtermöglichkeiten kann die Suche konkretisiert werden. Des Weiteren bietet der Dialog ein Eingabefeld, indem der Nutzer sein Suchwort eingeben kann. Die Suchergebnisse werden gleichzeitig zur Eingabe bei jedem eingegebenen Buchstaben oder bei jeder eingegebenen Zahl gefiltert und angezeigt. Dazu werden die anklickbaren Suchergebnisse in Form einer Liste unter die Eingabemaske platziert.

Um die Informationen und auf der Benutzeroberfläche zu reduzieren wurde für die Filterfunktionen eine *Collapsible* verwendet (siehe Kapitel 4.4.4). Da Filterfunktionen nicht immer verwendet werden, kann der Nutzer diese nur bei Bedarf ausklappen. Dies steigert die Übersichtlichkeit der Benutzeroberfläche (siehe Kapitel 3.2.3).

Abbildung 4.20: Mockups für einen generischen Such-Dialog

4.5 User-Interface Styleguide

In diesem Kapitel werden anwendungsspezifische Styleguideregeln definiert, die bei der Implementierung der Benutzeroberfläche eingehalten werden. Somit wird die Einheitlichkeit und Konsistenz des User-Interfaces gewährleistet [Off12]. Das erste Kapitel 4.5.1 beschreibt die Eigenschaften der in der mobilen Anwendung verwendeten Schrift. Des Weiteren wird in Abschnitt 4.5.2 das Farbschema erläutert. Abschließend umfasst Kapitel 4.5.3 den Entwurf und die Umsetzung der eingesetzten Icons.

4.5.1 Schrift

Die Auswahl der richtigen Schriftart und Schriftgröße ist sehr wichtig in mobilen Anwendungen. Dadurch das die Bildschirmoberfläche wenig Platz bietet, muss darauf geachtet werden, dass die Schriftgröße nicht zu klein wird und lesbar bleibt. Des Weiteren sind einige Schriftarten ungeeignet für den Einsatz in einer Smartphone Applikation, da sie dem Nutzer ein unangenehmes Lesegefühl bereiten. Beispielsweise sollten eine serifenlose Schrift verwendet werden, da diese als angenehm wahrgenommen werden [And13]. In der mobilen Anwendung wurde die *Google Font PT Sans Narrow* verwendet. Die *PT Sans Narrow* verbindet das traditionelle und konservative Design der *Sans Serif* mit modernen Aspekten. Zusätzlich besitzt sie den Vorteil der verbesserten Lesbarkeit, was insbesondere auf einer kleinen Bildschirmoberfläche bedeutungsvoll ist [Typ13]. Die nachstehende Abbildung 4.21 zeigt einen Beispielausschnitt, in der der Schrifttyp sichtbar ist. Für die Kopfzeile wird die Schrift in der Variante *bold/16px* verwendet und alle Buchstaben werden zu Großbuchstaben transferiert (siehe Markierung 1). Die Überschrift (siehe Markierung 2) besitzt ebenso die Variante *bold/16px*, jedoch werden die Buchstaben nicht zu Großbuchstaben transferiert. Letztendlich wird für die Überschrift der zweiten Ebene (siehe Markierung 3) die Schriftgröße auf *12px* minimiert.

Abbildung 4.21: PT Sans Narrow

4.5.2 Farbschema

Der Einsatz geeigneter Farben ist sehr wichtig, da Farben einen Wiedererkennungswert haben und als Informationsträger dienen können. In der zu entwickelnden mobilen

Anwendung sollen zwei Grundfarben eingesetzt werden. Aufgrund des bereits ausgearbeiteten Projektlogos (siehe Abbildung 4.22), wurde die Farbe blau gewählt. Diese Farbe entspricht demselben Farbton, der im Projektlogo verwendet wird.

Abbildung 4.22: ProCollab Logo

Zusätzlich wurde zu diesem Farbton noch eine weitere Variante in dunklerer Abstufung verwendet, wodurch ein Hell-Dunkel Kontrast entsteht. Der Hell-Dunkel Kontrast dient zur Unterscheidung von bedienbaren Oberflächenelementen und nicht bedienbaren Oberflächenelementen (siehe Abbildung 4.23). Wird das bedienbare Oberflächenelement aktiviert, wird es für die Zeitspanne der Aktivierung mit der Farbe #ededed (Hex) markiert. Des Weiteren wird ein türkis-grün verwendet, um Akzente zu setzen. Um das Farbschema einheitlich zu gestalten, wird das türkis-grün nur für Icons und Buttons verwendet. Letztendlich werden in der mobilen Anwendung einerseits die Blautöne #068ec8 (Hex) und #009fe3 (Hex) verwendet und andererseits das Türkis-grün #5bc994 (Hex).

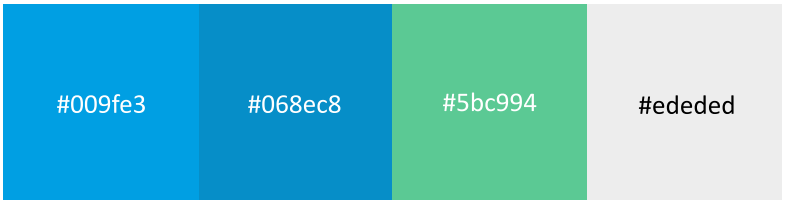


Abbildung 4.23: Farbschema

4.5.3 Icons

Icons sind kleine Symbole, die ein wichtiger Bestandteil der Programmoberfläche sind. Sie sind Repräsentanten einer Funktion der Anwendung, welche aktiviert werden können. Die Icons ersetzen eine Beschriftung und reduzieren dadurch die benötigte Bildfläche.

Dieser Aspekt ist im Bereich der Entwicklung einer mobilen Anwendung relevant, da auf den kleinen Bildschirmen wenig Platz für viele Informationen vorhanden ist. Ein weiterer Vorteil ist, dass Nutzer die Bedeutung eines Icons mit einem Blick erfassen, was die Produktivität erhöht [Hor94]. Im Laufe der Zeit haben sich bestimmte Symbole für Funktionen durchgesetzt, die dem Anwender Orientierungshilfe bieten. Daher werden vorwiegend etablierte Icons verwendet, welche dem Anwender intuitiv die hinterlegte Funktionalität suggerieren [Hor94]. Insgesamt sind die eindimensionalen Icons im Design minimalistisch gestaltet. Abbildung 4.24 zeigt einige Entwürfe verschiedener Icons.

Abbildung 4.24: Entwürfe einiger Icons

Im Anschluss an die Entwicklung der Entwürfe jeglicher Icons, wurden aufgrund von Kriterien wie Einheitlichkeit, Konsistenz und Funktionalität, die in Abbildung 4.25 visualisierten Icons gewählt. Das erste Icon bildet einen Nutzer ab (siehe Markierung 1). Dieses Icon wird einerseits in den Textfeldern der Nutzerdaten verwendet und andererseits im Hauptmenü, um auf die Daten des eigenen Accounts zuzugreifen. Markierung 2 stellt das Icon für die Organisation dar, welches in den entsprechenden Textfeldern zum Einsatz kommt. Da eine Organisation aus mehreren Mitarbeitern besteht, welche zusammenarbeiten, zeigt das Icon drei Nutzer, die visuell miteinander verbunden sind. Anschließend zeigt die Abbildung einen Schlüssel (siehe Markierung 3), der in allen Textfeldern, in denen das Passwort erfragt wird, angezeigt wird. Im alltäglichen Gebrauch dient der Schlüssel als Werkzeug, um Schlösser zu öffnen. Somit gewährleistet der Schlüssel die Sicherheit vor einer Fremdöffnung des Schlosses. Aufgrund dessen wird das Schlüsselicon als Passwortsymbol verwendet, da ein Passwort ebenso als Sicher-

heit, vor der Fremdnutzung eines System schützt. Das nächste Icon (siehe Markierung 4) bildet einen Briefumschlag ab, was als gängiges Symbol für die E-Mail-Adresse eingesetzt wird. Markierung 5 visualisiert eine Wolke, die als Icon für den organisatorischen Rahmen fungieren soll. Der organisatorische Rahmen kann untergeordnete organisatorische Rahmen, Checklisten und Einträge enthalten, weshalb er die übergeordnete Instanz verkörpert. Die Wolke symbolisiert die Zusammenfassung des Inhalts und der untergeordneten Elemente. Des Weiteren wurde ein Checklistenicon (siehe Markierung 6) entwickelt, welches eine Liste von Einträgen zeigt. Um die Konsistenz der Icons zu gewährleisten, entspricht das nachfolgende Icon (siehe Markierung 7) nahezu dem des Checklistenicons. Der einzige Unterschied besteht darin, dass mit einem Pfeil, auf einen Eintrag gezeigt wird, weshalb dieses Icon einen Checklisteneintrag symbolisiert. Das letzte grünfarbige Icon stellt eine Lupe (siehe Markierung 8) dar, was die Suchfunktion symbolisiert. Die drei weiß gefärbten Icons werden nur in der Kopfzeile verwendet. Der Pfeil (siehe Markierung 9) steht für den Zurück Button, das Haus (siehe Markierung 10) symbolisiert das Hauptmenü und Markierung 11 dient als Button für die Abmeldung.

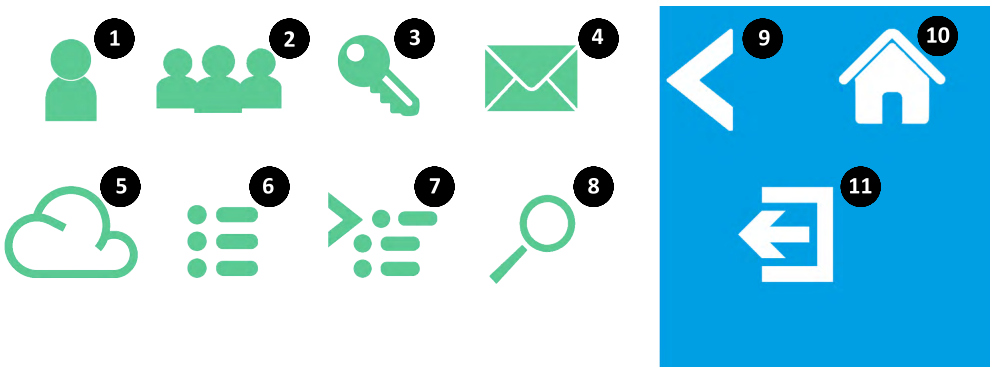


Abbildung 4.25: Icons der proCollab Smartphone-Anwendung

5

Implementierung

Die Aufgabe der Implementierung besteht darin, den Entwurf auf Basis der erhobenen Anforderungen bestmöglich umzusetzen. Hierfür ist die Auswahl geeigneter Implementierungstechnologien und Frameworks ein wesentlicher Aspekt [Bal09]. Im Kapitel 5.1 werden drei verschiedene Implementierungstechnologien vorgestellt und abgewägt, um eine passende Auswahl zu treffen. Zudem werden die verwendeten Frameworks *PhoneGap*, in Kapitel 5.2 und *JQuery Mobile*, in Kapitel 5.3 beschrieben. Schließlich werden in Kapitel 5.4 die vorher beschriebenen Technologien, anhand von einigen Aspekten der Implementierung aufgezeigt. Einen Überblick über alle Kapitel bietet Abbildung 5.1.

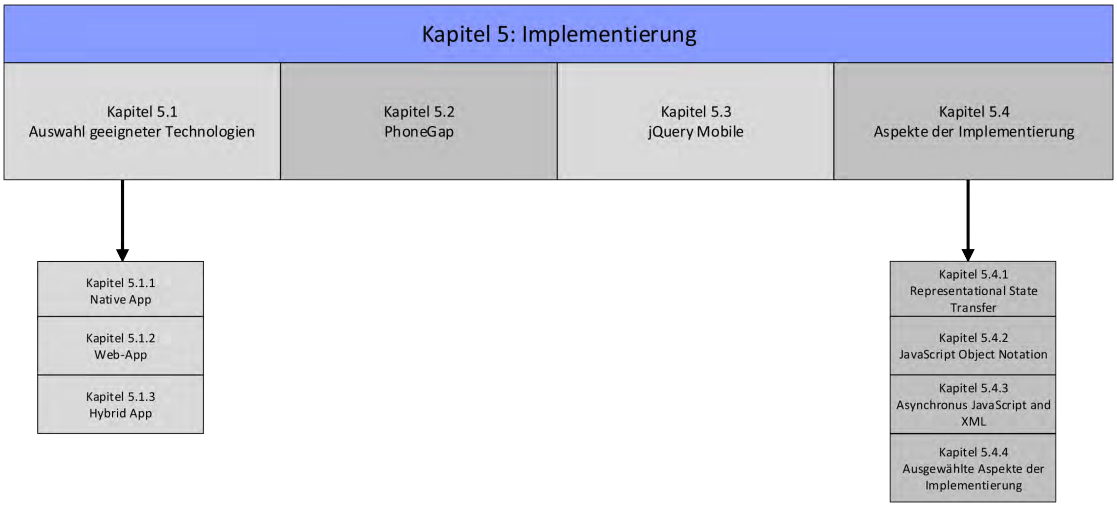


Abbildung 5.1: Aufbau des Kapitels Implementierung

5.1 Auswahl geeigneter Technologien

Zu Beginn der Implementierung ist es notwendig sich für eine bestimmte Entwicklungsvariante zu entscheiden. Grundsätzlich kann man bei der Entwicklung einer mobilen Anwendung drei verschiedene Technologieansätze unterscheiden: die *native App*, die *Web-App* und die *Hybrid App*. Wie üblich gibt es bei jedem dieser drei Ansätze Vor- und Nachteile, welche im Folgenden abgewägt werden. Ziel ist es, durch eine Gegenüberstellung der positiven und negativen Aspekte passende Technologien für die Entwicklung der mobilen Anwendung zu finden.

5.1.1 Native App

Eine native Applikation symbolisiert eine individuell für ein mobiles Gerät entwickelte Applikation, beispielsweise eine Applikation rein fürs Android-Gerät, welche mit JAVA programmiert werden kann [And13]. Eine native App bietet den Vorteil, dass sie auf viele Hardwarefunktionen zugreifen können. Weiterhin profitieren grafiklastige Anwendungen von dieser Technologie, da eine Applikation die für einen Plattformtyp programmiert wurde, durch den direkten Zugriff auf Betriebssystemfunktionen einen Performancevorteil

bietet. Die native App bringt jedoch nicht nur positive Aspekte mit sich. Der Entwicklungsaufwand einer solchen App ist sehr hoch im Bezug darauf, dass diese später nur auf einer Zielplattform funktionieren wird [HS11].

5.1.2 Web-App

Im Gegensatz zu einer nativen Anwendung müssen Webanwendungen keine Genehmigungsprozedur durchlaufen. Sie können einfach den Code veröffentlichen und die Anwendung ist weltweit auf jedem Gerät über einen Browser verfügbar. Damit ist die Webanwendung nicht an eine bestimmte Plattform gebunden. Sie wird üblicherweise mit den Webtechnologien HTML, CSS und JavaScript entwickelt und kann im Browser ausgeführt werden. Mit Hilfe dieser Webtechnologien können die Entwickler einfacher, schneller und sogar günstiger eine Anwendung implementieren. Ein Nachteil gegenüber der nativen App ist, dass Web-Apps keinen Zugriff auf verschiedene Hardwarefunktionen haben und diese deshalb nicht nutzen können. Beispielsweise der Zugriff auf die Kamera eines Mobiltelefons ist mit einer Web-App nicht möglich. Des Weiteren besitzen Web-Apps keine Offline-Funktionalität, was bei einem Abbruch der Internetverbindung die Nutzung der Applikation meist verhindert [FI10].

5.1.3 Hybrid App

Ein besonderer Typ einer Applikation ist die sogenannte *Hybrid App*. Dieser Typ basiert ebenfalls auf den erwähnten Webtechnologien und muss nicht für jedes Betriebssystem gesondert entwickelt werden (siehe Kapitel 5.1.2). Die Anwendung wird wie eine Web-App entwickelt, jedoch läuft sie in einer Umgebung, die speziell für die Plattform angepasst wurde. Die Hybrid App ist damit eine Verbindung einer Web-App und einer nativen App. Die Entwicklung mit HTML, CSS und JavaScript gewährleistet die Vorteile einer Web-App, die im vorherigen Abschnitt erläutert wurden. Zusätzlich kann eine Hybrid App auf mehr Hardwarefunktionen zugreifen als eine Web-App und sich wie eine native App verhalten [HS11]. Jedoch benötigt die Hybrid App noch eine zusätzliche Schicht zwischen der Implementierung mit HTML, CSS und JavaScript und dem Betriebs-

system, um auf die Hardwarefunktionen zuzugreifen. Ferner kann es zu Problemen mit Web-Standards kommen. Der Grund für diese Probleme ist, dass die Applikationen meist in einem Container gekapselt sind, der ein abgewandter, plattformoptimierter Browser ist.

Trotz dieser negativen Aspekten, soll die zu implementierende Anwendung als Hybrid App umgesetzt werden. Durch die Entwicklung mit den Webtechnologien kann die mobile Anwendung schnell und effizient umgesetzt werden. Des Weiteren soll dem späteren Nutzer ein möglichst natives Verhalten der Applikation geboten werden. Zusätzlich wird die konzipierte Architektur für die mobile Anwendung (siehe Kapitel 4.1.1) durch diesen Technologieansatz unterstützt. Die *View*-Komponente kann somit mit den Webtechnologien HTML und CSS realisiert werden. Darüber hinaus können die Komponenten *Controller* und *Core*, die für die Steuerung und Manipulation von Daten zuständig sind, mit der Programmiersprache JavaScript implementiert werden. Schon in Kapitel 3.5 wurde aufgrund der Vielfalt der späteren Betriebssysteme und der mobilen Endgeräte zu einer plattformunabhängigen Entwicklung tendiert. Unterstützt wird die Umsetzung durch das Framework *PhoneGap*, welches im Anschluss beschrieben wird.

5.2 PhoneGap

PhoneGap ist ein Open-Source-Framework, das die Entwicklung plattformübergreifender Applikationen unterstützt [FI10]. 2011 wurde dieses von Adobe aufgekauft, weshalb es nun unter dem Namen *Apache Cordova* bekannt ist. *PhoneGap* kombiniert die beiden technischen Ansätze der Web-App (siehe Kapitel 5.1.2) und der nativen App (siehe Kapitel 5.1.1) und nutzt somit die Vorteile beider Entwicklungsmöglichkeiten [FI10]. Anwender können mit Hilfe der Webtechnologien HTML5, CSS und JavaScript mobile Anwendungen erstellen. Mit Hilfe dieses Frameworks können Web-Apps innerhalb einer nach außen hin nativen Anwendung publiziert werden. Darüber hinaus kann man auf die verschiedenen Hardwarekomponenten eines Smartphones verbessert zugreifen. Dies ist ein wesentlicher Vorteil gegenüber einer herkömmlichen Web-App. Die Architektur des Frameworks wird in Abbildung 5.2 visualisiert. Beginnend mit dem *WebApp Code*,

der auf die *Hardware API* aufsetzt, wird die Applikation unter Verwendung der *Software SDKs* zu einer native App transformiert.

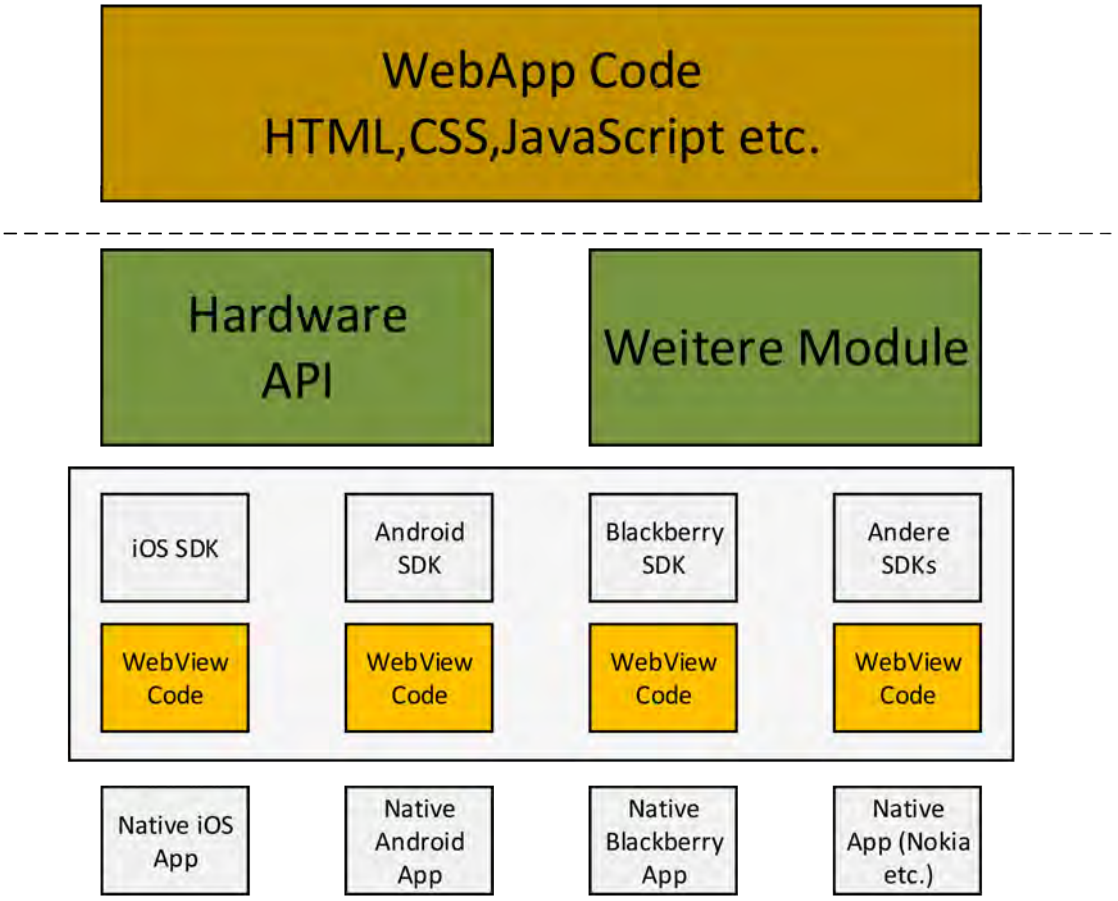


Abbildung 5.2: Das PhoneGap-Modell [HS11]

Außer dem Framework *PhoneGap* gibt es noch zwei weitere Ansätze eine Hybrid App zu entwickeln. Eine weitere Möglichkeit wäre die Anwendung mit *Xamarin* zu entwickeln [Xam13]. Hier programmieren die Entwickler größtenteils mit C#. Das User-Interface wird jedoch für jede Plattform gesondert entwickelt, um das Look and Feel einer nativen App zu gewährleisten. Dadurch entsteht ein beachtlicher Mehraufwand, da sich ein Entwickler mit der jeweiligen Programmiersprache auseinandersetzen muss. Dies war der Grund für die Entscheidung das Framework *Xamarin* nicht zu nutzen.

Der zweite Ansatz wäre die Anwendung mit dem Framework *Titanium* zu entwickeln. Dieses Framework deckt jedoch weniger Plattformen ab und besitzt einen geringeren Reifegrad. Der Hauptgrund für die Entscheidung für *PhoneGap* bestand also darin, dass die wichtigsten mobilen Plattformen abgedeckt sind. Des Weiteren wird mit *PhoneGap* eine mobile Seite entwickelt und in eine App transformiert, was bei anderen Frameworks nicht der Fall ist. Zusätzlich erleichtert ein einfacher Einstieg und der hohe Reifegrad das Arbeiten mit *PhoneGap* stark [HS11]. Außer dem Framework *PhoneGap* wurde noch ein weiteres Interface-Framework verwendet, welches im Folgenden beschrieben wird.

5.3 jQuery Mobile

JQuery Mobile erschien im Jahr 2011 das erste Mal und dient als Unterstützung bei der Implementierung von mobilen Anwendungen. Das Interface-Framework ist die Weiterentwicklung der Bibliothek jQuery. Mit Hilfe von diesem, auf mobile Endgeräte optimiertem, Framework werden die Entwickler einer Web-App, vor allem bei der Implementierung der Bedienoberfläche unterstützt. Die Gestaltung bestimmter Dialogelemente ist bereits für sämtliche Smartphones und ihre Bildschirmgrößen definiert und optimiert [FI10]. *JQuery Mobile* basiert auf den Web-Standards HTML, CSS und JavaScript. Im Gegensatz zum User-Interface-Framework *Sencha Touch* setzt *jQuery Mobile* auf einen deklarativen Ansatz: „*D. h. die Programmierung erfolgt innerhalb des HTML-Markups, wodurch entsprechende JavaScript-Methoden angesteuert werden*“ [Jas11]. Im Bezug auf die Architektur der mobilen Anwendung dient *jQuery Mobile* also zur optimierten Umsetzung der *View*-Komponente und deren Dialogelemente, sowie der *Controller*- und *Core*-Komponente, mit Hilfe von *jQuery Mobile Events und Methods* (siehe Kapitel 4.1.1).

5.4 Aspekte der Implementierung

Auf Basis des konzipierten Datenmodells (siehe Kapitel 4.2) und der Architektur des Projektes (siehe Kapitel 4.1), wurde die Anwendung mit Hilfe von einigen Program-

mieraspekten entwickelt. Es werden die angewandten Programmieraspekte *Representational State Transfer*, *JavaScript Object Notation* und *Asynchronous JavaScript and XML* beschrieben. Des Weiteren wird anhand von ausgewählten Programmieraspekten aufgezeigt, wie diese in der Umsetzung zum Einsatz kommen. Darüber hinaus wird in diesem Kapitel anhand der Programmieraspekte aufgezeigt, wie einige Mockups aus Kapitel 4.4 implementiert wurden.

5.4.1 Representational State Transfer

Representational State Transfer (REST) ist ein Architekturstil, der für Webanwendungen konzipiert wurde. Durch die Verwendung von *Hypertext Transfer Protocol* (HTTP) kann auf verschiedene Dienste und Ressourcen des Servers zugegriffen werden. Das Anbieten von verschiedenen Diensten im Internet entspricht dem Architekturmuster *service-oriented architecture*. Für das Projekt proCollab (siehe Kapitel 2.2) und die Anbindung der mobilen Applikation werden serverseitig verschiedene REST-Schnittstellen zur Verfügung gestellt. Aufbauend auf der Architektur in Kapitel 4.1.1 wird somit die Kommunikation mit dem Server realisiert. Insgesamt muss REST die folgenden fünf verschiedene Eigenschaften beinhalten [Fie00].

Adressierbarkeit
Durch den *Uniform Resource Locator* (URL) muss jeder REST-Dienst eindeutig adressierbar sein.

Unterschiedliche Repräsentation
Nach einer Anfrage des Clients ist es möglich, dass der Server verschiedene Repräsentationen der Ressource ausliefert (z.B. JSON, siehe Kapitel 5.4.2). Der *Uniform Resource Locator* (URL) repräsentiert diese Ressource.

Zustandslosigkeit
REST setzt auf einem Client-Server Protokoll, üblicherweise *HTTP* oder *HTTPS* auf. Sendet der Client eine Anfrage an den Server mittels *HTTP*, enthält dieser nur Daten über die der Server informiert werden muss. Zwischen zwei Anfragen werden somit keine Zustandsinformationen gespeichert.

Operationen

Es ist wichtig, dass mehrere Operationen zur Verfügung stehen, die auf die Ressourcen angewandt werden können. *HTTP* stellt eine Menge von Operationen bereit, u.a. *GET*, *POST*, *PUT* und *DELETE*. Die *GET*-Operation ist verpflichtend und beschreibt die Anforderung der Informationen. *POST* stellt eine Anfrage dar, die für die Realisierung von Sub-Ressourcen zuständig ist. Um eine Ressource anzulegen oder zu ändern sollte man die *PUT* Operation verwenden. Letztendlich wird mit der *DELETE* Operation eine Ressource auf dem Server gelöscht. Laut *HTTP* Spezifikation müssen alle Operationen, außer *GET*, idempotent sein.

Verwendung von Hypermedia

Unter der Verwendung von Hypermedia versteht man die Darstellung von den erlangten Informationen. Dies wird meist durch die Verwendung der Markup-Sprache HTML erreicht [Fie00].

5.4.2 JavaScript Object Notation

JavaScript Object Notation (JSON) ist ein Datenformat, das für die Maschine und den Menschen leicht lesbar ist und zur Serialisierung von strukturierten Daten dient [Cro06]. JSON ist sehr einfach aufgebaut und kann vier verschiedene primitive Datentypen und zwei strukturierte Datentypen darstellen: *strings*, *numbers*, *booleans*, *null*, *objects* und *arrays* [Cro06]. Die strukturierten Datentypen ermöglichen es, dass eine JSON-Datei eine variable Tiefe besitzt und somit beliebig oft hierarchisch verschachtelt werden kann. Üblicherweise wird JSON im Zusammenhang mit *Asynchronous JavaScript and XML* (siehe Kapitel 5.4.3) verwendet, um Daten zwischen Client und Server auszutauschen. Im Anschluss wird die Verwendung von JSON in der mobilen Anwendung erläutert.

Verwendung von JSON

JSON wird in diesem Kontext eingesetzt, um den Datenaustausch zwischen Server und Client, insbesondere zwischen Delivery Layer und Application Layer, zu realisieren (siehe Kapitel 4.1). Die auszutauschenden Daten entsprechen dem Datenformat JSON. Das Listing 5.1 zeigt eine kurze und einfache JSON-Datei der Nutzerverwaltung. In dieser

Datei sind alle relevanten Informationen zu einem Nutzer gespeichert. In Zeile 2 beginnt die Auflistung der Attribute mit der *id*, welche als eindeutige Identifizierung unter allen registrierten Nutzern dient. Weiterhin enthält die JSON-Datei Informationen, wie E-Mail, Vorname, Nachname, Organisation und die *Uniform Resource Locator* der Organisation.

```
1  {
2    "id":123,
3    "email": "sabrina@mail.de",
4    "prename": "Sabrina",
5    "surname": "Geiger",
6    "organization": {
7      "name": "organisation",
8      "url": "http://www.organisation.de"
9    }
10 }
```

Listing 5.1: JSON-Datei aus der Nutzerverwaltung

Eine komplexere JSON-Datei wird in Listing 5.2 dargestellt. In diesem Beispiel handelt es sich um eine JSON-Datei, die die Daten einer organisatorischen Rahmeninstanz enthält. Dies ist erkennbar an dem Attribut *"class":"frame"* der Zeile 3. Dieses Attribut macht es auf dem Client einfacher die Daten zu unterscheiden und erhöht die Effizienz der implementierten Methoden, die dieses Attribut benötigen. In Zeile 12 beginnt die Verschachtelung der JSON-Datei mit der Angabe der involvierten Nutzer (*"involvedUsers":[]*). Dieses Array besitzt Tupel, die beschreiben, welchen Personen es erlaubt ist mit der jeweiligen organisatorischen Rahmeninstanz zu arbeiten. Zusätzlich besitzt die JSON-Datei ab Zeile 18 weitere Kindknoten (*"containedLists":[]*), die die verschiedenen untergeordneten Checklisteninstanzen und Eintragsinstanzen darstellen. Die Subinstanzen wiederum können nun weitere Kindknoten (*"containedLists":[]*) enthalten, wodurch die variable Tiefe der Subinstanzen einer JSON-Datei gewährleistet ist.

```
1  {
2          "id": 1,
3          "class": "frame",
```

```
4           "name": "First Frame",
5           "description": "The first Frame we created",
6           "state": "initialized",
7           "goal": "Test the application",
8           "duration": 42,
9           "creationDate": "Sat Jan 26 06:24:07 GMT 2013",
10          "startDate": "Sat Jan 28 06:38:00 GMT 20131",
11          "endDate": "Sat Jan 30 09:24:07 GMT 2013",
12          "involvedUsers": [
13              {
14                  "email":"sabrina.geiger@uni.de",
15                  "role": "manager"
16              },
17          ],
18          "containedLists": [
19              {
20                  "class": "list",
21                  "id": 3,
22                  "name": "List3 Name",
23                  "instance": true,
24                  "description": "An other cool list",
25                  "state": "initialized",
26                  "children": []
27               }
28      ]
29  }
```

Listing 5.2: JSON-Datei aus der organisatorischen Rahmenverwaltung

5.4.3 Asynchronous JavaScript and XML

Asynchronous JavaScript and XML (AJAX) ist eine Variante der Hybriden Programmierung und dient zur dynamischen Datenübertragung zwischen Client und Server [KKW07]. Durch die Verwendung von AJAX müssen die Inhalte einer Website nicht jedesmal komplett neu geladen werden, sondern sie werden einfach ausgetauscht (siehe Abbildung 5.3). Dies hat die Vorteile, dass die Anwendung schnell reagiert und dass der

Server und das Netzwerk entlastet werden [KKW07]. Im Kontext der zu entwickelnden Anwendung wurde AJAX verwendet, um den Datenaustausch zwischen Server und Client zu realisieren. Mittels AJAX wird eine asynchrone HTTP Anfrage ausgeführt, die serverseitig bearbeitet und beantwortet wird. Im Folgenden wird die Verwendung von AJAX in der zu entwickelnden mobilen Anwendung beschrieben.

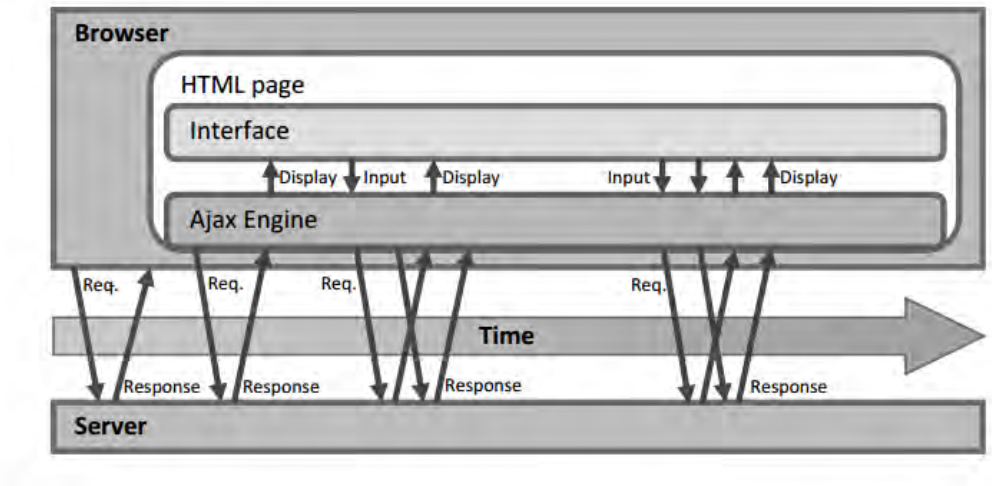


Abbildung 5.3: Interaktionsmuster bei AJAX [KKW07]

Verwendung von AJAX

AJAX ist in der mobilen Anwendung dafür Zuständig, die JSON-Daten zwischen dem Application Layer und dem Delivery Layer zu übertragen (siehe Kapitel 4.1.1). Das Listing 5.3 zeigt eine JavaScript Funktion, die eine asynchrone HTTP (AJAX) Anfrage ausführt. Insbesondere wird eine Anfrage über HTTP-GET ausgeführt, was in Zeile 5 (*type: 'GET'*) festgelegt ist. Der Wert in Zeile 3 (*url: serverURL+interface*) legt fest, welche JSON-Datei gefordert wird bzw. auf welches REST-Interface zugegriffen wird (siehe Kapitel 5.4.1). Weiterhin wird mit der Angabe *dataType: 'json'* in Zeile 6 vorgeschrieben, was für einen Datentyp der Client vom Server als Antwort erwartet, was in diesem Fall eine JSON-Datei ist. Die Callback-Funktionen in Zeile 7 (*success : function(result)*) und Zeile 11 (*error: function(jqXHR, textStatus, errorThrown)*) sind zuständig für die Bearbeitung der Anfrage. Die Funktion *success* wird bei einer erfolgreichen

Anfrage aufgerufen, wohingegen die Funktion *error* nur dann aufgerufen wird, wenn die Anfrage fehl schlägt. Somit können Fehlschläge einfach behandelt werden, sowie, in der entsprechenden Funktion, die HTTP Response Codes ausgewertet werden können. Damit eine Anfrage erfolgreich durchgeführt wird, müssen verschiedene Interfaces (*url: serverURL+interface*) auf Seiten des Servers angeboten werden.

```
1  function get(interface, handler){
2    $.ajax({
3      url: serverURL+interface,
4      type: 'GET',
5      dataType: 'json',
6      success : function(result)
7      {
8        handler(result);
9      },
10     error: function(jqXHR, textStatus, errorThrown)
11     {
12     }
13   });
14 };
```

Listing 5.3: Beispiel einer Anfrage per AJAX

5.4.4 Ausgewählte Aspekte der Implementierung

Auf Basis der eingeführten Technologien und des konzipierten Datenmodells (siehe Kapitel 4.2) werden im Folgenden einige ausgewählte Aspekte der Implementierung beschrieben. Beginnend mit der Bestimmung der Subtypen von organisatorischen Rahmentypen werden Screenshots und Implementierungsausschnitte vorgestellt. Zudem wird die Umsetzung der Breadcrumbnavigation erläutert.

Bestimmung der Subtypen von organisatorischen Rahmentypen

Bei der Instanziierung von organisatorischen Rahmentypen wird dem Nutzer eine Übersicht über alle Subtypen ermöglicht. Die erste Seite, nach Auswahl eines bestimmten organisatorischen Rahmentyps zeigt die Subtypen, die wiederum Subtypen besitzen können. In der nachstehenden Grafik 5.4 ist ein Screenshot der Übersicht über die Subtypen abgebildet. Dieser Screenshot zeigt die Umsetzung der Mockups aus Kapitel 4.4.4.

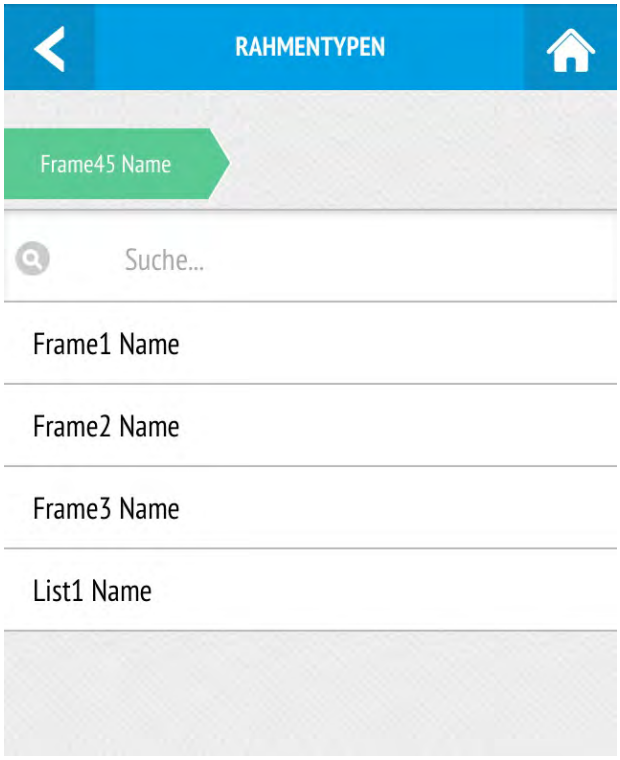


Abbildung 5.4: Screenshot der Übersicht über die Subtypen

Listing 5.4 zeigt einen Implementierungsausschnitt, der zeigt wie die Subtypen des gewählten organisatorischen Rahmentyps über die REST-Schnittstelle angefordert werden und danach an eine Liste angehängt werden. In Zeile 1 wird die Funktion *getFrameType* mit der *id* des angeklickten organisatorischen Rahmentyps aufgerufen. Danach werden in Zeile 2, mit einer for-Schleife alle Subtypen durchlaufen und jeweils ab Zeile 3 in einem *<a>*-Tag verankert mit dem *class*-Attribut *frametype*. Des Weiteren wird in Zeile 4 über

die *data()*-Methode das jeweilige *<a>*-Tag gespeichert und in Zeile 6 an ein Listenelement gehängt (*$cont.append($el);*). Insbesondere werden in Zeile 7 alle Listenelemente noch an die komplette Liste mit der *id = "#content_frame"* gehängt. Zusätzlich ist es notwendig, bei einer Listenoperation eine Aktualisierung durchzuführen, um die neu hinzugefügten Listenelemente auf dem Bildschirm sichtbar zu machen, was in Zeile 9 über die *refresh*-Methode ausgeführt wird.

```
1  rest.getFrameType(id,function(frames){
2    for (var i = 0, end = frames.children.length; i < end; i++){
3      $el = $('<a class="frametype">'+ frames.children[i].name +'</a>');
4      $el.data("data",frames.children[i]);
5      $cont = $('<li id="listElementFrameType'+i+'" data-ajax="false"></li>');
6      $cont.append($el);
7      $('#content_frame').append($cont);
8    };
9    $('#content_frame').listview('refresh');
10 });
```

Listing 5.4: Subtypen eines organisatorischen Rahmentyps

Der Nutzer muss außerdem die Möglichkeit besitzen, über einen Klick auf die momentan angezeigten Subtypen, die Subtypen des angeklickten Listenelements anzusehen. Dies wird über die Methode *getChildren(element,listviewUl,classAttr,idAttr)* realisiert (siehe Listing 5.6). Diese Methode wird aufgerufen, wenn ein Listenelement mit dem *class*-Attribut *frametype* angeklickt wurde (siehe Listing 5.5).

```
1  $(document).on("click",".frametype", function(){
2    var element= $(this).data("data");
3    getChildren(element,$('#frame'),"frametype","listElementFrameType");
4  });
```

Listing 5.5: Methode nach Anklicken eines Subtypen

Um das Verständnis der Methode *getChildren()* (siehe Listing 5.6) zu erhöhen, wird diese im Folgenden beschrieben. Zu Beginn wird in Zeile 4 geprüft, ob das angeklickte Listenelement noch weitere Subtypen besitzt. Ist dies nicht der Fall, wird ein Listenelement

mit der Beschriftung *Keine Kinder mehr vorhanden* an die Liste gehängt (siehe Zeile 5). Besitzt der Subtyp weitere Kindknoten, werden diese anhand einer for-Schleife in Zeile 7 durchlaufen. Alle Kindknoten werden nun in einem *<a>*-Tag verankert, das wiederum das *class*-Attribut *frametype* besitzt. Grund für dieselbe Benennung des *class*-Attributs ist, dass bei Anklicken eines Subtyps immer dieselbe Methode in Listing 5.5 aufgerufen wird. Jedoch ändern sich hierbei die Übergabeparameter, da der angeklickte Subtyp in der Variable *element* gespeichert wird und an die Methode *getChildren()* übergeben wird. Ein weiterer wichtiger Aspekt wird in Zeile 11 gezeigt, in welcher die Überschreibung des *data*-Attributs mit der *data()*-Methode implementiert ist. Hierbei wird das aktuelle Kindelement gespeichert und das alte überschrieben, wodurch der rekursive Aspekt gewährleistet wird. Um die Navigationswege in dem organisatorischen Rahmentyp zu verfolgen und bei Bedarf wiederherzustellen, wurde eine Breadcrumbnavigation konzipiert, die im Folgenden beschrieben wird.

```
1  function getChildren(element,listviewUl,classAttr,idAttr){
2    if(element != undefined){
3      listviewUl.empty();
4      if(element.children.length==0){
5        $('<li>Keine Kinder mehr vorhanden</li>').appendTo(listviewUl);
6      }
7      for (var i = 0, end = element.children.length; i < end; i++){
8        $el =
9        $('<a class="'+ classAttr +'" data-ajax="false">'+
10       element.children[i].name +'</a>');
11       $el.data("data",element.children[i]);
12       $cont =
13       $('<li id="'+idAttr +i+'" data-ajax="false"></li>');
14       $cont.append($el);
15       listviewUl.append($cont);
16     };
17     listviewUl.listview('refresh');
18   }
19 }
```

Listing 5.6: Methode zur Bestimmung der Subtypen von aktuellen Subtypen

Breadcrumbnavigation

Eine Breadcrumbnavigation ermöglicht dem Nutzer, eine Übersicht über seine aktuelle Position sowie über seine vorherig gewählten Navigationswege. Des Weiteren ist es jederzeit möglich, einen alten Navigationszustand wieder aufzurufen oder in den ursprünglichen Zustand zu navigieren. Die Abbildung 5.5 zeigt die implementierte Breadcrumbnavigation. Die Abbildung der Breadcrumbnavigation ist die Umsetzung der Mockups aus Kapitel 4.4.6.

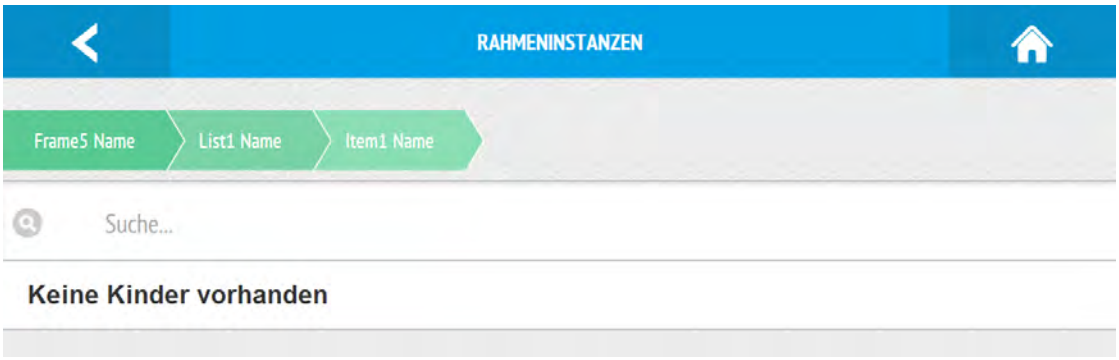


Abbildung 5.5: Screenshot der Breadcrumbnavigation

Für die zu entwickelnde mobile Applikation wurde die Breadcrumbnavigation mit Hilfe des Implementierungsausschnittes in Listing 5.7 realisiert. Beginnend mit Zeile 1 bis 3 wird der aktuelle organisatorische Rahmenname mit dem *class*-Attribut *breadCrumbNav* an ein * *-Tag mit der *id = breadCrumb* gehängt. Somit wird der Name des organisatorischen Rahmens auf der weißen * *-Tag Oberfläche angezeigt und bei weiteren Navigationswegen mit den folgenden Namen erweitert. Zusätzlich werden diese Daten über die *data()*-Methode in das Attribut *"breadCrumbData"* gespeichert (siehe Zeile 4). Möchte der Nutzer nun einen alten Navigationszustand wiederherstellen, indem er auf den entsprechenden organisatorischen Rahmennamen klickt, wird die Methode in Zeile 6 aufgerufen. Hier wird in der Zeile 7 bis 8 das aktuell angewählte Element ermittelt und in die Variable *clickedbreadCrumbNavElement* gespeichert. Daraufhin müssen die nachfolgenden Elemente der Breadcrumbnavigation wieder entfernt werden, um die Lokation des Nutzers korrekt anzuzeigen. Aufgrund dessen wird in Zeile 9 das Element in

der Variable *nextElements* gespeichert, woraufhin in Zeile 10 über die Methode *nextAll()* alle nachfolgenden Elemente angesprochen werden und letztendlich mit der Methode *remove()* entfernt werden. Der Inhalt des angewählten organisatorischen Rahmennamens wird nun durch die Methode *getChildren()* wiederhergestellt (siehe Abschnitt 5.4.4).

```
1  $breadCrumbRootEl =
2  $('<a class="breadCrumbNav" data-ajax="false">'+frames.name+
3  '</a>').appendTo("#breadCrumb");
4  $breadCrumbRootEl.data("breadCrumbData",frames);
5
6  $(document).on("click",".breadCrumbNav", function(){
7    var clickedbreadCrumbNavElement =
8    $(this).data("breadCrumbData");
9    var nextElements = $(this);
10   nextElements.nextAll().remove();
11   getChildren(clickedbreadCrumbNavElement,
12   $('#content_frame'),"frametype","listElementFrameType");
13 });
```

Listing 5.7: Breadcrumbnavigation

6

Zusammenfassung

Abschließend erfolgt ein Überblick über die vorgestellten Inhalte der Arbeit in Kapitel 6.1, ein zukunftsorientierter Ausblick über noch mögliche Erweiterungen oder Ausbaumöglichkeiten der mobilen Anwendung in Kapitel 6.2 und ein Schlusswort in Kapitel 6.3. Einen Überblick über alle Kapitel bietet die Abbildung 6.1.

Abbildung 6.1: Aufbau des Kapitels Zusammenfassung

6.1 Zusammenfassung

Im Rahmen dieser Bachelorarbeit wurde die Konzeption und Entwicklung einer auf Smartphones optimierten mobilen Anwendung für kollaboratives Checklisten-Management beschrieben. Beginnend in Kapitel 2 mit den Einführungen in die Thematik und den Grundlagen orientierte sich die weitere Vorgehensweise am *Referenzmodell Usability Engineering*. Die zwei folgenden Kapitel entsprechen den Phasen *Anforderungsanalyse* und *User-Interface Entwurf* dieses Referenzmodells.

Auf Basis der Grundlagen wurden in **Kapitel 3** die Anforderungen analysiert und festgelegt. Mit Hilfe der Ist-Stand Analyse, Benutzerprofilanalyse, Aufgabenanalyse, Analyse der Umgebungsbedingungen und den Hardware und Software Randbedingungen wurden die Anforderungen erarbeitet. Insbesondere die Ist-Stand Analyse bildete die Basis für erste Entwurfsideen und Umsetzungsmöglichkeiten.

Daraufhin erfolgte in **Kapitel 4** die Entwurfsphase, in welcher die technischen Aspekte der Projektarchitektur und der Datenmodellierung beschrieben wurden. Weiterhin wurden die charakterisierten Anforderungen anhand von User-Interface Entwürfen detailliert. Der Abschnitt der User-Interface Entwürfe begann mit der Entwicklung eines konzeptuellen User-Interface Modells, welches dann mit ersten Designentwürfen und Mockups erweitert wurde. Anschließend wurden im letzten Abschnitt diesen Kapitels, verschiedene Styleguideregeln definiert.

Nachdem die Analyse- und Entwurfsphase abgeschlossen waren, begann die Implementierungsphase in **Kapitel 5**. Hier wurden verschiedenen Implementierungstechnologien beschrieben, gegenübergestellt und abgewägt. Nachdem wurden die verwendeten Frameworks vorgestellt, die die eigentliche Implementierung unterstützten. Letztendlich wurden die drei Programmieraspekte REST, JSON und AJAX erläutert und demnach anhand von Programmierbeispielen näher beschrieben.

6.2 Ausblick

Während der gesamten Konzipierung und Entwicklung des Prototyps haben sich weitere Ausbaumöglichkeiten und Ideen ergeben, die aufgrund von einigen Randbedingungen im Verlauf der Arbeit nicht realisiert werden konnten. Im Folgenden werden diese Aspekte vorgestellt:

Funktionale Erweiterungen:
Werden alle Einträge und Subchecklisten aus einer Checkliste entfernt, erfüllt die Checkliste nicht die definierten Anforderungen, dass mindestens ein Eintrag oder eine Subcheckliste enthalten sein muss. Ein möglicher Lösungsansatz wäre, dass eine Checkliste in diesem Fall zu einem Eintrag gewandelt wird.

Die erstellten Instanzen werden nach der Bearbeitung als abgeschlossen gekennzeichnet. Eine sinnvolle Ausbaumöglichkeit besteht darin, Instanzen, welche sich als optimale Lösung bewährt haben, zu archivieren. Darüber hinaus sollen die Nutzer nun auf die archivierten Instanzen zurückgreifen können und diese in ihrem ursprünglichen Zustand, für weitere Anwendungsfälle verwenden können.

Derzeit besitzen die verschiedenen organisatorische Rahmeninstanzen, Checklisteninstanzen und Eintragsinstanzen keinen Nachweis aus welchen Typen sie abgeleitet worden sind, oder ob sie individuell erzeugt wurden. Eine Übersicht oder ein textueller Nachweis, der die Herkunft der Instanzen zeigt, wäre eine Erweiterung, die später noch realisiert werden sollte. Dadurch können sich die Nutzer nachträglich informieren, ob eine Instanz, die nun wieder erstellt werden soll, auf einem Typ basiert. Ist dies der Fall kann der Nutzer nach diesem Typ suchen und ihn weitere Male verwenden.

Die organisatorische Rahmeninstanzen, Checklisten und Einträge können derzeit entweder den Status *nicht bearbeitet* oder den Status *abgeschlossen* besitzen. Eine weitere Ausbaumöglichkeit wäre, diese beiden Stati weiter zu verfeinern. Denkbar wären die

Optionen *initialisiert*, *gestartet*, *in Bearbeitung* oder aber auch nach dem Abschließen ein zusätzlicher Status *archiviert*.

Momentan wurde ein vereinfachtes Management für die Nutzerrollen ausgearbeitet. Die Nutzer können entweder die Rolle eines Standardnutzers einnehmen, der nur Leserechte besitzt, oder die eines Verwalters. Diese Nutzerrollen könnten ausgearbeitet werden, indem ein spezifischeres und feingranulareres Management der Nutzerrollen erfolgt.

Möglicherweise können beim Datenaustausch hohe Datenraten anfallen, woraus eine lange Ladezeit resultiert. Diese Ladezeit soll jedoch gering gehalten werden, was durch eine Zwischenspeicherung der Daten zu erreichen wäre. Dies wäre weiterhin von Vorteil, falls es zum Abbruch der Internetverbindung käme, da die Daten erst synchronisiert werden, wenn die Verbindung wieder hergestellt wurde. Somit würde die eingeschränkte Nutzung bei fehlender Internetverbindung weiterhin gewährleistet sein.

Allgemeine Erweiterungen:
Die mobile Anwendung muss später den Nutzern die Möglichkeit einer Sprachauswahl bieten. Da die kollaborative Wissensarbeit auf internationaler Ebene vertreten ist, sollen fremdsprachige Wissensarbeiter beispielsweise die Möglichkeit haben, in der mobilen Anwendung die Sprache Englisch auszuwählen.

Da jeder Nutzer ein anderes Farbempfinden hat und jeder individuelle Vorlieben und Abneigungen bezüglich der Farbwahl besitzt, soll die mobile Anwendung dem Nutzer eine Auswahl an verschiedenen Themes zur Verfügung stellen.

6.3 Schlusswort

Die Konzeption und Entwicklung der mobilen Anwendung für das kollaborative Checklisten-Management stellt einen ersten Schritt in die Richtung der Problemlösung von kollabora-

tiven Wissensarbeiten und ihren Prozessen dar. Aufbauend auf diesem Prototyp werden weiterführende Konzipierungen und Entwicklungen des Projektes proCollab zeigen, ob hiermit die kollaborative Wissensarbeit optimal unterstützt werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[And13] *Android Guidelines*. http://developer.android.com/guide/topics/ui/index.html. Version: 2013, Abruf: 24.10.2013

[Any13a] *Beispieldialog Any.Do*. http://www.darwiszone.com/2013/01/a-must-have-note-taking-app-for-android-any.do.html. Version: 2013, Abruf: 29.07.2013

[Any13b] *Der Taskmanager Any.Do*. http://www.any.do/. Version: 2013, Abruf: 21.05.2013

[Bal09] BALZERT, Helmut: *Lehrbuch der Software-Technik*. Heidelberg : Spektrum, 2009

[Boe13] *Boeing 747-400 normal procedures checklist*. http://www.onebag.com/popups/747checklist.pdf. Version: 2013, Abruf: 07.08.2013

[Cro06] CROCKFORD, Douglas: *RFC 4627- The application/json Media Type for JavaScript Object Notation (JSON)*. http://www.ietf.org/rfc/rfc4627.txt. Version: Juli 2006, Abruf: 25.10.2013

[Doo13] *IBM Rational DOORS*. http://www-03.ibm.com/software/products/de/de/ratidoor/. Version: 2013, Abruf: 19.10.2013

[Dud13] *Der Begriff Rahmen im Duden*. http://www.duden.de/rechtschreibung/Rahmen. Version: 2013, Abruf: 29.06.2013

[DW93] DEGANI, Asaf ; WIENER, Earl: Cockpit checklists: Concepts, design, and use. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 35 (1993), June, Nr. 2, S. 345–359

[ENI06] *Norm EN ISO 9241 Teil 110: Grundslätze der Dialoggestaltung*. Berlin : Beuth, 2006

[EPK07] ENNKER, Jürgen ; PIETROWSKI, Detlef ; KLEINE, Peter: *Risikomanagement in der operativen Medizin: Mit 18 Tabellen*. Darmstadt : Steinkopff, 2007

[FI10] FRANKE, Florian ; IPPEN, Johannes: *Apps mit HTML5 und CSS3*. Bonn : Galileo Press, 2010

[Fie00] FIELDING, Thomas: *Architectural Styles and the Design of Network-based Software Architectures*. Irvine, University of California, Diss., 2000. `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`, Abruf: 26.09.2013

[Gaw11] GAWANDE, Atul: *The checklist manifesto: How to get things right*. London : Profile Books, 2011

[Har05] HARTIG, Sabine: *Evaluation der methodischen Qualität von Leitlinien der medizinischen Versorgung aus dem System der Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften in Deutschland*. Marburg, Philipps-Universität Marburg, Diss., 2005. `http://archiv.ub.uni-marburg.de/diss/z2006/0382/view.html`

[Hor94] HORTON, Wiliam: *Das ICON-Buch*. München : Addison Wesley, 1994

[HS11] HAIGES, Sven ; SPIERING, Markus: *HTML5-Apps für iPhone und Android*. Haar : Franzis, 2011

[Hub05] HUBE, Gerhard: *Beitrag zur Beschreibung und Analyse von Wissensarbeit*, Universität Stuttgart, Diss., 2005

[HWB+09] HAYNES, Alex B. ; WEISER, Thomas G. ; BERRY, William R. ; LIPSITZ, Stuart R. ; BREIZAT, Abdel-Hadi S. ; DELLINGER, E. P. ; HERBOSA, Teodoro ; JOSEPH, Sudhir ; KIBATALA, Pascience L. ; LAPITAN, Marie Carmela M. ; MERRY, Alan F. ; MOORTHY, Krishna ; REZNICK, Richard K. ; TAYLOR, Bryce ; GAWANDE, Atul A.: A Surgical Safety Checklist to Reduce Morbidity and Mortality in a Global Population. In: *New england journal of medicine* 360 (2009), January, Nr. 5, S. 491–499

[Jas11] JASER, Michael: *Evaluation, Bewertung und Implementierung verschiedener Cross-Platform Development Ansätze für Mobile Internet Devices auf Basis von Web-Technologien*. Augsburg, University of Applied Sciences, Bachelorarbeit, 2011

[KHA10] KANKI, Barbara G. ; HELMREICH, Robert L. ; ANCA, José M.: *Crew Resource Management*. San Diego : Academic Press/Elsevier, 2010

[KKW07] KLUGE, Jonas ; KARGL, Frank ; WEBER, Michael: The effects of the ajax technology on web application usability. In: *3rd International Conference on Web Information Systems and Technologies (WebIST 2007)*. Barcelona, Spain, 2007, S. 289–294

[KS09] KOWALEWSKI, Julia ; STILLER, Silvia: Strukturwandel im deutschen Verarbeitenden Gewerbe. In: *Wirtschaftsdienst* (2009), Nr. 8, S. 548–555

[Mal13a] *Beispieldialog Maler Checklisten App*. `http://a5.mzstatic.com/us/r1000/075/Purple/v4/ea/5a/67/ea5a67ed-969e-b32e-1a58-dcef8315cd62/mzl.vummucvc.320x480-75.jpg`. Version: 2013, Abruf: 29.07.2013

[Mal13b] *Die Maler Checklisten App*. `http://www.malerchecklistenapp.de/`. Version: 2013, Abruf: 21.05.2013

[Mar13] *Checkliste zur Marktanalyse*. `http://www.controlling-wiki.com/de/images/9/96/Checkliste_zur_Marktanalyse.JPG`. Version: 2013, Abruf: 07.08.2013

[MKR12] MUNDBROD, Nicolas ; KOLB, Jens ; REICHERT, Manfred: Towards a System Support of Collaborative Knowledge Work. In: *Business Process Management Workshops* Bd. 132, Springer, 2012, S. 31–42

[Mun12] MUNDBROD, Nicolas: *Business Process Support for Collaborative Knowledge Workers*. Ulm, Universität Ulm, Masterarbeit, 2012

[Mun13] MUNDBROD, Nicolas: *Interner Bericht: Nutzung von Checklisten zur Unterstützung der Entwicklung mechatronischer Systeme im Automobilbereich*. Universität Ulm, 2013

[NB12] NIELSEN, Jakob ; BUDIU, Raluca: *Mobile Usability*. Berkeley : New Riders, 2012

[Nie93] NIELSEN, Jakob: *Usability Engineering*. San Francisco : Morgan Kaufmann Publishers, 1993

[Off12] OFFERGELD, Michael: *Skript zur Vorlesung Usability Engineering*. Ulm, Universität Ulm, 2012

[OKF01] OLLENSCHLÄGER, Günter ; KIRCHNER, Hanna ; FIENE, Michael: Leitlinien in der Medizin - scheitern sie an der praktischen Umsetzung? In: *Der Internist* 42 (2001), Nr. 4, S. 473–483

[Paw98] PAWLOWSKY, Peter: *Wissensmanagement*. Wiesbaden : Gabler, 1998

[Pro13] *proCollab - Process-aware Support for Collaborative Knowledge Workers*. `http://www.uni-ulm.de/in/iui-dbis/forschung/projekte/procollab.html`. Version: 2013, Abruf: 05.07.2013

[Sch91] SCHNEIDER, Hans-Jochen: *Lexikon der Informatik und Datenverarbeitung*. 3. München : R. Oldenbourg, 1991

[SE11] SCHEIDERER, Joachim ; EBERMANN, Hans-Joachim: *Human Factors im Cockpit*. Heidelberg : Springer, 2011

[Ska06] SKACHKOVA, Mariya: *Einsatz von Checklisten in der Analytischen Qualitätssicherung*. Hannover, Gottfried Wilhelm Leibniz Universität Hannover, Bachelorarbeit, 2006

[Sur09] *Surgical Safety Checklist*. `http://public-health-kompakt.frappant.ch/wp-content/uploads/2012/04/Web.Abb_.3.4.1.png`. Version: 2009, Abruf: 07.08.2013

[Typ13] *Typekit*. `https://typekit.com/fonts/pt-sans-narrow`. Version: 2013, Abruf: 21.09.2013

[Wun13a] *Beispieldialog Wunderlist*. `http://i2.wp.com/www.mobiflip.de/wp-content/uploads/2012/12/wunderlist_2_screenshots.jpg`. Version: 2013, Abruf: 29.07.2013

[Wun13b] *Welcome to Wunderlist*. `https://www.wunderlist.com/de/`. Version: 2013, Abruf: 20.05.2013

[Xam13] *Xamarin*. `http://xamarin.com/`. Version: 2013, Abruf: 29.07.2013

Name: Sabrina Geiger Matrikelnummer: 724339

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Sabrina Geiger